Report D2.7

"TRO Final Release"

Grant Agreement: 958371

OntoCommons - Ontology-driven data documentation for Industry Commons, has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement no. 958371.

Project Title	Ontology-driven data documentation for Industry Commons
Project Acronym	OntoCommons
Project Number	958371
Type of project	CSA - Coordination and support action
Topics	DT-NMBP-39-2020 - Towards Standardised Documentation of Data through taxonomies and ontologies (CSA)
Starting date of Project	01 November 2020
Duration of the project	36 months
Website	www.ontocommons.eu

Report D2.7

"TRO Final Release"

Work Package	WP2 | TOP REFERENCE ONTOLOGY
Task	T2.4 | TRO Development
Lead author	Claudio Masolo (CNR)
Main Contributors	Francesco A. Zaccarini (UNIBO), Francesco Compagno (CNR)
Contributors	Stefano Borgo (CNR), Emanuele Ghedini (UNIBO), Nicola Guarino (CNR), Emilio M. Sanfilippo (CNR), Luca Biccheri (CNR)
Peer reviewers	Hedi Karray (ENIT), Nadja Adamovic (TU WIEN)
Version	Final
Submission Date	07/09/2023

Versioning History

Revision	Date	Editors	Comments
0.1	03/07/2023	Claudio Masolo (CNR), Francesco A. Zaccarini (UNIBO)	First complete draft
0.2	14/07/2023	Stefano Borgo (CNR), Emanuele Ghedini (UNIBO)	Improvement of the document
0.3	21/07/2023	Claudio Masolo (CNR), Francesco A. Zaccarini (UNIBO)	Improvement of the document
0.4	28/07/2023	Hedi Karray (ENIT)	Improvement of the document
0.5	31/07/2023	Nadja Adamovic (TU Wien, Coordinator)	Final approval and submission

Glossary of terms

Item	Description
BFO	Basic Formal Ontology
CL	Common Logic
DLO	Domain-Level Ontology
DOLCE	Descriptive Ontology for Linguistic and Cognitive Engineering
EMMO	Elementary Multiperspective Material Ontology
FOL	First-Order Logic
MLO	Middle-Level Ontology
OCES	Ontology Commons EcoSystem
OWL	Web Ontology Language
TLO	Top-Level Ontology
TRO	Top Reference Ontology

Keywords

Ontology; Top-Level Ontology, Top Reference ontology, BFO, DOLCE, EMMO, Alignment, Mapping

Disclaimer

Executive Summary

This report presents the alignment between the Top-Level Ontologies (TLOs) BFO, DOLCE, and EMMO available in first-order logic and OWL and used in the NMBP work programme domains of interest as individuated by the OntoCommons project. The report presents today's state of these ontologies and provides formal alignments between them both in first-order logic and in OWL.

Notation

We use expressions (**a**x), (**t**x) and (**d**x) to label axioms, theorems, and syntactic definitions, respectively. More specifically,

- ($a_d x$), ($t_d x$), ($d_d x$) are used for axioms, theorems, and definitions of the DOLCE ontology;
- ($a_b x$), ($t_b x$), ($d_b x$) are used for axioms, theorems, and definitions of the BFO ontology;
- ($a_e x$), ($t_e x$), ($d_e x$) are used for axioms, theorems, and definitions of the EMMO ontology;
- ($a_{db} x$), ($t_{db} x$), ($d_{db} x$) are used for axioms, theorems, and definitions used in the mapping from DOLCE to BFO, and
- ($a_{bd} x$), ($t_{bd} x$), ($d_{bd} x$) are used for axioms, theorems, and definitions used in the mapping from BFO to DOLCE.
- ($a_{ed} x$), ($t_{ed} x$), ($d_{ed} x$) are used for axioms, theorems, and definitions used in the mapping from EMMO to DOLCE, and
- ($a_{de} x$), ($t_{de} x$), ($d_{de} x$) are used for axioms, theorems, and definitions used in the mapping from DOLCE to EMMO.

Some of the notions used in DOLCE and EMMO are represented with the same predicate. In these cases, we maintained the original notation of EMMO and we added the subscript 'd' for the predicates of DOLCE.

Contents

1 Introduction

The vision of the OntoCommons project to foster the use of ontology in the NMBP areas focuses on a network of formal ontologies, called Ontology Commons EcoSystem (OCES). This network of ontologies is not existent today. Yet, over the last 30 years the ontology community has developed several elements that are needed to build this network. These elements are of different kinds: (a) ontologies in some formal language (starting from first-order logic, FOL, and its computational extensions to the Web Ontology Language, OWL) (b) software for ontology development, management, and update; (c) techniques for alignments across ontologies; (d) techniques for ontology modularization and integration. This report presents one of the missing components foreseen in OntoCommons for the creation of the OCES (see below). In this sense, it provides a building block for OCES.

OntoCommons adopts a distinction of ontologies in three levels: top-level ontologies, middle-level ontologies and domain level ontologies. Top-level ontologies (TLO) are general systems that provide a coherent and explicit way to read reality. They formalize very general terms (like object, event, property, constitution, dependence and so on) and are built according to a coherent set of principles. Due to their guiding role in understanding reality, top-level ontologies tend to be richly axiomatized and to be written in expressive languages, typically FOL (with or without modality). TLOs are always presented as independent conceptual systems, i.e., they are self-contained and self-sufficient.

Middle-level ontologies (MLO) are conceptual systems that aim to organize and characterize the vocabulary used in a discipline or area of interest like physics, engineering, medicine etc. Formally, MLOs can be given in any formal language, typically they are presented in FOL (or some fragment of it) or some version of OWL. MLOs are presented either as independent systems, in this case they assume an informal understanding of general concepts like object and event, or as extensions of TLOs. In the latter case, the main classes of the MLO are introduced as subclasses of the chosen TLO, and the main relations as specializations of relations in that TLO.

Domain-level ontologies (DLO) are conceptual systems that focus on a specific domain or application. Their goal is (a) to collect and organize the terminology needed by agents (human or artificial, including software) to properly operate in that domain or with that application; and (b) to add constraints to (automatically or semi-automatically) reason with those terms. A DLO assumes that the agents using the ontology already share a common view on how to interpret elements and happenings in that domain.

This report provides a presentation and alignment of the first-order versions of the TLOs selected by OntoCommons as backbones for the OCES system. The TLOs of interest are BFO, DOLCE, and EMMO.

1.1 Methodology

The purpose of the report is to present the consolidated formal systems of the three covered TLOs and the alignment across these systems. Indirectly, it also serves as

a guideline for the alignment of other TLOs that might be interested in joining the ontology network.

Keeping these targets in mind, the report was developed following these steps:

1. The versions in First Order Logic (FOL) of BFO, DOLCE, and EMMO ontologies were collected from the groups that developed them. When available, i.e., for BFO and DOLCE, we considered the Common Logic version. Common Logic is a consolidated standard (https://www.iso.org/standard/66249.html) whose visibility and stability increases users' trust in the OCES system.
2. Ontologies with a stable reference version were compared to the version in Common Logic to highlight relevant changes. This case applies to DOLCE whose reference version was released in 2003 in first-oder modal logic. The version in Common Logic is an adaptation due to the reduced expressivity of Common Logic.
3. Formal primitive relations were listed.
4. The FOL axiomatization was analyzed to ensure proper understanding of the intended interpretation.
5. Methodological choices were established and documented.
6. Formal mappings from one ontology to the other (and vice versa) were proposed and key expected consequences were tested using theorem provers whenever possible, manually otherwise.

2 BFO (BFO-2020 version 12 Nov 2021)

This section reports the axioms of BFO that are relevant for the mappings from DOLCE to BFO. In Sect. 6.1 we discuss the fact that some notions of BFO are not definable in DOLCE. It is important to observe that for the BFO to DOLCE mapping we consider the full BFO, see in particular the material in Sec. 7.2.

With BFO we refer here to the version of BFO 2020 of the 12 November 2021. Since formulas tend to be complex, for the primitive relations we adopt the predicates listed in Table 1 while for the universals we adopt the individual constants in Table 2. The taxonomy of BFO is depicted in Figure 1 (vertical lines represent ISA relationships, when solid they indicate a partition).

First note that inverse relations (e.g., hasContinuantPart is the inverse of continuantPartOf) is here captured by inverting the order of arguments, except for the temporal argument which is always the last one (when present), e.g., $\mathtt{cP}(x,y,t)$ vs. $\mathtt{cP}(y,x,t)$. Second, 'AtSomeTime' and 'AtAllTimes' relations are not included because they are 'defined' but never used in BFO-CL (the Common Logic version of BFO); note that they can be easily defined in FOL. Third, for relevant relations introduced in BFO-CL with 'if and only if' clauses, we introduce corresponding syntactic definitions: (d_b4) for continuantProperPartOf, (d_b6) for occurrentProperPartOf, (d_b8) for temporalProperPartOf, and (d_b10) for inheresIn. Fourth, to improve the readability of formulas, we introduce some syntactic definitions: (d_b1), (d_b2), (d_b3), (d_b5), (d_b7), (d_b9).

In the following, with $\mathfrak{B}$ we indicate the logical theory consisting of all the axioms in the Common Logic (Prover9) version of BFO 2020 / 12 November 2021.

The identifier of the axiom in the Common Logic version is indicated between square brackets.

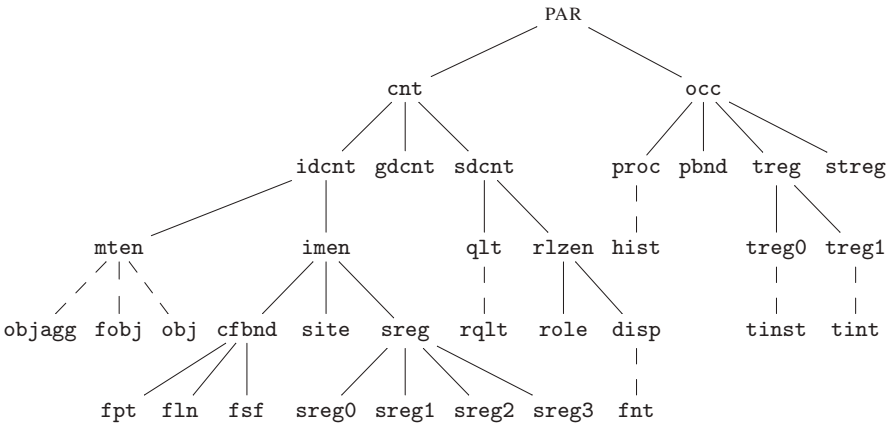

Figure 1: Taxonomy of BFO (vertical lines represent ISA relationships, when solid they indicate a partition).

$x ::_t u$	x is an instance of u at time t
$\mathtt{EX}(x,t)$	x exists at time t
$\mathtt{cP}(x,y,t)$	x is a continuant part of y at time t
$\mathtt{oP}(x,y)$	x is an occurrent part of y
$\mathtt{mP}(x,y)$	x is a member part of y
$\mathtt{tmP}(x,y)$	x is a temporal part of y
$\mathtt{STREG}(x,y)$	x occupies spatiotemporal region y
$\mathtt{SREG}(x,y,t)$	x occupies spatial region y at time t
$\mathtt{TREG}(x,y)$	x occupies temporal region y
$\mathtt{TPROJ}(x,y)$	x temporally projects onto y
$\mathtt{SPROJ}(x,y,t)$	x spatially projects onto y at time t
$\mathtt{OCCIN}(x,y)$	x occurs in y
$\mathtt{LOC}(x,y,t)$	x is located in y at time t
$\mathtt{SDEP}(x,y)$	x specifically depends on y
$\mathtt{CONCR}(x,y)$	x concretizes y
$\mathtt{GDEP}(x,y,t)$	x generically depends on y at time t
$\mathtt{PTC}(x,y,t)$	x participates in y at time t
$\mathtt{REAL}(x,y)$	x realizes y
$\mathtt{HIST}(x,y)$	x is the history of y
$\mathtt{PREC}(x,y)$	x precedes y
$\mathtt{FINST}(x,y)$	x is the first instant of y
$\mathtt{LINST}(x,y)$	x is the last instant of y
$\mathtt{MBAS}(x,y,t)$	x is the material basis of y at time t

Table 1: Primitive relations of BFO.

2.1 Particulars and Universals

- $\mathrm{PAR}(x)$ "x is a particular"
- $\mathrm{UNI}(x)$ "x is a universal"
- $\mathrm{ENT}(x)$ "x is an entity"

$\mathbf{a_b}$1 $\mathrm{PAR}(x) \vee \mathrm{UNI}(x)$ [eto-1]

$\mathbf{a_b}$2 $\mathrm{PAR}(x) \rightarrow \neg\mathrm{UNI}(x)$ [qkp-1]

2.2 instanceOf and existsAt

$\mathbf{d_b}$1 $\mathrm{TM}(x) \coloneqq x ::_x \mathtt{treg}$ [added]

$\mathbf{d_b}$2 $x :: u \coloneqq \exists t(x ::_t u)$ [added]

$\mathbf{d_b}$3 $\mathtt{ISA}(x,y) \coloneqq \forall zt(z ::_t x \rightarrow z ::_t y)$ [added]

$\mathbf{a_b}$3 $x ::_t u \rightarrow \mathrm{PAR}(x) \wedge \mathrm{UNI}(u) \wedge \mathrm{TM}(t)$ [ekp-1]

$\mathbf{a_b}$4 $\mathtt{EX}(x,t) \rightarrow \mathrm{PAR}(x) \wedge \mathrm{TM}(t)$ [oap-1]

cfbnd	Continuant Fiat Boundary	qlt	Quality
cnt	Continuant	qlt	Relational Quality
disp	Disposition	rlzen	Realizable Entity
fln	Fiat Line	role	Role
fobj	Fiat Object	sdcnt	Specifically Dep. Continuant
fpt	Fiat Point	site	Site
fsf	Fiat Surface	sreg	Spatial Region
fnt	Function	sreg0	0d Spatial Region
gdcnt	Generically Dep. Continuant	sreg1	1d Spatial Region
hist	History	sreg2	2d Spatial Region
idcnt	Independent Continuant	sreg3	3d Spatial Region
imen	Immaterial Entity	treg	Temporal Region
mten	Material Entity	treg0	0d Temporal Region
obj	Object	treg1	1d Temporal Region
objagg	Object Aggregate	streg	Spatiotemporal Region
PAR	Particular	tinst	Temporal Instant
pbnd	Process Boundary	tint	Temporal Interval
proc	Process		

Table 2: Categories of BFO.

a_b5 $\mathtt{EX}(x,t) \leftrightarrow \exists u(x ::_t u)$ [bee-1]

a_b6 $\mathrm{UNI}(u) \rightarrow \exists xt(x ::_t u)$ [mbf-1]

a_b7 $\mathrm{PAR}(x) \rightarrow \exists t(\mathtt{EX}(x,t))$ [nmq-1]

a_b8 $x ::\mathtt{mten} \rightarrow \exists t(t ::_t \mathtt{treg1} \wedge \mathtt{EX}(x,t))$ [zuw-1]

a_b9 $\mathrm{TM}(t) \rightarrow \exists xu(x \neq t \wedge x ::_t u)$ [nis-1]

a_b10 $x ::_t u \wedge \mathtt{tmP}(t',t) \rightarrow x ::_{t'} u$ [qaf-1]

a_b11 $(x ::\mathtt{cnt} \vee x ::\mathtt{occ}) \rightarrow \mathrm{ENT}(x)$ [vgn-1]

t_b1 $\mathrm{PAR}(x) \rightarrow \exists ut(x ::_t u)$ [added]
Proof. Trivially from (a_b5) and (a_b7). It a sort of dual of (a_b6). □

t_b2 $\mathtt{EX}(x,t) \wedge \mathtt{tmP}(t',t) \rightarrow \mathtt{EX}(x,t')$ [added]
Proof. By (a_b5) $\mathtt{EX}(x,t) \rightarrow \exists u(x ::_t u)$. By ($a_b10$), from $x ::_t u \wedge \mathtt{tmP}(t',t)$ we have $x ::_{t'} u$, i.e., $\exists u(x ::_{t'} u)$ that, by (a_b5), implies $\mathtt{EX}(x,t')$. □

2.3 continuantPartOf

d_b4 $\mathtt{cPP}(x,y,t) \coloneqq \mathtt{cP}(x,y,t) \wedge \neg\mathtt{cP}(y,x,t)$ [sls-1]

d_b5 $\mathtt{cO}(x,y,t) \coloneqq \exists z(\mathtt{cP}(z,x,t) \wedge \mathtt{cP}(z,y,t))$ [added]

a_b12 $\mathtt{cP}(x,y,t) \rightarrow x ::_t \mathtt{cnt} \wedge y ::_t \mathtt{cnt} \wedge \mathrm{TM}(t)$ [bdd-1]

a_b13 $x ::_t \mathtt{idcnt} \rightarrow \mathtt{cP}(x,x,t)$ [mcd-1]

a_b14 $\mathtt{cP}(x,y,t) \wedge \mathtt{cP}(y,z,t') \wedge \mathtt{tmP}(t,t') \rightarrow \mathtt{cP}(x,z,t)$ [plp-1]

$\mathbf{a_b15}$ $\mathtt{cP}(x,y,t) \wedge x \neq y \rightarrow \exists z(\mathtt{cP}(z,y,t) \wedge z \neq y \wedge \neg \mathtt{cO}(z,x,t))$ [fyf-1]

$\mathbf{a_b16}$ $\mathtt{cO}(x,y,t) \rightarrow \exists z(\forall w(\mathtt{cP}(w,z,t) \leftrightarrow \mathtt{cP}(w,x,t) \wedge \mathtt{cP}(w,y,t)))$ [gzr-1]

$\mathbf{a_b17}$ $\mathtt{cP}(x,y,t) \wedge \mathtt{tmP}(t',t) \rightarrow \mathtt{cP}(x,y,t')$ [mqp-1]

$\mathbf{a_b18}$ $\mathtt{cP}(x,y,t) \wedge \mathtt{EX}(y,t) \rightarrow \mathtt{EX}(x,t)$ [uns-1]

$\mathbf{a_b19}$ $\exists t(x ::_t \mathtt{objagg} \wedge \mathtt{cP}(x,y,t) \wedge \mathtt{cP}(y,x,t)) \wedge \forall t(\mathtt{cP}(x,y,t) \leftrightarrow \mathtt{cP}(y,x,t)) \rightarrow x = y$ [glc-1]

$\mathbf{a_b20}$ $\exists t(x ::_t \mathtt{idcnt} \wedge y ::_t \mathtt{idcnt} \wedge \neg(x ::_t \mathtt{objagg}) \wedge \neg(y ::_t \mathtt{objagg}) \wedge \mathtt{cP}(x,y,t) \wedge \mathtt{cP}(y,x,t)) \rightarrow x = y$ [tab-1]

$\mathbf{t_b3}$ $\mathtt{cP}(x,y,t) \rightarrow \mathtt{EX}(y,t) \wedge \mathtt{EX}(x,t)$ [added]

Proof. Assume $\mathtt{cP}(x,y,t)$, by ($\mathbf{a_b}$12), we have $x ::_t \mathtt{cnt}$ and $x ::_t \mathtt{cnt}$. By ($\mathbf{a_b}$5) $\mathtt{EX}(x,t)$ and $\mathtt{EX}(y,t)$ hold. □

continuantPartOf across universals

$\mathbf{a_b21}$ $x ::_t \mathtt{fobj} \leftrightarrow \neg(x ::_t \mathtt{imen}) \wedge \exists y(y ::_t \mathtt{obj} \wedge \mathtt{cPP}(x,y,t))$ [yir-1]

$\mathbf{a_b22}$ $\mathtt{cP}(x,y,t) \rightarrow (x ::_t \mathtt{idcnt} \leftrightarrow y ::_t \mathtt{idcnt})$ [cez-1]

$\mathbf{a_b23}$ $x ::_t \mathtt{mten} \wedge \exists y(\mathtt{cP}(y,x,t) \wedge x \neq y) \rightarrow \exists y(\mathtt{cP}(y,x,t) \wedge x \neq y \wedge \neg(y ::_t \mathtt{imen}))$ [adm-1]

$\mathbf{a_b24}$ $x ::_t \mathtt{mten} \wedge \mathtt{cP}(y,x,t) \rightarrow (y ::_t \mathtt{mten} \vee y ::_t \mathtt{site} \vee y ::_t \mathtt{cfbnd})$ [mic-1]

$\mathbf{a_b25}$ $x ::_t \mathtt{mten} \wedge \mathtt{cP}(x,y,t) \rightarrow y ::_t \mathtt{mten}$ [dok-1]

$\mathbf{a_b26}$ $x ::_t \mathtt{site} \wedge \mathtt{cP}(y,x,t) \rightarrow (y ::_t \mathtt{site} \vee y ::_t \mathtt{cfbnd})$ [mjj-1]

$\mathbf{a_b27}$ $x ::_t \mathtt{site} \wedge \mathtt{cP}(x,y,t) \rightarrow (y ::_t \mathtt{site} \vee y ::_t \mathtt{mten})$ [izr-1]

$\mathbf{a_b28}$ $\mathtt{cP}(x,y,t) \rightarrow (x ::_t \mathtt{sreg} \leftrightarrow y ::_t \mathtt{sreg})$ [kbr-1]

$\mathbf{a_b29}$ $x ::_t \mathtt{sreg3} \wedge \mathtt{cP}(y,x,t) \rightarrow y ::_t \mathtt{sreg}$ [fzg-1]

$\mathbf{a_b30}$ $x ::_t \mathtt{sreg2} \wedge \mathtt{cP}(y,x,t) \rightarrow (y ::_t \mathtt{sreg2} \vee y ::_t \mathtt{sreg1} \vee y ::_t \mathtt{sreg0})$ [hbn-1]

$\mathbf{a_b31}$ $x ::_t \mathtt{sreg1} \wedge \mathtt{cP}(y,x,t) \rightarrow (y ::_t \mathtt{sreg1} \vee y ::_t \mathtt{sreg0})$ [wne-1]

$\mathbf{a_b32}$ $x ::_t \mathtt{sreg0} \wedge \mathtt{cP}(y,x,t) \rightarrow (y ::_t \mathtt{sreg0})$ [bfv-1]

$\mathbf{a_b33}$ $x ::_t \mathtt{cfbnd} \wedge \mathtt{cP}(y,x,t) \rightarrow y ::_t \mathtt{cfbnd}$ [ixo-1]

$\mathbf{a_b34}$ $x ::_t \mathtt{fsf} \wedge \mathtt{cP}(y,x,t) \rightarrow y ::_t \mathtt{cfbnd}$ [ysp-1]

$\mathbf{a_b35}$ $x ::_t \mathtt{fln} \wedge \mathtt{cP}(y,x,t) \rightarrow (y ::_t \mathtt{fln} \vee y ::_t \mathtt{fpt})$ [cwp-1]

$\mathbf{a_b36}$ $x ::_t \mathtt{fpt} \wedge \mathtt{cP}(y,x,t) \rightarrow x = y$ [jgo-1]

2.4 occurrentPartOf

$\mathbf{d_b6}$ $\mathtt{oPP}(x,y) \coloneqq \mathtt{oP}(x,y) \wedge x \neq y$ [okr-1]

$\mathbf{d_b7}$ $\mathtt{oO}(x,y) \coloneqq \exists z(\mathtt{oP}(z,x) \wedge \mathtt{oP}(z,y))$ [added]

$\mathbf{a_b37}$ $\mathtt{oP}(x,y) \rightarrow x ::\mathtt{occ} \wedge y ::\mathtt{occ}$ [zmr-1]

$\mathbf{a_b38}$ $x ::\mathtt{occ} \rightarrow \mathtt{oP}(x,x)$ [hbj-1]

$\mathbf{a_b39}$ $\mathtt{oP}(x,y) \wedge \mathtt{oP}(y,x) \rightarrow x = y$ [xlu-1]

$\mathbf{a_b40}$ $\mathtt{oP}(x,y) \wedge \mathtt{oP}(y,z) \rightarrow \mathtt{oP}(x,z)$ [kad-1]

$\mathbf{a_b41}$ $\mathtt{oO}(x,y) \rightarrow \exists z(\forall w(\mathtt{oP}(w,z) \leftrightarrow \mathtt{oP}(w,x) \wedge \mathtt{oP}(w,y)))$ [hpc-1]

$\mathbf{a_b42}$ $\mathtt{oP}(x,y) \wedge \mathtt{EX}(x,t) \rightarrow \mathtt{EX}(y,t)$ [ybr-1]

occurentPartOf across universals

$\mathbf{a_b43}$ $x::\mathtt{proc} \wedge \mathtt{oP}(x,y) \rightarrow y::\mathtt{proc}$ [csk-1]

$\mathbf{a_b44}$ $x::\mathtt{proc} \wedge \mathtt{oP}(y,x) \rightarrow (y::\mathtt{proc} \vee y::\mathtt{pbnd})$ [ccz-1]

$\mathbf{a_b45}$ $x::\mathtt{proc} \rightarrow \exists y(y::\mathtt{pbnd} \wedge \mathtt{oP}(y,x))$ [aff-1]

$\mathbf{a_b46}$ $x::\mathtt{pbnd} \wedge \mathtt{oP}(x,y) \rightarrow (y::\mathtt{pbnd} \vee y::\mathtt{proc})$ [ptm-1]

$\mathbf{a_b47}$ $x::\mathtt{pbnd} \wedge \mathtt{oP}(y,x) \rightarrow y::\mathtt{pbnd}$ [hdk-1]

$\mathbf{a_b48}$ $\mathtt{oP}(x,y) \rightarrow (x::\mathtt{treg} \leftrightarrow y::\mathtt{treg})$ [gjl-1]

$\mathbf{a_b49}$ $\mathtt{oP}(x,y) \rightarrow (x::\mathtt{streg} \leftrightarrow y::\mathtt{streg})$ [myl-1]

2.5 temporalPartOf and occupiesTemporalRegion

$\mathbf{d_b8}$ $\mathtt{tmPP}(x,y) \coloneqq \mathtt{tmP}(x,y) \wedge x \neq y$ [aeu-1]

$\mathbf{d_b9}$ $\mathtt{tmO}(x,y) \coloneqq \exists z(\mathtt{tmP}(z,x) \wedge \mathtt{tmP}(z,y))$ added

$\mathbf{a_b50}$ $\mathtt{tmP}(x,y) \rightarrow x::\mathtt{occ} \wedge y::\mathtt{occ}$ [ruj-1]

$\mathbf{a_b51}$ $\mathtt{TREG}(x,t) \rightarrow (x::\mathtt{proc} \vee x::\mathtt{pbnd}) \wedge \mathrm{TM}(t)$ [lyx-1]

$\mathbf{a_b52}$ $x::\mathtt{occ} \rightarrow \mathtt{tmP}(x,x)$ [bvr-1]

$\mathbf{a_b53}$ $\mathtt{tmP}(x,y) \wedge \mathtt{tmP}(y,x) \rightarrow x = y$ [zdq-1]

$\mathbf{a_b54}$ $\mathtt{tmP}(x,y) \wedge \mathtt{tmP}(y,z) \rightarrow \mathtt{tmP}(x,z)$ [bfq-1]

$\mathbf{a_b55}$ $x::_x\mathtt{treg} \wedge y::_y\mathtt{treg} \wedge \mathtt{tmPP}(x,y) \rightarrow \exists z(\mathtt{tmPP}(z,y) \wedge \neg\mathtt{tmO}(z,x))$ [vbw-1]

$\mathbf{a_b56}$ $\mathtt{tmP}(x,y) \rightarrow \mathtt{oP}(x,y)$ [bal-1]

$\mathbf{a_b57}$ $\mathtt{tmP}(x,y) \wedge \mathtt{EX}(x,t) \rightarrow \mathtt{EX}(y,t)$ [jqz-1]

$\mathbf{a_b58}$ $x::\mathtt{treg} \wedge y::\mathtt{treg} \rightarrow (\mathtt{tmP}(x,y) \leftrightarrow \mathtt{oP}(x,y))$ [cmy-1]

$\mathbf{a_b59}$ $t' ::_t \mathtt{treg} \rightarrow \mathtt{tmP}(t,t')$ [njq-1]

$\mathbf{a_b60}$ $x::\mathtt{pbnd} \leftrightarrow \exists y(\mathtt{tmP}(x,y) \wedge y::\mathtt{proc}) \wedge \exists t(\mathtt{TREG}(x,t) \wedge t::_t\mathtt{tinst})$ [esh-1]

$\mathbf{a_b61}$ $x::\mathtt{pbnd} \wedge \mathtt{TREG}(x,t) \rightarrow t::_t\mathtt{tinst}$ [atz-1]

$\mathbf{a_b62}$ $x::\mathtt{proc} \wedge \mathtt{TREG}(x,t) \rightarrow \exists t'(t'::_{t'}\mathtt{tint} \wedge \mathtt{tmP}(t',t))$ [fzy-1]

$\mathbf{a_b63}$ $(x::\mathtt{proc} \vee x::\mathtt{pbnd}) \wedge \mathrm{TM}(t) \rightarrow$
$(\mathtt{TREG}(x,t) \leftrightarrow \forall yt'(\mathtt{oP}(y,x) \wedge \mathtt{TREG}(y,t') \rightarrow \mathtt{oP}(t',t)) \wedge$
$\neg\exists t'(\mathtt{oPP}(t',t) \wedge \mathtt{TREG}(x,t')))$ [tao-1]

$\mathbf{a_b64}$ $\mathtt{oP}(x,y) \wedge \mathtt{TREG}(x,t) \wedge \mathtt{TREG}(y,t') \rightarrow \mathtt{oP}(t,t')$ [jiv-1]

$\mathbf{a_b65}$ $\mathtt{tmPP}(x,y) \rightarrow \neg\mathtt{tmPP}(y,x)$ [aqu-1]

$\mathbf{a_b66}$ $\mathtt{tmPP}(x,y) \wedge \mathtt{tmPP}(y,z) \rightarrow \mathtt{tmPP}(x,z)$ [mns-1]

2.6 memberPartOf

a$_b$67 $\mathtt{mP}(x,y,t) \rightarrow x ::_t \mathtt{obj} \wedge y ::_t \mathtt{objagg} \wedge \mathrm{TM}(t)$ [dvq-1]

a$_b$68 $\mathtt{mP}(x,y,t) \wedge \mathtt{tmP}(t',t) \rightarrow \mathtt{mP}(x,y,t')$ [yip-1]

a$_b$69 $x :: \mathtt{objagg} \rightarrow \exists yzt(y \neq z \wedge \mathtt{mP}(y,x,t) \wedge \mathtt{mP}(z,x,t))$ [ibd-1]

a$_b$70 $x ::_t \mathtt{objagg} \rightarrow \exists y(\mathtt{mP}(y,x,t))$ [uhs-1]

a$_b$71 $\mathtt{mP}(x,y,t) \leftrightarrow$ [evk-1]
$x ::_t \mathtt{obj} \wedge y ::_t \mathtt{objagg} \wedge \mathtt{cP}(x,y,t) \wedge \forall z(\mathtt{mP}(z,y,t) \rightarrow (z = x \vee \neg \mathtt{cO}(z,x,t)))$

a$_b$72 $\mathtt{cPP}(x,y,t) \wedge y ::_t \mathtt{objagg} \rightarrow \exists z(\mathtt{mP}(z,y,t) \wedge \mathtt{cO}(z,x,t))$ [fsy-1]

2.7 occupiesSpatialRegion and locatedIn

a$_b$73 $\mathtt{SREG}(x,y,t) \rightarrow x ::_t \mathtt{idcnt} \wedge \neg(x ::_t \mathtt{sreg}) \wedge y ::_t \mathtt{sreg} \wedge \mathrm{TM}(t)$ [lzw-1]

a$_b$74 $\mathtt{LOC}(x,y,t) \rightarrow x ::_t \mathtt{idcnt} \wedge \neg(x ::_t \mathtt{sreg}) \wedge y ::_t \mathtt{idcnt} \wedge \neg(y ::_t \mathtt{sreg}) \wedge \mathrm{TM}(t)$ [bge-1]

a$_b$75 $\mathtt{SREG}(x,y,t) \wedge \mathtt{SREG}(x,z,t) \rightarrow y = z$ [zls-1]

a$_b$76 $\mathtt{SREG}(x,y,t) \wedge \mathtt{tmP}(t',t) \rightarrow \mathtt{SREG}(x,y,t')$ [mud-1]

a$_b$77 $\mathtt{LOC}(x,y,t) \rightarrow \exists t'rs(\mathtt{tmP}(t',t) \wedge \mathtt{SREG}(x,r,t') \wedge \mathtt{SREG}(y,s,t') \wedge \mathtt{cP}(r,s,t'))$ [uas-1]

a$_b$78 $x ::_t \mathtt{mten} \wedge y ::_t \mathtt{mten} \wedge \mathtt{SREG}(x,r,t) \wedge \mathtt{SREG}(y,r,t) \rightarrow \mathtt{cP}(x,y,t) \wedge \mathtt{cP}(y,x,t)$ [scr-1]

a$_b$79 $\mathtt{LOC}(x,y,t) \wedge \mathtt{tmP}(t',t) \rightarrow \mathtt{LOC}(x,y,t')$ [put-1]

a$_b$80 $\mathtt{LOC}(x,y,t) \wedge \mathtt{LOC}(y,z,t') \wedge \mathtt{tmP}(t,t') \rightarrow \mathtt{LOC}(x,z,t)$ [xlm-1]

a$_b$81 $\mathtt{LOC}(x,y,t) \wedge \mathtt{cP}(y,z,t) \rightarrow \mathtt{LOC}(x,z,t)$ [evu-1]

a$_b$82 $\mathtt{LOC}(x,y,t) \wedge \mathtt{cP}(z,x,t) \rightarrow \mathtt{LOC}(z,y,t)$ [wty-1]

a$_b$83 $x ::_t \mathtt{idcnt} \wedge \neg(x ::_t \mathtt{sreg}) \wedge y ::_t \mathtt{idcnt} \wedge \neg(y ::_t \mathtt{sreg}) \wedge \mathtt{cP}(x,y,t) \rightarrow \mathtt{LOC}(x,y,t)$ [bao-1]

2.8 occupiesSpatiotemporalRegion, temporallyProjectsOnto, spatiallyProjectsOnto, and occursIn

a$_b$84 $\mathtt{STREG}(x,y) \rightarrow (x :: \mathtt{proc} \vee x :: \mathtt{pbnd}) \wedge y :: \mathtt{streg}$ [vvo-1]

a$_b$85 $\mathtt{TPROJ}(x,y) \rightarrow x :: \mathtt{streg} \wedge \mathrm{TM}(y)$ [cvr-1]

a$_b$86 $\mathtt{SPROJ}(x,y,t) \rightarrow x ::_t \mathtt{streg} \wedge y ::_t \mathtt{sreg} \wedge \mathrm{TM}(t)$ [blj-1]

a$_b$87 $\mathtt{OCCIN}(x,y) \rightarrow (x :: \mathtt{proc} \vee x :: \mathtt{pbnd}) \wedge (y :: \mathtt{mten} \vee y :: \mathtt{site})$ [tfw-1]

a$_b$88 $(x :: \mathtt{proc} \vee x :: \mathtt{pbnd}) \rightarrow \exists r(\mathtt{STREG}(x,r))$ [qyy-1]

a$_b$89 $\mathtt{STREG}(x,y) \wedge \mathtt{STREG}(x,z) \rightarrow y = z$ [uqt-1]

a$_b$90 $\mathtt{oP}(x,y) \wedge \mathtt{STREG}(x,r) \wedge \mathtt{STREG}(y,s) \rightarrow \mathtt{oP}(r,s)$ [iqe-1]

a$_b$91 $(x :: \mathtt{proc} \vee x :: \mathtt{pbnd}) \wedge (y :: \mathtt{proc} \vee y :: \mathtt{pbnd}) \rightarrow$ [kqv-1]
$(\mathtt{oP}(x,y) \leftrightarrow \exists rs(\mathtt{STREG}(x,r) \wedge \mathtt{STREG}(y,s) \wedge \mathtt{oP}(r,s)))$

$\mathbf{a_b92}$ $\texttt{SPROJ}(x,s,t) \wedge \texttt{SPROJ}(x,r,t) \rightarrow s = r$ [fdb-1]

$\mathbf{a_b93}$ $x ::_t \texttt{streg} \rightarrow \exists st'(\texttt{tmP}(t',t) \wedge \texttt{SPROJ}(x,s,t'))$ [geq-1]

$\mathbf{a_b94}$ $x :: \texttt{streg} \wedge y :: \texttt{streg} \rightarrow (\texttt{oP}(x,y) \leftrightarrow$ [txf-1]
$\exists tu(\texttt{TPROJ}(x,t) \wedge \texttt{TPROJ}(y,u) \wedge \texttt{tmP}(t,u)) \wedge$
$\forall v(\texttt{EX}(x,v) \rightarrow \exists wsr(\texttt{tmP}(w,v) \wedge \texttt{SPROJ}(x,s,w) \wedge \texttt{SPROJ}(y,r,w) \wedge \texttt{cP}(s,r,w))))$

$\mathbf{a_b95}$ $\texttt{OCCIN}(x,y) \wedge \texttt{EX}(x,t) \rightarrow \texttt{EX}(y,t)$ [dxv-1]

$\mathbf{a_b96}$ $\texttt{OCCIN}(x,y) \wedge \texttt{oP}(z,x) \rightarrow \texttt{OCCIN}(z,y)$ [jil-1]

$\mathbf{a_b97}$ $\texttt{OCCIN}(x,y) \wedge \forall t(\texttt{EX}(x,t) \leftrightarrow \texttt{cP}(y,z,t)) \rightarrow \texttt{OCCIN}(x,z)$ [czc-1]

$\mathbf{a_b98}$ $\texttt{OCCIN}(x,y) \wedge \forall t(\texttt{EX}(x,t) \leftrightarrow \texttt{LOC}(y,z,t)) \rightarrow \texttt{OCCIN}(x,z)$ [yex-1]

2.9 specificallyDependsOn

$\mathbf{d_b10}$ $\texttt{INH}(x,y) \coloneqq \texttt{SDEP}(x,y) \wedge x :: \texttt{sdcnt} \wedge y :: \texttt{idcnt} \wedge \neg(y :: \texttt{sreg})$ (inherence) [tht-1]

$\mathbf{a_b99}$ $\texttt{SDEP}(x,y) \rightarrow x :: \texttt{sdcnt} \wedge (y :: \texttt{sdcnt} \vee (y :: \texttt{idcnt} \wedge \neg(y :: \texttt{sreg})))$ [kkl-1]

$\mathbf{a_b100}$ $\texttt{SDEP}(x,y) \rightarrow \neg\exists t(\texttt{cO}(x,y,t))$ [nfe-1]

$\mathbf{a_b101}$ $\texttt{SDEP}(x,y) \wedge \texttt{SDEP}(y,z) \wedge x \neq z \rightarrow \texttt{SDEP}(x,z)$ [myu-1]

$\mathbf{a_b102}$ $\texttt{SDEP}(x,y) \rightarrow \exists t(\texttt{EX}(x,t) \wedge \texttt{EX}(y,t)) \wedge \forall t(\texttt{EX}(x,t) \rightarrow \texttt{EX}(y,t))$ [iyu-1]

$\mathbf{a_b103}$ $x ::_t \texttt{idcnt} \leftrightarrow x ::_t \texttt{cnt} \wedge \neg\exists yt'(\texttt{SDEP}(x,y) \vee \texttt{GDEP}(x,y,t'))$ [ilw-1]

$\mathbf{a_b104}$ $x ::_t \texttt{sdcnt} \leftrightarrow x ::_t \texttt{cnt} \wedge \exists y(y ::_t \texttt{idcnt} \wedge \neg(y ::_t \texttt{sreg}) \wedge \texttt{SDEP}(x,y))$ [akq-1]

2.10 concretizes and genericallyDependsOn

$\mathbf{a_b105}$ $\texttt{CONCR}(x,y,t) \rightarrow (x ::_t \texttt{sdcnt} \vee x ::_t \texttt{proc}) \wedge y ::_t \texttt{gdcnt} \wedge \texttt{TM}(t)$ [rog-1]

$\mathbf{a_b106}$ $\texttt{GDEP}(x,y,t) \rightarrow x ::_t \texttt{gdcnt} \wedge y ::_t \texttt{idcnt} \wedge \neg(y ::_t \texttt{sreg}) \wedge \texttt{TM}(t)$ [ekp-1]

$\mathbf{a_b107}$ $x ::_t \texttt{gdcnt} \rightarrow \exists yt'(\texttt{tmP}(t',t) \wedge \texttt{CONCR}(y,x,t'))$ [ibk-1]

$\mathbf{a_b108}$ $\texttt{CONCR}(x,y,t) \wedge \texttt{tmP}(t',t) \rightarrow \texttt{CONCR}(x,y,t')$ [nyz-1]

$\mathbf{a_b109}$ $\texttt{GDEP}(x,y,t) \rightarrow \exists zt'(\texttt{tmP}(t',t) \wedge \texttt{INH}(z,y) \wedge \texttt{CONCR}(z,x,t'))$ [otx-1]

$\mathbf{a_b110}$ $\texttt{CONCR}(x,y,t) \wedge \texttt{INH}(x,z) \rightarrow \texttt{GDEP}(y,z,t)$ [cik-1]

$\mathbf{a_b111}$ $x ::_t \texttt{gdcnt} \wedge \texttt{PTC}(x,p,t) \rightarrow \exists yt'(\texttt{tmP}(t',t) \wedge \texttt{CONCR}(y,x,t') \wedge$
$((y ::_{t'} \texttt{sdcnt} \wedge \exists z(\texttt{SDEP}(y,z) \wedge \texttt{PTC}(z,p,t'))) \vee (\texttt{oP}(y,p) \wedge \texttt{EX}(y,t'))))$ [fmm-1]

2.11 participatesIn

$\mathbf{a_b112}$ $\texttt{PTC}(x,y,t) \rightarrow x ::_t \texttt{cnt} \wedge \neg(x ::_t \texttt{sreg}) \wedge y ::_t \texttt{proc} \wedge \texttt{TM}(t)$ [ild-1]

$\mathbf{a_b113}$ $x ::_t \texttt{proc} \rightarrow \exists y(\texttt{PTC}(y,x,t))$ [trl-1]

$\mathbf{a_b114}$ $\texttt{PTC}(x,y,t) \wedge \texttt{tmP}(t',t) \rightarrow \texttt{PTC}(x,y,t')$ [yjm-1]

$\mathbf{a_b115}$ $x ::_t \texttt{sdcnt} \wedge \texttt{PTC}(x,p,t) \rightarrow$ [cgn-1]
$\exists yt'(\texttt{tmP}(t',t) \wedge y ::_{t'} \texttt{idcnt} \wedge \neg(y ::_{t'} \texttt{sreg}) \wedge \texttt{SDEP}(x,y) \wedge \texttt{PTC}(y,p,t'))$

2.12 historyOf

a$_b$116 $\texttt{HIST}(x,y) \rightarrow x::\texttt{hist} \wedge y::\texttt{mten}$ [rph-1]

a$_b$117 $\texttt{HIST}(x,y) \wedge \texttt{HIST}(x,z) \rightarrow y = z$ [zek-1]

a$_b$118 $\texttt{HIST}(x,z) \wedge \texttt{HIST}(y,z) \rightarrow x = y$ [woe-1]

a$_b$119 $x::_t\texttt{hist} \rightarrow \exists y(\texttt{HIST}(x,y))$ [vvy-1]

a$_b$120 $x::_t\texttt{mten} \rightarrow \exists y(\texttt{HIST}(y,x))$ [okt-1]

a$_b$121 $\texttt{HIST}(x,y) \wedge \texttt{EX}(y,t) \rightarrow \texttt{PTC}(y,x,t)$ [lga-1]

a$_b$122 $\texttt{HIST}(x,y) \rightarrow (\texttt{EX}(x,t) \leftrightarrow \texttt{EX}(y,t))$ [uzz-1]

2.13 realizes

a$_b$123 $\texttt{REAL}(x,y) \rightarrow x::\texttt{proc} \wedge y::\texttt{rlzen}$ [oot-1]

a$_b$124 $\texttt{REAL}(x,y) \wedge \texttt{INH}(y,z) \rightarrow \exists t(\texttt{PTC}(z,x,t))$ [grx-1]

2.14 Taxonomy of universals

In BFO all the universals in the taxonomy of BFO are 'rigid' in the sense captured in (a$_b$125) for the continuants except obj, fobj, and objagg. In addition, all the universals directly subsumed by a given universal cover the whole root and are disjoint except obj, fobj, and objagg for which the disjointness is not explicitly stated. Here we consider only the axioms concerning the universals of BFO that are involved in the mappings.

2.14.1 Rigidity

a$_b$125 $x::\texttt{cnt} \rightarrow \forall t(\texttt{EX}(x,t) \rightarrow x::_t\texttt{cnt})$

a$_b$126 $x::\texttt{occ} \rightarrow \forall t(\texttt{EX}(x,t) \rightarrow x::_t\texttt{occ})$

a$_b$127 $x::\texttt{idcnt} \rightarrow \forall t(\texttt{EX}(x,t) \rightarrow x::_t\texttt{idcnt})$

a$_b$128 $x::\texttt{gdcnt} \rightarrow \forall t(\texttt{EX}(x,t) \rightarrow x::_t\texttt{gdcnt})$

a$_b$129 $x::\texttt{sdcnt} \rightarrow \forall t(\texttt{EX}(x,t) \rightarrow x::_t\texttt{sdcnt})$

a$_b$130 $x::\texttt{mten} \rightarrow \forall t(\texttt{EX}(x,t) \rightarrow x::_t\texttt{mten})$

a$_b$131 $x::\texttt{imen} \rightarrow \forall t(\texttt{EX}(x,t) \rightarrow x::_t\texttt{imen})$

a$_b$132 $x::\texttt{site} \rightarrow \forall t(\texttt{EX}(x,t) \rightarrow x::_t\texttt{site})$

a$_b$133 $x::\texttt{cfbnd} \rightarrow \forall t(\texttt{EX}(x,t) \rightarrow x::_t\texttt{cfbnd})$

a$_b$134 $x::\texttt{sreg} \rightarrow \forall t(\texttt{EX}(x,t) \rightarrow x::_t\texttt{sreg})$

a$_b$135 $x::\texttt{proc} \rightarrow \forall t(\texttt{EX}(x,t) \rightarrow x::_t\texttt{proc})$

a$_b$136 $x::\texttt{pbnd} \rightarrow \forall t(\texttt{EX}(x,t) \rightarrow x::_t\texttt{pbnd})$

a$_b$137 $x::\texttt{treg} \rightarrow \forall t(\texttt{EX}(x,t) \rightarrow x::_t\texttt{treg})$

a$_b$138 $x::\texttt{streg} \rightarrow \forall t(\texttt{EX}(x,t) \rightarrow x::_t\texttt{streg})$

$\mathbf{a_b139}$ $x::\texttt{tint} \rightarrow \forall t(\texttt{EX}(x,t) \rightarrow x::_t \texttt{tint})$

$\mathbf{a_b140}$ $x::\texttt{tinst} \rightarrow \forall t(\texttt{EX}(x,t) \rightarrow x::_t \texttt{tinst})$

2.14.2 Disjointness

$\mathbf{a_b141}$ $\neg\exists x(x::\texttt{cnt} \wedge x::\texttt{occ})$

$\mathbf{a_b142}$ $\neg\exists x(x::\texttt{idcnt} \wedge x::\texttt{gdcnt})$

$\mathbf{a_b143}$ $\neg\exists x(x::\texttt{idcnt} \wedge x::\texttt{sdcnt})$

$\mathbf{a_b144}$ $\neg\exists x(x::\texttt{sdcnt} \wedge x::\texttt{gdcnt})$

$\mathbf{a_b145}$ $\neg\exists x(x::\texttt{mten} \wedge x::\texttt{imen})$

$\mathbf{a_b146}$ $\neg\exists x(x::\texttt{site} \wedge x::\texttt{cfbnd})$

$\mathbf{a_b147}$ $\neg\exists x(x::\texttt{site} \wedge x::\texttt{sreg})$

$\mathbf{a_b148}$ $\neg\exists x(x::\texttt{sreg} \wedge x::\texttt{cfbnd})$

$\mathbf{a_b149}$ $\neg\exists x(x::\texttt{proc} \wedge x::\texttt{pbnd})$

$\mathbf{a_b150}$ $\neg\exists x(x::\texttt{proc} \wedge x::\texttt{treg})$

$\mathbf{a_b151}$ $\neg\exists x(x::\texttt{proc} \wedge x::\texttt{streg})$

$\mathbf{a_b152}$ $\neg\exists x(x::\texttt{pbnd} \wedge x::\texttt{treg})$

$\mathbf{a_b153}$ $\neg\exists x(x::\texttt{pbnd} \wedge x::\texttt{streg})$

$\mathbf{a_b154}$ $\neg\exists x(x::\texttt{treg} \wedge x::\texttt{streg})$

$\mathbf{a_b155}$ $\neg\exists x(x::\texttt{tinst} \wedge x::\texttt{tint})$

2.14.3 Inclusions

$\mathbf{a_b156}$ $x::\texttt{cnt} \leftrightarrow (x::\texttt{idcnt} \vee x::\texttt{gdcnt} \vee x::\texttt{sdcnt})$

$\mathbf{a_b157}$ $x::\texttt{occ} \leftrightarrow (x::\texttt{proc} \vee x::\texttt{pbnd} \vee x::\texttt{treg} \vee x::\texttt{streg})$

$\mathbf{a_b158}$ $x::\texttt{idcnt} \leftrightarrow (x::\texttt{mten} \vee x::\texttt{imen})$

$\mathbf{a_b159}$ $x::\texttt{imen} \leftrightarrow (x::\texttt{site} \vee x::\texttt{cfbnd} \vee x::\texttt{sreg})$

$\mathbf{a_b160}$ $x::\texttt{tint} \rightarrow x::\texttt{treg}$

$\mathbf{a_b161}$ $x::\texttt{tinst} \rightarrow x::\texttt{treg}$

3 DOLCE (extension of the ISO-CL version)

With respect to DOLCE-D18, DOLCE-ISO has two main simplifications:

(1) Modality operators are not available in the language of Common Logic.
Formal consequences: the reference version of DOLCE (DOLCE-D18) is a first order *modal* theory relying on the modal logic QS5 with constant domain, without modal operators the intended models of DOLCE change;

(2) The mereological fusion operator is not available in the language of Common Logic.
Formal consequences: given a property expressible in the theory, the reference version of DOLCE (DOLCE-D18) assumes the existence of the mereological sum of all the entities that satisfy this property, this kind of entity cannot be ensured to exist in logics without the fusion operator.

The first simplification dramatically weakens several notions of dependence that are strongly grounded on modality. To partially overcome this problem, we introduce the additional primitive of temporary existential dependence (EXD) that, however, differs from the existential dependence in DOLCE-D18 in several aspects as we will see in Section 3.9.

Concerning the second simplification, to partially overcome the lack of the fusion operation we add the axiom of strong supplementation for $\mathtt{P_d}$ (parthood simpliciter) and $\mathtt{tP_d}$ (temporary parthood) obtaining a theory based on extensional mereologies (EM) rather than general extensional mereologies (GEM) as in the case of DOLCE-D18. Notice that axioms guaranteeing the closure of the domain under binary sums and products are not included in DOLCE. The user is free to add the sum and product operators whenever needed. Second, some important notions defined in DOLCE-D18 by using mereological fusion are now introduced as primitives; in particular, this applies to $\mathtt{TLC}(x,t)$, read as "x is (exactly) located at time t", and $\mathtt{SLC}(x,s,t)$, "at time t, x is (exactly) located at space s".

Following DOLCE-ISO, two additional simplifications are adopted:

– only direct qualities are considered (DQT);

– (Ad56), (Ad57), (Ad63), and (Ad64) in DOLCE-D18 are instantiated only by temporal locations (TL) and time intervals (T) and by spatial locations (SL) and space regions (S), i.e., all the quality leaves explicitly introduced in D18.

In the following, with DOLCE we indicate the logical theory $\mathfrak{D}$ consisting of the axioms (a_d1)-(a_d131) together with the syntactic definitions (d_d1)-(d_d10) introduced in the Sections 3.1-3.11. Table 3 lists the primitive relations of DOLCE, Table 4 lists the categories of DOLCE, and Figure 2 shows the taxonomy of DOLCE (vertical lines represent ISA relationships, when solid they indicate a partition).

We introduce also some theorems. The proofs (generated with the help of theorem provers), are listed in Sect. 11.1.

$\mathtt{DQT}(x,y)$	x is a direct quality of y
$\mathtt{EXD}(x,y,t)$	x is existentially dependent on y at time t
$\mathtt{K}(x,y,t)$	x constitutes y at time t
$\mathtt{P_d}(x,y)$	x is part of y
$\mathtt{PC}(x,y,t)$	x participates in y at time t
$\mathtt{QL}(x,y)$	x is the immediate quale of y
$\mathtt{SLC}(x,y,t)$	x is (exactly) located at space y at time t
$\mathtt{TLC}(x,t)$	x is (exactly) located at time t
$\mathtt{tP_d}(x,y,t)$	x is part of y at time t
$\mathtt{tQL}(x,y,t)$	x is the temporary quale of y at time t

Table 3: Primitive relations of DOLCE.

AB	Abstract
ACC	Accomplishment
ACH	Achievement
APO	Agentive Physical Object
AQ	Abstract Quality
AR	Abstract Region
AS	Arbitrary Sum
ASO	Agentive Social Object
ED	Endurant
EV	Event
F	Feature
M	Amount Of Matter
MOB	Mental Object
NAPO	Non-agentive Physical Object
NASO	Non-agentive Social Object
NPED	Non-physical Endurant
NPOB	Non-physical Object
PD	Perdurant

PED	Physical Endurant
POB	Physical Object
PQ	Physical Quality
PR	Physical Region
PRO	Process
Q	Quality
R	Region
S	Space Region
SAG	Social Agent
SC	Society
SOB	Social Object
SL	Spatial Location
ST	State
STV	Stative
T	Time Interval
TQ	Temporal Quality
TL	Temporal Location
TR	Temporal Region

Table 4: Categories of DOLCE.

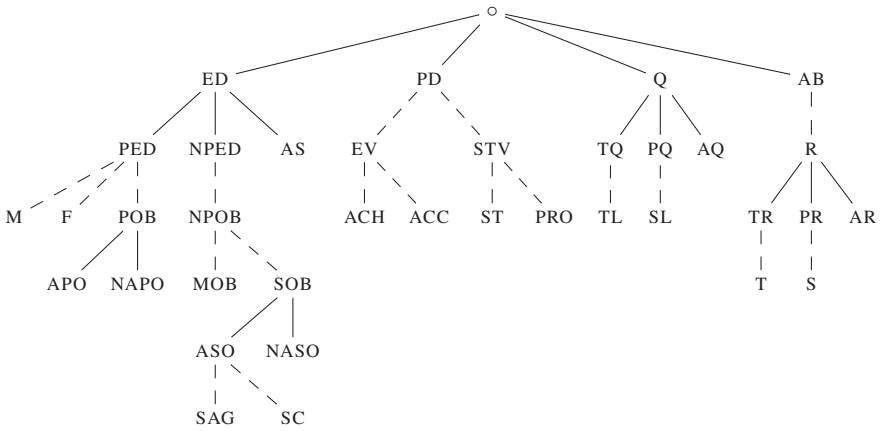

Figure 2: Taxonomy of DOLCE (vertical lines represent ISA relationships, when solid they indicate a partition).

3.1 Mereology

d$_d$1 $\mathtt{PP}_d(x,y) \coloneqq \mathtt{P}_d(x,y) \wedge \neg \mathtt{P}_d(y,x)$

d$_d$2 $\mathtt{O}_d(x,y) \coloneqq \exists z(\mathtt{P}_d(z,x) \wedge \mathtt{P}_d(z,y))$

d$_d$3 $\mathtt{AT}(x) \coloneqq (\mathrm{PD}(x) \vee \mathrm{AB}(x)) \wedge \neg \exists y(\mathtt{PP}_d(y,x))$

d$_d$4 $\mathtt{SUM}_d(s,x,y) \coloneqq \forall z(\mathtt{O}_d(z,s) \leftrightarrow (\mathtt{O}_d(z,x) \vee \mathtt{O}_d(z,y)))$

a$_d$1 $\mathtt{P}_d(x,y) \rightarrow (\mathrm{AB}(x) \wedge \mathrm{AB}(y)) \vee (\mathrm{PD}(x) \wedge \mathrm{PD}(y))$

a$_d$2 $\mathtt{P}_d(x,y) \rightarrow (\mathrm{T}(x) \leftrightarrow \mathrm{T}(y))$

a$_d$3 $\mathtt{P}_d(x,y) \rightarrow (\mathrm{S}(x) \leftrightarrow \mathrm{S}(y))$

a$_d$4 $(\mathrm{AB}(x) \vee \mathrm{PD}(x)) \rightarrow \mathtt{P}_d(x,x)$

a$_d$5 $\mathtt{P}_d(x,y) \wedge \mathtt{P}_d(y,x) \rightarrow x = y$

a$_d$6 $\mathtt{P}_d(x,y) \wedge \mathtt{P}_d(y,z) \rightarrow \mathtt{P}_d(x,z)$

a$_d$7 $(\mathrm{AB}(x) \vee \mathrm{PD}(x)) \wedge \neg \mathtt{P}_d(x,y) \rightarrow \exists z(\mathtt{P}_d(z,x) \wedge \neg \mathtt{O}_d(z,y))$

3.2 Direct Quality

a$_d$8 $\mathtt{DQT}(x,y) \rightarrow (\mathrm{TQ}(x) \wedge \mathrm{PD}(y)) \vee (\mathrm{PQ}(x) \wedge \mathrm{PED}(y)) \vee (\mathrm{AQ}(x) \wedge \mathrm{NPED}(y))$

a$_d$9 $\mathtt{DQT}(x,y) \wedge \mathtt{DQT}(x,z) \rightarrow y = z$

a$_d$10 $\mathrm{Q}(x) \rightarrow \exists y(\mathtt{DQT}(x,y))$

a$_d$11 $\mathrm{PD}(x) \leftrightarrow \exists y(\mathtt{DQT}(y,x) \wedge \mathrm{TL}(y))$

a$_d$12 $\mathrm{PED}(x) \leftrightarrow \exists y(\mathtt{DQT}(y,x) \wedge \mathrm{SL}(y))$

t$_d$1 $\mathtt{DQT}(x,y) \rightarrow x \neq y$

$\mathbf{t_d2}$ $\mathtt{DQT}(x,y) \wedge \mathrm{AQ}(x) \rightarrow \mathrm{NPED}(y)$

$\mathbf{t_d3}$ $\mathtt{DQT}(x,y) \wedge \mathrm{PQ}(x) \rightarrow \mathrm{PED}(y)$

$\mathbf{t_d4}$ $\mathtt{DQT}(x,y) \wedge \mathrm{TQ}(x) \rightarrow \mathrm{PD}(y)$

3.3 Immediate Quale

$\mathbf{a_d13}$ $\mathtt{QL}(x,y) \rightarrow \mathrm{TR}(x) \wedge \mathrm{TQ}(y)$

$\mathbf{a_d14}$ $\mathrm{TQ}(x) \rightarrow \exists y(\mathtt{QL}(y,x))$

$\mathbf{a_d15}$ $\mathtt{QL}(x,y) \wedge \mathrm{T}(x) \rightarrow \mathrm{TL}(y)$

$\mathbf{a_d16}$ $\mathrm{TL}(x) \rightarrow \exists y(\mathrm{T}(y) \wedge \mathtt{QL}(y,x))$

$\mathbf{a_d17}$ $\mathtt{QL}(x,y) \wedge \mathtt{QL}(z,y) \wedge \mathrm{T}(x) \wedge \mathrm{T}(z) \rightarrow x = z$

3.4 Temporal Location and Present At

$\mathbf{d_d5}$ $\mathtt{PRE}(x,t) \coloneqq \exists u(\mathtt{TLC}(x,u) \wedge \mathrm{P_d}(t,u))$

$\mathbf{a_d18}$ $\mathtt{TLC}(x,t) \rightarrow (\mathrm{ED}(x) \vee \mathrm{PD}(x) \vee \mathrm{Q}(x)) \wedge \mathrm{T}(t)$

$\mathbf{a_d19}$ $(\mathrm{ED}(x) \vee \mathrm{PD}(x) \vee \mathrm{Q}(x)) \rightarrow \exists t(\mathtt{TLC}(x,t))$

$\mathbf{a_d20}$ $\mathtt{TLC}(x,t) \wedge \mathtt{TLC}(x,u) \rightarrow t = u$

$\mathbf{a_d21}$ $\mathrm{P_d}(x,y) \wedge \mathtt{TLC}(x,t) \wedge \mathtt{TLC}(y,u) \rightarrow \mathrm{P_d}(t,u)$

$\mathbf{a_d22}$ $\mathtt{DQT}(x,y) \wedge \mathtt{TLC}(x,t) \wedge \mathtt{TLC}(y,u) \rightarrow \mathrm{P_d}(t,u)$

$\mathbf{a_d23}$ $\mathrm{PD}(x) \rightarrow (\mathtt{TLC}(x,t) \leftrightarrow \exists y(\mathtt{DQT}(y,x) \wedge \mathtt{QL}(t,y) \wedge \mathrm{T}(t)))$

$\mathbf{a_d24}$ $\mathrm{SL}(x) \wedge \mathtt{DQT}(x,y) \wedge \mathtt{TLC}(x,t) \wedge \mathtt{TLC}(y,u) \rightarrow t = u$

$\mathbf{t_d5}$ $\mathtt{PRE}(x,t) \rightarrow (\mathrm{ED}(x) \vee \mathrm{PD}(x) \vee \mathrm{Q}(x)) \wedge \mathrm{T}(t)$

$\mathbf{t_d6}$ $\mathtt{PRE}(x,t) \wedge \mathrm{P_d}(s,t) \rightarrow \mathtt{PRE}(x,s)$

$\mathbf{t_d7}$ $\mathtt{PRE}(x,t) \wedge \mathrm{P_d}(x,y) \rightarrow \mathtt{PRE}(y,t)$

$\mathbf{t_d8}$ $\mathtt{DQT}(x,y) \wedge \mathtt{PRE}(x,t) \rightarrow \mathtt{PRE}(y,t)$

$\mathbf{t_d9}$ $(\mathrm{ED}(x) \vee \mathrm{PD}(x) \vee \mathrm{Q}(x)) \rightarrow \exists t(\mathtt{PRE}(x,t))$

3.5 Temporary Quale

$\mathbf{a_d25}$ $\mathtt{tQL}(x,y,t) \rightarrow ((\mathrm{PR}(x) \wedge \mathrm{PQ}(y)) \vee (\mathrm{AR}(x) \wedge \mathrm{AQ}(y))) \wedge \mathrm{T}(t)$

$\mathbf{a_d26}$ $\mathtt{tQL}(x,y,t) \rightarrow \mathtt{PRE}(y,t)$

$\mathbf{a_d27}$ $(\mathrm{PQ}(x) \vee \mathrm{AQ}(x)) \wedge \mathtt{PRE}(x,t) \rightarrow \exists y(\mathtt{tQL}(y,x,t))$

$\mathbf{a_d28}$ $\mathtt{tQL}(x,y,t) \wedge \mathrm{P_d}(u,t) \rightarrow \exists z(\mathtt{tQL}(z,y,u) \wedge \mathrm{P_d}(z,x))$

$\mathbf{a_d29}$ $\mathtt{tQL}(x,y,t) \wedge \mathrm{S}(x) \rightarrow \mathrm{SL}(y)$

$\mathbf{a_d30}$ $\mathrm{SL}(x) \wedge \mathtt{PRE}(x,t) \rightarrow \exists y(\mathrm{S}(y) \wedge \mathtt{tQL}(y,x,t))$

$\mathbf{a_d31}$ $\mathtt{tQL}(x,y,t) \wedge \mathtt{tQL}(z,y,t) \wedge \mathrm{S}(x) \wedge \mathrm{S}(z) \rightarrow x = z$

3.6 Temporary Mereology

d_d6 $\mathtt{tO}_\mathtt{d}(x,y,t) \coloneqq \exists z(\mathtt{tP}_\mathtt{d}(z,x,t) \wedge \mathtt{tP}_\mathtt{d}(z,y,t))$

d_d7 $\mathtt{tPP}_\mathtt{d}(x,y,t) \coloneqq \mathtt{tP}_\mathtt{d}(x,y,t) \wedge \neg\mathtt{tP}_\mathtt{d}(y,x,t)$

d_d8 $\mathtt{CP}(x,y) \coloneqq \mathrm{ED}(x) \wedge \forall t(\mathtt{PRE}(x,t) \rightarrow \mathtt{tP}_\mathtt{d}(x,y,t))$

a_d32 $\mathtt{tP}_\mathtt{d}(x,y,t) \rightarrow \mathrm{ED}(x) \wedge \mathrm{ED}(y) \wedge \mathrm{T}(t)$

a_d33 $\mathtt{tP}_\mathtt{d}(x,y,t) \rightarrow \mathtt{PRE}(x,t) \wedge \mathtt{PRE}(y,t)$

a_d34 $\mathtt{tP}_\mathtt{d}(x,y,t) \wedge \mathtt{P}_\mathtt{d}(u,t) \rightarrow \mathtt{tP}_\mathtt{d}(x,y,u)$

a_d35 $\mathtt{tP}_\mathtt{d}(x,y,t) \wedge \mathtt{tP}_\mathtt{d}(x,y,u) \wedge \mathtt{SUM}_\mathtt{d}(s,t,u) \rightarrow \mathtt{tP}_\mathtt{d}(x,y,s)$

a_d36 $\mathrm{ED}(x) \wedge \mathtt{PRE}(x,t) \rightarrow \mathtt{tP}_\mathtt{d}(x,x,t)$

a_d37 $\mathtt{tP}_\mathtt{d}(x,y,t) \wedge \mathtt{tP}_\mathtt{d}(y,z,t) \rightarrow \mathtt{tP}_\mathtt{d}(x,z,t)$

a_d38 $\mathrm{ED}(x) \wedge \mathrm{ED}(y) \wedge \mathtt{PRE}(x,t) \wedge \mathtt{PRE}(y,t) \wedge \neg\mathtt{tP}_\mathtt{d}(x,y,t) \rightarrow \exists z(\mathtt{tP}_\mathtt{d}(z,x,t) \wedge \neg\mathtt{tO}_\mathtt{d}(z,y,t))$

a_d39 $\mathrm{PED}(x) \leftrightarrow \exists yt(\mathtt{tP}_\mathtt{d}(y,x,t)) \wedge \forall yt(\mathtt{tP}_\mathtt{d}(y,x,t) \rightarrow \mathrm{PED}(y))$

a_d40 $\mathrm{NPED}(x) \leftrightarrow \exists yt(\mathtt{tP}_\mathtt{d}(y,x,t)) \wedge \forall yt(\mathtt{tP}_\mathtt{d}(y,x,t) \rightarrow \mathrm{NPED}(y))$

a_d41 $\mathrm{AS}(x) \rightarrow \exists yzut(\mathtt{tP}_\mathtt{d}(y,x,t) \wedge \mathrm{PED}(y) \wedge \mathtt{tP}_\mathtt{d}(z,x,u) \wedge \mathrm{NPED}(z))$

a_d42 $\mathrm{M}(x) \wedge \mathtt{tP}_\mathtt{d}(y,x,t) \wedge \mathtt{PRE}(x,u) \rightarrow \mathtt{tP}_\mathtt{d}(y,x,u)$

t_d10 $\exists yzut(\mathtt{tP}_\mathtt{d}(y,x,t) \wedge \mathrm{PED}(y) \wedge \mathtt{tP}_\mathtt{d}(z,x,u) \wedge \mathrm{NPED}(z)) \rightarrow \mathrm{AS}(x)$

t_d11 $\mathfrak{D}_\mathrm{e} \vdash \mathrm{ED}(x) \rightarrow \mathtt{CP}(x,x)$
Proof. Directly by (d_d8) and (a_d36). □

t_d12 $\mathfrak{D}_\mathrm{e} \nvdash \mathtt{CP}(x,y) \wedge \mathtt{CP}(y,x) \rightarrow x = y$
Proof. Nothing excluded the possibility to have two different endurants that mereologically coincides at every time of their existence. □

t_d13 $\mathfrak{D}_\mathrm{e} \vdash \mathtt{CP}(x,y) \wedge \mathtt{CP}(y,z) \rightarrow \mathtt{CP}(x,z)$
Proof. Directly by (d_d8), (a_d32), (a_d33), and (a_d37). □

t_d14 $\mathfrak{D}_\mathrm{e} \nvdash \mathrm{ED}(x) \wedge \neg\mathtt{CP}(x,y) \rightarrow \exists z(\mathtt{CP}(z,x) \wedge \neg\exists w(\mathtt{CP}(w,z) \wedge \mathtt{CP}(w,y)))$
Proof. Consider a model of DOLCE with only three physical objects (POB) x, y, and z such that (i) x and y are present only at time t; (ii) z is present at times t and u; and (iii) at t, y and z are disjoint and they compose x, i.e., $\mathtt{tP}_\mathtt{d}(y,x,t)$, $\mathtt{tP}_\mathtt{d}(z,x,t)$, $\neg\mathtt{tP}_\mathtt{d}(x,y,t)$, $\neg\mathtt{tP}_\mathtt{d}(x,z,t)$, and $\neg\mathtt{tO}_\mathtt{d}(y,z,t)$ (plus all the reflexive $\mathtt{tP}_\mathtt{d}$-relationships). In this case we have that $\mathrm{ED}(x) \wedge \neg\mathtt{CP}(x,y)$ but the only constant parts of x are x and y that however both share a common constant part, namely y, i.e., $\mathtt{CP}(y,y) \wedge \mathtt{CP}(y,x)$ (z is not a constant part of x because z, but not x, is present at time u). □

3.7 Constitution

a_d43 $\mathrm{K}(x,y,t) \rightarrow ((\mathrm{PED}(x) \wedge \mathrm{PED}(y)) \vee (\mathrm{NPED}(x) \wedge \mathrm{NPED}(y)) \vee (\mathrm{PD}(x) \wedge \mathrm{PD}(y))) \wedge \mathrm{T}(t)$

a_d44 $\mathrm{K}(x,y,t) \rightarrow \mathtt{PRE}(x,t) \wedge \mathtt{PRE}(y,t)$

a_d45 $\mathrm{K}(x,y,t) \wedge \mathtt{P}_\mathtt{d}(u,t) \rightarrow \mathrm{K}(x,y,u)$

$\mathbf{a_d46}$ $\mathtt{K}(x,y,t) \wedge \mathtt{K}(x,y,u) \wedge \mathtt{SUM_d}(s,t,u) \rightarrow \mathtt{K}(x,y,s)$

$\mathbf{a_d47}$ $\mathtt{K}(x,y,u) \wedge \mathtt{PRE}(y,t) \rightarrow \exists zv(\mathtt{P_d}(v,t) \wedge \mathtt{K}(z,y,v))$

$\mathbf{a_d48}$ $\mathtt{K}(x,y,t) \rightarrow \neg \mathtt{K}(y,x,t)$

$\mathbf{a_d49}$ $\mathtt{K}(x,y,t) \wedge \mathtt{K}(y,z,t) \rightarrow \mathtt{K}(x,z,t)$

$\mathbf{a_d50}$ $\mathtt{K}(x,y,t) \wedge \mathtt{tPP_d}(z,y,t) \rightarrow \exists w(\mathtt{tPP_d}(w,x,t) \wedge \mathtt{K}(w,z,t))$

$\mathbf{a_d51}$ $\mathtt{K}(x,y,t) \wedge \mathtt{K}(z,y,t) \rightarrow (x = z \vee \mathtt{K}(x,z,t) \vee \mathtt{K}(z,x,t))$

$\mathbf{t_d15}$ $\mathtt{K}(x,y,t) \rightarrow x \neq y$

$\mathbf{t_d16}$ $\mathtt{K}(x,y,u) \wedge \mathtt{PRE}(y,t) \wedge \mathtt{P_d}(v,t) \rightarrow \exists wz(\mathtt{P_d}(w,v) \wedge \mathtt{K}(z,y,w))$

3.8 Participation

$\mathbf{a_d52}$ $\mathtt{PC}(x,y,t) \rightarrow \mathrm{ED}(x) \wedge \mathrm{PD}(y) \wedge \mathrm{T}(t)$

$\mathbf{a_d53}$ $\mathtt{PC}(x,y,t) \rightarrow \mathtt{PRE}(x,t) \wedge \mathtt{PRE}(y,t)$

$\mathbf{a_d54}$ $\mathtt{PC}(x,y,t) \wedge \mathtt{P_d}(u,t) \rightarrow \mathtt{PC}(x,y,u)$

$\mathbf{a_d55}$ $\mathtt{PC}(x,y,t) \wedge \mathtt{PC}(x,y,u) \wedge \mathtt{SUM_d}(s,t,u) \rightarrow \mathtt{PC}(x,y,s)$

$\mathbf{a_d56}$ $\mathrm{PD}(x) \wedge \mathtt{PRE}(x,t) \rightarrow \exists yu(\mathtt{P_d}(u,t) \wedge \mathtt{PC}(y,x,u))$

$\mathbf{a_d57}$ $\mathrm{ED}(x) \wedge \mathtt{PRE}(x,t) \rightarrow \exists yu(\mathtt{P_d}(u,t) \wedge \mathtt{PC}(x,y,u))$

$\mathbf{a_d58}$ $\mathtt{PC}(x,y,t) \wedge \mathtt{P_d}(y,z) \rightarrow \mathtt{PC}(x,z,t)$

$\mathbf{t_d17}$ $\mathrm{PD}(x) \wedge \mathtt{PRE}(x,t) \wedge \mathtt{P_d}(v,t) \rightarrow \exists yu(\mathtt{P_d}(u,v) \wedge \mathtt{PC}(y,x,u))$

$\mathbf{t_d18}$ $\mathrm{ED}(x) \wedge \mathtt{PRE}(x,t) \wedge \mathtt{P_d}(v,t) \rightarrow \exists yu(\mathtt{P_d}(u,v) \wedge \mathtt{PC}(x,y,u))$

$\mathbf{t_d19}$ $\mathtt{PC}(x,y,u) \wedge \mathtt{PRE}(y,t) \rightarrow \exists zu(\mathtt{P_d}(u,t) \wedge \mathtt{PC}(z,y,u))$

$\mathbf{t_d20}$ $\mathtt{PC}(x,y,u) \wedge \mathtt{PRE}(x,t) \rightarrow \exists zu(\mathtt{P_d}(u,t) \wedge \mathtt{PC}(x,z,u))$

$\mathbf{t_d21}$ $\mathtt{PC}(x,y,t) \rightarrow x \neq y$

3.9 Existential Dependence

$\mathbf{d_d9}$ $\mathtt{SD}(x,y) \coloneqq \exists t(\mathtt{PRE}(x,t)) \wedge \forall t(\mathtt{PRE}(x,t) \rightarrow \mathtt{EXD}(x,y,t))$

$\mathbf{a_d59}$ $\mathtt{EXD}(x,y,t) \rightarrow \mathtt{PRE}(x,t) \wedge \mathtt{PRE}(y,t)$

$\mathbf{a_d60}$ $\mathtt{EXD}(x,y,t) \wedge \mathtt{P_d}(u,t) \rightarrow \mathtt{EXD}(x,y,u)$

$\mathbf{a_d61}$ $\mathtt{EXD}(x,y,t) \wedge \mathtt{EXD}(x,y,u) \wedge \mathtt{SUM_d}(s,t,u) \rightarrow \mathtt{EXD}(x,y,s)$

$\mathbf{a_d62}$ $\mathtt{EXD}(x,y,u) \wedge \mathtt{PRE}(x,t) \rightarrow \exists zv(\mathtt{P_d}(v,t) \wedge \mathtt{EXD}(x,z,v))$

$\mathbf{a_d63}$ $\mathtt{K}(x,y,t) \rightarrow \mathtt{EXD}(y,x,t)$

$\mathbf{a_d64}$ $\mathtt{PC}(x,y,t) \rightarrow \mathtt{EXD}(x,y,t) \wedge \mathtt{EXD}(y,x,t)$

$\mathbf{a_d65}$ $\mathtt{DQT}(x,y) \rightarrow \mathtt{SD}(x,y)$

$\mathbf{a_d66}$ $\mathrm{NPED}(x) \wedge \mathtt{PRE}(x,t) \rightarrow \exists yu(\mathrm{PED}(y) \wedge \mathtt{P_d}(u,t) \wedge \mathtt{EXD}(x,y,u))$

$\mathbf{a_d67}$ $\mathrm{MOB}(x) \rightarrow \exists y(\mathrm{APO}(y) \wedge \mathtt{SD}(x,y))$

$\mathbf{a_d68}$ $\text{NASO}(x) \wedge \text{PRE}(x,t) \rightarrow \exists yu(\text{SC}(y) \wedge \text{P}_\text{d}(u,t) \wedge \text{EXD}(x,y,u))$

$\mathbf{a_d69}$ $\text{SC}(x) \wedge \text{PRE}(x,t) \rightarrow \exists yu(\text{SAG}(y) \wedge \text{P}_\text{d}(u,t) \wedge \text{K}(y,x,u))$

$\mathbf{a_d70}$ $\text{SAG}(x) \wedge \text{PRE}(x,t) \rightarrow \exists yu(\text{APO}(y) \wedge \text{P}_\text{d}(u,t) \wedge \text{EXD}(x,y,u))$

$\mathbf{a_d71}$ $\text{F}(x) \wedge \text{PRE}(x,t) \rightarrow \exists yu(\text{NAPO}(y) \wedge \text{P}_\text{d}(u,t) \wedge \text{EXD}(x,y,u))$

$\mathbf{a_d72}$ $\text{APO}(x) \wedge \text{PRE}(x,t) \rightarrow \exists yu(\text{NAPO}(y) \wedge \text{P}_\text{d}(u,t) \wedge \text{K}(y,x,u))$

$\mathbf{a_d73}$ $\text{NAPO}(x) \wedge \text{PRE}(x,t) \rightarrow \exists yu(\text{M}(y) \wedge \text{P}_\text{d}(u,t) \wedge \text{K}(y,x,u))$

3.10 Spatial Location

$\mathbf{d_d10}$ $\text{sPRE}(x,s,t) \coloneqq \exists r(\text{SLC}(x,r,t) \wedge \text{P}_\text{d}(s,r))$

$\mathbf{a_d74}$ $\text{SLC}(x,s,t) \rightarrow ((\text{ED}(x) \wedge \neg \text{AS}(x)) \vee \text{PQ}(x) \vee \text{PD}(x)) \wedge \text{S}(s)$

$\mathbf{a_d75}$ $\text{SLC}(x,s,t) \rightarrow \text{PRE}(x,t)$

$\mathbf{a_d76}$ $\text{SLC}(x,s,t) \wedge \text{SLC}(x,r,t) \rightarrow s = r$

$\mathbf{a_d77}$ $\text{SLC}(x,s,u) \wedge \text{SLC}(x,r,t) \wedge \text{P}_\text{d}(u,t) \rightarrow \text{P}_\text{d}(s,r)$

$\mathbf{a_d78}$ $\text{SLC}(x,s,u) \wedge \text{PRE}(x,t) \rightarrow \exists r(\text{SLC}(x,r,t))$

$\mathbf{a_d79}$ $\text{PED}(x) \rightarrow (\text{SLC}(x,s,t) \leftrightarrow \exists y(\text{DQT}(y,x) \wedge \text{SL}(y) \wedge \text{tQL}(s,y,t)))$

$\mathbf{a_d80}$ $\text{PQ}(x) \rightarrow (\text{SLC}(x,s,t) \leftrightarrow \exists y(\text{DQT}(x,y) \wedge \text{SLC}(y,s,t)))$

$\mathbf{a_d81}$ $\text{NPED}(x) \wedge \text{SLC}(x,s,t) \rightarrow \exists y(\text{PED}(y) \wedge \text{EXD}(x,y,t) \wedge \text{SLC}(y,s,t))$

$\mathbf{a_d82}$ $\text{PC}(x,y,t) \wedge \text{SLC}(x,s,t) \rightarrow \exists r(\text{SLC}(y,r,t))$

$\mathbf{a_d83}$ $\text{PC}(x,y,t) \wedge \text{SLC}(x,r,t) \wedge \text{SLC}(y,s,t) \rightarrow \text{P}_\text{d}(r,s)$

$\mathbf{a_d84}$ $\text{tP}_\text{d}(x,y,t) \wedge \text{SLC}(x,s,t) \rightarrow \exists r(\text{SLC}(y,r,t))$

$\mathbf{a_d85}$ $\text{tP}_\text{d}(x,y,t) \wedge \text{SLC}(x,r,t) \wedge \text{SLC}(y,s,t) \rightarrow \text{P}_\text{d}(r,s)$

$\mathbf{a_d86}$ $\text{P}_\text{d}(x,y) \wedge \text{SLC}(x,s,t) \rightarrow \exists r(\text{SLC}(y,r,t))$

$\mathbf{a_d87}$ $\text{P}_\text{d}(x,y) \wedge \text{SLC}(x,r,t) \wedge \text{SLC}(y,s,t) \rightarrow \text{P}_\text{d}(r,s)$

$\mathbf{a_d88}$ $\text{K}(x,y,t) \rightarrow \forall s(\text{SLC}(x,s,t) \leftrightarrow \text{SLC}(y,s,t))$

$\mathbf{t_d22}$ $\text{PED}(x) \wedge \text{PRE}(x,t) \rightarrow \exists s(\text{SLC}(x,s,t))$

3.11 Taxonomy

3.11.1 Inclusions

$\mathbf{a_d89}$ $\text{AB}(x) \vee \text{ED}(x) \vee \text{PD}(x) \vee \text{Q}(x)$

$\mathbf{a_d90}$ $(\text{AS}(x) \vee \text{NPED}(x) \vee \text{PED}(x)) \leftrightarrow \text{ED}(x)$

$\mathbf{a_d91}$ $(\text{PRO}(x) \vee \text{ST}(x)) \rightarrow \text{STV}(x)$

$\mathbf{a_d92}$ $(\text{EV}(x) \vee \text{STV}(x)) \rightarrow \text{PD}(x)$

$\mathbf{a_d93}$ $(\text{AQ}(x) \vee \text{PQ}(x) \vee \text{TQ}(x)) \leftrightarrow \text{Q}(x)$

$\mathbf{a_d94}$ $(\text{ACC}(x) \vee \text{ACH}(x)) \rightarrow \text{EV}(x)$

$\mathbf{a_d95}$ $(\mathrm{APO}(x) \vee \mathrm{NAPO}(x)) \leftrightarrow \mathrm{POB}(x)$
$\mathbf{a_d96}$ $(\mathrm{SAG}(x) \vee \mathrm{SC}(x)) \rightarrow \mathrm{ASO}(x)$
$\mathbf{a_d97}$ $(\mathrm{ASO}(x) \vee \mathrm{NASO}(x)) \leftrightarrow \mathrm{SOB}(x)$
$\mathbf{a_d98}$ $(\mathrm{SOB}(x) \vee \mathrm{MOB}(x)) \rightarrow \mathrm{NPOB}(x)$
$\mathbf{a_d99}$ $(\mathrm{AR}(x) \vee \mathrm{PR}(x) \vee \mathrm{TR}(x)) \leftrightarrow \mathrm{R}(x)$
$\mathbf{a_d100}$ $\mathrm{R}(x) \rightarrow \mathrm{AB}(x)$
$\mathbf{a_d101}$ $(\mathrm{F}(x) \vee \mathrm{M}(x) \vee \mathrm{POB}(x)) \rightarrow \mathrm{PED}(x)$
$\mathbf{a_d102}$ $\mathrm{NPOB}(x) \rightarrow \mathrm{NPED}(x)$
$\mathbf{a_d103}$ $\mathrm{S}(x) \rightarrow \mathrm{PR}(x)$
$\mathbf{a_d104}$ $\mathrm{SL}(x) \rightarrow \mathrm{PQ}(x)$
$\mathbf{a_d105}$ $\mathrm{T}(x) \rightarrow \mathrm{TR}(x)$
$\mathbf{a_d106}$ $\mathrm{TL}(x) \rightarrow \mathrm{TQ}(x)$
$\mathbf{a_d107}$ $\neg\exists x(\mathrm{AB}(x) \wedge \mathrm{ED}(x))$
$\mathbf{a_d108}$ $\neg\exists x(\mathrm{AB}(x) \wedge \mathrm{PD}(x))$
$\mathbf{a_d109}$ $\neg\exists x(\mathrm{AB}(x) \wedge \mathrm{Q}(x))$
$\mathbf{a_d110}$ $\neg\exists x(\mathrm{ED}(x) \wedge \mathrm{PD}(x))$
$\mathbf{a_d111}$ $\neg\exists x(\mathrm{PD}(x) \wedge \mathrm{Q}(x))$
$\mathbf{a_d112}$ $\neg\exists x(\mathrm{ED}(x) \wedge \mathrm{Q}(x))$
$\mathbf{a_d113}$ $\neg\exists x(\mathrm{PED}(x) \wedge \mathrm{NPED}(x))$
$\mathbf{a_d114}$ $\neg\exists x(\mathrm{PED}(x) \wedge \mathrm{AS}(x))$
$\mathbf{a_d115}$ $\neg\exists x(\mathrm{NPED}(x) \wedge \mathrm{AS}(x))$
$\mathbf{a_d116}$ $\neg\exists x(\mathrm{M}(x) \wedge \mathrm{F}(x))$
$\mathbf{a_d117}$ $\neg\exists x(\mathrm{F}(x) \wedge \mathrm{POB}(x))$
$\mathbf{a_d118}$ $\neg\exists x(\mathrm{M}(x) \wedge \mathrm{POB}(x))$
$\mathbf{a_d119}$ $\neg\exists x(\mathrm{MOB}(x) \wedge \mathrm{SOB}(x))$
$\mathbf{a_d120}$ $\neg\exists x(\mathrm{ASO}(x) \wedge \mathrm{NASO}(x))$
$\mathbf{a_d121}$ $\neg\exists x(\mathrm{SAG}(x) \wedge \mathrm{SC}(x))$
$\mathbf{a_d122}$ $\neg\exists x(\mathrm{APO}(x) \wedge \mathrm{NAPO}(x))$
$\mathbf{a_d123}$ $\neg\exists x(\mathrm{EV}(x) \wedge \mathrm{STV}(x))$
$\mathbf{a_d124}$ $\neg\exists x(\mathrm{ACH}(x) \wedge \mathrm{ACC}(x))$
$\mathbf{a_d125}$ $\neg\exists x(\mathrm{ST}(x) \wedge \mathrm{PRO}(x))$
$\mathbf{a_d126}$ $\neg\exists x(\mathrm{TQ}(x) \wedge \mathrm{PQ}(x))$
$\mathbf{a_d127}$ $\neg\exists x(\mathrm{PQ}(x) \wedge \mathrm{AQ}(x))$
$\mathbf{a_d128}$ $\neg\exists x(\mathrm{TQ}(x) \wedge \mathrm{AQ}(x))$

$\mathbf{a_d 129}$ $\neg\exists x(\mathrm{TR}(x) \wedge \mathrm{PR}(x))$

$\mathbf{a_d 130}$ $\neg\exists x(\mathrm{PR}(x) \wedge \mathrm{AR}(x))$

$\mathbf{a_d 131}$ $\neg\exists x(\mathrm{TR}(x) \wedge \mathrm{AR}(x))$

4 EMMO

EMMO is a foundational ontology developed with the aim of grounding the representation of knowledge and data from applied sciences, with a focus on materials modeling.

EMMO endorses a form of *ontological naturalism* and a pragmatic *reductionistic* stance, electing the standard model of particle physics as the theory of choice: given the preferred interpretation, the world is seen as made up entirely of fundamental (non-virtual) elementary particles (e.g., electrons, photons), and sums of the latter, understood as portions of the world individuated via a linguistic label. Specifically, EMMO's focus is on non-virtual elementary particles in-between interactions, given the endorsement of a form of perdurantism to be understood in terms of persistence across change (rather than time), and the further assumption that change is intrinsically related to causal interactions mimicking Feynman diagrams at the micro level. EMMO supports *nominalism* across the board, as e.g., abstract entities and sets have no place in EMMO's domain; the ontology also favors a "materialistic", "physicalistic", worldview.

EMMO's architecture is divided into two parts: (1) a common core, comprising of the mereocausal module plus other, more specific, commitments borrowed from the sciences, and (2) a plurality of "*perspectives*", each an expression of a salient conceptual schema and/or offering tools to partition the logical space (see Sect. 4.2 on extensions and perspectives).

EMMO's uppermost mereocausal module is built upon formal ontology and naturalistic constraints; it establishes clear and objective numerical identity conditions, sets up the preconditions for a multi-scale approach and provides the means to deal with spatio-temporal reasoning. It is not necessary to go into the details regarding the naturalistic extension, since, as we shall see in a moment, the module won't figure in the version of the ontology considered for alignment purposes.

The perspectives are not mutually incompatible. The choice of the perspectives is guided by pragmatic principles and not metaphysical adequacy; they make EMMO inherently pluralistic, and greatly enhance its expressiveness – dealing with the core limitations of reductionistic approaches.

There are two core differences between the standard version of EMMO and the one considered for alignment:

(1) most perspectives and the naturalistic extension were not considered on the grounds that they include categories which are not properly "top-level", and that their FOL axiomatization is, at the present time, still unstable.

(2) Three extensions were crafted in the context of the OntoCommons project as preconditions for the possibility of an alignment, namely the Time Extension, the Temporal Slices extension, and the Concepts Extension. These modular extensions are not alignment-specific and the proposed constructions have more general applicability; nevertheless, it should be noted that, at the present time, they are not officially part of the EMMO, nor do they have analogues in the OWL version of the ontology.

$\mathtt{P}(x,y)$	x is part of y
$\mathtt{C}(x,y)$	x causes y

Table 5: Primitive relations of the mereocausal module of EMMO.

`CslPth`	Causal Path
`CslStr`	Causal Structure
`CslSys`	Causal System
`COLL`	Collection
`MR`	Interpretant
`INT`	Interpreter
`ITEM`	Item

`Obj`	Object
`Prc`	Process
`Q`	Quantum
`SemObj`	Semiotic Object
`sPR`	Semiotic Process
`Sign`	Sign

Table 6: Categories of EMMO considered for the alignment.

4.1 Mereocausal Module

4.1.1 Mereology

$\mathbf{d_e1}$ $\mathtt{O}(x,y) \coloneqq \exists z(\mathtt{P}(z,x) \wedge \mathtt{P}(z,y))$ (*Overlap*)

$\mathbf{d_e2}$ $\sigma x\langle\phi(x)\rangle \coloneqq \iota z(\forall y(\mathtt{O}(y,z) \leftrightarrow \exists x(\phi(x) \wedge \mathtt{O}(x,y))))$ (*Fusion*)

$\mathbf{d_e3}$ $\mathtt{SUM}(x,y,z) \coloneqq x = \sigma w\langle\mathtt{P}(w,y) \vee \mathtt{P}(w,z)\rangle$ (*Binary Sum*)

$\mathbf{d_e4}$ $\mathtt{PRD}(x,y,z) \coloneqq x = \sigma w\langle\mathtt{P}(w,y) \wedge \mathtt{P}(w,z)\rangle$ (*Binary Product*)

$\mathbf{d_e5}$ $\mathtt{DIF}(x,y,z) \coloneqq x = \sigma w\langle\mathtt{P}(w,y) \wedge \neg\mathtt{O}(w,z)\rangle$ (*Binary Difference*)

$\mathbf{d_e6}$ $\mathtt{u} \coloneqq \sigma x\langle\mathtt{P}(x,x)\rangle$ (*Universe*)

$\mathbf{d_e7}$ $\mathtt{Q}(x) \coloneqq \neg\exists y(\mathtt{PP}(y,x))$ (*Quantum* [*Mereological Atom*])

$\mathbf{d_e8}$ $\mathtt{qP}(x,y) \coloneqq \mathtt{P}(x,y) \wedge \mathtt{Q}(x)$ (*Quantum Part*)

$\mathbf{a_e1}$ $\mathtt{P}(x,x)$ (*Parthood: Reflexivity*)

$\mathbf{a_e2}$ $\mathtt{P}(x,y) \wedge \mathtt{P}(y,x) \rightarrow x = y$ (*Parthood: Antisymmetry*)

$\mathbf{a_e3}$ $\mathtt{P}(x,y) \wedge \mathtt{P}(y,z) \rightarrow \mathtt{P}(x,z)$ (*Parthood: Transitivity*)

$\mathbf{a_e4}$ $\neg\mathtt{P}(y,x) \rightarrow \exists z(\mathtt{P}(z,y) \wedge \neg\mathtt{O}(z,x))$ (*Strong Supplementation*)

$\mathbf{a_e5}$ $\exists x(\phi(x)) \rightarrow \exists y(y = \sigma x\langle\phi(x)\rangle)$ (*Unrestricted Composition*)

$\mathbf{a_e6}$ $\exists y(\mathtt{qP}(y,x))$ (*Atomicity*)

$\mathbf{t_e1}$ $\mathtt{P}(x,y) \leftrightarrow \forall z(\mathtt{qP}(z,x) \rightarrow \mathtt{qP}(z,y))$ (*Quantum Extensionality*)

$\mathbf{t_e2}$ $\mathtt{SUM}(s,x,y) \leftrightarrow \forall z(\mathtt{qP}(z,s) \leftrightarrow (\mathtt{qP}(z,x) \vee \mathtt{qP}(z,y)))$ (*Binary Sum's Atoms*)

$\mathbf{t_e3}$ $\mathtt{SUM}(s,x,y) \rightarrow \mathtt{P}(x,s) \wedge \mathtt{P}(y,s)$ (*Binary Sum's Parts*)

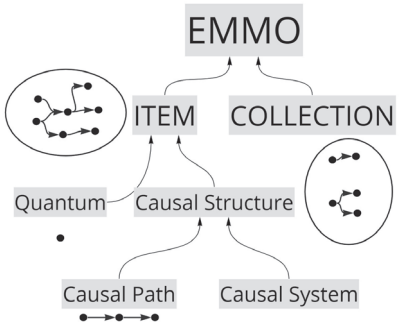

Figure 3: EMMO: Mereocausality Module's taxonomy.

4.1.2 Causation

d$_e$9 $\mathtt{dC}(x,y) \coloneqq \mathtt{C}(x,y) \wedge \neg\exists z(\mathtt{C}(x,z) \wedge \mathtt{C}(z,y))$ (*Direct Causation*)

d$_e$10 $\mathtt{iC}(x,y) \coloneqq \mathtt{C}(x,y) \wedge \neg\mathtt{dC}(x,y)$ (*Indirect Causation*)

d$_e$11 $\mathtt{CC}(x,y) \coloneqq \mathtt{C}(x,y) \vee \mathtt{C}(y,x)$ (*Causal Connection*)

d$_e$12 $\mathtt{MC}(x,y) \coloneqq \neg\mathtt{O}(x,y) \wedge \exists wz(\mathtt{qP}(w,x) \wedge \mathtt{qP}(z,y) \wedge \mathtt{C}(w,z))$ (*Macro Causation*)

d$_e$13 $\mathtt{MdC}(x,y) \coloneqq \neg\mathtt{O}(x,y) \wedge \exists wz(\mathtt{qP}(w,x) \wedge \mathtt{qP}(z,y) \wedge \mathtt{dC}(w,z))$ (*Macro Direct Caus.*)

d$_e$14 $\mathtt{ITEM}(x) \coloneqq \forall yz(\mathtt{SUM}(x,y,z) \wedge \neg\mathtt{O}(y,z) \rightarrow (\mathtt{MdC}(y,z) \vee \mathtt{MdC}(z,y)))$ (*Item*)

a$_e$7 $\mathtt{C}(x,y) \rightarrow \mathtt{Q}(x) \wedge \mathtt{Q}(y)$ (*Quantum Causation*)

a$_e$8 $\neg\mathtt{C}(x,x)$ (*Causation: Irreflexivity*)

a$_e$9 $\mathtt{C}(x,y) \wedge \mathtt{C}(y,z) \rightarrow \mathtt{C}(x,z)$ (*Causation: Transitivity*)

a$_e$10 $\mathtt{C}(x,y) \rightarrow \mathtt{dC}(x,y) \vee \exists zw(\mathtt{C}(x,z) \wedge \mathtt{dC}(z,y) \wedge \mathtt{dC}(x,w) \wedge \mathtt{C}(w,y))$
(*Causation: Discreteness / Direct Causation's Necessity*)

a$_e$11 $\mathtt{ITEM}(\mathtt{u})$ (*Self-Connected Universe*)

a$_e$12 $\mathtt{dC}(x,y) \rightarrow \exists z((\mathtt{dC}(x,z) \vee \mathtt{dC}(z,y)) \wedge y \neq z \wedge x \neq z)$ (*Minimal Causal Structure*)

a$_e$13 $\mathtt{dC}(x,y) \wedge \mathtt{dC}(x,z) \wedge \mathtt{dC}(w,y) \rightarrow \mathtt{dC}(w,z)$ (*Locality*)

4.1.3 Core Taxonomical distinctions

d$_e$15 $\mathtt{COLL}(x) \coloneqq \neg\mathtt{ITEM}(x)$ (*Collection*)

d$_e$16 $\mathtt{CslStr}(x) \coloneqq \mathtt{ITEM}(x) \wedge \neg\mathtt{Q}(x)$ (*Causal Object*)

d$_e$17 $\mathtt{CslPth}(x) \coloneqq \mathtt{CslStr}(x) \wedge \forall yz(y \neq z \wedge \mathtt{qP}(y,x) \wedge \mathtt{qP}(z,x) \rightarrow \mathtt{CC}(y,z))$
(*Causal Path*)

d$_e$18 $\mathtt{CslSys}(x) \coloneqq \mathtt{CslStr}(x) \wedge \neg\mathtt{CslPth}(x)$ (*Causal System*)

t$_e$4 $\mathtt{Q}(x) \rightarrow \mathtt{ITEM}(x)$ (*Hierarchy:* $\mathtt{Q} \subset \mathtt{ITEM}$)

t$_e$5 $\texttt{CslPth}(x) \rightarrow \texttt{CslStr}(x)$ (*Hierarchy:* $\texttt{CslPth} \subset \texttt{CslStr}$)

t$_e$6 $\texttt{CslSys}(x) \rightarrow \texttt{CslStr}(x)$ (*Hierarchy:* $\texttt{CslSys} \subset \texttt{CslStr}$)

t$_e$7 $\texttt{CslStr}(x) \rightarrow \texttt{ITEM}(x)$ (*Hierarchy:* $\texttt{CslStr} \subset \texttt{ITEM}$)

t$_e$8 $\texttt{COLL}(x) \leftrightarrow \neg\texttt{ITEM}(x)$ (*Hierarchy: domain's partition in items and collections*)

t$_e$9 $\texttt{COLL}(x) \vee \texttt{ITEM}(x)$ (*Items and collections: full domain coverage*)

t$_e$10 $\texttt{ITEM}(x) \leftrightarrow \texttt{Q}(x) \vee \texttt{CslPth}(x) \vee \texttt{CslSys}(x) \wedge (\texttt{Q}(x) \rightarrow \neg(\texttt{CslPth}(x) \vee \texttt{CslSys}(x))) \wedge (\texttt{CslPth}(x) \rightarrow \neg(\texttt{Q}(x) \vee \texttt{CslSys}(x))) \wedge (\texttt{CslSys}(x) \rightarrow \neg(\texttt{CslPth}(x) \vee \texttt{Q}(x)))$ (*Hierarchy:* ITEM*s' partition*)

4.1.4 Additional definitions

d$_e$19 $\texttt{SRC}(x,y) \coloneqq \texttt{PP}(x,y) \wedge \texttt{ITEM}(x) \wedge \neg\exists z(\texttt{PP}(z,y) \wedge \texttt{MC}(z,x))$ (*Source*)

d$_e$20 $\texttt{qSRC}(x,y) \coloneqq \texttt{qP}(x,y) \wedge \texttt{SRC}(x,y)$ (*Quantum Source*)

d$_e$21 $\texttt{SNK}(x,y) \coloneqq \texttt{PP}(x,y) \wedge \texttt{ITEM}(x) \wedge \neg\exists z(\texttt{PP}(z,y) \wedge \texttt{MC}(x,z))$ (*Sink*)

d$_e$22 $\texttt{qSNK}(x,y) \coloneqq \texttt{qP}(x,y) \wedge \texttt{SNK}(x,y)$ (*Quantum Sink*)

d$_e$23 $\texttt{TP}(x,y) \coloneqq \texttt{ITEM}(y) \wedge \texttt{PP}(x,y) \wedge \forall z(\texttt{iMMB}(z,y-x) \rightarrow \texttt{tCNT}(z,x) \vee \texttt{tCNT}(x,z))$ (*Temporal Part*)

d$_e$24 $\texttt{TiP}(x,y) \coloneqq \texttt{TP}(x,y) \wedge \texttt{ITEM}(x)$ (*Temporal Item Part*)

d$_e$25 $\texttt{TsP}(x,y) \coloneqq \texttt{TP}(x,y) \wedge \forall z(\texttt{TP}(z,y) \wedge \neg\texttt{O}(z,x) \rightarrow \texttt{MC}(x,z) \vee \texttt{MC}(z,x))$ (*Temporal Slice Part*)

d$_e$26 $\texttt{TsiP}(x,y) \coloneqq \texttt{TsP}(x,y) \wedge \texttt{TiP}(x,y)$ (*Temporal Slice Item Part*)

4.2 Extensions

4.2.1 General principles

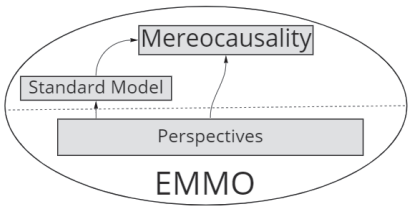

Figure 4: EMMO: abstract architecture.

Let EMMO$_{[MC]}$ be EMMO's mereocausal module's theory. The theory can be extended with the introduction of primitives and axioms constraining said primitives (entire ontologies' theories could, in principle, be "implanted" in EMMO to specialize it).

For the sake of clarity we introduce the notions of *extensions, perspectives, and views*. Given a theory $\mathfrak{T}$ such that EMMO$_{[MC]} \subseteq \mathfrak{T}$, an *extension* $\mathfrak{E}$ of $\mathfrak{T}$ is a theory consistent with $\mathfrak{T}$, i.e., $\mathfrak{E} \cup \mathfrak{T}$ is consistent[1], such that its signature[2] includes the one of $\mathfrak{T}$, i.e., $\sigma(\mathfrak{E}) \supseteq \sigma(\mathfrak{T})$, and where the (finite) predicates in $\sigma(\mathfrak{E})$ satisfy (a_e14).

a_e14 $\bigwedge_{\phi \in \sigma(\mathfrak{E})}(\phi(x_1,\ldots,x_n) \rightarrow (\texttt{ITEM}(x_1) \vee \texttt{COLL}(x_1)) \wedge \ldots \wedge (\texttt{ITEM}(x_n) \vee \texttt{COLL}(x_n)))$
(where ϕ is n-ary) (Extensions' range)

We assume that, in addition to EMMO$_{[MC]}$, $\mathfrak{T}$ can include only other extensions. The general idea is that an extension $\mathfrak{E}$ offers new (with respect to the imported theories) predicates to classify at least some of the entities in EMMO$_{[MC]}$, i.e., extensions needn't necessarily fully cover the whole domain of EMMO$_{[MC]}$. As limit cases, we accept *definitional extensions*, i.e., extensions containing only syntactic definitions based on the predicates in $\sigma(\mathfrak{T})$.

Given an extension $\mathfrak{E}$ of $\mathfrak{T}$, we distinguish the set $\mathbf{t}(\mathfrak{E}) \subseteq (\sigma(\mathfrak{E}) \setminus \sigma(\mathfrak{T}))$ of newly introduced unary predicates called *types*, from the set $\mathbf{r}(\mathfrak{E}) \subseteq (\sigma(\mathfrak{E}) \setminus \sigma(\mathfrak{T}))$ of newly introduced n-ary, with $n > 1$, predicates called *relations*. As a further constraint, given two extensions $\mathfrak{E}_1$ and $\mathfrak{E}_2$, we assume that $(\mathbf{t}(\mathfrak{E}_1) \cup \mathbf{r}(\mathfrak{E}_1)) \cap (\mathbf{t}(\mathfrak{E}_2) \cup \mathbf{r}(\mathfrak{E}_2)) = \emptyset$, i.e., the predicates newly introduced by different extensions are disjoint.

Notice that all the above conditions are only necessary conditions for being an extension. Extensions are provided by the users (or developers) and every extension is then the outcome of a decision process undertaken by the users to delimit the signature and the axioms and/or definitions in the extension.

Among the extensions one can distinguish the ones provided, with a foundational goal, by EMMO developers, from the ones (called here *views*) provided by independent users for practical or applicative purposes.

Among the first type of extensions, *perspectives* play a central role. Perspectives are mutually independent theories satisfying (a_e15) and (a_e16), i.e., they cover the whole class of causal structures (and only such class).

a_e15 $\bigwedge_{\phi \in \sigma(\mathfrak{E})}(\phi(x_1,\ldots,x_n) \rightarrow \texttt{CslStr}(x_1) \wedge \ldots \wedge \texttt{CslStr}(x_n))$
(where ϕ is n-ary) (Perspectives only CslStr)

a_e16 $\texttt{CslStr}(x) \rightarrow \bigvee_{\phi \in \mathbf{t}(\mathfrak{E})} \phi(x)$ (Perspectives cover the Domain)

It might also be worth to distinguish other sub-kinds of extensions satisfying constraints related to the types they introduce, the covering of the domain of EMMO$_{[MC]}$, etc. For instance, outside the class of perspectives, the *naturalistic* extension refines EMMO$_{[MC]}$ by distinguishing different kinds of quanta, i.e., by 'interpreting' quanta in terms of the physical particles accepted by the Standard Model of Particle Physics. The *naturalistic* extension and the physicalistic perspective can be distinguished from other extensions from a meta-theoretical point of view, as the former stand for commitments endorsed by EMMO on epistemological grounds, following a broadly Quinean methodology contaminated by pragmatical elements, whereas the latter ideally stand

[1] The introduction of 'if and only if' axioms to reduce a predicate in the extension to the predicates of the imported theories is always possible. In this case, the consistency of the extensions is not guaranteed.

[2] Here we include in the signature not only the primitive predicates but also the ones syntactically defined.

for conceptual schemata reflecting human categorization adopted to meet expressive needs.

Another distinction can be drawn, for instance, between toolbox-like extensions and more classical taxonomy-like extensions, where the former make substantive use of parametrically defined classes/relations to partition the logical space, contra the latter. For instance, the persistence and reductionistic perspectives are toolbox-like extensions: the relations partitioning the logical space are parametrical: one depending on the ϕ with respect to which an entity is considered; the other being relative to a specific interpreter. The already cited physical perspective, instead, is closer to a standard taxonomy.

Among the views, constructions via definitional extensions to the end of alignment and the definition of other ontologies' primitives in terms of EMMO's, are particularly salient, e.g., the Time and the Temporal slice extensions.

For practical purposes, it might be useful to individuate the set of types newly introduced by a class Δ of extensions:

– $\mathbf{t}(\Delta) = \{\phi \mid \text{there exists } \mathfrak{E} \in \Delta \text{ s.t. } \phi \in \mathbf{t}(\mathfrak{E})\}$.

In particular, let us also consider $\text{EMMO}_{[\text{ALL}]}$, the set of all the extensions (including the views), $\text{EMMO}_{[\text{DEV}]}$ the set of all the extensions provided by EMMO developers (including all the perspectives), and $\text{EMMO}_{[\text{VIEW}]}$, i.e., $\text{EMMO}_{[\text{ALL}]}/\text{EMMO}_{[\text{DEV}]}$, the set of all the extensions provided by users.

4.2.2 Semiotic Perspective

EMMO's semiotic perspective, noted $\text{EMMO}_{[\text{SMS}]}$, takes inspiration from Peirce's theory of signs (and, specifically, Peirce's semiotic triangle and his classifications of sign), putting more emphasis on the role of (single) interpreters, codices, and imposing stricter causal connections between the relata.

Meta-theoretically, semiotic processes are considered in EMMO shortcuts for complex causal processes and patterns which cannot, practically, if not in principle, be expressed by means of employing a reductionistic approach. Conversely, semiosis serves to enhance the ontology's expressive power: showcasing EMMO's nominalist stance within the ontology, while abstract objects and properties proper are *de facto* not allowed in EMMO's domain, the relevant scenarios can be represented as involving semiotic processes (d_e33) in which a connection between two "concrete" objects, one taking the role of a sign (d_e27), and the other of a semiotic object (d_e28), is established by an interpreter (d_e29) via an interpretant (i.e., roughly, an internal representation, understood as physically/mechanistically grounded) (d_e30). Notably, the entities should be understood as having these roles only within the context of a specific semiosis.

It should be noted that it was decided to consider a simplified version of the semiotic perspective in the alignment, including only the more general categories, a single primitive $\texttt{SMS}(z,s,o,i,m)$ – which reads "in (the semiotic process) z, a connection between (the sign) s and the (semiotic object) o is established by (the interpreter) i on the grounds of (the interpretant) m" – and the minimal and weak axioms (a_e17)-(a_e21) (related to mereocausal aspects), which are likely to be retained in future versions.

These choices make it extremely likely that the alignment will preserve its value and applicability in the future, despite possible changes.

In line with what has been said above, the inclusion of the semiotic perspective (at least in this simplified version) was considered a precondition for the possibility of mappings concerning properties and/or qualities (tropes) – however they are represented in other ontologies.

$\texttt{SMS}(z,s,o,i,m)$	Semiosis

Table 7: Primitive relations of the semiotic perspective of EMMO.

d$_e$27 $\texttt{Sign}(s) \coloneqq \exists z\, oim(\texttt{SMS}(z,s,o,i,m))$ (*Sign*)

d$_e$28 $\texttt{SemObj}(o) \coloneqq \exists z\, sim(\texttt{SMS}(z,s,o,i,m))$ (*Semiotic Object*)

d$_e$29 $\texttt{INT}(i) \coloneqq \exists z\, som(\texttt{SMS}(z,s,o,i,m))$ (*Interpreter*)

d$_e$30 $\texttt{MR}(m) \coloneqq \exists z\, soi(\texttt{SMS}(z,s,o,i,m))$ (*Interpretant*)

d$_e$31 $\texttt{Rf}(i,o) \coloneqq \exists z\, sm(\texttt{SMS}(z,s,o,i,m))$ (*Intentional Reference*)

d$_e$32 $\texttt{Conc}(m,o) \coloneqq \exists z\, si(\texttt{SMS}(z,s,o,i,m))$ (*Pragmatic Conceptual Rel*)

d$_e$33 $\texttt{sPR}(z) \coloneqq \exists\, soim(\texttt{SMS}(z,s,o,i,m))$ (*Semiotic Process*)

a$_e$17 $\texttt{SMS}(z,s,o,i,m) \rightarrow \texttt{O}(z,s) \wedge \texttt{O}(z,o) \wedge \texttt{O}(z,m)$ (*SemPrc Composition*)

a$_e$18 $\texttt{SMS}(z,s,o,i,m) \rightarrow (\texttt{MC}(o,i) \wedge (\texttt{MC}(i,s) \vee (\texttt{MC}(s,i) \wedge \neg\texttt{MC}(o,s)))) \vee (\texttt{MC}(o,s) \wedge \texttt{MC}(s,i))$ (*Semiosis w.r.t. Causation*)

a$_e$19 $\texttt{MR}(m) \rightarrow \exists!\, i(\texttt{INT}(i) \wedge \texttt{P}(m,i))$ (*unique* INT *for an* MR)

a$_e$20 $\texttt{SMS}(z,s,o,i,m) \wedge \texttt{SMS}(z,s',o',i',m') \rightarrow (s = s' \wedge o = o' \wedge i = i' \wedge m = m')$ (SMS *Identification via* sPR)

a$_e$21 $\texttt{SMS}(z,s,o,i,m) \rightarrow (z \neq s \wedge z \neq o \wedge z \neq i \wedge \neg\texttt{O}(s,o) \wedge \neg\texttt{O}(s,i) \wedge \neg\texttt{O}(o,i))$ (*Identity constraints in a* sPR)

4.2.3 Concepts Extension

To the end of the mapping, it was deemed necessary to produce an extension pertaining to the construction of intersubjective concepts. The extension was then validated by EMMO's developers (ontologically) and through the alignment itself.

The extensions was made necessary by the gap between the approaches of the two ontologies in the representation of properties, understood broadly and from an ontological point of view. In fact, in DOLCE qualities can be seen as abstractions of tropes. Qualities are particulars inhering in individuals and they can be seen as the basic entities that are perceived or measured. In DOLCE, a quality (e.g., the color of a specific rose) is connected (at a given time) to its "value" (e.g., a particular shade of red), called a quale, describing the position of the quality within a certain quality space associated with the type of qualities under analysis and having at least an intersubjective nature.

In EMMO properties are represented as the establishment of a semiotic connection between two "material entities", one taking the role of a sign, and the other of a semiotic object, mediated by an internal representation (an interpretant) for a given subject (the interpreter). Not only properties do not make up a self-standing ontological category, but the connection between "property" and "individual" is established by subjects and, thus, not metaphysically grounded, nor intrisinc to them. While it could be argued that metatheoretically some objective elements are retained through the causal connections necessary for the semiotic process to obtain, and the mechanistic rules which specifically characterize certain kind of semiosis (e.g., observations), the semiotic perspective, per se, is strongly rooted in subjectivity and nominalism. It is also worth pointing out that in EMMO, a given entity can be both e.g., a semiotic object and a sign in different semiotic processes, that a certain entity can be a sign more than one semiotic process, resting on different interpretants (and possibly different interpreters), hence not having a determinate "meaning" intrinsically associated with them; interpretants themselves are private, and interpreters enter in semiotic processes only with a given set of entities, thus not allowing global comparison based on usage.

The concepts extension, noted $\text{EMMO}_{[\text{CPT}]}$, attempts to provide a dimension of intersubjectivity (though not objectivity) to EMMO, building intersubjective concepts (d_e36), by focusing on local comparisons of (self-reported) interpretants' applicability to given semiotic objects (d_e34), to narrow the gap with DOLCE's stance. Having introduced a relation between concepts themselves and subsumed semiotic objects (d_e37), we then provide notions capable of supporting approximative alignments with qualia and regions: (d_e38), (d_e39), (d_e40), (d_e41) (see Sect. 8.1 for the discussion of the alignments themselves). Given the complex problems outlined above, the extension has limits related to extensionality, as we were lacking the expressive resources to provide intensional definitions; nevertheless, it should serve as a practical approximation for standard use cases.

d_e34 $\mathtt{cEQ}(i_1,m_1,i_2,m_2) \coloneqq$ *(Pragmatic Conceptual Equivalence)*
$\neg\mathtt{O}(i_1,i_2) \wedge \exists z_1 s_1 z_2 s_2 o(\mathtt{SMS}(z_1,s_1,o,i_1,m_1) \wedge \mathtt{SMS}(z_2,s_2,o,i_2,m_2)) \wedge$
$\forall o_1 o_2(\mathtt{Rf}(i_1,o_1) \wedge \mathtt{Rf}(i_2,o_2) \rightarrow$
$\forall z_1 z_2 s_1 s_2 o_3 o_4(\mathtt{SMS}(z_1,s_1,o_3,i_1,m_1) \leftrightarrow \mathtt{SMS}(z_2,s_2,o_4,i_2,m_2)))$

d_e35 $\mathtt{CPT}^*(x) \coloneqq \forall a(\mathtt{qP}(a,x) \rightarrow \exists y(\mathtt{MR}(y) \wedge \mathtt{P}(y,x) \wedge \mathtt{qP}(a,y))) \wedge$
$\forall m_1 m_2(\mathtt{MR}(m_1) \wedge \mathtt{P}(m_1,x) \wedge \mathtt{MR}(m_2) \wedge \mathtt{P}(m_2,x) \wedge m_1 \neq m_2 \rightarrow$
$\exists i_1 i_2(\mathtt{cEQ}(i_1,m_1,i_2,m_2)))$

d_e36 $\mathtt{CPT}(x) \coloneqq \mathtt{CPT}^*(x) \wedge \neg\exists y(\mathtt{CPT}^*(y) \wedge \mathtt{PP}(x,y))$ *(Concept)*

d_e37 $\mathtt{CF}(c,o) \coloneqq \mathtt{CPT}(c) \wedge \exists z sim(\mathtt{SMS}(z,s,o,i,m) \wedge \mathtt{P}(m,c))$ *(Classification)*

d_e38 $\mathtt{SPC}(x,y) \coloneqq \mathtt{CPT}(x) \wedge \mathtt{CPT}(y) \wedge \forall o(\mathtt{CF}(x,o) \rightarrow \mathtt{CF}(y,o))$ *(Specialization)*

d_e39 $\mathtt{SPPC}(x,y) \coloneqq \mathtt{SPC}(x,y) \wedge \neg\mathtt{SPC}(y,x)$ *(Proper Specialization)*

d_e40 $\mathtt{TOP}(x) \coloneqq \mathtt{CPT}(x) \wedge \neg\exists y(\mathtt{SPPC}(x,y) \wedge y \neq x)$ *(Top Concept)*

d_e41 $\mathtt{MIN}(c,x) \coloneqq \mathtt{CF}(c,x) \wedge \forall d(\mathtt{SPPC}(d,c) \rightarrow \neg\mathtt{CF}(d,x))$ *(Minimal Concept)*

It is worth pointing out that (t_e11) doesn't hold. While, trivially, in the reflexive case, $\mathtt{SPC}(x,x) \rightarrow \mathtt{P}(x,x)$, ($d_e36$) excludes all the cases in which $\mathtt{PP}(x,y) \wedge \mathtt{CPT}(y)$

(it would follow that $\neg\texttt{CPT}(x)$). This is an intended result, since the MRs respectively making two concepts c_1 and c_2, should intuitively be different if $(\texttt{SPC}(c_1,c_2) \vee \texttt{SPC}(c_2,c_1)) \wedge c_1 \neq c_2$.

t_e11 $\nvdash \texttt{SPC}(x,y) \rightarrow \texttt{P}(x,y)$

Proof. Consider the model of EMMO in which $\texttt{SMS}(z,s,o,i,m_1) \wedge \texttt{SMS}(z,s,o,i,m_2) \wedge \neg\texttt{O}(m_1,m_2) \wedge \texttt{CPT}(m_1) \wedge \texttt{CPT}(m_2)$. Since, trivially, $\texttt{O}(i,i)$, then $\neg\texttt{cEQ}(m_1,m_2)$. However, by ($d_e$38), $\texttt{SPC}(m_1,m_2)$ and vice versa (assuming only o enters a semiotic relation with m_1 and m_2). □

4.2.4 Time Extension

To the end of the mapping, an extension concerning the construction of time, called EMMO$_{[\text{TM}]}$, was put forward and validated by EMMO's developers (ontologically) and on independent grounds (philosophically/formally).

The proposed construction of time rests on the idea that, in EMMO, (direct or indirect) causation between quanta intuitively implies complete temporal precedence, i.e., the effect-quantum starts to be present after the cause-quantum ceases to be present. This assumption is supported by EMMO's perdurantist stance and relative consequences for quanta, as well as by the fact that dC is characterized as intransitive and loop-less, ruling out causal connections among causes (effects) of a given interaction. In fact, given the productive nature of causation in EMMO and the lack of other relevant entities in the domain, it seems appropriate to go a bit further, and assume that, in a given interaction, the "(direct) causes cease to be present when effects appear", as the former transform into the latter. Remember that given time's emergence from causality, it makes no sense to speak of interactions' duration, so that degree of freedom shouldn't be considered in the construction.

To build time, we proceed by generalising causation to sums of quanta strengthening the notion of macro causation introduced in (d_e13) to assure that the cause completely precedes the effect, where causes and effects are now generic sums of quanta. The new notion, called *macro causal precedence* (d_e42), assumes that all the quantum parts of the cause are linked by a causation relation to (and hence, intuitively, temporally precede) all the quantum parts of the effect.

Times, our temporal individuals, are built as *sort of equivalence classes* (technically they are mereological sums) of sums of quanta that are *indistinguishable* with respect to macro causal precedence, i.e., following the reductionist stance of EMMO, times are reduced to specific sums of quanta. Notice that (i) causal chains of several quanta can be part of times and (ii) times can be proper parts (PP) of other times. Intuitively, (i) and (ii) indicate that the ontological nature of times is close to the one of periods/intervals (as opposed to points), i.e., times can have a "duration".

A reconstruction of time cannot stop at temporal individuals, as it should cover the individuals' temporal arrangement. We have just seen that there are several options concerning the nature of temporal individuals (e.g., points vs. periods); the same holds, *mutatis mutandis*, when it comes to the nature of the structuring relations of times (e.g, discrete vs. continuous orders). Hence, we introduce a precedence relation and a parthood relation on times (see respectively (d_e48) and (d_e49)) and we study their

properties. In particular, we show that they satisfy the formal requirements assumed by van Benthem in his book "The Logic of Time"[3] for a theory of time based on periods. This formally validates our construction of time from mereocausation.

In line with what has been said above, first we define the notion of *macro causal precedence*, (d_e42), where $\mathtt{pC}(x,y)$ stands for "x macro causally precedes y". Intuitively this relation assures that y starts to be present after x ceases to be present, even when macro-objects are concerned. Theorems (t_e12)-(t_e18) show that pC is a discrete partial order which is monotone with respect to P, possibly left/right bounded, and possibly non-linear.

d_e42 $\mathtt{pC}(x,y) \coloneqq \forall uv(\mathtt{qP}(u,x) \wedge \mathtt{qP}(v,y) \rightarrow \mathtt{C}(u,v))$ ((*Macro*) *Causal Precedence*)

t_e12 $\neg\mathtt{pC}(x,x)$ (pC*: Irreflexivity*)
Proof. By contradiction, assume $\mathtt{pC}(x,x)$. From (d_e42) it follows that $\forall q(\mathtt{qP}(q,x) \rightarrow \mathtt{C}(q,q))$; given ($a_e$6), $\exists q(\mathtt{qP}(q,x) \wedge \mathtt{C}(q,q))$ that contradicts (a_e8). □

t_e13 $\mathtt{pC}(x,y) \wedge \mathtt{pC}(y,z) \rightarrow \mathtt{pC}(x,z)$ (pC*: Transitivity*)
Proof. It follows trivially from (a_e9) and (d_e42). □

t_e14 $\mathtt{pC}(x,y) \rightarrow \exists z(\mathtt{pC}(x,z) \wedge \neg\exists u(\mathtt{pC}(x,u) \wedge \mathtt{pC}(u,z))) \wedge$
$\mathtt{pC}(x,y) \rightarrow \exists z(\mathtt{pC}(z,y) \wedge \neg\exists u(\mathtt{pC}(z,u) \wedge \mathtt{pC}(u,y)))$ (pC*: Discreteness*)
Proof. Consider the first conjunct. By (d_e42) we have that $\mathtt{pC}(x,y) \rightarrow \forall uv(\mathtt{qP}(u,x) \wedge \mathtt{qP}(v,y) \rightarrow \mathtt{C}(u,v))$. Thus, by ($a_e$6), $\mathtt{pC}(x,y) \rightarrow \exists v(\mathtt{qP}(v,y) \wedge \forall u(\mathtt{qP}(u,x) \rightarrow \mathtt{C}(u,v)))$, and then $\mathtt{pC}(x,y) \rightarrow \exists v(\forall u(\mathtt{qP}(u,x) \rightarrow \mathtt{C}(u,v)))$. By ($a_e$5), $\mathtt{pC}(x,y) \rightarrow \exists z(z = \sigma v\langle\forall u(\mathtt{qP}(u,x) \rightarrow \mathtt{C}(u,v))\rangle)$. From the hypotheses, by (a_e6), (a_e8), (d_e42), and the construction of z, we have that $\mathtt{pC}(x,z) \wedge \forall u(\mathtt{pC}(x,u) \rightarrow \mathtt{P}(u,z))$ and then $\mathtt{pC}(x,z) \wedge \forall u(\mathtt{pC}(x,u) \rightarrow \neg\mathtt{pC}(u,z))$. Similarly for the second conjunct.

t_e15 $\mathtt{pC}(x,y) \wedge \mathtt{P}(u,x) \wedge \mathtt{P}(v,y) \rightarrow \mathtt{pC}(u,v)$ (pC*: Monotonicity*)
Proof. Directly from (d_e42) and (a_e3) by observing that all the quanta of u are also quanta of x and all the quanta of v are also quanta of y. □

t_e16 $\mathtt{pC}(x,z) \wedge \mathtt{pC}(y,z) \wedge \mathtt{SUM}(s,x,y) \rightarrow \mathtt{pC}(s,z) \wedge$
$\mathtt{pC}(z,x) \wedge \mathtt{pC}(z,y) \wedge \mathtt{SUM}(s,x,y) \rightarrow \mathtt{pC}(z,s)$ (pC *preservation w.r.t Sum*)
Proof. Consider the first conjunct. By (d_e42) we have that $\mathtt{pC}(x,z) \rightarrow \forall uv(\mathtt{qP}(u,x) \wedge \mathtt{qP}(v,z) \rightarrow \mathtt{C}(u,v))$, and analogously for $\mathtt{pC}(y,z)$. By (d_e2), (d_e3), and extensionality (t_e1) together with (a_e2), $\mathtt{SUM}(s,x,y) \rightarrow \forall q((\mathtt{qP}(q,x) \vee \mathtt{qP}(q,y)) \leftrightarrow \mathtt{qP}(q,s))$, thus $\mathtt{pC}(x,z) \wedge \mathtt{pC}(y,z) \wedge \mathtt{SUM}(s,x,y) \rightarrow \forall uv(\mathtt{qP}(u,s) \wedge \mathtt{qP}(v,z) \rightarrow \mathtt{C}(s,z))$. The conclusion follows from (d_e42). Similarly for the second conjunct.

t_e17 $\nvdash \exists y(\mathtt{pC}(y,x)) \vee \exists y(\mathtt{pC}(x,y))$ (pC*: Existence of the Predecessor/Successor*)
Proof. Consider $\mathtt{dC}(1,2)$, $\mathtt{dC}(1,3)$, $\mathtt{dC}(2,4)$, $\mathtt{dC}(3,4)$ where $1,2,3,4$ are all different. $\neg\exists y(\mathtt{pC}(y,1))$. Consider $\mathtt{dC}(1,2)$, $\mathtt{dC}(1,3)$, $\mathtt{dC}(2,4)$, $\mathtt{dC}(3,4)$ where $1,2,3,4$ are all different. $\neg\exists y(\mathtt{pC}(4,y))$. □

t_e18 $\nvdash \mathtt{pC}(x,y) \vee \mathtt{pC}(y,x) \vee \mathtt{O}(x,y)$ (pC*: Linearity*)
Proof. Consider the situation where $\mathtt{dC}(0,1)$, $\mathtt{dC}(0,2)$, $\mathtt{dC}(1,3)$, $\mathtt{dC}(1,4)$, $\mathtt{dC}(2,5)$, and $\mathtt{dC}(2,6)$. We have $\neg\mathtt{pC}(1,2)$, $\neg\mathtt{pC}(2,1)$, and $\neg\mathtt{O}(1,2)$. □

[3] van Benthem J. The Logic of Time: A Model-Theoretic Investigation Into the Varieties of Temporal Ontology and Temporal Discourse. Dordrecht, Netherland:Kluwer Academic Publishers; 1982.

The second step to build the temporal individuals consists in the introduction of a relation of *temporal equivalence* (noted EQ), intuitively holding between temporally co-extensional sums of quanta. Our idea is to reduce EQ to indistinguishability with respect to pC. Technically, we introduce a notion of *temporal inclusion* on items (i.e., self-connected macro-entities made up of quanta, see (d_e14)) in (d_e43) which is successively extended to general, possibly non self-connected, sums of quanta in (d_e44). Temporal equivalence is then characterised as mutual temporal inclusion (d_e45). Two things are worth mentioning on this constructive step. First, note that (d_e43) does not work for generic sums of quanta, as a causation-chain c of quanta would result temporally included in any proper part of c containing the "first" – and "last" – quantum/a of c. Second, (d_e43) and (d_e44) also apply to sums of quanta including qSRCs or qSNKs of the universe (u), i.e., quanta intuitively at the beginning/end of time. Other solutions might have been equally valid. Geometrical approaches closer to Robb's orthodoxy do not have to take a stance, as the relevant models have no boundaries; thus, in this respect, EMMO offers a more neutral framework. Theorems (t_e19)-(t_e35) prove some important properties of iIN, IN, and EQ. In particular they show that EQ is an equivalence relation that does not collapse on numerical identity and that IN is a partial order on sums of quanta that is more general than parthood, preserves pC, and is preserved by P and SUM.

d_e43 $\mathtt{iIN}(x,y) \coloneqq \mathtt{ITEM}(x) \wedge \mathtt{ITEM}(y) \wedge \forall z(\mathtt{pC}(z,y) \rightarrow \mathtt{pC}(z,x)) \wedge \forall z(\mathtt{pC}(y,z) \rightarrow \mathtt{pC}(x,z))$ (*Temporal Inclusion among Items*)

d_e44 $\mathtt{IN}(x,y) \coloneqq \forall u(\mathtt{ITEM}(u) \wedge \mathtt{P}(u,x) \rightarrow \exists v(\mathtt{P}(v,y) \wedge \mathtt{iIN}(u,v)))$ (*Temporal Inclusion*)

d_e45 $\mathtt{EQ}(x,y) \coloneqq \mathtt{IN}(x,y) \wedge \mathtt{IN}(y,x)$ (*Temporal Equivalence*)

t_e19 $\mathtt{iIN}(x,x)$ (iIN*: Reflexivity*)
Proof. Directly from (d_e43). □

t_e20 $\mathtt{iIN}(x,y) \wedge \mathtt{iIN}(y,z) \rightarrow \mathtt{iIN}(x,z)$ (iIN*: Transitivity*)
Proof. From the hypothesis, by (d_e43), $\mathtt{ITEM}(x) \wedge \mathtt{ITEM}(z)$. Consider a such that $\mathtt{pC}(a,z)$. From $\mathtt{iIN}(y,z)$, by (d_e43), $\mathtt{pC}(a,y)$ that from $\mathtt{iIN}(x,y)$, by (d_e43), implies $\mathtt{pC}(a,x)$. Similarly for a such that $\mathtt{pC}(z,a)$. □

t_e21 $\mathtt{iIN}(x,y) \wedge \mathtt{P}(y,z) \wedge \mathtt{ITEM}(z) \rightarrow \mathtt{iIN}(x,z)$ (iIN *preservation w.r.t.* P)
Proof. By contradiction assume $\neg\mathtt{iIN}(x,z)$. If $\neg\mathtt{ITEM}(x)$ or $\neg\mathtt{ITEM}(z)$ we have a contradiction. Assume then that $\mathtt{ITEM}(x)$ and $\mathtt{ITEM}(z)$. From $\neg\mathtt{iIN}(x,z)$, by (d_e43), there exists u s.t. $\mathtt{pC}(u,z) \wedge \neg\mathtt{pC}(u,x)$ or $\mathtt{pC}(z,u) \wedge \neg\mathtt{pC}(x,u)$. Then, assuming $\mathtt{P}(y,z)$, by (t_e15) and the reflexivity of P, $\mathtt{pC}(u,y) \wedge \neg\mathtt{pC}(u,x)$ or $\mathtt{pC}(y,u) \wedge \neg\mathtt{pC}(x,u)$, that contradicts $\mathtt{iIN}(x,y)$. □

t_e22 $\mathtt{pC}(x,y) \wedge \mathtt{iIN}(u,x) \wedge \mathtt{iIN}(v,y) \rightarrow \mathtt{pC}(u,v)$ (pC *preservation w.r.t.* iIN)
Proof. From $\mathtt{iIN}(u,x)$, by (d_e43), $\mathtt{pC}(x,a) \rightarrow \mathtt{pC}(u,a)$ then, from $\mathtt{pC}(x,y)$, we have $\mathtt{pC}(u,y)$. From $\mathtt{iIN}(v,y)$, by (d_e43), $\mathtt{pC}(a,y) \rightarrow \mathtt{pC}(a,v)$ then, from $\mathtt{pC}(u,y)$, we have $\mathtt{pC}(u,v)$. □

t_e23 $\mathtt{IN}(x,x)$ (IN*: Reflexivity*)
Proof. Directly from (d_e44), (t_e19) and the reflexivity of P (a_e1). □

t_e24 $\mathtt{IN}(x,y) \wedge \mathtt{IN}(y,z) \rightarrow \mathtt{IN}(x,z)$ (IN*: Transitivity*)

Proof. From $\texttt{IN}(x,y)$, by (d_e44), if $\texttt{ITEM}(u) \wedge \texttt{P}(u,y)$ then there exists v s.t. $\texttt{ITEM}(v) \wedge \texttt{P}(v,y) \wedge \texttt{iIN}(u,v)$. Then, from $\texttt{IN}(y,z)$, by (d_e44), there exists w s.t. $\texttt{ITEM}(w) \wedge \texttt{P}(w,z) \wedge \texttt{iIN}(v,w)$. The thesis follows from the iIN transivity (t_e20). □

t_e25 $\texttt{P}(x,y) \rightarrow \texttt{IN}(x,y)$ (P *specializes* IN)
Proof. Consider u s.t. $\texttt{ITEM}(u) \wedge \texttt{P}(u,x)$. By the transitivity of P (a_e3), from $\texttt{P}(u,x) \wedge \texttt{P}(x,y)$, we have $\texttt{P}(u,y)$ and, by (t_e19), $\texttt{P}(u,y) \wedge \texttt{iIN}(u,u)$. □

t_e26 $\texttt{TME}(y) \wedge \texttt{IN}(x,y) \rightarrow \texttt{P}(x,y)$
Proof. By contradiction assume $\texttt{TME}(y) \wedge \texttt{IN}(x,y) \wedge \neg\texttt{P}(x,y)$. From $\neg\texttt{P}(x,y)$, by (a_e4), (a_e6) and (t_e1), $\exists q(\texttt{qP}(q,x) \wedge \neg\texttt{qP}(q,y))$. From $\texttt{IN}(x,y) \wedge \texttt{qP}(q,x)$, by ($a_e$1) and ($t_e$27), $\texttt{IN}(q,y)$. From $\texttt{IN}(q,y)$, given that q is the only item in q (see (t_e4)), by (d_e44) and (d_e43), $\exists i(\texttt{ITEM}(i) \wedge \texttt{P}(i,y) \wedge \forall z(\texttt{pC}(z,i) \rightarrow \texttt{pC}(z,q)) \wedge \forall z(\texttt{pC}(i,z) \rightarrow \texttt{pC}(q,z))$. We prove that $\texttt{EQ}(q{+}i,i)$, i.e., by (d_e45), $\texttt{IN}(q{+}i,i) \wedge \texttt{IN}(i,q{+}i)$. By construction $\texttt{IN}(q,i)$ and, by (t_e23), $\texttt{IN}(i,i)$ thus, by (t_e28), $\texttt{IN}(q{+}i,i)$. $\texttt{IN}(i,q{+}i)$ follows directly by (t_e25) given that $\texttt{P}(i,q{+}i)$.
From $\texttt{P}(i,y) \wedge \texttt{EQ}(q{+}i,i)$, by ($t_e$35), $\texttt{EQ}(y,y{+}q)$ thus, given that $\texttt{TME}(y)$, by (d_e47), (d_e46) and (t_e36), $\texttt{P}(y{+}q,y)$ and then $\texttt{qP}(q,y)$. Contradiction. □

t_e27 $\texttt{P}(z,x) \wedge \texttt{P}(y,u) \wedge \texttt{IN}(x,y) \rightarrow \texttt{IN}(z,u)$ (IN *preservation w.r.t.* P)
Proof. Directly from (t_e24) and (t_e25). □

t_e28 $\texttt{IN}(x,z) \wedge \texttt{IN}(y,z) \wedge \texttt{SUM}(s,x,y) \rightarrow \texttt{IN}(s,z)$ (IN *preservation w.r.t.* SUM)
Proof. By contradiction. From $\neg\texttt{IN}(s,z)$, by (d_e44), $\exists u(\texttt{P}(u,s) \wedge \forall v(\texttt{P}(v,z) \rightarrow \neg\texttt{iIN}(u,v)))$, i.e., by ($d_e$43), $\texttt{P}(u,s) \wedge \forall v(\texttt{P}(v,z) \rightarrow ([1]\ \exists a(\texttt{pC}(a,v) \wedge \neg\texttt{pC}(a,u)) \vee [2]\ \exists a(\texttt{pC}(v,a) \wedge \neg\texttt{pC}(u,a))))$. If u is part of x or y, trivially $\neg\texttt{IN}(x,z)$ or $\neg\texttt{IN}(y,z)$ and then we have a contradiction. Assume then that u overlaps both x and y.
Consider [1]. From $\neg\texttt{pC}(a,u)$, by (t_e16), the hypothesis that u overlaps both x and y, the existence of the mereological product (that follows from (a_e5) and the fact that u overlaps both x and y), and the fact that $\texttt{P}(u,s) \wedge \texttt{SUM}(s,x,y)$, we have [1.1] $\neg\texttt{pC}(a,u{\times}x)$ or [1.2] $\neg\texttt{pC}(a,u{\times}y)$.
Consider [1.1], i.e., $\exists u(\texttt{P}(u,s) \wedge \forall v(\texttt{P}(v,z) \rightarrow \exists a(\texttt{pC}(a,v) \wedge \neg\texttt{pC}(a,u{\times}x)))$, i.e., by ($d_e$43), $\exists u(\texttt{P}(u,s) \wedge \forall v(\texttt{P}(v,z) \rightarrow \neg\texttt{iIN}(v,u{\times}x)))$. Consider now a q s.t. $\texttt{qP}(q,u{\times}x)$. From $\texttt{qP}(q,u{\times}x) \wedge \neg\texttt{iIN}(v,u{\times}x)$, by ($t_e$21), $\neg\texttt{iIN}(v,q)$, then $\exists q(\texttt{qP}(q,x) \wedge \texttt{ITEM}(q) \wedge \forall v(\texttt{P}(v,z) \rightarrow \neg\texttt{iIN}(v,q)))$, i.e., by ($d_e$44), $\neg\texttt{IN}(x,z)$. Contradiction.
Consider [1.2], i.e., $\exists u(\texttt{P}(u,s) \wedge \forall v(\texttt{P}(v,z) \rightarrow \exists a(\texttt{pC}(a,v) \wedge \neg\texttt{pC}(a,u{\times}y)))$. Following the reasoning done for [1.1] we obtain $\neg\texttt{IN}(y,z)$. Contradiction.
Consider [2]. From $\neg\texttt{pC}(u,a)$, by (t_e16), the hypothesis that u overlaps both x and y, the existence of the mereological product, and the fact that $\texttt{P}(u,s) \wedge \texttt{SUM}(s,x,y)$, we have [2.1] $\neg\texttt{pC}(u{\times}x,a)$ or [2.2] $\neg\texttt{pC}(u{\times}y,a)$.
Consider [2.1], i.e., $\exists u(\texttt{P}(u,s) \wedge \forall v(\texttt{P}(v,z) \rightarrow \exists a(\texttt{pC}(a,v) \wedge \neg\texttt{pC}(u{\times}x,a)))$, i.e., by ($d_e$43), $\exists u(\texttt{P}(u,s) \wedge \forall v(\texttt{P}(v,z) \rightarrow \neg\texttt{iIN}(v,u{\times}x)))$. We obtain a contradiction following what done for the case [1.1].
Consider [2.2], i.e., $\exists u(\texttt{P}(u,s) \wedge \forall v(\texttt{P}(v,z) \rightarrow \exists a(\texttt{pC}(a,v) \wedge \neg\texttt{pC}(u{\times}y,a)))$, i.e., by ($d_e$43), $\exists u(\texttt{P}(u,s) \wedge \forall v(\texttt{P}(v,z) \rightarrow \neg\texttt{iIN}(v,u{\times}y)))$. We obtain a contradiction following what done for the case [1.2]. □

t_e29 $s = \sigma w\langle\texttt{IN}(w,z)\rangle \rightarrow \texttt{IN}(s,z)$ (IN *preservation w.r.t. u. comp.*)

Proof. By contradiction. From $\neg\mathtt{IN}(s,z)$, by ($\mathrm{d_e}$44), $\exists u(\mathtt{P}(u,s)\wedge\forall v(\mathtt{P}(v,z)\rightarrow\neg\mathtt{iIN}(u,v)))$, i.e., by ($\mathrm{d_e}$43), $\mathtt{P}(u,s)\wedge\forall v(\mathtt{P}(v,z)\rightarrow([1]\ \exists a(\mathtt{pC}(a,v)\wedge\neg\mathtt{pC}(a,u))\vee[2]\ \exists a(\mathtt{pC}(v,a)\wedge\neg\mathtt{pC}(u,a))))$. If u is part of one of the ws, trivially $\neg\mathtt{IN}(w,z)$ and then we have a contradiction. Assume then that u overlaps several ws: at the very least, it has to overlap two of them; let us thus assume that u overlaps x and y.
Consider [1]. From $\neg\mathtt{pC}(a,u)$, by ($\mathrm{t_e}$16), the hypothesis that u overlaps both x and y, the existence of the mereological product (that follows from ($\mathrm{a_e}$5) and the fact that u overlaps both x and y), the fact that $\mathtt{P}(u,s)\wedge s=\sigma w\langle(\mathtt{IN}(w,z))\rangle$, and that x and y are ws, we have [1.1] $\neg\mathtt{pC}(a,u{\times}x)$ or [1.2] $\neg\mathtt{pC}(a,u{\times}y)$.
Consider [1.1], i.e., $\exists u(\mathtt{P}(u,s)\wedge\forall v(\mathtt{P}(v,z)\rightarrow\exists a(\mathtt{pC}(a,v)\wedge\neg\mathtt{pC}(a,u{\times}x)))$, i.e., by ($\mathrm{d_e}$43), $\exists u(\mathtt{P}(u,s)\wedge\forall v(\mathtt{P}(v,z)\rightarrow\neg\mathtt{iIN}(v,u{\times}x)))$. Consider now a q s.t. $\mathtt{qP}(q,u{\times}x)$. From $\mathtt{qP}(q,u{\times}x)\wedge\neg\mathtt{iIN}(v,u{\times}x)$, by ($\mathrm{t_e}$21), $\neg\mathtt{iIN}(v,q)$, then $\exists q(\mathtt{qP}(q,x)\wedge\mathtt{ITEM}(q)\wedge\forall v(\mathtt{P}(v,z)\rightarrow\neg\mathtt{iIN}(v,q)))$, i.e., by ($\mathrm{d_e}$44), $\neg\mathtt{IN}(x,z)$. Contradiction.
Consider [1.2], i.e., $\exists u(\mathtt{P}(u,s)\wedge\forall v(\mathtt{P}(v,z)\rightarrow\exists a(\mathtt{pC}(a,v)\wedge\neg\mathtt{pC}(a,u{\times}y)))$. Following the reasoning done for [1.1] we obtain $\neg\mathtt{IN}(y,z)$. Contradiction.
Consider [2]. From $\neg\mathtt{pC}(u,a)$, by ($\mathrm{t_e}$16), the hypothesis that u overlaps both x and y, the existence of the mereological product, the fact that $\mathtt{P}(u,s)\wedge s=\sigma w\langle(\mathtt{IN}(w,z))\rangle$, and that x and y are ws we have [2.1] $\neg\mathtt{pC}(u{\times}x,a)$ or [2.2] $\neg\mathtt{pC}(u{\times}y,a)$.
Consider [2.1], i.e., $\exists u(\mathtt{P}(u,s)\wedge\forall v(\mathtt{P}(v,z)\rightarrow\exists a(\mathtt{pC}(a,v)\wedge\neg\mathtt{pC}(u{\times}x,a)))$, i.e., by ($\mathrm{d_e}$43), $\exists u(\mathtt{P}(u,s)\wedge\forall v(\mathtt{P}(v,z)\rightarrow\neg\mathtt{iIN}(v,u{\times}x)))$. We obtain a contradiction following what done for the case [1.1].
Consider [2.2], i.e., $\exists u(\mathtt{P}(u,s)\wedge\forall v(\mathtt{P}(v,z)\rightarrow\exists a(\mathtt{pC}(a,v)\wedge\neg\mathtt{pC}(u{\times}y,a)))$, i.e., by ($\mathrm{d_e}$43), $\exists u(\mathtt{P}(u,s)\wedge\forall v(\mathtt{P}(v,z)\rightarrow\neg\mathtt{iIN}(v,u{\times}y)))$. We obtain a contradiction following what done for the case [1.2]. □

$\mathrm{t_e}$30 $\mathtt{pC}(x,y)\wedge\mathtt{IN}(u,x)\wedge\mathtt{IN}(v,y)\rightarrow\mathtt{pC}(u,v)$ (pC *preservation w.r.t.* IN)
Proof. Consider a,b s.t. $\mathtt{qP}(a,u)$ and $\mathtt{qP}(b,v)$. Given ($\mathrm{d_e}$42), we need to prove that $\mathtt{C}(a,b)$. From $\mathtt{IN}(u,x)$ and $\mathtt{qP}(a,u)$, by ($\mathrm{d_e}$44), there exists $\bar{a}$ s.t. $\mathtt{P}(\bar{a},x)\wedge\mathtt{iIN}(a,\bar{a})$. From $\mathtt{IN}(v,y)$ and $\mathtt{qP}(b,v)$, by ($\mathrm{d_e}$44), there exists $\bar{b}$ s.t. $\mathtt{P}(\bar{b},y)\wedge\mathtt{iIN}(b,\bar{b})$. From $\mathtt{P}(\bar{a},x)\wedge\mathtt{P}(\bar{b},y)\wedge\mathtt{pC}(x,y)$, by ($\mathrm{t_e}$15), $\mathtt{pC}(\bar{a},\bar{b})$. From $\mathtt{pC}(\bar{a},\bar{b})\wedge\mathtt{iIN}(a,\bar{a})\wedge\mathtt{iIN}(b,\bar{b})$, by ($\mathrm{t_e}$22), $\mathtt{pC}(a,b)$ that, given the atomicity of a and b, implies $\mathtt{C}(a,b)$. □

$\mathrm{t_e}$31 $\nvdash\mathtt{EQ}(x,y)\rightarrow x=y$ (EQ*: No Collapse on Numerical Identity*)
Proof. Consider $\mathtt{dC}(1,2)$, $\mathtt{dC}(1,4)$, $\mathtt{dC}(3,4)$, $\mathtt{dC}(3,2)$ where $1,2,3,4$ are all different. We have $\mathtt{EQ}(1,3)$ but $1\neq 3$. □

$\mathrm{t_e}$32 $\mathtt{EQ}(x,x)$ (EQ*: Reflexivity*)
Proof. Directly from ($\mathrm{d_e}$45) and ($\mathrm{t_e}$23). □

$\mathrm{t_e}$33 $\mathtt{EQ}(x,y)\rightarrow\mathtt{EQ}(y,x)$ (EQ*: Simmetry*)
Proof. Directly from ($\mathrm{d_e}$45). □

$\mathrm{t_e}$34 $\mathtt{EQ}(x,y)\wedge\mathtt{EQ}(y,z)\rightarrow\mathtt{EQ}(x,z)$ (EQ*: Transitivity*)
Proof. Directly from ($\mathrm{d_e}$45) and ($\mathrm{t_e}$24). □

$\mathrm{t_e}$35 $\mathtt{P}(a,x)\wedge\mathtt{EQ}(b,a)\rightarrow\mathtt{EQ}(x,(x{-}a){+}b)$
(EQ *preservation w.r.t.* EQ*-Parts Substitution*)

Proof. By (d_e45), we prove that [A] $\mathtt{IN}(x,(x{-}a){+}b)$ and [B] $\mathtt{IN}((x{-}a){+}b,x)$.
Consider [A]. By (t_e32), $\mathtt{EQ}(x{-}a,x{-}a)$ and, by hypothesis, $\mathtt{EQ}(a,b)$. By (d_e45), $\mathtt{IN}(x{-}a,x{-}a)$ and $\mathtt{IN}(a,b)$. From $\mathtt{IN}(x{-}a,x{-}a)$ and $\mathtt{P}(x{-}a,(x{-}a){+}b)$, by ($t_e$27), $\mathtt{IN}(x{-}a,(x{-}a){+}b)$. From $\mathtt{IN}(a,b)$ and $\mathtt{P}(b,(x{-}a){+}b)$, by ($t_e$27), $\mathtt{IN}(a,(x{-}a){+}b)$. From $\mathtt{IN}(x{-}a,(x{-}a){+}b)$ and $\mathtt{IN}(a,(x{-}a){+}b)$, by ($t_e$28) and by observing that $\mathtt{P}(a,x)\rightarrow x=(x{-}a){+}a$, $\mathtt{IN}(x,(x{-}a){+}b)$.
Consider [B]. Following what done for the case [A], from $\mathtt{IN}(x{-}a,x{-}a)$ and $\mathtt{P}(x{-}a,x)$, by (t_e27), $\mathtt{IN}(x{-}a,x)$. From $\mathtt{IN}(b,a)$ and $\mathtt{P}(a,x)$, by (t_e27), $\mathtt{IN}(b,x)$. From $\mathtt{IN}(x{-}a,x)$ and $\mathtt{IN}(b,x)$, by (t_e28), $\mathtt{IN}((x{-}a){+}b,x)$. □

In the last step of the construction of temporal individuals, *times* are defined as maximal sums of temporally equivalent entities, i.e., times can be seen as sort of equivalence classes (EQ) of sums of quanta. We follow a procedure which is standard in mathematics: first we introduce the sum of quanta x obtained by summing up all the entities temporally equivalent to a given entity a, see (d_e46) and (t_e36). This corresponds to the definition $[a]_{\mathtt{EQ}}=\{x\in D:\mathtt{EQ}(x,a)\}$ (where D is our domain). Then, we collect under times (TME) all the EQ equivalent classes, see (d_e47), i.e., TME corresponds to the quotient set $D/\mathtt{EQ}$.

d_e46 $\mathtt{EC}(x,a)\coloneqq\forall z(\mathtt{qP}(z,x)\leftrightarrow\exists w(\mathtt{EQ}(w,a)\wedge\mathtt{qP}(z,w)))$ (*Maximal Sum of* EQs)

d_e47 $\mathtt{TME}(x)\coloneqq\exists y(\mathtt{EC}(x,y))$ (*Time*)

t_e36 $\mathtt{EC}(x,a)\leftrightarrow x=\sigma y\langle\mathtt{EQ}(y,a)\rangle$ (*Fusion-based definition of* EC)
Proof. By the definitions of fusion and EC, we need to prove that $\forall u(\mathtt{qP}(u,x)\leftrightarrow\exists v(\mathtt{EQ}(v,a)\wedge\mathtt{qP}(u,v)))$ if and only if $\forall u(\mathtt{O}(u,x)\leftrightarrow\exists v(\mathtt{EQ}(v,a)\wedge\mathtt{O}(v,u)))$.
Consider ($\rightarrow$). Assume $\mathtt{O}(u,x)$, by (a_e6) and the definition of overlap, $\exists q(\mathtt{qP}(q,u)\wedge\mathtt{qP}(q,x))$. From $\mathtt{qP}(q,x)$, by the hypothesis, $\exists v(\mathtt{EQ}(v,a)\wedge\mathtt{qP}(q,v))$. Then we have that $\mathtt{qP}(q,v)$ and $\mathtt{qP}(q,u)$, i.e., $\mathtt{O}(v,u)$. Assume $\exists v(\mathtt{EQ}(v,a)\wedge\mathtt{O}(v,u))$, by ($a_e$6) and the definition of overlap, $\exists q(\mathtt{qP}(q,v)\wedge\mathtt{qP}(q,u))$. From $\exists v(\mathtt{EQ}(v,a)\wedge\mathtt{qP}(q,v))$, by the hypothesis, $\mathtt{qP}(q,x)$ that, together with $\mathtt{qP}(q,u)$, implies $\mathtt{O}(u,x)$.
Consider ($\leftarrow$). Assume $\mathtt{qP}(u,x)$, then $\mathtt{O}(u,x)$, and then, by the hypothesis, $\exists v(\mathtt{EQ}(v,a)\wedge\mathtt{O}(v,u))$ that, by the atomicity of u, implies $\mathtt{qP}(u,v)$. Assume $\exists v(\mathtt{EQ}(v,a)\wedge\mathtt{qP}(u,v))$, then $\exists v(\mathtt{EQ}(v,a)\wedge\mathtt{O}(v,u))$ and, by the hypothesis, $\mathtt{O}(u,x)$, and because u is atomic, $\mathtt{qP}(u,x)$. □

t_e37 $\mathtt{EC}(x,a)\rightarrow\mathtt{P}(a,x)$
Proof. Directly from (t_e36) and (t_e32). □

t_e38 $\mathtt{EC}(x,a)\rightarrow\mathtt{EQ}(x,a)$
Proof. From the hypotheses, by (t_e37) and (t_e25) it follows that $\mathtt{IN}(a,x)$. From the hypotheses, by (d_e46) and (d_e45), $\forall q(\mathtt{qP}(q,x)\rightarrow\exists w(\mathtt{IN}(w,a)\wedge\mathtt{qP}(q,w))$ then, by (t_e25) and (t_e24), $\forall q(\mathtt{qP}(q,x)\rightarrow\mathtt{IN}(q,a))$ and then, by (t_e29), $\mathtt{IN}(x,a)$. The conclusion follows from (d_e45). □

We now turn to the structure of time by introducing a precedence relation (tpC) and a parthood relation (tP) on times; see (d_e48) and (d_e49). These new notions are trivial restrictions of pC and P to times. Concerning overlap between times (tO), (d_e50) and (t_e40) demonstrate that it is equivalent to define this notion starting from P-part or tP-part.

It is now possible to check what constraints are satisfied by tP and tpC, among the ones considered by van Benthem in the Logic of Time as relevant for the characterization of temporal structures based on periods. Theorems (t_e41)-(t_e45) show that tP is an extensional mereology closed under product, while (t_e46) and (t_e47) show that tpC is a partial order. (t_e48), (t_e49), and (t_e50) correspond to the constraints called, respectively, *separatedness*, *monotonicity*, and MOND in the Logic of Time of van Benthem. Van Benthem assumes that these constraints are sufficient to (minimally) characterise a structure of periods.[4] Theorems (t_e52)-(t_e55) better characterise the nature of the time resulting from our construction: namely, boundness, linearity and directness do not hold and times are not necessarily convex.

d_e48 $\mathtt{tpC}(x,y) \coloneqq \mathtt{TME}(x) \wedge \mathtt{TME}(y) \wedge \mathtt{pC}(x,y)$ (*Macro Causal Precedence btw Times*)

d_e49 $\mathtt{tP}(x,y) \coloneqq \mathtt{TME}(x) \wedge \mathtt{TME}(y) \wedge \mathtt{P}(x,y)$ (*Parthood between Times*)

d_e50 $\mathtt{tO}(x,y) \coloneqq \mathtt{TME}(x) \wedge \mathtt{TME}(y) \wedge \mathtt{O}(x,y)$ (*Overlap between Times*)

t_e39 $\mathtt{TME}(x) \wedge \mathtt{qP}(a,x) \wedge s = \sigma y \langle \mathtt{EQ}(y,a) \rangle \rightarrow \mathtt{tP}(s,x)$
Proof. First, note that, from $s = \sigma y \langle \mathtt{EQ}(y,a) \rangle$, by ($t_e$36), we have $\mathtt{EC}(s,a)$ and then, by (d_e47), $\mathtt{TME}(s)$. From $\mathtt{TME}(x)$, by (t_e32) and (d_e46), we have $\mathtt{EC}(x,x)$. We prove that $\mathtt{EC}(s,a) \wedge \mathtt{EC}(x,x) \wedge \mathtt{qP}(a,x) \rightarrow \mathtt{P}(s,x)$, i.e., given ($t_e$1), that all the quantum parts of s are parts of x. Consider q such that $\mathtt{qP}(q,s)$. From $\mathtt{EC}(s,a)$, by (d_e46), $\exists w(\mathtt{EQ}(w,a) \wedge \mathtt{qP}(q,w))$. From $\mathtt{qP}(a,x) \wedge \mathtt{EQ}(w,a)$, by ($t_e$33) and ($t_e$35), $\mathtt{EQ}((x-a)+w,x)$. We prove that $\mathtt{EQ}((x-a)+w,x) \wedge \mathtt{EC}(x,x) \rightarrow \mathtt{P}((x-a)+w,x)$. Consider a c such that $\mathtt{qP}(c,(x-a)+w)$. From $\mathtt{EQ}((x-a)+w,x) \wedge \mathtt{qP}(c,(x-a)+w) \wedge \mathtt{EC}(x,x)$, by ($d_e$46), $\mathtt{qP}(c,x)$. This proves that $\mathtt{P}((x-a)+w,x)$ and then, from $\mathtt{qP}(q,w)$, by (t_e3) and (a_e3), $\mathtt{qP}(q,x)$. This proves that $\mathtt{P}(s,x)$. The thesis follows directly from (d_e49). □

t_e40 $\mathtt{tO}(x,y) \rightarrow \exists z(\mathtt{tP}(z,x) \wedge \mathtt{tP}(z,y))$ (tO*'s relation with* tP)
Proof. From the hypotheses, by (d_e50) and (a_e6), $\exists a(\mathtt{qP}(a,x) \wedge \mathtt{qP}(a,y))$. By ($a_e$5) and ($t_e$32), there exists $z = \sigma u \langle \mathtt{EQ}(u,a) \rangle$ and, by (t_e39), $\mathtt{tP}(z,x) \wedge \mathtt{tP}(z,y)$. □

t_e41 $\mathtt{TME}(x) \rightarrow \mathtt{tP}(x,x)$ (TME*: Reflexivity*)
Proof. It follows trivially from (a_e1). □

t_e42 $\mathtt{tP}(x,y) \wedge \mathtt{tP}(y,x) \rightarrow x = y$ (TME*: Antisymmetry*)
Proof. It follows trivially from (a_e2). □

t_e43 $\mathtt{tP}(x,y) \wedge \mathtt{tP}(y,z) \rightarrow \mathtt{tP}(x,z)$ (TME*: Transitivity*)
Proof. It follows trivially from (a_e3). □

t_e44 $\mathtt{TME}(x) \wedge \mathtt{TME}(y) \wedge \neg\mathtt{tP}(x,y) \rightarrow \exists z(\mathtt{tP}(z,x) \wedge \neg\mathtt{tO}(z,y))$ (*Strong Suppl. btw* TME*s*)
Proof. From the hypotheses, by (a_e4) and (d_e49), we have that $\exists z(\mathtt{P}(z,x) \wedge \neg\mathtt{O}(z,y))$ and then, by (a_e6), $\exists a(\mathtt{qP}(a,x) \wedge \neg\mathtt{O}(a,y))$. By ($a_e$5) and ($t_e$32), there exists $s = \sigma u \langle \mathtt{EQ}(u,a) \rangle$ and, by (t_e39), $\mathtt{tP}(s,x)$. It remains to prove that $\neg\mathtt{O}(s,y)$. By contradiction assume that $\mathtt{O}(s,y)$. By (d_e1) and (a_e6), $\exists b(\mathtt{qP}(b,s) \wedge \mathtt{qP}(b,y))$. From the definition of the fusion s, we have that $\mathtt{EQ}(a,b)$ and, by the unicity of

[4] To be precise, strong supplementation of tP is not among the requirements listed by van Benthem.

the fusion, $s = \sigma u\langle \texttt{EQ}(u,b)\rangle$. Then, from $\texttt{TME}(y) \wedge \texttt{qP}(b,y) \wedge s = \sigma u\langle \texttt{EQ}(u,b)\rangle$, by ($t_e$39), $\texttt{tP}(s,y)$ and then $\texttt{qP}(a,y)$ against $\neg\texttt{O}(a,y)$. □

t_e45 $\texttt{tO}(x,y) \rightarrow \exists z(\texttt{TME}(z) \wedge \texttt{PRD}(z,x,y))$ (*Existence of the* TME *Product*)
Proof. From the hypothesis, by (d_e50), $\texttt{O}(x,y)$, and thus, by (a_e5) and the definition of product, $\exists z(\texttt{PRD}(z,x,y))$. We need to prove that $\texttt{TME}(z)$. To do so, we prove $\texttt{EC}(z,z)$, i.e., [1] $\forall k(\texttt{qP}(k,z) \rightarrow \exists w(\texttt{EQ}(w,z) \wedge \texttt{qP}(k,w)))$ and [2] $\forall k(\exists w(\texttt{EQ}(w,z) \wedge \texttt{qP}(k,w)) \rightarrow \texttt{qP}(k,z))$. Consider [1]. It follows from (t_e32) and the hypotheses. Consider [2]. By (d_e4), $\texttt{P}(z,x)$. Then from $\texttt{P}(z,x) \wedge \texttt{EQ}(w,z)$, by ($t_e$35), $\texttt{EQ}(x,(x{-}z){+}w)$. From $\texttt{qP}(k,w)$, by the definition of sum, $\texttt{qP}(k,(x{-}z){+}w)$, thus from $\texttt{TME}(x) \wedge \texttt{EQ}(x,(x{-}z){+}w) \wedge \texttt{qP}(k,(x{-}z){+}w)$, by ($d_e$46) and ($d_e$47), $\texttt{qP}(k,x)$. By the definition of the product, we also have that $\texttt{P}(z,y)$. Following the same reasoning we also have that $\texttt{qP}(k,y)$ and then, by the definition of the product, $\texttt{qP}(k,z)$. □

t_e46 $\neg\texttt{tpC}(x,x)$ (tpC*: Irreflexivity*)
Proof. It follows trivially from (t_e12). □

t_e47 $\texttt{tpC}(x,y) \wedge \texttt{tpC}(y,z) \rightarrow \texttt{tpC}(x,z)$ (tpC*: Transitivity*)
Proof. It follows trivially from (t_e13). □

t_e48 $\texttt{tpC}(x,y) \rightarrow \neg\texttt{tO}(x,y)$ (tpC*: Separatedness*)
Proof. By contradiction assume that there exists z s.t. $\texttt{qP}(z,x) \wedge \texttt{qP}(z,y)$. By ($a_e$8) we have $\neg\texttt{C}(z,z)$ against (d_e42). □

t_e49 $\texttt{tpC}(x,y) \wedge \texttt{tP}(z,x) \rightarrow \texttt{tpC}(z,y) \wedge$
$\texttt{tpC}(x,y) \wedge \texttt{tP}(z,y) \rightarrow \texttt{tpC}(x,z)$ (tpC*: Monotonicity*)
Proof. Directly from the definitions of tpC and tP by (t_e15) and (a_e1). □

t_e50 $\texttt{tpC}(y,x) \wedge \texttt{tpC}(z,x) \wedge \texttt{SUM}(s,y,z) \wedge \texttt{TME}(s) \rightarrow \texttt{tpC}(s,x) \wedge$
$\texttt{tpC}(x,y) \wedge \texttt{tpC}(x,z) \wedge \texttt{SUM}(s,y,z) \wedge \texttt{TME}(s) \rightarrow \texttt{tpC}(x,s)$ (tpC: MOND)
Proof. Directly from the definitions of tpC and tP by (t_e16). □

t_e51 $\texttt{TME}(\texttt{u})$ (the Universe is a TME)
Proof. By (a_e11) $\texttt{ITEM}(\texttt{u})$. By the definition of u we have that EC(u,u) because u contains all the existing quanta. □

t_e52 $\nvdash \texttt{TME}(x) \rightarrow \exists y(\texttt{tpC}(y,x) \vee \texttt{tpC}(x,y))$ (tpC*: Existence Predecessor/Successor*)
Proof. By (t_e51) and the definition of the universe; consider $x = \texttt{u}$. □

t_e53 $\nvdash \exists z(\texttt{tpC}(x,z) \wedge \texttt{tpC}(y,z)) \wedge \exists z(\texttt{tpC}(z,x) \wedge \texttt{tpC}(z,y))$ (tpC*: Directness*)
Proof. By (t_e51) and the definition of the universe; consider $x = \texttt{u}$. □

t_e54 $\nvdash \texttt{TME}(x) \wedge \texttt{TME}(y) \rightarrow \texttt{tpC}(x,y) \vee \texttt{tpC}(y,x) \vee \texttt{tO}(x,y)$ (tpC*: Linearity*)
Proof. Consider the case $\texttt{dC}(0,1)$, $\texttt{dC}(0,2)$, $\texttt{dC}(1,3)$, $\texttt{dC}(1,4)$, $\texttt{dC}(2,5)$, $\texttt{dC}(2,6)$. We have $\texttt{TME}(1)$ and $\texttt{TME}(2)$ but 1 and 2 are not in an tpC or tO relation. □

t_e55 $\nvdash \texttt{tpC}(x,y) \wedge \texttt{tpC}(y,z) \wedge \texttt{tP}(x,u) \wedge \texttt{tP}(z,u) \rightarrow \texttt{tP}(y,u)$ (tpC*: Convexity*)
Proof. Consider the situation $\texttt{dC}(0,1)$, $\texttt{dC}(0,2)$, $\texttt{dC}(1,3)$, $\texttt{dC}(1,4)$. We have $\texttt{TME}(0)$, $\texttt{TME}(1)$, $\texttt{TME}(3{+}4)$, $\texttt{TME}(0{+}3{+}4)$. Furthermore we have, $\texttt{tpC}(0,1)$, $\texttt{tpC}(1,3{+}4)$, $\texttt{tP}(0,0{+}3{+}4)$, $\texttt{tP}(3{+}4,0{+}3{+}4)$ but $\neg\texttt{tP}(1,0{+}3{+}4)$. □

Furthermore, time is not atomic.

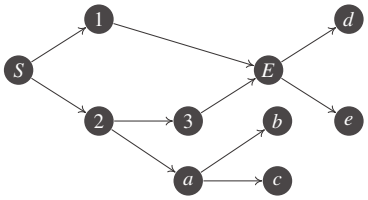

Figure 5: The sum of quanta temporally equivalent to a quantum is not an atomic time

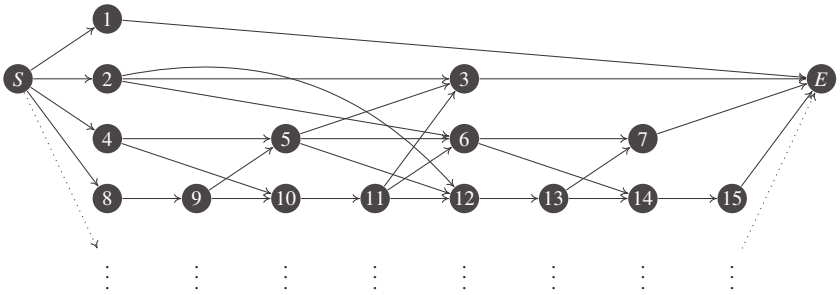

Figure 6: Time is not atomic

d_e51 $\mathtt{aTME}(x) \coloneqq \forall y(\mathtt{tP}(y,x) \rightarrow y = x)$ (*Atomic time*)

t_e56 $\mathtt{aTME}(t) \wedge \mathtt{qP}(a,t) \wedge s = \sigma y\langle \mathtt{EQ}(y,a)\rangle \rightarrow s = t$
Proof. From the hypotheses, by (t_e39), $\mathtt{tP}(s,t)$ and then, by (d_e51), the thesis. □

t_e57 $\mathtt{aTME}(t) \rightarrow \exists a(\mathtt{qP}(a,t) \wedge t = \sigma y\langle \mathtt{EQ}(y,a)\rangle)$
Proof. By (a_e5) and (a_e6), $\exists as(\mathtt{qP}(a,t) \wedge s = \sigma y\langle \mathtt{EQ}(y,a)\rangle)$ and then, by (t_e56), $s = t$. □

t_e58 $\nvdash \mathtt{Q}(a) \wedge t = \sigma y\langle \mathtt{EQ}(y,a)\rangle \rightarrow \mathtt{aTME}(t)$
Proof. Consider the model of EMMO depicted in Fig. 5. We have $\mathtt{Q}(1)$, $123 = \sigma y\langle \mathtt{EQ}(y,1)\rangle$, $\mathtt{tP}(3,123)$, and $3 \neq 123$. □

t_e59 $\nvdash \mathtt{TME}(x) \rightarrow \exists y(\mathtt{aTME}(y) \wedge \mathtt{tP}(y,x))$ (*Non Atomicity of Time*)
Proof. (Sketch) Consider the infinite model of EMMO in Fig.6 and assume $\mathtt{EC}(t_1,1)$, $\mathtt{EC}(t_2,2)$, and $\mathtt{EC}(t_3,3)$. We have that $t_1 = 1+t_2+t_3$, $t_1 \neq t_2$, and $t_1 \neq t_3$, i.e., t_1 is not an atomic interval. Furthermore, 1 is a quantum that is not a time, therefore the eventual atomic subintervals of t_1 must be part of t_2 or t_3. Given the construction of the model in Fig.6, t_2 and t_3 have the same structure of t_1, i.e., t_2 and t_3 are not atomic intervals and they are the sum of a quantum (that is not an interval) and two intervals with the same structure of t_1. The reasoning can be iterated ad infinitum and all the subintervals of t_1 are not atomic. □

4.2.5 Existence at a Time and Temporal Slice extension

This extension, noted EMMO$_{[TS]}$, introduces two important notions: the notion of existence in time and the notion of temporal slice.

(d_e52) defines existence in time ($\texttt{EX}(x,t)$ stands for "*x exists* at time *t*"). (t_e60)-(t_e66) show some useful properties of EX. Note that quanta can be extended in time, i.e., they can exist at different times. For instance, in Fig. 5, $\texttt{EX}(1,123) \wedge \texttt{TME}(123)$, $\texttt{P}(2,123)$, and $\texttt{P}(3,123)$, i.e., according to (d_e52), $\texttt{EX}(1,2)$ and $\texttt{EX}(1,3)$.

In Sect. 8.2.4 we will analyze the link between EX and the DOLCE PRE defined in (d_d5) on the basis of TLC and $\texttt{P}_\texttt{d}$.

d_e52 $\texttt{EX}(x,t) \coloneqq \texttt{TME}(t) \wedge \exists u(\texttt{TME}(u) \wedge \texttt{EQ}(x,u) \wedge \texttt{P}(t,u))$ (*Existence in time*)

t_e60 $\texttt{EX}(x,t) \wedge \texttt{P}(u,t) \wedge \texttt{TME}(u) \rightarrow \texttt{EX}(x,u)$ (*Dissectivity of* EX)
Proof. Directly from (d_e52). □

t_e61 $\texttt{EX}(x,t) \wedge \texttt{EX}(x,u) \wedge \texttt{TME}(t+u) \rightarrow \texttt{EX}(x,t+u)$
Proof. From the hypothesis, by (d_e52) and (t_e66), we have that $\texttt{TME}(t) \wedge \texttt{TME}(u) \wedge \texttt{TME}(t+u) \wedge \exists s(\texttt{TLC}(x,s) \wedge \texttt{P}(t,s) \wedge \texttt{P}(u,s))$. Then from ($d_e$3), $\texttt{TME}(t+u) \wedge \exists s(\texttt{TLC}(x,s) \wedge \texttt{P}(t+u,s))$. □

t_e62 $\texttt{EX}(x,t) \rightarrow \texttt{IN}(t,x)$
Proof. From $\texttt{EX}(x,t)$, by (d_e52), $\exists u(\texttt{EQ}(x,u) \wedge \texttt{P}(t,u))$. From $\texttt{P}(t,u)$, by (t_e25), $\texttt{IN}(t,u)$. From $\texttt{IN}(t,u) \wedge \texttt{EQ}(x,u)$, by ($d_e$45) and ($t_e$24), $\texttt{IN}(t,x)$. □

t_e63 $\texttt{TME}(t) \wedge \texttt{IN}(t,x) \rightarrow \texttt{EX}(x,t)$
Proof. Consider *u* such that $\texttt{EC}(u,x)$ (such *u* always exists because of (t_e32),(t_e36), and (a_e5)). By (d_e47) and (t_e38), $\texttt{TME}(u) \wedge \texttt{EQ}(u,x)$. From $\texttt{IN}(t,x) \wedge \texttt{EQ}(u,x)$, by ($d_e$45) and ($t_e$24), $\texttt{IN}(t,u)$. From $\texttt{TME}(u) \wedge \texttt{IN}(t,u)$, by ($t_e$26), $\texttt{P}(t,u)$. Then, there exists *u* s.t. $\texttt{TME}(u) \wedge \texttt{EQ}(u,x) \wedge \texttt{P}(t,u)$. The thesis follows directly from (d_e52). □

t_e64 $\texttt{EX}(z,t) \wedge \texttt{P}(z,y) \rightarrow \texttt{EX}(y,t)$
Proof. From the hypothesis, by (d_e52), $\texttt{TME}(t)$. From $\texttt{P}(z,y)$, by (t_e25), $\texttt{IN}(z,y)$. From $\texttt{EX}(z,t)$, by (t_e62), $\texttt{IN}(t,z)$. From $\texttt{IN}(z,y)$ and $\texttt{IN}(t,z)$, by (t_e24), $\texttt{IN}(t,y)$. From $\texttt{IN}(t,y)$ and $\texttt{TME}(t)$, by (t_e63), $\texttt{EX}(y,t)$. □

t_e65 $\nvdash \texttt{EX}(x,t) \rightarrow \texttt{O}(x,t)$
Proof. Consider the model of EMMO in Fig. 5: we have $\texttt{EQ}(1,123)$ and $\texttt{tP}(3,123)$, i.e., $\texttt{EX}(1,3)$, but $\neg\texttt{O}(1,3)$. □

t_e66 $\texttt{TME}(t) \wedge \texttt{TME}(u) \wedge \texttt{EQ}(x,t) \wedge \texttt{EQ}(x,u) \rightarrow t = u$
Proof. The thesis follows from (d_e47), (d_e46), (t_e36) and (t_e1) (or directly by noticing that EQ is an equivalence relation). □

As said, EMMO endorses a form of perdurantism mainy intended in terms of persistence across change (rather than time). However, the Time extension developed in Sect. 4.2.4 introduces a notion of time built from causation and change allowing then to study how the form of perdurantism endorsed by EMMO is linked to more standard versions of perdurantism (like the one of Sider) characterized in terms of time and *temporal slices*. As we will see, temporal slices play also a fundamental role in the

mapping from EMMO to DOLCE, especially (but not exclusively) for characterizing the distinction between endurants and perdurants.

If one considers the standard definition of temporal slice (i.e., the temporal slice of y at t is the maximal part of y that exists at t):

– $\mathtt{TS}(x,y,t) \coloneqq \mathtt{TME}(t) \wedge \mathtt{EQ}(x,t) \wedge \mathtt{P}(x,y) \wedge \forall z(\mathtt{P}(z,y) \wedge \mathtt{EX}(z,t) \rightarrow \mathtt{O}(z,x))$

then, given the construction of time in Sect. 4.2.4, entities including a persistent quantum do not have temporal slices at every time they exist. For instance, consider the model of EMMO in Fig. 7 and assume $y = 345$ and $t = 1$. We have that $\mathtt{TME}(1)$, $\mathtt{TME}(345) \wedge \mathtt{TME}(12345) \wedge \mathtt{EQ}(345, 12345)$, and $\mathtt{tP}(1, 12345)$, then $\mathtt{EX}(345, 1)$ but there is no part of 345 temporally equivalent to 1.

The situation does not improve by considering the following definitions:

– $\mathtt{TS}(x,y,t) \coloneqq \mathtt{TME}(t) \wedge x = \sigma a \langle \mathtt{qP}(a,y) \wedge \mathtt{EX}(a,t) \rangle$

– $\mathtt{TS}(x,y,t) \coloneqq \mathtt{TME}(t) \wedge x = \sigma a \langle \mathtt{qP}(a,y) \wedge \exists u(\mathtt{TME}(u) \wedge \mathtt{EQ}(x,u) \wedge \mathtt{O}(t,u)) \rangle$

– $\mathtt{TS}(x,y,t) \coloneqq \mathtt{TME}(t) \wedge x = \sigma a \langle \mathtt{qP}(a,y) \wedge \mathtt{IN}(a,t) \rangle$

it is enough to consider again the model in Fig. 7 and assume $y = 345$ and $t = 1$.

Given the nature of the constructed time (in particular the non-linearity), to guarantee that $\mathtt{EX}(y,t) \rightarrow \exists x(\mathtt{TS}(x,y,t))$—a property that is at the basis of the Sider's characterization of perdurantism and that is crucial, as we will see, for the ED/PD distinction—the notion of temporal slice must be thought in a different (and weaker) way.

The definition of temporal slice is given in two steps.

1. We introduce a notion of 'instantaneous' temporal slice, i.e., a temporal slice at an 'instantaneous' time, a time built starting from a quantum, see (d_e53). Instantaneous times are not necessarily atomic (see (t_e70) and (t_e72)) and they can contain quanta that are not equivalent to the time itself (see (t_e71)). Interestingly, every time can be obtained by summing up all their instantaneous parts (see (t_e69)). The temporal slice of y at the instantaneous time t collects all the parts of y that cannot be temporally ordered with respect to t, see (d_e54). In the example in Fig. 7 we have $\mathtt{TS}(345, 345, 1)$. As an intuitive explanation, the model in Fig. 7 can be read in terms of its compatibility with respect to different linear situations. For instance, according to the temporal interpretation of causation adopted in the construction of time, 1 and 3 begin together, 1 ends before the end of 5, and (in a linear time) 1 could temporally overlap only 3, both 3 and 4, or 3, 4, and 5. (d_e54) is maximally inclusive, it collects all the parts of 345 that (temporally) overlap 1 at least in one possible linear interpretation of the model.
2. We define the temporal slice of y at a time t as the sum of all the temporal slices of y at the instantaneous times part of t, see (d_e55).

In this framework (t_e77) holds, i.e., all the entities of EMMO are perdurants in the sense of Sider. (t_e80) shows that the temporal slice of y at t is a part of y, while (t_e81) guarantees the unicity of temporal slices. (t_e82) and (t_e84) prove that the temporal slice is monotonic with respect times and hosts. However, there are also some counterintuitive consequences. E.g., the temporal slice at t could have a temporal location bigger than t, see (t_e94), and an entity can have the same temporal slice at different (possibly disjoint) times, see (t_e95).

d$_e$53 $\mathtt{iTME}(t) \coloneqq \exists a(\mathtt{Q}(a) \wedge \mathtt{EC}(t,a))$ *(Instantaneous Time)*

d$_e$54 $\mathtt{iTS}(x,y,t) \coloneqq \mathtt{EX}(y,t) \wedge \mathtt{iTME}(t) \wedge x = \sigma a\langle \mathtt{qP}(a,y) \wedge \neg \mathtt{pC}(a,t) \wedge \neg \mathtt{pC}(t,a)\rangle$ *(Instantaneous Temporal Slice)*

d$_e$55 $\mathtt{TS}(x,y,t) \coloneqq \mathtt{EX}(y,t) \wedge x = \sigma a\langle \exists s(\mathtt{P}(s,t) \wedge \mathtt{iTS}(a,y,s))\rangle$ *(Temporal Slice)*

t$_e$67 $\mathtt{iTME}(t) \rightarrow \mathtt{TME}(t)$
Proof. Directly by (d$_e$53) and (d$_e$47). □

t$_e$68 $\mathtt{TME}(t) \rightarrow \exists s(\mathtt{iTME}(s) \wedge \mathtt{P}(s,t))$
Proof. By (a$_e$6), $\exists q(\mathtt{qP}(q,t))$, and then by (t$_e$32), (a$_e$5), and (t$_e$36), $\exists s(\mathtt{EC}(s,q))$ and then, by (d$_e$53), $\mathtt{iTME}(s)$. $\mathtt{P}(s,t)$ follows directly by (t$_e$36) and (t$_e$39). □

t$_e$69 $\mathtt{TME}(t) \wedge s = \sigma r\langle \mathtt{iTME}(r) \wedge \mathtt{P}(r,t)\rangle \rightarrow s = t$
Proof. First observe that, by (t$_e$68) and (a$_e$5), s exists. We prove that $\mathtt{P}(s,t) \wedge \mathtt{P}(t,s)$; the conclusion follows from (a$_e$2). $\mathtt{P}(s,t)$ follows directly from (d$_e$2). For $\mathtt{P}(t,s)$, consider an a s.t. $\mathtt{qP}(a,t)$ and assume that $\mathtt{EC}(w,a)$ (by (t$_e$32), (t$_e$36), and (a$_e$5), such w always exists). By (d$_e$53), (t$_e$32), (t$_e$36), and (t$_e$39), $\mathtt{iTME}(w) \wedge \mathtt{P}(w,t) \wedge \mathtt{qP}(a,w)$ then, by the construction of s and (a$_e$3), $\mathtt{qP}(a,s)$ and, by (t$_e$1), $\mathtt{P}(t,s)$. □

t$_e$70 $\nvdash \mathtt{iTME}(t) \rightarrow \mathtt{aTME}(t)$
Proof. Analogous to (t$_e$58). □

t$_e$71 $\nvdash \mathtt{iTME}(t) \wedge \mathtt{qP}(a,t) \rightarrow \mathtt{EC}(t,a)$
Proof. Consider the model of EMMO depicted in Fig. 5. We have $\mathtt{iTME}(123)$ (it is generated by 1), $\mathtt{qP}(2,123)$, and $\mathtt{EC}(2,2)$. □

t$_e$72 $\nvdash \mathtt{iTME}(t) \wedge \mathtt{iTME}(s) \wedge \mathtt{P}(t,s) \rightarrow t = s$
Proof. Consider the model of EMMO depicted in Fig. 5. We have $\mathtt{iTME}(123)$ (it is generated by 1), $\mathtt{iTME}(2)$ (it is generated by 2), $\mathtt{P}(2,123)$, and $2 \neq 123$. □

t$_e$73 $\mathtt{iTME}(t) \wedge \mathtt{qP}(a,t) \wedge \mathtt{EC}(s,a) \rightarrow \mathtt{P}(s,t)$
Proof. Directly from (t$_e$67), (t$_e$36), and (t$_e$39), □

t$_e$74 $\mathtt{TME}(t) \wedge \mathtt{TME}(u) \wedge \mathtt{TME}(t+u) \wedge \mathtt{iTME}(i) \rightarrow (\mathtt{P}(i,t+u) \leftrightarrow (\mathtt{P}(i,t) \vee \mathtt{P}(i,u)))$.
Proof. ($\rightarrow$) By (d$_e$53), $\exists a(\mathtt{Q}(a) \wedge \mathtt{EC}(i,a))$ and, by (t$_e$37), $\mathtt{qP}(a,i)$. From $\mathtt{qP}(a,i) \wedge \mathtt{P}(i,t+u)$, by (a$_e$3) and (d$_e$3), $\mathtt{qP}(a,t) \vee \mathtt{qP}(a,u)$. Assume $\mathtt{qP}(a,t)$. From $\mathtt{TME}(t) \wedge \mathtt{qP}(a,t) \wedge \mathtt{EC}(i,a)$, by (t$_e$36) and (t$_e$39), $\mathtt{P}(i,t)$. Analogously for $\mathtt{qP}(a,u)$.
($\leftarrow$) Directly from $\mathtt{P}(i,t) \vee \mathtt{P}(i,u)$ by (d$_e$3). □

t$_e$75 $\mathtt{iTS}(x,y,t) \wedge \mathtt{iTS}(z,y,u) \wedge \mathtt{P}(t,u) \rightarrow \mathtt{P}(x,z)$
Proof. We prove that $\forall c(\mathtt{qP}(c,x) \rightarrow \mathtt{qP}(c,z))$, the thesis follows directly from (t$_e$1). From the hypotheses, by (d$_e$54), $x = \sigma a\langle \mathtt{qP}(a,y) \wedge \neg \mathtt{pC}(a,t) \wedge \neg \mathtt{pC}(t,a)\rangle \wedge z = \sigma b\langle \mathtt{qP}(b,y) \wedge \neg \mathtt{pC}(b,u) \wedge \neg \mathtt{pC}(u,b)\rangle \wedge \mathtt{P}(t,u)$. Then, from $\mathtt{qP}(c,x)$ and by (d$_e$2), we have $\mathtt{qP}(c,y) \wedge \neg \mathtt{pC}(c,t) \wedge \neg \mathtt{pC}(t,c)$. From $\neg \mathtt{pC}(c,t) \wedge \mathtt{P}(t,u)$, by (t$_e$15), $\neg \mathtt{pC}(c,u)$. From $\neg \mathtt{pC}(t,c) \wedge \mathtt{P}(t,u)$, by (t$_e$15), $\neg \mathtt{pC}(u,t)$. Then we have that $\mathtt{qP}(c,y) \wedge \neg \mathtt{pC}(c,u) \wedge \neg \mathtt{pC}(t,u)$ and, by the definition of z, $\mathtt{qP}(c,z)$. □

t$_e$76 $\mathtt{EX}(y,t) \wedge \mathtt{iTME}(t) \rightarrow \exists x(\mathtt{iTS}(x,y,t))$
Proof. We prove that $\exists a(\mathtt{qP}(a,y) \wedge \neg \mathtt{pC}(a,t) \wedge \neg \mathtt{pC}(t,a))$, the thesis follows from (a$_e$5). By contradiction assume $\neg \exists a(\mathtt{qP}(a,y) \wedge \neg \mathtt{pC}(a,t) \wedge \neg \mathtt{pC}(t,a))$. Then either (1) $\forall a(\mathtt{qP}(a,y) \rightarrow \mathtt{pC}(a,t))$; or (2) $\forall a(\mathtt{qP}(a,y) \rightarrow \mathtt{pC}(t,a))$; or (3) $\forall a(\mathtt{qP}(a,y) \rightarrow$

$(\mathtt{pC}(a,t) \vee \mathtt{pC}(t,a))$ and it is not the case that (1) or (2) then. Note that in case (3), by (a_e5) and (a_e6), there exist s_1 and s_2 such that $s_1 = \sigma a\langle \mathtt{qP}(a,y) \wedge \mathtt{pC}(a,t)\rangle$ and $s_2 = \sigma a\langle \mathtt{qP}(a,y) \wedge \mathtt{pC}(t,a)\rangle$ and, by (t_e1), $\mathtt{SUM}(y,s_1,s_2)$. Furthermore we have that $\neg\mathtt{MdC}(s_1,s_2)$ because $\forall ab(\mathtt{qP}(a,s_1) \wedge \mathtt{qP}(b,s_2))$, by the definitions of s_1 and s_2, $\mathtt{pC}(a,t) \wedge \mathtt{pC}(t,b)$ and then there exists at least a c such that $\mathtt{qP}(c,t) \wedge \mathtt{C}(a,c) \wedge \mathtt{C}(c,b)$, i.e., by ($d_e$9), $\neg\mathtt{dC}(a,b)$ and then, by (d_e13), $\neg\mathtt{MdC}(s_1,s_2)$.
We show that $\neg\mathtt{IN}(t,y)$ follows. To see this, consider (d_e44) and assume, by contradiction, that $\mathtt{IN}(t,y)$. In this case, $\forall u(\mathtt{qP}(u,t) \rightarrow \exists v(\mathtt{P}(v,y) \wedge \mathtt{iIN}(u,v)))$, i.e., by ($d_e$43), $\forall z(\mathtt{pC}(z,v) \rightarrow \mathtt{pC}(z,u)) \wedge \forall z(\mathtt{pC}(v,z) \rightarrow \mathtt{pC}(u,z))$. Consider $z = u$, by (t_e12), in case (1) we have that $\mathtt{pC}(v,u)$ but $\neg\mathtt{pC}(u,u)$; in case (2) we have that $\mathtt{pC}(u,v)$ but $\neg\mathtt{pC}(u,u)$. In case (3), by (d_e14) and the facts that $\neg\mathtt{MdC}(s_1,s_2)$ and that $\mathtt{SUM}(y,s_1,s_2)$, the only ITEMs v part of y are wholly included in s_1 or wholly included in s_2 and then we can reapply the argument used for the cases (1) and (2). We then obtain a contradiction and we have that $\neg\mathtt{IN}(t,y)$. But from $\mathtt{EX}(y,t)$, by (t_e62), $\mathtt{IN}(t,y)$. Contradiction.

t_e77 $\mathtt{EX}(y,t) \rightarrow \exists x(\mathtt{TS}(x,y,t))$
Proof. From $\mathtt{EX}(y,t)$, by (t_e68), $\exists s(\mathtt{iTME}(s) \wedge \mathtt{P}(s,t))$ then, by (t_e67) and (t_e60), $\mathtt{EX}(y,s) \wedge \mathtt{iTME}(s)$ and then, by (t_e76), $\exists ws(\mathtt{iTS}(w,y,s) \wedge \mathtt{P}(s,t))$. The thesis follows from (d_e55) and (a_e5). □

t_e78 $\mathtt{iTME}(t) \rightarrow (\mathtt{iTS}(x,y,t) \leftrightarrow \mathtt{TS}(x,y,t))$
Proof. ($\rightarrow$) From the hypothesis and (a_e1), $\mathtt{P}(t,t) \wedge \mathtt{iTS}(x,y,t)$ then, by (d_e55), x is part of the generalized temporal part of y at t. Consider now $\mathtt{P}(s,t) \wedge \mathtt{iTS}(a,y,s)$ then, by (t_e75), $\mathtt{P}(a,x)$, i.e., according to (d_e55), nothing can be added to x.
($\leftarrow$) We prove that $\mathtt{iTS}(x,y,t) \wedge \mathtt{TS}(x',y,t) \rightarrow x = x'$, i.e., by ($t_e$1) and ($a_e$2), that $\forall a(\mathtt{qP}(a,x) \leftrightarrow \mathtt{qP}(a,x'))$ where $x = \sigma a\langle \mathtt{qP}(a,y) \wedge \neg\mathtt{pC}(a,t) \wedge \neg\mathtt{pC}(t,a)\rangle$ and $x' = \sigma b\langle \exists s(\mathtt{P}(s,t) \wedge \mathtt{iTS}(b,y,s))\rangle$. By considering $\mathtt{P}(t,t) \wedge \mathtt{iTS}(x,y,t)$ then, by the definition of x', we obtain that x is part of x' and then, by (t_e1), $\mathtt{qP}(a,x) \rightarrow \mathtt{qP}(a,x')$. Assume $\mathtt{qP}(a,x')$. By the definition of x', $\exists sw(\mathtt{P}(s,t) \wedge \mathtt{iTS}(w,y,s) \wedge \mathtt{qP}(a,w))$ where $w = \sigma c\langle \mathtt{qP}(c,y) \wedge \neg\mathtt{pC}(c,s) \wedge \neg\mathtt{pC}(s,c)\rangle$. Because we have $\mathtt{qP}(a,w)$, a is among the c considered in the definition of w. From $\neg\mathtt{pC}(a,s) \wedge \mathtt{P}(s,t)$, by ($t_e$15), $\neg\mathtt{pC}(a,t)$. From $\neg\mathtt{pC}(s,a) \wedge \mathtt{P}(s,t)$, by ($t_e$15), $\neg\mathtt{pC}(t,a)$. Then, $\mathtt{qP}(a,w) \wedge \neg\mathtt{pC}(a,t) \wedge \neg\mathtt{pC}(t,a)$ then, by the definition of x, $\mathtt{qP}(a,x)$. □

t_e79 $\mathtt{TS}(x,y,t) \rightarrow \mathtt{IN}(t,y)$
Proof. Directly from (d_e55) and (t_e62). □

t_e80 $\mathtt{TS}(x,y,t) \rightarrow \mathtt{P}(x,y)$
Proof. Directly from (d_e54) and (d_e55). □

t_e81 $\mathtt{TS}(x,y,t) \wedge \mathtt{TS}(z,y,t) \rightarrow x = z$
Proof. Directly from (d_e54), (d_e55) and (t_e1). □

t_e82 $\mathtt{TS}(x,y,t) \wedge \mathtt{TS}(z,y,u) \wedge \mathtt{P}(t,u) \rightarrow \mathtt{P}(x,z)$
Proof. Directly from (d_e55) by observing that $\exists s(\mathtt{P}(s,t) \wedge \mathtt{iTS}(a,y,s)) \wedge \mathtt{P}(t,u)$ implies, by (a_e3), $\exists s(\mathtt{P}(s,u) \wedge \mathtt{iTS}(a,y,s))$. □

t_e83 $\mathtt{iTS}(x,y,t) \wedge \mathtt{iTS}(z,w,t) \wedge \mathtt{P}(y,w) \rightarrow \mathtt{P}(x,z)$

Proof. We prove that $\forall c(\mathtt{qP}(c,x) \rightarrow \mathtt{qP}(c,z))$, the thesis follows directly from (t_e1). From the hypotheses, by (d_e54), $x = \sigma a\langle \mathtt{qP}(a,y) \wedge \neg \mathtt{pC}(a,t) \wedge \neg \mathtt{pC}(t,a)\rangle \wedge z = \sigma b\langle \mathtt{qP}(b,w) \wedge \neg \mathtt{pC}(b,t) \wedge \neg \mathtt{pC}(t,b)\rangle$. From $\mathtt{qP}(c,x)$, by the definition of x, $\mathtt{qP}(c,y) \wedge \neg \mathtt{pC}(a,t) \wedge \neg \mathtt{pC}(t,a)$. From $\mathtt{qP}(c,y) \wedge \mathtt{P}(y,w)$, by ($a_e$3), $\mathtt{qP}(c,w)$, and then $\mathtt{qP}(c,w) \wedge \neg \mathtt{pC}(c,t) \wedge \neg \mathtt{pC}(t,c)$ then, by the definition of z, $\mathtt{qP}(c,z)$. □

t_e84 $\mathtt{TS}(x,y,t) \wedge \mathtt{TS}(z,w,t) \wedge \mathtt{P}(y,w) \rightarrow \mathtt{P}(x,z)$
Proof. From $\mathtt{TS}(x,y,t)$, by (d_e55), x is the sum of all the entities a s.t. $\exists s(\mathtt{P}(s,t) \wedge \mathtt{iTS}(a,y,s))$. From $\mathtt{TS}(z,w,t) \wedge \mathtt{P}(s,t)$, by ($d_e$55) and ($t_e$60), $\mathtt{EX}(w,s)$ and then, by (t_e76), $\exists b(\mathtt{iTS}(b,w,s) \wedge \mathtt{P}(b,t))$. From $\mathtt{iTS}(a,y,s) \wedge \mathtt{iTS}(b,w,s) \wedge \mathtt{P}(y,w)$, by ($t_e$83), $\mathtt{P}(a,b)$. Then, by the definition of z, all the (quantum) parts of x are also (quantum) part of z and, by (t_e1), $\mathtt{P}(x,z)$. □

t_e85 $\mathtt{iTS}(x,y,t) \wedge \mathtt{iTS}(z,w,t) \wedge \mathtt{P}(x,w) \rightarrow \mathtt{P}(x,z)$
Proof. We prove that $\forall c(\mathtt{qP}(c,x) \rightarrow \mathtt{qP}(c,z))$, the thesis follows directly from (t_e1). From the hypotheses, by (d_e54), $x = \sigma a\langle \mathtt{qP}(a,y) \wedge \neg \mathtt{pC}(a,t) \wedge \neg \mathtt{pC}(t,a)\rangle \wedge z = \sigma b\langle \mathtt{qP}(b,w) \wedge \neg \mathtt{pC}(b,t) \wedge \neg \mathtt{pC}(t,b)\rangle$. From $\mathtt{qP}(c,x)$, by the definition of x, $\neg \mathtt{pC}(a,t) \wedge \neg \mathtt{pC}(t,a)$. From $\mathtt{qP}(c,x) \wedge \mathtt{P}(x,w)$, by ($a_e$3), $\mathtt{qP}(c,w)$, and then $\mathtt{qP}(c,w) \wedge \neg \mathtt{pC}(c,t) \wedge \neg \mathtt{pC}(t,c)$ then, by the definition of z, $\mathtt{qP}(c,z)$. □

t_e86 $\mathtt{TS}(x,y,t) \wedge \mathtt{TS}(z,w,t) \wedge \mathtt{P}(x,w) \rightarrow \mathtt{P}(x,z)$
Proof. From $\mathtt{TS}(x,y,t)$, by (d_e55), x is the sum of all the entities a s.t. $\exists s(\mathtt{P}(s,t) \wedge \mathtt{iTS}(a,y,s))$. From $\mathtt{TS}(z,w,t) \wedge \mathtt{P}(s,t)$, by ($d_e$55) and ($t_e$60), $\mathtt{EX}(w,s)$ and then, by (t_e76), $\exists b(\mathtt{iTS}(b,w,s) \wedge \mathtt{P}(b,t))$. From the definition of sum, $\mathtt{P}(a,x)$ and then, from $\mathtt{P}(a,x) \wedge \mathtt{P}(x,w)$, by ($a_e$3), $\mathtt{P}(a,w)$. From $\mathtt{iTS}(a,y,s) \wedge \mathtt{iTS}(b,w,s) \wedge \mathtt{P}(a,w)$, by ($t_e$85), $\mathtt{P}(a,b)$. Then, by the definition of z, all the (quantum) parts of x are also (quantum) part of z and, by (t_e1), $\mathtt{P}(x,z)$. □

t_e87 $\mathtt{TS}(x,y,t) \wedge \mathtt{TS}(z,y,u) \wedge \mathtt{TME}(t{+}u) \rightarrow \mathtt{TS}(x{+}z,y,t{+}u)$
Proof. First notice that $\mathtt{EX}(y,t)$ and $\mathtt{EX}(y,u)$ follow directly from (d_e55); then, by (t_e61), $\mathtt{EX}(y,t{+}u)$, and by (t_e77), $\exists w\mathtt{TS}(w,y,t{+}u)$. We need to prove that $w = x{+}z$. By (d_e55), $w = \sigma a\langle \exists s(\mathtt{P}(s,t{+}u) \wedge \mathtt{iTS}(a,y,s))\rangle$. By ($d_e$54) and ($t_e$74), $\mathtt{P}(s,t{+}u) \wedge \mathtt{iTS}(a,y,s)$ if and only if $(\mathtt{P}(s,t) \wedge \mathtt{iTS}(a,y,s)) \vee (\mathtt{P}(s,u) \wedge \mathtt{iTS}(a,y,s))$. The thesis follows by considering the definitions of x and z. □

t_e88 $\nvdash \mathtt{IN}(t,y) \wedge \mathtt{TME}(t) \rightarrow \exists x(\mathtt{P}(x,y) \wedge \mathtt{EQ}(t,x))$
Proof. Consider Fig. 7: $t = 4$, $y = 1{+}2$. $\neg(\mathtt{EQ}(1{+}2,4) \vee \mathtt{EQ}(1,4) \vee \mathtt{EQ}(2,4))$. □

t_e89 $\nvdash \mathtt{IN}(t,y) \rightarrow \exists x(\mathtt{P}(x,y) \wedge \mathtt{EQ}(t,x))$
Proof. Directly from (t_e88). □

t_e90 $\mathtt{IN}(t,y) \wedge \mathtt{iTS}(x,y,t) \rightarrow \mathtt{IN}(t,x)$
Proof. By contradiction let us assume that $\mathtt{IN}(t,y) \wedge \mathtt{iTS}(x,y,t) \wedge \neg \mathtt{IN}(t,x)$. By the hypothesis, ($d_e$54) and ($d_e$53), then $\mathtt{IN}(t,y) \wedge \exists q(\mathtt{Q}(q) \wedge \mathtt{EC}(t,q) \wedge x = \sigma a\langle \mathtt{qP}(a,y) \wedge \neg \mathtt{pC}(a,t) \wedge \neg \mathtt{pC}(t,a)\rangle) \wedge \neg \mathtt{IN}(t,x)$. From $\mathtt{EC}(t,q)$, by (t_e38), $\mathtt{EQ}(t,q)$, i.e., by (d_e45), $\mathtt{IN}(t,q) \wedge \mathtt{IN}(q,t)$. From $\mathtt{IN}(t,y) \wedge \neg \mathtt{IN}(t,x) \wedge \mathtt{IN}(t,q) \wedge \mathtt{IN}(q,t)$, by ($t_e$24), then $\mathtt{IN}(q,y) \wedge \neg \mathtt{IN}(q,x)$.
From $\mathtt{EC}(t,q)$, by (t_e32) and (t_e36), $\mathtt{qP}(q,t)$ and thus, by (d_e42), (a_e1) and (a_e8), it follows that $\neg \mathtt{pC}(q,t) \wedge \neg \mathtt{pC}(t,q)$. From the construction of x, $\neg \mathtt{pC}(q,t) \wedge \neg \mathtt{pC}(t,q)$, and $\mathtt{qP}(q,y)$, by (d_e2), it follows that $\mathtt{P}(q,x)$. From $\mathtt{P}(q,x)$, by (t_e25), then $\mathtt{IN}(q,x)$. Contradiction. □

t$_e$91 $\mathtt{iTS}(x,y,t) \rightarrow \mathtt{EX}(x,t)$
Proof. From the hypothesis, by (d$_e$54) and (t$_e$67), $\mathtt{TME}(t)$. The thesis follows directly from (t$_e$90) and (t$_e$63). □

t$_e$92 $\mathtt{IN}(t,y) \wedge \mathtt{TS}(x,y,t) \rightarrow \mathtt{IN}(t,x)$
Proof. From the hypothesis, by (d$_e$55), $\mathtt{IN}(t,y) \wedge \mathtt{EX}(y,t) \wedge x = \sigma a \langle \exists s(\mathtt{P}(s,t) \wedge \mathtt{iTS}(a,y,s)) \rangle$. Consider an s s.t. $\mathtt{P}(s,t) \wedge \mathtt{iTS}(a,y,s)$. From $\mathtt{IN}(t,y) \wedge \mathtt{P}(s,t)$, by (t$_e$25) and (t$_e$24), $\mathtt{IN}(s,y)$. From $\mathtt{IN}(s,y) \wedge \mathtt{iTS}(a,y,s)$, by (t$_e$90) and (d$_e$54), $\mathtt{IN}(s,a) \wedge \mathtt{iTME}(s)$. Given the definition of x, $\mathtt{P}(a,x)$ and then, by (t$_e$25), $\mathtt{IN}(a,x)$. From $\mathtt{IN}(s,a) \wedge \mathtt{IN}(a,x)$, by (t$_e$24), $\mathtt{IN}(s,x)$. From the hypotheses, by (t$_e$69), t is the mereological fusion of such ss, thus, by (t$_e$29), $\mathtt{IN}(t,x)$. □

t$_e$93 $\mathtt{TS}(x,y,t) \rightarrow \mathtt{EX}(x,t)$
Proof. From the hypothesis, by (d$_e$52) and (d$_e$55), $\mathtt{TME}(t)$. The thesis follows directly from (t$_e$92) and (t$_e$63). □

t$_e$94 $\nvdash \mathtt{TS}(x,y,t) \rightarrow \mathtt{IN}(x,t)$
Proof. In Fig. 7 we have $\mathtt{TS}(345,345,1) \wedge \neg\mathtt{IN}(345,1)$. To see this, first observe that $\neg\mathtt{iIN}(5,1)$ because $\mathtt{pC}(1,2)$ but $\neg\mathtt{pC}(5,2)$ (see (d$_e$43)). But then, by applying (d$_e$44) to the case $\mathtt{IN}(345,1)$, from $\mathtt{P}(5,345)$ one needs to find a part v of 1 such that $\mathtt{iIN}(5,v)$. But the only part of 1 is 1 itself and we have $\neg\mathtt{iIN}(5,1)$. □

t$_e$95 $\nvdash \mathtt{TS}(x,y,t) \wedge \mathtt{TS}(x,y,u) \rightarrow t = u$
Proof. In Fig. 7 we have $\mathtt{TS}(345,345,1)$, $\mathtt{TS}(345,345,2)$, $\mathtt{TS}(345,345,12345)$. □

t$_e$96 $\vdash \mathtt{iTS}(z,x,t) \wedge \mathtt{iTS}(z,y,t) \wedge \mathtt{iTS}(w,x,u) \wedge \mathtt{iTS}(v,y,u) \wedge \mathtt{P}(u,t) \rightarrow w = v$
Proof. From the hypothesis, by (t$_e$75), $\mathtt{P}(w,z) \wedge \mathtt{P}(v,z)$. From the hypothesis, by (t$_e$80), (d$_e$55) and (d$_e$53), then $\mathtt{P}(z,x) \wedge \mathtt{P}(z,y)$. From $\mathtt{iTS}(z,x,t) \wedge \mathtt{P}(u,t)$, by (t$_e$91) and (t$_e$60), $\mathtt{EX}(z,u)$. From $\mathtt{iTS}(w,x,u) \wedge \mathtt{P}(z,x) \wedge \mathtt{P}(w,z)$, by (d$_e$54) and (t$_e$1), $\mathtt{EX}(x,u) \wedge \mathtt{iTME}(u) \wedge w = \sigma a \langle \mathtt{qP}(a,x) \wedge \neg\mathtt{pC}(a,u) \wedge \neg\mathtt{pC}(u,a) \rangle \wedge \forall q(\mathtt{qP}(q,z) \rightarrow \mathtt{qP}(q,x)) \wedge \forall k(\mathtt{qP}(k,w) \rightarrow \mathtt{qP}(k,z))$. Let $f = \sigma b \langle \mathtt{qP}(b,z) \wedge \neg\mathtt{pC}(b,u) \wedge \neg\mathtt{pC}(u,a) \rangle$: trivially, from $\forall q(\mathtt{qP}(q,z) \rightarrow \mathtt{qP}(q,x))$, the bs are a subset of the as, and by $\forall k(\mathtt{qP}(k,w) \rightarrow \mathtt{qP}(k,z))$ and the definition of w there is no member of the as which is not one of the bs. Thus, $f = w$, and, given $\mathtt{EX}(z,u)$, by (d$_e$54), then $\mathtt{iTS}(w,z,u)$. Analogously, from $\mathtt{iTS}(v,y,u) \wedge \mathtt{P}(v,z) \wedge \mathtt{P}(z,y)$, it follows that $\mathtt{iTS}(v,z,u)$. The conclusion follows directly from $\mathtt{iTS}(v,z,u)$ and $\mathtt{iTS}(w,z,u)$ by (t$_e$81), (d$_e$55) and (d$_e$53). □

t$_e$97 $\vdash \mathtt{TS}(z,x,t) \wedge \mathtt{TS}(z,y,t) \wedge \mathtt{TS}(w,x,u) \wedge \mathtt{TS}(v,y,u) \wedge \mathtt{P}(u,t) \rightarrow w = v$
Proof. From the hypothesis, by (t$_e$82), $\mathtt{P}(w,z) \wedge \mathtt{P}(v,z)$. From the hypothesis, by (t$_e$80), then $\mathtt{P}(z,x) \wedge \mathtt{P}(z,y)$. The conclusion follows from $\mathtt{TS}(w,x,u) \wedge \mathtt{TS}(v,y,u) \wedge \mathtt{P}(w,z) \wedge \mathtt{P}(v,z) \wedge \mathtt{P}(z,x) \wedge \mathtt{P}(z,y)$ by (d$_e$55) and (t$_e$96), and (t$_e$1). □

Temporal dissectivity and non-dissectivity Consider a type newly introduced in an extension of EMMO, i.e., a $\phi \in \mathbf{t}(\Delta)$. (d$_e$56) and (d$_e$57) define, respectively, the notions of temporal dissectivity and temporal non-dissectivity of an entity x with respect to such type ϕ. $\mathtt{tDSC}\langle\phi\rangle(x)$ holds when x is temporally homogeneous with respect to ϕ,

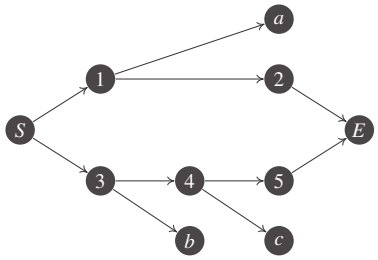

Figure 7: Problematic Temporal slices.

i.e., all the temporal slices of x (including x itself) satisfy ϕ. Vice versa $\mathtt{tnDSC}\langle\phi\rangle(x)$ holds when x satisfies ϕ but some of its temporal slices do not satisfy ϕ.

The adopted notation underlines the fact that ϕ is a sort of 'parameter'. However, technically, one has to introduce a different definition for each $\phi \in \mathbf{t}(\Delta)$ but all these definitions have the same structure provided in (d_e56) and (d_e57).

d_e56 $\mathtt{tDSC}\langle\phi\rangle(x) \coloneqq \phi(x) \wedge \forall yt(\mathtt{TS}(y,x,t) \rightarrow \phi(y))$ (*Temporal dissectiviy*)

d_e57 $\mathtt{tnDSC}\langle\phi\rangle(x) \coloneqq \phi(x) \wedge \exists yt(\mathtt{TS}(y,x,t) \rightarrow \neg\phi(y))$ (*Temporal non-dissectivity*)

We will see that temporal dissectiviy and non-dissectivity play a crucial role in the definition of endurants and perdurants starting from EMMO.

Temporary coincidence and existential dependencies Temporary coincidence is another important notion for the mapping. Two entities coincide at time t when they have the same temporal slice at t, i.e., at t, they are indistinguishable (see (d_e58)). Temporary coincidence is dissective and additive with respect to the temporal argument, see (t_e98) and (t_e99).

(d_e59) is a new 'parametric' definition, based on CNC. $\mathtt{CDEP}\langle\phi,\psi\rangle$ holds when every time there exists at t and instance x of ϕ then there exists also an instance of ψ that temporally coincides at t with x. This notion is used to define the DOLCE constitution starting from EMMO. $\mathtt{EDEP}\langle\phi,\psi\rangle$, see ($d_e$60), generalizes $\mathtt{CDEP}\langle\phi,\psi\rangle$ by discarding the coincidence condition and it is used to define the DOLCE existential dependence starting from EMMO.

d_e58 $\mathtt{CNC}(x,y,t) \coloneqq x \neq y \wedge \exists z(\mathtt{TS}(z,x,t) \wedge \mathtt{TS}(z,y,t))$ (*Temporary coincidence*)

d_e59 $\mathtt{CDEP}\langle\phi,\psi\rangle \coloneqq \exists x(\phi(x)) \wedge \forall xt(\phi(x) \wedge \mathtt{EX}(x,t) \rightarrow \exists y(\psi(y) \wedge \mathtt{CNC}(x,y,t)))$

d_e60 $\mathtt{EDEP}\langle\phi,\psi\rangle \coloneqq \exists x(\phi(x)) \wedge \forall xt(\phi(x) \wedge \mathtt{EX}(x,t) \rightarrow \exists y(\psi(y) \wedge \mathtt{EX}(y,t)))$

t_e98 $\vdash \mathtt{CNC}(x,y,t) \wedge \mathtt{P}(u,t) \wedge \mathtt{TME}(u) \rightarrow \mathtt{CNC}(x,y,u)$
Proof. From the hypotheses, by (d_e58), $x \neq y \wedge \exists z(\mathtt{TS}(z,x,t) \wedge \mathtt{TS}(z,y,t)) \wedge \mathtt{P}(u,t) \wedge \mathtt{TME}(u)$ then, by (t_e93), (t_e60) and (t_e77), $\exists wv(\mathtt{TS}(w,x,u) \wedge \mathtt{TS}(v,y,u))$ and, by (t_e97), $w = v$, then $x \neq y \wedge \exists w(\mathtt{TS}(w,x,u) \wedge \mathtt{TS}(w,y,u))$. □

t_e99 $\vdash \mathtt{CNC}(x,y,t) \wedge \mathtt{CNC}(x,y,u) \wedge \mathtt{TME}(t+u) \rightarrow \mathtt{CNC}(x,y,t+u)$

Proof. From the hypotheses, by (d_e58), $x \neq y \wedge \exists z(\mathtt{TS}(z,x,t) \wedge \mathtt{TS}(z,y,t)) \wedge \exists v(\mathtt{TS}(v,x,u) \wedge \mathtt{TS}(v,y,u)) \wedge \mathtt{TME}(t{+}u)$ and then, by (t_e87), $\mathtt{TS}(z{+}v,x,t{+}u) \wedge \mathtt{TS}(z{+}v,y,t{+}u) \wedge x \neq y$, i.e., by ($d_e$58), $\mathtt{CNC}(x,y,t{+}u)$. □

5 Preliminary considerations and general strategy for the alignment of TLOs

In the previous sections we have introduced the FOL versions of BFO, DOLCE, and EMMO. In this part of the report we introduce the assumptions, the general strategy and related considerations that we follow to develop the alignment and verify the correctness.

We start from three assumptions:

(M1) The informal correspondences between the presentations of the notions in the two ontologies resulting from the analysis of their documentation and the examples therein should be used to suggest class and relation mappings. For instance, BFO *occurrents* (*continuants*) are described and used in a way that is similar to DOLCE *perdurants* (*endurants*); at first sight, the BFO relation *continuantPartOf* is applied coherently to the use of the DOLCE relation *temporary parthood*, etc.

(M2) The ideal goal of the mapping is to have all the domain of quantification of the source ontology included in the domain of quantification of the target ontology. In other terms, the purpose is to maximize the coverage of the entities in one ontology (called *source* ontology, namely, DOLCE in the case analyzed in Sect. 6) which are modeled in the other (called *target* ontology, namely, BFO in Sect. 6). Given this scenario, the study of which axioms of the target ontology hold or not after optimizing the mappings from point (M1) can highlight the differences between the ontological commitments of the two ontologies, at least as formalized in FOL, and what are the entities of the source ontology that are problematic to model in, or even incompatible with, the target ontology.

(M3) Only mappings that can be formalized in FOL are to be considered. This means that we explore the connection between the entities of the source ontology that can be mapped to into the domain of the target ontology. Meta-modeling techniques that require the application of abstraction processes or set-theoretical (second-order) constructions are not considered. These techniques usually require to enrich the domain of the source ontology with additional entities. This change raises concerns about the actual correspondence between the ontological commitments of the source ontology and that of the theory with the enriched domain.

Assumptions (M2) and (M3) are clearly interrelated and must be seen in the perspective of an interactive development of mappings across ontologies where the analytical step (M1) is a prerequisite. In Sect. 6.3, Sect. 7.3, and Sect. 8.3 we list some alternative mappings that modify, by restricting or expanding those based on (M1), the way some notions of the target ontology are seen in terms of the ones of the source ontology. These alternative mappings are proposed to solve some misalignments raised by the work in Sect. 6, Sect. 7, and Sect. 8. We will also mention alternative (definitely more complex) constructions which extend the domain of the source ontology to match the domain of the target ontology (possibly allowing to define primitives of the target ontology via these new entities). Given the few primitives of EMMO (parthood and causation), the mapping from DOLCE to EMMO results quite straightforward.

In the context of the OCES framework, the alignment strategy to be exploited is

exemplified by the mapping from BFO to DOLCE, the one from DOLCE to BFO, the one from EMMO to DOLCE, and the one from DOLCE to EMMO carried out in Sect. 6, Sect. 7, Sect. 8, and Sect. 9. Given the time constraints and the complexity of the mapping from EMMO to DOLCE we did not manage to introduce direct BFO-EMMO mappings. That said, DOLCE can act as an intermediary, linking indirectly (and mediately) BFO and EMMO.

In order to clarify some technical background on assumption (M3), we discuss two ways to extend a theory via definitions. The goal is to have formulas in the source ontology which match (or best approximate) the concepts in the target ontology. As an example, we take DOLCE as the source ontology and BFO as the target ontology. Analogous considerations hold for the other mappings. The theory $\mathfrak{D}$ introduced in Sect. 3 is extended with a set of syntactic definitions—the mappings in Sect. 6.1—which inject BFO notions into DOLCE, i.e., these definitions capture the way BFO notions can be 'understood' in the DOLCE perspective. Syntactic definitions are labelled with the symbol $:=$ and are treated as 'parametric macros': we syntactically substitute the *definendum* with the *definiens* by suitably matching the parameters. For instance, suppose we aim to define BFO parthood on occurrents via formula $\mathtt{oP}(x,y) := \phi(x,y)$, where ϕ is an expression in the language of DOLCE containing x and y. In this case x and y are treated as parameters. The formula $\forall zw(\mathtt{oP}(z,w) \wedge \mathtt{oP}(w,z) \rightarrow z = w)$ in the language of BFO becomes formula $\forall zw(\phi(z,w) \wedge \phi(w,z) \rightarrow z = w)$ in the language of DOLCE. Similarly, BFO-formula $\mathtt{oP}(\mathtt{proc\#1}, \mathtt{proc\#2})$ becomes DOLCE-formula $\phi(\mathtt{proc\#1}, \mathtt{proc\#2})$.

One could decide to include these as new formulas in the source ontology obtaining a theory which is an 'extension by definitions'. This is obtained by using biconditional conditions ('if and only if') rather than syntactic definitions as described above. E.g., in the previous example one would add the entire formula $\forall xy(\mathtt{oP}(x,y) \leftrightarrow \phi(x,y))$ directly into the source theory. The two approaches are quite similar. The advantage of using syntactic definitions instead of extensions by definitions is that the vocabulary of the theory in which we introduce the syntactic definition does not change. Using the 'if and only if' clause one automatically adds the defined predicates to the ontology language (e.g., we would add the BFO-predicate oP into the DOLCE vocabulary when using the biconditional in the previous example). In Sect. 5.1 we will see that this latter approach may add new ontological commitments with important consequences in particular on the view of universals.

Once DOLCE has been extended with these definitions (and possibly further syntactic definitions already present BFO)—we dub $\mathfrak{D}_\mathtt{b}$ such extension—it is possible to check how much of the theory $\mathfrak{B}$ introduced in Sect. 2 is preserved. This is done in Sect. 6.2 where each BFO axiom expressible[5] in $\mathfrak{D}_\mathtt{b}$ is proved or disproved in $\mathfrak{D}_\mathtt{b}$. The analysis in Sect. 6.3 of the results obtained in Sect. 6.2 allows, modulo the introduced mapping, to highlight and understand similarities and divergences between the two top-levels. The mapping in the other direction, i.e., the extension $\mathfrak{B}_\mathtt{d}$ with the definitions of DOLCE notions in terms of BFO-primitives, is formalized in Sect. 7. The combined analysis of the mappings DOLCE-into-BFO and BFO-into-DOLCE points out genuine ontological differences as well as problematic aspects of the adopted mapping technique. This analysis opens the possibility to refine the mappings in an iterative

[5] We will see in Sect. 6.1 that this mapping is only partial.

process to solve, as far as possible, false cases of agreement and disagreement. Similar analyses are done in for EMMO-DOLCE mappings in Sect. 8 and Sect. 9.

Before formally introducing the mappings and the subsequent process of axiom verification, we add a few preliminary remarks about two problems. The first problem is related to the technical choices made to represent categories in BFO, on the one hand, and in DOLCE and EMMO, on the other hand (see Sect. 5.1). The second problem concerns the resolution adopted by the considered TLOs, i.e., the different level at which they investigare some classes (see Sect. 5.2).

5.1 Representation of categories

To each category in the DOLCE or EMMO taxonomy corresponds a FOL unary predicate. For instance, to state that an entity x is an endurant, one writes $\mathrm{ED}(x)$. These categories are assumed to correspond to rigid properties, that is, an entity cannot change its taxonomic classification. In other terms, if an entity x belongs to a category, it must belong to that category from the time it starts to exist to the time it ceases to exist (if any).

In BFO, universals are in the domain of quantification and a temporary *instance-of* primitive relation (written $::$) is introduced to represent when a particular is an instance of an universal at a given time. For instance, the fact that "at time t, x is a continuant" is formally represented as $x ::_t \mathtt{cnt}$, where `cnt` corresponds to the universal *being a continuant*. All the universals considered in the BFO taxonomy are non-empty and rigid through time (see Sect. 2.14.1) with the exception of `obj`, `objagg`, and `fobj`. For instance, a material entity could belong to `obj` for some time, and then to `objagg` (or `fobj`) and change again later, provided at each point in time it is classified by one of these three universals.

These different representational choices raise an initial problem because, according to (M3), one should individuate to which kind of DOLCE or EMMO entities the BFO universals correspond (Sect. 6.3 adds further considerations on this point). Here we focus on the BFO-DOLCE mappings but similar considerations apply to the EMMO-DOLCE mappings that however are here considered only via the mediation of DOLCE.

We consider only the BFO-universals present in the taxonomy, and start from the subset that can be defined in DOLCE. For each universal $\mathtt{u}$ we introduce a syntactic definition of form: $x ::_t \mathtt{u} \coloneqq \phi(x,t)$. Note that here the 'parameters' are x and t, not $\mathtt{u}$, which is a constant. This means that, first we do not define the notion of instance-of but we define only the instantiation of a given $\mathtt{u}$. That is, in general, the definitions of $x ::_t \mathtt{u}_1$ and $x ::_t \mathtt{u}_2$ could be different. Second, as discussed earlier, $\mathtt{u}$ will not appear in $\phi(x,t)$ and thus will not become an individual in the domain of DOLCE. If we were to use the biconditional conditions, writing $x ::_t \mathtt{u} \leftrightarrow \phi(x,t)$ in the theory, both the predicate $::$ and the individual constant $\mathtt{u}$ would be added to the vocabulary of DOLCE, and this would imply to have some universal in the domain of DOLCE. One alternative way to avoid this problem is to substitute $\mathtt{U}(x,t) \coloneqq \phi(x,t)$ for $x ::_t \mathtt{u} \leftrightarrow \phi(x,t)$, i.e., by mapping the needed BFO categories to predicates and the instance-of relation to (logical) predication. We adopt the first strategy because it is closer to the formalization of BFO, and it avoids to include universals in the domain of DOLCE as well as the relation instance-of among the primitives of DOLCE.

In this perspective, BFO axioms quantifying over universals (see, especially, Sect. 2.2) are disregarded by the mappings. Weaker versions of these axioms are introduced by considering the (finite) set $\{u_1, \ldots, u_n\}$ of those universals explicitly used to generate the mappings. For instance, (a_b6) can be substituted by $\exists xt(x ::_t u_1) \wedge \ldots \wedge \exists x(x ::_t u_n)$.

5.2 Different ontological focus and resolution

When it comes to comparing BFO, DOLCE and EMMO, it is natural to start from the first two, as they share many similarities. In fact, the informal categorical distinctions in DOLCE and BFO are quite similar. The presentations of BFO occurrents and DOLCE perdurants are very close and this conclusion is further supported by the fact that both the ontologies introduce a primitive of parthood simpliciter for these entities (this parthood relation is called P_d in DOLCE and oP in BFO). Analogously, we notice a similarity between BFO continuants and DOLCE endurants, and both ontologies conceive the related temporary parthood relation as a primitive (tP_d in DOLCE and cP in BFO). Things are slightly different for temporal/spatial regions and qualities. While these categories might appear *prima facie* similar, formally, they are classified in different ways in the two ontologies. BFO classifies qualities (and, more generally, specific dependent continuants) and spatial regions under continuants while temporal regions are classified under occurrents. In DOLCE, temporal regions, spatial regions, and qualities are neither endurants nor perdurants.[6] In particular, temporal regions and spatial regions are abstract entities in DOLCE. This difference –mainly grounded on the way the distinction between endurants/continuants and perdurants/occurents is characterized– complicates the comparison and needs to be taken into account in defining the mappings (see for instance the definition (d_{db}1) in $\mathfrak{D}_b$ of the *existsAt* primitive (EX) of BFO).

The way in which the most general categories (endurants/continuants on the one hand, perdurants/occurrents on the other hand) are specialized diverges considerably in the two ontologies:

On endurants vs continuants. In DOLCE perdurants are specialized mainly on the basis of two notions, both extensively discussed in the linguistic and philosophical literature: *homeomericity* (roughly, the parts of a perdurant of a certain kind are also of the same kind) and *cumulativity* (roughly, the mereological sum of perdurants all of which are of a certain kind is of the same kind, too); in BFO processes are distinguished from process boundaries mainly on the basis of the dimensionality of their temporal locations and on the fact that they are temporal proper parts of other occurrents.

On perdurants vs occurrents. The spatial and material dimensions of this kind of entities play a central role in both DOLCE and BFO (but see Sect. 6.3 where subtle differences are discussed). DOLCE presents a finer taxonomy mainly aimed to cover the notions of agentivity and sociality. BFO is driven by the distinction between fiat vs. bona fide entities, explicitly introduces the notion of aggregate, and relies on the dimensionality of the spatial (rather than temporal) location to distinguish sites and continuant fiat boundaries.

On qualities. The branch of the taxonomy for qualities in DOLCE is detached from that of endurants/continuants, differently from BFO. DOLCE's organization of qualities

[6] We will come back to this topic *infra*; a discussion of spatiotemporal regions will then be provided.

rests on the *comparability* principle: it makes sense to compare the color of a rose and the color of a vase, while it does not make sense to compare the color of a rose and the weight of a vase. Qualities are clustered in terms of maximal comparability, e.g., colors, weights, lengths, shapes, etc. form different quality classes. Quality spaces, each having a structure, are associated to qualities following the same principle, e.g., for colors, the space has regions for red, blue, green, etc. Instead, BFO distinguishes classes of specifically dependent continuants according to the existence, the nature of, and the participants involved in their realization processes.

These different ways of classifying entities are not incompatible *per se*. For instance, one could consider cumulativity, agentivity, comparability in BFO, and dimensionality of temporal/spatial regions and realization processes in DOLCE. But, clearly, this approach requires to extend BFO and DOLCE, a choice that should be pondered carefully. This kind of extension is not exploited in this deliverable. The consequence is that some primitive notions and categories of the target ontology cannot be defined in terms of those in the source ontology. As a result of this choice, the mapping focuses on the general notions and categories of the two ontologies.

As it should be evident from the brief introduction of EMMO put forward in Sect. 4, the differences between EMMO and BFO/DOLCE are much more pronounced.

Core Ontological Sorts. The core categorical distinction in EMMO are grounded in formal ontology, and, specifically, on notions which can be understood in terms of causal self-connectedness and dimensionality. While it is possible to draw some analogies with the partition of immaterial entities in BFO, the scope of the two are completely different, and EMMO downright rejects a multiplicative approach with respect to co-location. In fact, the formal principles partitioning the logical space introduce no strictly ontological distinctions, and, thus, all the entities in EMMO can be considered as falling under the same ontological sort. Partly as a consequence of that, while mereology plays an important role in all the foundational ontologies considered, in EMMO it is downright pivotal.

Architecture. While BFO and DOLCE are (mostly) organized as simple trees, EMMO introduces a number of perspectives connected to the mereocausal module, which, in principle, could be investigated as standalone ontologies. Not even the perspectives introduce strictly ontological distinctions, instead providing parametric tools or other kind of formal classifications, with a few exceptions related to commitments extrapolated from the sciences, in line with EMMO's focus on material modeling. Thus, while entities can be seen as falling under 4 major ontological sorts in DOLCE, and under 2 in BFO, there is only one ontological sort for all EMMO's entities.

Persistence. That said, even EMMO contains an analogue to the core distinction shared by BFO and DOLCE, i.e., a way to distinguish entities depending on how they persist in time (via the classes Object and Process in the persistence perspective). However, as it has been anticipated, in EMMO this distinction is parametric, having an epistemological, rather than ontological, flavor. The differences are just as marked when it comes to resolution, as EMMO immediately introduces practical classification without focusing on high-level criteria. Again, the tool to distinguish between Objects and Processes is ultimately grounded on primitives introduced in the mereocausal module, with all the related limitations.

On "qualities". Even qualities find a place in EMMO, in the semiotic perspective, as anticipated in Sect 4.2.2. Without going back into the details, even in this case the core distinctions are strongly rooted in the mereocausal module, and the taxonomical distinctions serve to characterize semiotic processes. The focus is on observational processes, the establishment of extrinsic connections between entities and functional distinctions, rather than on putative ontological distinctions among said entities. Signs group together entities which are taken to be quite different in BFO and DOLCE, providing formal or empirically-grounded distinctions in their stead.

All in all, and despite all the differences, EMMO can be compared with BFO and DOLCE under many respects: in general, the core issue is individuating the strategies adopted by the involved ontologies to meet practical expressive needs. To provide an example beside the ones cited above, while EMMO does not support forms of ontological dependence, of the kind grounding, for instance, BFO's classification of continuants, semiosis can be understood in terms of the establishment of representational dependences.

Other perspectives of EMMO introduce further toolboxes and conceptual schemata, following, more or less the same principles. In some cases, these perspectives have no direct analogues in BFO and DOLCE, while in some other cases, some connections could in principle be draw. We won't discuss this point here since it is not relevant for the sake of the proposed alignment.

The peculiarities of EMMO make for complex alignments, often requiring systematic approximations, on pain of obtaining minimal links. Given EMMO's reductionistic approach, and non-standard mereocausal and semiotic modules, it is especially difficult to recreate EMMO's primitives in other ontologies. Likewise, EMMO's unique ontological sort, and its contained number of taxonomical distinctions, can support only the definition of general notions from other ontologies.

6 The mapping from DOLCE to BFO

This section presents the technical results of the establishment of a mapping from DOLCE to BFO, that is, it shows how to define in DOLCE the categories and relations of BFO. As discussed earlier, the mapping does not cover all categories and relations.

The first part of this section, Sect. 6.1, reports the elements covered by the mapping and the conceptual and formal reasons for the limitation. This part makes technically clear the theoretical and formal barriers for a complete coverage of the ontology when this kind of mappings is exploited. It ends with the list of primitive relations and the list of categories of BFO which are covered by the mappings followed by the corresponding syntactic definitions in DOLCE.

The second part of this section, Sect. 6.2, verifies whether the axioms of BFO hold in DOLCE when extended with the given syntactic definitions. The axioms are clustered according to the relations or categories they characterize. This part is essentially a list of theorems and their proofs. The proofs have been tested using state of the art theorem provers.

The third and final part, Sect. 6.3, provides an in-depth analysis of the achieved results, the limitations and possible alternative strategies.

6.1 Mappings

In this section we establish how the BFO notions can be mapped to DOLCE: according to our previous discussion and point (M3) of Sect. 5, we introduce syntactic definitions of BFO notions in terms of DOLCE primitives. Unfortunately, not all the BFO notions can be defined in DOLCE in this way. We clarify the main reasons for this limitation when it happens.

First of all, DOLCE cannot (without suitable extensions) capture the distinctions grounded on the dimensionality of the instances of a given category. This implies that:

1. all the subcategories of `sreg` and of `cfbnd` are ruled out;
2. the distinction between `site` and `cfbnd` cannot be captured (sites are three-dimensional while continuant fiat boundaries are two-, one-, or zero-dimensional);
3. the distinction between `tint` and `tinst` can be only roughly characterized.

Concerning (ii) we introduce a definition only for the disjunction of `site` and `cfbnd` (written `site`$\cup$`cfbnd`), see (d_{db}12). `site`$\cup$`cfbnd` is clearly not among the categories in the BFO taxonomy and it might not be acceptable as a BFO universal. However, it is used in the mapping with a purely technical role: $x ::_t$ `site`$\cup$`cfbnd` is a shortcut for $x ::_t$ `site` $\vee\, x ::_t$ `cfbnd`. Concerning (iii) we identify temporal instants (intervals) with DOLCE atomic (non-atomic) time intervals, see (d_{db}9) and (d_{db}10). Admittedly, this is a very rough characterization. First, temporal atoms can have the same dimensionality of the times they are part of. Second, in BFO the (finite) sum of zero-dimensional temporal regions is still zero-dimensional while the sum of atomic times is always not atomic, i.e., according to (d_{db}9) and (d_{db}10), it is an interval. Third, BFO time intervals are convex while DOLCE non-atomic time intervals are not necessarily convex. Thus, (d_{db}10) seems to approximate the category `treg1` better than `tint`. At the same time, if all the non-atomic entities are instances of `treg1`, then `treg0` and `tinst` collapse. For these reasons, we omit `treg1` and `treg0` and consider only the rough definition of `tinst` and `tint`. Note that, in BFO, the convexity of intervals is characterized by means of the primitive of precedence (`PREC`). To define `PREC` in DOLCE one should extend the theory with an order relation defined on time intervals. Similarly for `FINST` (firstInstantOf) and `LINST` (lastInstantOf). We then need to rule out `PREC`, `FINST`, and `LINST`.

As anticipated earlier, following point (M1) of Sect. 5 the category of specifically dependent continuants (`sdcnt`) seem to correspond to the DOLCE category of qualities. However, as observed in Sect. 5.2, BFO and DOLCE distinguish the subclasses of these categories on the basis of different criteria. In BFO, the subcategories of `sdcnt` are mainly characterized by means of the primitives `REAL` (realizes) and `MBAS` (materialBasisOf) for which, however, the characterization is missing.[7] As a consequence, `REAL`, `MBAS`, and the subcategories of `sdcnt` are not included in the mapping.

[7] Some necessary conditions can be collected for `REAL` but these are not enough to characterize the relation.

A similar situation holds for the subcategories of material entities (mten). In particular, objects aggregates are intimately connected to the memberOf (mP) relation. None of these notions can be defined without an extension of DOLCE. It follows that mP and the subcategories of mten are not covered by the mapping.

Other cases are more subtle. The primitive historyOf (HIST) seems to be naturally defined in DOLCE as:

$$\mathtt{HIST}(x,y) \coloneqq \forall z(\mathtt{P_d}(z,x) \leftrightarrow \forall t(\mathtt{PRE}(z,t) \rightarrow \mathtt{PC}(y,z,t))$$

However the effectiveness of this definition strongly depends on the existential assumptions on perdurants. For instance, if y just participates to a perdurant p but only during a part t of the temporal extension of p and the temporal part of p during t does not exist—and DOLCE does not commit to the existence of all the temporal slices of a perdurant—then y would not have an history. These technical difficulties, and the fact that histories seem to have a marginal role in the BFO ontology, suggest to leave out HIST and hist. (This choice might be revised in the finale release of the deliverable.)

Spatiotemporal regions are not explicitly present in DOLCE. One could introduce them among the regions together with the correspondent qualities. Links to the spatial and temporal projections (corresponding to the BFO TPROJ and SPROJ) would be also necessary. Alternatively, violating (M2), spatiotemporal regions could be built (at the semantic level, for instance) as sets of couples ⟨time interval, space region⟩. These options could be investigated but, since they are technical extensions, we do not consider them here. However, it seems that spatiotemporal regions do not add expressive power so the problem might be less controversial from the technical viewpoint. Sect. 6.2.3 delineates how one can avoid spatiotemporal regions by rewriting BFO axioms concerning spatiotemporal regions/locations in terms of DOLCE time intervals/TLC and space regions/SLC.

Finally, following the discussion in Sect. 5.1, instance-of (::) is not directly defined. We define in DOLCE the instantiation of the needed universals.

Summing up, among the BFO notions, in the following we introduce:

1. syntactic definitions for the primitive relations: EX, cP, oP, tmP, SREG, TREG, OCCIN, LOC, SDEP, CONCR, GDEP, PTC; and
2. syntactic definitions with form $x ::_t \mathtt{u}$ for each category $\mathtt{u} \in \{$cnt, occ, idcnt, gdcnt, sdcnt, mten, imen, site∪cfbnd, sreg, proc, pbnd, treg, tinst, tint$\}$.

Below we list these syntactic definitions together with a short informal description. A deeper analysis about these definitions and their impact on the preservation of the axioms of BFO is done in Sect. 6.3.

$\mathbf{d_{db}1}$ $\mathtt{EX}(x,t) \coloneqq \mathtt{PRE}(x,t) \vee (\mathtt{T}(x) \wedge \mathtt{P_d}(t,x)) \vee (\mathtt{AB}(x) \wedge \neg\mathtt{T}(x) \wedge \mathtt{T}(t))$.

EX extends PRE: for EDs, PDs, and Qs, EX and PRE coincide but EX applies also to ABs. Time intervals EX-exist at every subinterval of themselves while the other abstracts entities EX-exist at every time. This extension try to match the fact that, in BFO, both temporal regions and spatial regions (that seem very close to DOLCE time intervals and space regions, respectively, see ($\mathrm{d_{db}}$7), ($\mathrm{d_{db}}$13)) are in time.

$\mathbf{d_{db}2}$ $\mathtt{oP}(x,y) \coloneqq \mathtt{P_d}(x,y) \wedge ((\mathtt{PD}(x) \wedge \mathtt{PD}(y)) \vee (\mathtt{T}(x) \wedge \mathtt{T}(y)))$

oP restricts $\mathtt{P_d}$ that originally applies to all ABs. In this way we try to preserve the fact that in BFO oP is defined only on occurrents (that include temporal regions while spatial regions are classified under continuants).

$\mathbf{d_{db}3}$ $\mathtt{tmP}(x,y) \coloneqq \mathtt{oP}(x,y) \wedge \forall z(\mathtt{oP}(z,y) \wedge \forall t(\mathtt{EX}(z,t) \rightarrow \mathtt{EX}(x,t)) \rightarrow \mathtt{oP}(z,x))$

This is the classical definition of temporal part/slice, i.e., x is the maximal part of y during the temporal extension of x.

$\mathbf{d_{db}4}$ $\mathtt{TREG}(x,t) \coloneqq \mathrm{PD}(x) \wedge \mathtt{TLC}(x,t)$

TREG is the restriction of TLC to PDs (in DOLCE TLC is defined for all the entities present in time, i.e., also for EDs and QS).

$\mathbf{d_{db}5}$ $x ::_t \mathtt{pbnd} \coloneqq \mathrm{PD}(x) \wedge \exists y(\mathtt{tmPP}(x,y)) \wedge \mathtt{TLC}(x,t) \wedge \mathtt{AT}(t)$

A process boundary is a temporally atomic perdurant that is a temporal proper part of at least another perdurant. As said, the atomicity of the temporal location of a perdurant is an approximation of its instantaneity.

$\mathbf{d_{db}6}$ $x ::_t \mathtt{proc} \coloneqq \mathrm{PD}(x) \wedge \mathtt{PRE}(x,t) \wedge \neg(x ::_t \mathtt{pbnd})$

Processes are perdurants that are not process boundaries (all the perdurants are present at some times ($\mathrm{t_d}$9)).

$\mathbf{d_{db}7}$ $x ::_t \mathtt{treg} \coloneqq \mathrm{T}(x) \wedge \mathtt{EX}(x,t)$

Temporal regions coincide with DOLCE time intervals.

$\mathbf{d_{db}8}$ $x ::_t \mathtt{occ} \coloneqq x ::_t \mathtt{pbnd} \vee x ::_t \mathtt{proc} \vee x ::_t \mathtt{treg}$

As said before, spatiotemporal regions are ruled out (but see Sect. 6.2.3).

$\mathbf{d_{db}9}$ $x ::_t \mathtt{tinst} \coloneqq x ::_t \mathtt{treg} \wedge \mathtt{AT}(x)$

Temporal instants are atomic time intervals.

$\mathbf{d_{db}10}$ $x ::_t \mathtt{tint} \coloneqq x ::_t \mathtt{treg} \wedge \neg\mathtt{AT}(x)$

Temporal intervals are non-atomic time intervals (in BFO they are self-connected but this property cannot be defined in DOLCE).

$\mathbf{d_{db}11}$ $x ::_t \mathtt{mten} \coloneqq \mathrm{ED}(x) \wedge \mathtt{PRE}(x,t) \wedge \exists u(\mathtt{PRE}(x,u) \wedge \neg\mathtt{AT}(u)) \wedge \forall u(\mathtt{PRE}(x,u) \rightarrow \exists y s r(\mathrm{M}(y) \wedge \mathtt{SLC}(x,s,u) \wedge \mathtt{SLC}(y,r,u) \wedge \mathrm{P_d}(r,s)))$

Material entities are non instantaneous (to match ($\mathrm{a_b}$8)) endurants that during their whole life are (at least partially) spatially co-localized with an amount of matter.

$\mathbf{d_{db}12}$ $x ::_t (\mathtt{site} \cup \mathtt{cfbnd}) \coloneqq \mathrm{F}(x) \wedge \mathtt{PRE}(x,t) \wedge \exists s(\mathtt{SLC}(x,s,t)) \wedge \forall u(\mathtt{PRE}(x,u) \rightarrow \neg\exists y s r(\mathrm{M}(y) \wedge \mathtt{SLC}(x,s,u) \wedge \mathtt{SLC}(y,r,u) \wedge \mathrm{P_d}(r,s)))$

Both sites and continuant fiat boundaries are DOLCE features localized in space that during their whole life are never (partially) spatially co-localized with an amount of matter.

$\mathbf{d_{db}13}$ $x ::_t \mathtt{sreg} \coloneqq \mathrm{S}(x) \wedge \mathtt{EX}(x,t)$

Spatial regions coincide with DOLCE space regions.

$\mathbf{d_{db}14}$ $x ::_t \mathtt{imen} \coloneqq x ::_t (\mathtt{site} \cup \mathtt{cfbnd}) \vee x ::_t \mathtt{sreg}$

$\mathbf{d_{db}15}$ $x ::_t \mathtt{idcnt} \coloneqq x ::_t \mathtt{mten} \vee x ::_t \mathtt{imen}$

d_{db}16 $x ::_t \texttt{sdcnt} \coloneqq \mathrm{Q}(x) \wedge \texttt{PRE}(x,t) \wedge \exists y(y ::_t \texttt{idcnt} \wedge \neg \mathrm{S}(y) \wedge \texttt{DQT}(x,y))$

Specifically dependent continuants are DOLCE qualities inhering (in the sense of DQT) in an independent continuant (as defined in (d_{db}15)) that is not a spatial region.

d_{db}17 $\texttt{CONCR}(x,y,t) \coloneqq x ::_t \texttt{sdcnt} \wedge \mathrm{NPED}(y) \wedge \neg\exists s(\texttt{SLC}(y,s,t)) \wedge \texttt{EXD}(y,x,t)$

Concretization is a form of DOLCE existential dependence (EXD) between a non-physical endurant y that does not have a spatial localization and a specifically dependent continuant (as defined in (d_{db}16)) x.

d_{db}18 $x ::_t \texttt{gdcnt} \coloneqq \texttt{PRE}(x,t) \wedge \forall u(\texttt{PRE}(x,u) \rightarrow \exists y(\texttt{CONCR}(y,x,u)))$

Generic dependent continuants are non-physical endurants (from (d_{db}17)) that are concretized during their whole life.

d_{db}19 $x ::_t \texttt{cnt} \coloneqq x ::_t \texttt{idcnt} \vee x ::_t \texttt{sdcnt} \vee x ::_t \texttt{gdcnt}$

d_{db}20 $\texttt{cP}(x,y,t) \coloneqq x ::_t \texttt{cnt} \wedge y ::_t \texttt{cnt} \wedge (\texttt{tP}_\texttt{d}(x,y,t) \vee \mathrm{P}_\mathrm{d}(x,y) \vee \exists zu(\texttt{DQT}(x,z) \wedge \texttt{DQT}(y,u) \wedge \texttt{tP}_\texttt{d}(z,u,t)))$

According to (d_{db}19), (d_{db}15), (d_{db}16), and (d_{db}18), continuants include EDs, SSs, and QSs. For EDs, cP coincides with $\texttt{tP}_\texttt{d}$. For PDs, cP coincides with P_d. For QSs, in DOLCE, there is not a parthood relation but we can use the parthood relation among the endurants these qualities inhere in (each quality inheres in a single entity (a_d9)).[8]

d_{db}21 $\texttt{SREG}(x,s,t) \coloneqq x ::_t \texttt{idcnt} \wedge \neg \mathrm{S}(x) \wedge \texttt{SLC}(x,s,t)$

SREG is the resctriction of SLC to independent continuants (as defined in (d_{db}15)) that are not spatial regions.

d_{db}22 $\texttt{PTC}(x,y,t) \coloneqq x ::_t \texttt{cnt} \wedge \neg \mathrm{S}(x) \wedge y ::_t \texttt{proc} \wedge \texttt{PC}(x,y,t)$

PTC is the resctriction of PC to continuants that are not spatial regions and to processes (as defined above).

d_{db}23 $\texttt{OCCIN}(x,y) \coloneqq \mathrm{PD}(x) \wedge \exists t(y ::_t \texttt{mten} \vee y ::_t \texttt{site} \cup \texttt{cfbnd}) \wedge \forall t(\texttt{PRE}(x,t) \rightarrow \exists sr(\texttt{SLC}(x,s,t) \wedge \texttt{SLC}(y,r,t) \wedge \mathrm{P}_\mathrm{d}(s,r)))$

$\texttt{OCCIN}(x,y)$ holds when the spatial location of the perdurant x is always included in the one of the material entity/site/continuant fiat boundary y (this definition closely corresponds to the one in the documentation of BFO).

d_{db}24 $\texttt{LOC}(x,y,t) \coloneqq x ::_t \texttt{idcnt} \wedge \neg \mathrm{S}(x) \wedge y ::_t \texttt{idcnt} \wedge \neg \mathrm{S}(y) \wedge \exists sr(\texttt{SLC}(x,s,t) \wedge \texttt{SLC}(y,r,t) \wedge \mathrm{P}_\mathrm{d}(s,r))$

At time t, and independent continuant (that is not a space region) is located in another independent continuant (that is not a space region) when the spatial location (at t) of the first continuant is included in the spatial location (at t) of the second continuant (this definition closely corresponds to the one in the documentation of BFO).

[8] Note that this definitions allows qualities of different kind to be one part of the other (for instance the color of an object is part of the temperature of a bigger object). To avoid these cases one could assume that every specialization of DOLCE has a (finite) set of leaf-types of qualities and require, in the last disjunct in (d_{db}20), x and y to be instances of the same leaf-type. We leave the implementation and the study of this new definition for future work.

d_{db}25 $\texttt{SDEP}(x,y) \coloneqq x :: \texttt{sdcnt} \wedge \texttt{DQT}(x,y)$

SDEP is the restriction of DQT to specifically dependent continuants (as defined in (d_{db}16).

d_{db}26 $\texttt{GDEP}(x,y,t) \coloneqq x ::_t \texttt{gdcnt} \wedge y ::_t \texttt{idcnt} \wedge \neg \texttt{S}(y) \wedge \exists z s(\texttt{P}_\texttt{d}(s,t) \wedge \texttt{DQT}(z,y) \wedge \texttt{CONCR}(z,x,s))$

At time t, the generic dependent continuant x generically depends on the independent continuant (that is not a spatial region) y, when it is concretized (during a part s of t) by a specifically dependent continuant z inhering in y.

d_{db}27 $\text{PAR}(x) \coloneqq \exists t(\texttt{EX}(x,t))$

Particulars are entities that EX-exist in time (to match (a_b4) and (a_b7)).

d_{db}28 $\text{UNI}(x) \coloneqq \neg \text{PAR}(x)$

Universals are non-particulars (to match (a_b1) and (a_b2)).

6.2 Check of the preservation of the original BFO axioms

As discussed in Sect. 5.2 (and made explicit in Sect. 6.1) some BFO-axioms involve relations or categories that have not been defined in the mappings. In what follows, when possible and relevant we consider approximations of these axioms as expressible in the DOLCE language, otherwise we set them aside.

In the rest of this section, $\mathfrak{D}_b$ indicates the 'extension' of $\mathfrak{D}$ with the mappings introduced in Sect. 6.1 together with the syntactic definitions introduced in Sect. 2, i.e., formally $\mathfrak{D}_b = \mathfrak{D} \cup \{(d_{db}1)\text{-}(d_{db}27)\} \cup \{(d_b1), (d_b4)\text{-}(d_b10)\}$.

Note that (d_b2) and (d_b3) are not included in $\mathfrak{D}_b$ because they require universals in the domain of quantification. Moreover, (d_b2) could be avoided by including the existential quantifier on time. Alternatively, for each predicate u representing a BFO-category, one could introduce the syntactic definition $x :: \texttt{u} \coloneqq \exists t(x ::_t \texttt{u})$. Definition ($d_b$3) is also not needed since it does not occur in the BFO axioms. This definition will be used in the mapping from BFO to DOLCE, see Sect. 7.1.

Several proofs has been verified by using theorems provers, they are reported in Sect. 11.3.

6.2.1 occurrentPartOf

t_{db}1 $\mathfrak{D}_b \vdash \texttt{oP}(x,y) \rightarrow \exists t(x ::_t \texttt{occ}) \wedge \exists t(y ::_t \texttt{occ})$ (a_b37)

Proof. See proof generated by theorem provers. □

t_{db}2 $\mathfrak{D}_b \vdash \exists t(x ::_t \texttt{occ}) \rightarrow \texttt{oP}(x,x)$ (a_b38)

Proof. See proof generated by theorem provers. □

t_{db}3 $\mathfrak{D}_b \vdash \texttt{oP}(x,y) \wedge \texttt{oP}(y,x) \rightarrow x = y$ (a_b39)

Proof. See proof generated by theorem provers. □

t_{db}4 $\mathfrak{D}_b \vdash \texttt{oP}(x,y) \wedge \texttt{oP}(y,z) \rightarrow \texttt{oP}(x,z)$ (a_b40)

Proof. See proof generated by theorem provers. □

t_{db}5 $\mathfrak{D}_b \nvdash \texttt{oO}(x,y) \rightarrow \exists z(\forall w(\texttt{oP}(w,z) \leftrightarrow \texttt{oP}(w,x) \wedge \texttt{oP}(w,y)))$ (a_b41)

Proof. See the counterexample on page 131 where $\mathtt{oO}(\mathtt{t123},\mathtt{t234})$, $\mathtt{oP}(\mathtt{t2},\mathtt{t123})$, $\mathtt{oP}(\mathtt{t2},\mathtt{t234})$, $\mathtt{oP}(\mathtt{t3},\mathtt{t123})$, $\mathtt{oP}(\mathtt{t3},\mathtt{t234})$, but the mereological sum of t2 and t3 does not exists. □

t_{db}6 $\mathfrak{D}_b \vdash \mathtt{oP}(x,y) \wedge \mathtt{EX}(x,t) \rightarrow \mathtt{EX}(y,t)$ (a_b42)
Proof. See proof generated by theorem provers. □

t_{db}7 $\mathfrak{D}_b \nvdash x ::_t \mathtt{proc} \wedge \mathtt{oP}(x,y) \rightarrow \exists t'(y ::_{t'} \mathtt{proc})$ (a_b43)
Proof. See the counterexample on page 131 where $\mathtt{p1} ::_{\mathtt{t1}} \mathtt{proc}$ (because it is not a temporal proper part of any process), $\mathtt{oP}(\mathtt{p1},\mathtt{p12})$ but $\mathtt{p12} ::_{\mathtt{t1}} \mathtt{pbnd}$ (it is instantaneous at it is a temporal proper part of p123). □

t_{db}8 $\mathfrak{D}_b \vdash x ::_t \mathtt{proc} \wedge \mathtt{oP}(y,x) \rightarrow \exists t'(y ::_{t'} \mathtt{proc} \vee y ::_{t'} \mathtt{pbnd})$ (a_b44)
Proof. See proof generated by theorem provers. □

t_{db}9 $\mathfrak{D}_b \nvdash x ::_t \mathtt{proc} \rightarrow \exists y t'(y ::_{t'} \mathtt{pbnd} \wedge \mathtt{oP}(y,x))$ (a_b45)
Proof. See the counterexample on page 132 where $\mathtt{p1} ::_{\mathtt{t1}} \mathtt{proc}$ (there are no processes of which p1 is a temporal proper part) and p1 is atomic. □

t_{db}10 $\mathfrak{D}_b \vdash x ::_t \mathtt{pbnd} \wedge \mathtt{oP}(x,y) \rightarrow \exists t'(y ::_{t'} \mathtt{pbnd} \vee y ::_{t'} \mathtt{proc})$ (a_b46)
Proof. See proof generated by theorem provers. □

t_{db}11 $\mathfrak{D}_b \nvdash x ::_t \mathtt{pbnd} \wedge \mathtt{oP}(y,x) \rightarrow \exists t'(y ::_{t'} \mathtt{pbnd})$ (a_b47)
Proof. See the counterexample on page 131 where $\mathtt{p12} ::_{\mathtt{t1}} \mathtt{pbnd}$ (it is instantaneous at it is a temporal proper part of p123) and $\mathtt{oP}(\mathtt{p1},\mathtt{p12})$ but $\mathtt{p1} ::_{\mathtt{t1}} \mathtt{proc}$ (it is not a temporal proper part of any process). □

t_{db}12 $\mathfrak{D}_b \vdash \mathtt{oP}(x,y) \rightarrow (\exists t(x ::_t \mathtt{treg}) \leftrightarrow \exists t(y ::_t \mathtt{treg}))$ (a_b48)
Proof. See proof generated by theorem provers. □

6.2.2 temporalPartOf

t_{db}13 $\mathfrak{D}_b \vdash \mathtt{tmP}(x,y) \rightarrow \exists t(x ::_t \mathtt{occ}) \wedge \exists t(y ::_t \mathtt{occ})$ (a_b50)
Proof. From $\mathtt{tmP}(x,y)$, by (d_{db}3), $\mathtt{oP}(x,y)$ and then, by (d_{db}2), $(\mathrm{PD}(x) \wedge \mathrm{PD}(y)) \vee (\mathrm{T}(x) \wedge \mathrm{T}(y))$. If $\mathrm{T}(x) \wedge \mathrm{T}(y)$, by ($d_{db}$1) and ($a_d$4), $\mathtt{EX}(x,x) \wedge \mathtt{EX}(y,y)$ and then, by (d_{db}7), $x ::_x \mathtt{treg} \wedge y ::_y \mathtt{treg}$. When $\mathrm{PD}(x) \wedge \mathrm{PD}(y)$ the thesis follows from (t_{db}94). □

t_{db}14 $\mathfrak{D}_b \vdash \exists t(x ::_t \mathtt{occ}) \rightarrow \mathtt{tmP}(x,x)$ (a_b52)
Proof. See proof generated by theorem provers. □

t_{db}15 $\mathfrak{D}_b \vdash \mathtt{tmP}(x,y) \wedge \mathtt{tmP}(y,x) \rightarrow x = y$ (a_b53)
Proof. See proof generated by theorem provers. □

t_{db}16 $\mathfrak{D}_b \vdash \mathtt{tmP}(x,y) \wedge \mathtt{tmP}(y,z) \rightarrow \mathtt{tmP}(x,z)$ (a_b54)
Proof. See proof generated with the help of theorem provers. □

t_{db}17 $\mathfrak{D}_b \vdash \exists t(x ::_t \mathtt{treg}) \wedge \exists t(y ::_t \mathtt{treg}) \rightarrow (\mathtt{tmP}(x,y) \leftrightarrow \mathtt{oP}(x,y))$ (a_b58)
Proof. See proof generated by theorem provers. □

t_{db}18 $\mathfrak{D}_b \vdash x ::_x \mathtt{treg} \wedge y ::_y \mathtt{treg} \wedge \mathtt{tmPP}(x,y) \rightarrow \exists z(\mathtt{tmPP}(z,y) \wedge \neg \mathtt{tmO}(z,x))$ (a_b55)
Proof. From the hypothesis, by (d_{db}7) and (t_{db}17), $\mathrm{T}(x) \wedge \mathrm{T}(y) \wedge \mathtt{oPP}(x,y)$ and then, by (d_{db}2) and (a_d5), $\neg \mathrm{P}_d(y,x)$. Then, from (a_d7), $\exists z(\mathrm{P}_d(z,y) \wedge \neg \mathrm{O}_d(z,x))$

and then, by (a_d2), $\mathtt{T}(z)$. From $\mathtt{T}(y) \wedge \mathtt{T}(z) \wedge \mathtt{P_d}(z,y)$, by ($d_{db}$2) and ($t_{db}$17), $\mathtt{tmP}(z,y)$. From $\mathtt{T}(x) \wedge \mathtt{T}(z) \wedge \neg \mathtt{O_d}(z,x)$, by ($d_{db}$2) and ($t_{db}$17), $\neg\mathtt{tmO}(z,x)$. It lacks then to show that $z \neq y$. By contradiction, assume $z = y$, then $\mathtt{tmPP}(x,y) \wedge \neg\mathtt{tmO}(z,x)$ we have $\mathtt{tmPP}(x,y) \wedge \neg\mathtt{tmO}(y,x)$. Contradiction. □

t_{db}19 $\mathfrak{D}_b \vdash \mathtt{tmP}(x,y) \rightarrow \mathtt{oP}(x,y)$ (a_b56)
Proof. Directly from (d_{db}3). □

t_{db}20 $\mathfrak{D}_b \vdash \mathtt{tmP}(x,y) \wedge \mathtt{EX}(x,t) \rightarrow \mathtt{EX}(y,t)$ (a_b57)
Proof. Directly from (d_{db}3). □

t_{db}21 $\mathfrak{D}_b \vdash t' ::_t \mathtt{treg} \rightarrow \mathtt{tmP}(t,t')$ (a_b59)
Proof. See proof generated by theorem provers. □

t_{db}22 $\mathfrak{D}_b \vdash \mathtt{tmPP}(x,y) \rightarrow \neg\mathtt{tmPP}(y,x)$ (a_b65)
Proof. See proof generated by theorem provers. □

t_{db}23 $\mathfrak{D}_b \vdash \mathtt{tmPP}(x,y) \wedge \mathtt{tmPP}(y,z) \rightarrow \mathtt{tmPP}(x,z)$ (a_b66)
Proof. See proof generated with the help of theorem provers. □

t_{db}24 $\mathfrak{D}_b \nvdash \exists t(x ::_t \mathtt{pbnd}) \leftrightarrow$
$\exists y(\mathtt{tmP}(x,y) \wedge \exists t(y ::_t \mathtt{proc})) \wedge \exists t(\mathtt{TREG}(x,t) \wedge t ::_t \mathtt{tinst})$ (a_b60)
Proof. See the counterexample on page 132 where $\mathtt{p1} ::_{\mathtt{t1}} \mathtt{proc}$ (there are no processes of which p1 is a temporal proper part), $\mathtt{TREG}(\mathtt{p1},\mathtt{t1})$ (because p1 is a perdurant and $\mathtt{TLC}(\mathtt{p1},\mathtt{t1})$), $\mathtt{t1} ::_{\mathtt{t1}} \mathtt{tinst}$, and $\mathtt{tmP}(\mathtt{p1},\mathtt{p1})$ but p1 is a process. □

t_{db}25 $\mathfrak{D}_b \vdash \exists t(x ::_t \mathtt{pbnd}) \wedge \mathtt{TREG}(x,t) \rightarrow t ::_t \mathtt{tinst}$ (a_b61)
Proof. See proof generated by theorem provers. □

t_{db}26 $\mathfrak{D}_b \nvdash \exists t(x ::_t \mathtt{proc}) \wedge \mathtt{TREG}(x,t) \rightarrow \exists t'(t' ::_{t'} \mathtt{tint} \wedge \mathtt{tmP}(t',t))$ (a_b62)
Proof. See the counterexample on page 132 where $\mathtt{p1} ::_{\mathtt{t1}} \mathtt{proc}$ (there are no processes of which p1 is a temporal proper part), $\mathtt{TREG}(\mathtt{p1},\mathtt{t1})$ (because p1 is a perdurant and $\mathtt{TLC}(\mathtt{p1},\mathtt{t1})$) but t1 is atomic (and therefore it is a temporal instant). □

t_{db}27 $\mathfrak{D}_b \vdash \exists t(x ::_t \mathtt{proc} \vee x ::_t \mathtt{pbnd}) \wedge \mathtt{TM}(t) \rightarrow$
$(\mathtt{TREG}(x,t) \leftrightarrow \forall y t'(\mathtt{oP}(y,x) \wedge \mathtt{TREG}(y,t') \rightarrow \mathtt{oP}(t',t)) \wedge$
$\neg\exists t'(\mathtt{oPP}(t',t) \wedge \mathtt{TREG}(x,t')))$
(a_b63)
Proof. See proof generated with the help of theorem provers. □

t_{db}28 $\mathfrak{D}_b \vdash \mathtt{oP}(x,y) \wedge \mathtt{TREG}(x,t) \wedge \mathtt{TREG}(y,t') \rightarrow \mathtt{oP}(t,t')$ (a_b64)
Proof. See proof generated by theorem provers. □

6.2.3 occupiesSpatiotemporalRegion

In the following we consider the rewriting of the axioms concerning spatiotemporal regions. These rewritings are based on the relation $\mathtt{SREG_0}$ defined in (d_{db}29) which is the analogous for occurrents of the relation SREG defined in (d_{db}21) for continuants. However note that $\mathtt{SREG_0}$ is not considered in BFO even though it can be easily defined on the basis of spatiotemporal regions and their projection on space.

d_{db}29 $\mathtt{SREG}_0(x,s,t) \coloneqq (x ::_t \mathtt{proc} \vee x ::_t \mathtt{pbnd}) \wedge \mathtt{SLC}(x,s,t)$

(a_b88) $\exists t(x ::_t \mathtt{proc} \vee x ::_t \mathtt{pbnd}) \rightarrow \exists r(\mathtt{STREG}(x,r))$

t_{db}29 $\mathfrak{D}_b \cup \{(d_{db}29)\} \nvdash \exists t(x ::_t \mathtt{proc} \vee x ::_t \mathtt{pbnd}) \rightarrow \forall t(\mathtt{EX}(x,t) \rightarrow \exists s(\mathtt{SREG}_0(x,s,t)))$
Proof. See the counterexample on page 132 where p2 is a process (because it is not a temporal proper part of other processes) but it does not have a spatial location (it only participant is a non-physical endurant). □

(a_b89) $\mathtt{STREG}(x,r) \wedge \mathtt{STREG}(x,s) \rightarrow r = s$

t_{db}30 $\mathfrak{D}_b \cup \{(d_{db}29)\} \vdash \mathtt{SREG}_0(x,r,t) \wedge \mathtt{SREG}_0(x,s,t) \rightarrow r = s$
Proof. See proof generated with the help of theorem provers. □

(a_b90) $\mathtt{oP}(x,y) \wedge \mathtt{STREG}(x,r) \wedge \mathtt{STREG}(y,s) \rightarrow \mathtt{oP}(r,s)$

t_{db}31 $\mathfrak{D}_b \cup \{(d_{db}29)\} \vdash \mathtt{oP}(x,y) \wedge \mathtt{SREG}_0(x,r,t) \wedge \mathtt{SREG}_0(y,s,t) \rightarrow \mathtt{cP}(r,s,t)$
Proof. See proof generated with the help of theorem provers. □

(a_b91) $\exists t(x ::_t \mathtt{proc} \vee x ::_t \mathtt{pbnd}) \wedge \exists t(y ::_t \mathtt{proc} \vee y ::_t \mathtt{pbnd}) \rightarrow$
$(\mathtt{oP}(x,y) \leftrightarrow \exists rs(\mathtt{STREG}(x,r) \wedge \mathtt{STREG}(y,s) \wedge \mathtt{oP}(r,s)))$

t_{db}32 $\mathfrak{D}_b \cup \{(d_{db}29)\} \nvdash \exists t(x ::_t \mathtt{proc} \vee x ::_t \mathtt{pbnd}) \wedge \exists t(y ::_t \mathtt{proc} \vee y ::_t \mathtt{pbnd}) \rightarrow$
$\mathtt{oP}(x,y) \leftrightarrow \forall t(\mathtt{EX}(x,t) \rightarrow \exists rs(\mathtt{SREG}_0(x,r,t) \wedge \mathtt{SREG}_0(y,s,t) \wedge \mathtt{cP}(r,s,t)))$
Proof. ($\rightarrow$) See the counterexample on page 133 where $\mathtt{p1} ::_{\mathtt{t1}} \mathtt{proc}$ (there are no processes of which p1 is a temporal proper part), $\mathtt{EX}(\mathtt{p1},\mathtt{t1})$, and $\mathtt{oP}(\mathtt{p1},\mathtt{p1})$ but p1 does not have a spatial location because its only participant is a non-physical endurant that does not have a spatial location (not all the NPEDs are necessarily located in space).
($\leftarrow$) See the counterexample on page 133 where p1 and p2 are spatiotemporally coincident but they are not part one of the other. □

t_{db}33 $\mathfrak{D}_b \vdash \exists t(x ::_t \mathtt{mten}) \rightarrow \exists t(t ::_t \mathtt{tint} \wedge \mathtt{EX}(x,t))$ $\sim$ (a_b8)
Proof. Directly from the condition $\exists u(\mathtt{PRE}(x,u) \wedge \neg \mathtt{AT}(u))$ in the definition of mten (d_{db}11). □

6.2.4 continuantPartOf

t_{db}34 $\mathfrak{D}_b \vdash \mathtt{cP}(x,y,t) \rightarrow x ::_t \mathtt{cnt} \wedge y ::_t \mathtt{cnt} \wedge \mathrm{TM}(t)$ (a_b12)
Proof. By (d_{db}19) $x ::_t \mathtt{cnt} \wedge y ::_t \mathtt{cnt}$. From $x ::_t \mathtt{cnt}$, by (t_{db}105) and (t_{db}102) we have $\mathrm{TM}(t)$. □

t_{db}35 $\mathfrak{D}_b \vdash x ::_t \mathtt{idcnt} \rightarrow \mathtt{cP}(x,x,t)$ (a_b13)
Proof. See proof generated by theorem provers. □

t_{db}36 $\mathfrak{D}_b \vdash \mathtt{cP}(x,y,t) \wedge \mathtt{cP}(y,z,u) \wedge \mathtt{tmP}(t,u) \rightarrow \mathtt{cP}(x,z,t)$ (a_b14)
Proof. By (d_{db}20), $x ::_t \mathtt{cnt} \wedge z ::_u \mathtt{cnt}$. From $z ::_u \mathtt{cnt} \wedge \mathtt{tmP}(t,u)$, by ($d_{db}$3), ($d_{db}$2), and ($t_{db}$106), $z ::_t \mathtt{cnt}$, thus we have $x ::_t \mathtt{cnt} \wedge z ::_t \mathtt{cnt}$.
– If $\mathtt{tP}_d(x,y,t)$, by (a_d32), $\mathrm{ED}(y)$ then, by (a_d107) and the fact that T is a subclass of AB, we also have $\mathtt{tP}_d(y,z,u)$. From $\mathtt{tP}_d(y,z,u) \wedge \mathtt{tmP}(t,u)$, by ($d_{db}$3), ($d_{db}$2), and ($a_d$34), we have $\mathtt{tP}_d(y,z,t)$ and by (a_d37) $\mathtt{tP}_d(x,z,t)$.
– If $\mathtt{P}_d(x,y)$, by (a_d1), $\mathrm{AB}(x) \wedge \mathrm{AB}(y)$ or $\mathrm{PD}(x) \wedge \mathrm{PD}(y)$, in both cases we have also $\mathtt{P}_d(y,z)$. By (a_d6), $\mathtt{P}_d(x,z)$.

– If $\exists vu(\texttt{DQT}(x,v)\wedge\texttt{DQT}(y,u)\wedge\texttt{tP}_\texttt{d}(v,u,t)))$ then, by (a_d8), x and y are qualities and then also z must be a quality and $\exists v'u'(\texttt{DQT}(y,v')\wedge\texttt{DQT}(z,u')\wedge\texttt{tP}_\texttt{d}(v',u',t)))$. From $\texttt{DQT}(y,u)\wedge\texttt{DQT}(y,v')$, by ($a_d$9), $u=v'$. $\texttt{tP}_\texttt{d}(v,u,t)\wedge\texttt{tP}_\texttt{d}(u,u',t)$, by ($a_d$37), $\texttt{tP}_\texttt{d}(v,u',t)$. Therefore we have that $\exists vu'(\texttt{DQT}(x,v)\wedge\texttt{DQT}(z,u')\wedge\texttt{tP}_\texttt{d}(v,u',t)))$. □

t_{db}37 $\mathfrak{D}_\texttt{b}\nvdash\texttt{cP}(x,y,t)\wedge x\neq y\rightarrow\exists z(\texttt{cP}(z,y,t)\wedge z\neq y\wedge\neg\texttt{cO}(z,x,t))$ (a_b15)
Proof. See the counterexample on page 134 where, at t1, e1 and e2 are both instances of site∪cfbnd (because they are feautures present only at t1 which are not partially spatially co-localized with an amount of matter). In addition $\texttt{e1}\neq\texttt{e2}$ but $\texttt{tP}_\texttt{d}(\texttt{e1},\texttt{e2},\texttt{t1})$ and $\texttt{tP}_\texttt{d}(\texttt{e2},\texttt{e1},\texttt{t1})$ hold, therefore, by the transitivity of $\texttt{tP}_\texttt{d}$, all the parts of e1 are also parts of e2 and vice versa. □

t_{db}38 $\mathfrak{D}_\texttt{b}\nvdash\texttt{cO}(x,y,t)\rightarrow\exists z(\forall w(\texttt{cP}(w,z,t)\leftrightarrow\texttt{cP}(w,x,t)\wedge\texttt{cP}(w,y,t)))$ (a_b16)
Proof. See the counterexample on page 134 where we have $\texttt{oO}(\texttt{s123},\texttt{s234})$, $\texttt{oP}(\texttt{s2},\texttt{s123})$, $\texttt{oP}(\texttt{s2},\texttt{s234})$, $\texttt{oP}(\texttt{s3},\texttt{s123})$, $\texttt{oP}(\texttt{s3},\texttt{s234})$ but the mereological sum of s2 and s3 does not exist. □

t_{db}39 $\mathfrak{D}_\texttt{b}\vdash\texttt{cP}(x,y,t)\wedge\texttt{tmP}(v,t)\rightarrow\texttt{cP}(x,y,v)$ (a_b17)
Proof. From $\texttt{cP}(x,y,t)$, by (d_{db}20), $x::_t\texttt{cnt}\wedge y::_t\texttt{cnt}$. From $\texttt{tmP}(v,t)$, by (d_{db}3) and (d_{db}2), $\texttt{P}_\texttt{d}(v,t)$. From $x::_t\texttt{cnt}\wedge y::_t\texttt{cnt}$ and $\texttt{P}_\texttt{d}(v,t)$, by ($t_{db}$107), $x::_v\texttt{cnt}\wedge y::_v\texttt{cnt}$.
From $\texttt{cP}(x,y,t)$, by (d_{db}20), $\texttt{tP}_\texttt{d}(x,y,t)\vee\texttt{P}_\texttt{d}(x,y)\vee\exists zu(\texttt{DQT}(x,z)\wedge\texttt{DQT}(y,u)\wedge\texttt{tP}_\texttt{d}(z,u,t))$. If $\texttt{tP}_\texttt{d}(x,y,t)$, then from $\texttt{tP}_\texttt{d}(x,y,t)\wedge\texttt{P}_\texttt{d}(v,t)$, by ($a_d$34), $\texttt{tP}_\texttt{d}(x,y,v)$ and then the thesis. If $\texttt{P}_\texttt{d}(x,y)$ the thesis follows directly. If $\exists zu(\texttt{DQT}(x,z)\wedge\texttt{DQT}(y,u)\wedge\texttt{tP}_\texttt{d}(z,u,t))$, then from $\texttt{tP}_\texttt{d}(z,u,t)\wedge\texttt{P}_\texttt{d}(v,t)$, by ($a_d$34), $\texttt{tP}_\texttt{d}(z,y,v)$ and then the thesis. □

t_{db}40 $\mathfrak{D}_\texttt{b}\vdash\texttt{cP}(x,y,t)\wedge\texttt{EX}(y,t)\rightarrow\texttt{EX}(x,t)$ (a_b18)
Proof. See proof generated by theorem provers. □

t_{db}41 $\mathfrak{D}_\texttt{b}\nvdash\exists t(x::_t\texttt{idcnt}\wedge y::_t\texttt{idcnt}\wedge\neg(x::_t\texttt{objagg})\wedge\neg(y::_t\texttt{objagg})\wedge\texttt{cP}(x,y,t)\wedge\texttt{cP}(y,x,t))\rightarrow x=y$ (a_b20)
Proof. See the counterexample of

– $\exists t(x::_t\texttt{imen}\wedge y::_t\texttt{imen}\wedge\texttt{cP}(x,y,t)\wedge\texttt{cP}(y,x,t))\rightarrow x=y$

on page 134 where $\texttt{tP}_\texttt{d}(\texttt{e1},\texttt{e2},\texttt{t1})$, $\texttt{tP}_\texttt{d}(\texttt{e2},\texttt{e1},\texttt{t1})$, but $\texttt{e1}\neq\texttt{e2}$. □

t_{db}42 $\mathfrak{D}_\texttt{b}\vdash\texttt{cP}(x,y,t)\rightarrow(x::_t\texttt{idcnt}\leftrightarrow y::_t\texttt{idcnt})$ (a_b22)

– $\mathfrak{D}_\texttt{b}\vdash\texttt{cP}(x,y,t)\wedge x::_t\texttt{idcnt}\rightarrow y::_t\texttt{idcnt}$
Proof. From $\texttt{cP}(x,y,t)$, by (d_{db}20), $y::_t\texttt{cnt}$.
Assume $x::_t\texttt{sreg}$ then, by (d_{db}20), $\texttt{P}_\texttt{d}(x,y)$ and, by (a_d3), $\texttt{S}(y)$. From $y::_t\texttt{cnt}$, by (t_{db}105), $\texttt{EX}(y,t)$. From $\texttt{S}(y)\wedge\texttt{EX}(y,t)$, by ($d_{db}$13), $y::_t\texttt{sreg}$.
Assume $x::_t\texttt{mten}$ or $x::_t\texttt{site∪cfbnd}$ then, by (d_{db}20), (d_{db}11), (d_{db}12), (a_d101) and (t_d22), $\texttt{tP}_\texttt{d}(x,y,t)$ and $\forall u(\texttt{PRE}(x,u)\rightarrow\exists s(\texttt{SLC}(x,s,u)))$ then, by (a_d32), (a_d33) and (a_d84), $\texttt{ED}(y)\wedge\texttt{PRE}(y,t)\wedge\exists s(\texttt{SLC}(y,s,t))$. It follows that y is neither an sdcnt (because it is not a quality) nor a gdcnt (because, against (d_{db}17), it has a spatial location at t). Then, by (d_{db}19), it is an idcnt. □

– $\mathfrak{D}_\texttt{b}\nvdash\texttt{cP}(x,y,t)\wedge y::_t\texttt{idcnt}\rightarrow x::_t\texttt{idcnt}$

Proof. See the counterexample on page 135 where, at t1, e23 is a material entity because it is always spatially co-localized with the amount of matter e1. Furthermore, $\mathtt{tP_d}(\mathtt{e3},\mathtt{e23},\mathtt{t1})$ and then, at t1, e3 is a generically dependent continuant (because $\mathtt{CONCR}(\mathtt{sl1},\mathtt{e3},\mathtt{t1})$, $\mathtt{CONCR}(\mathtt{sl1},\mathtt{e3},\mathtt{t2})$, and $\mathtt{CONCR}(\mathtt{sl1},\mathtt{e3},\mathtt{t12})$ and $\mathtt{PRE}(\mathtt{e3},\mathtt{t1})$). We then have that e3 is a continuant at t1, e23 is a independent continuant at t1, and $\mathtt{cP}(\mathtt{e3},\mathtt{e23},\mathtt{t1})$ but e3 is not an independent continuant at t1. □

t_{db}43 $\mathfrak{D}_b \nvdash x ::_t \mathtt{mten} \wedge \exists y(\mathtt{cP}(y,x,t) \wedge x \neq y) \rightarrow \exists z(\mathtt{cP}(z,x,t) \wedge x \neq z \wedge \neg(z ::_t \mathtt{imen}))$ (a_b23)

Proof. See the counterexample on page 136 where $\mathtt{e12} ::_{\mathtt{t12}} \mathtt{mten}$ (because $\mathrm{ED}(\mathtt{e12})$, $\mathtt{PRE}(\mathtt{e12},\mathtt{t12}) \wedge \neg\mathtt{AT}(\mathtt{t12})$, and at all times at which e12 is present (namely t1, t2, t12) the spatial location of the amount of matter m1 (i.e., s24) is part of the spatial location of e12 (i.e., s1234)). In addition we have $\mathtt{e1} ::_{\mathtt{t12}} \mathtt{site} \cup \mathtt{cfbnd}$ (because $\mathrm{F}(\mathtt{e1})$, $\mathtt{PRE}(\mathtt{e1},\mathtt{t12})$, and at all times at which e1 is present (namely t1, t2, t12) the spatial location of the only amount of matter m1 (i.e., s24) is not part of the spatial location of e1 (i.e., s12)). Similarly for e2. But $\mathtt{cP}(\mathtt{e1},\mathtt{e12},\mathtt{t12})$, $\mathtt{cP}(\mathtt{e2},\mathtt{e12},\mathtt{t12})$ (because $\mathtt{tP_d}(\mathtt{e1},\mathtt{e12},\mathtt{t12})$, $\mathtt{tP_d}(\mathtt{e2},\mathtt{e12},\mathtt{t12})$, and e1, e2, e12 are all instances of cnt at t12), $\mathtt{e1} \neq \mathtt{e12}$, $\mathtt{e2} \neq \mathtt{e12}$, and e12 does not have additional proper parts, i..e., all the continuant proper parts of e12 are immaterial entities. □

t_{db}44 $\mathfrak{D}_b \vdash \mathtt{cP}(x,y,t) \rightarrow (x ::_t \mathtt{sreg} \leftrightarrow y ::_t \mathtt{sreg})$ (a_b28)

Proof. See proof generated by theorem provers. □

t_{db}45 $\mathfrak{D}_b \nvdash x ::_t \mathtt{mten} \wedge \mathtt{cP}(y,x,t) \rightarrow (y ::_t \mathtt{mten} \vee y ::_t \mathtt{site} \cup \mathtt{cfbnd})$ (a_b24)

Proof. See the counterexample on page 135 where, at t1, e23 is a material entity because it is always spatially co-localized with the amount of matter e1. Furthermore, $\mathtt{cP}(\mathtt{e3},\mathtt{e23},\mathtt{t1})$ and e3 is a generically dependent continuant at t1 (because $\mathtt{CONCR}(\mathtt{sl1},\mathtt{e3},\mathtt{t1})$, $\mathtt{CONCR}(\mathtt{sl1},\mathtt{e3},\mathtt{t2})$, and $\mathtt{CONCR}(\mathtt{sl1},\mathtt{e3},\mathtt{t12})$ and $\mathtt{PRE}(\mathtt{e3},\mathtt{t1})$). Therefore e3 is a generically dep. continuant at t1, e23 is a material entity at t1, and $\mathtt{cP}(\mathtt{e3},\mathtt{e23},\mathtt{t1})$. □

t_{db}46 $\mathfrak{D}_b \vdash x ::_t \mathtt{mten} \wedge \mathtt{cP}(x,y,t) \rightarrow y ::_t \mathtt{mten}$ (a_b25)

Proof. From $\mathtt{cP}(x,y,t)$, by (d_{db}20), $y ::_t \mathtt{cnt}$. From $x ::_t \mathtt{mten}$, by (d_{db}20) and (d_{db}11), $\mathtt{tP_d}(x,y,t)$ and $\forall u(\mathtt{PRE}(x,u) \rightarrow \exists s(\mathtt{SLC}(x,s,u)))$ then, by (a_d32), (a_d33) and (a_d84), $\mathrm{ED}(y) \wedge \mathtt{PRE}(y,t) \wedge \exists s(\mathtt{SLC}(y,s,t))$. It follows that y is neither an sreg (because it is not a region), nor an sdcnt (because it is not a quality), nor a gdcnt (because, against (d_{db}17), it has a spatial location at t). By (a_d85) the spatial location of x at t is part of the the spatial location of y at t and then, by (a_d6) and (d_{db}11), at t, y is partially co-located with an amount of matter. By (d_{db}12), it follows that $y ::_t \mathtt{site} \cup \mathtt{cfbnd}$ does not hold. By (d_{db}19) the only remaining option is $y ::_t \mathtt{mten}$. □

6.2.5 occupiesSpatialRegion and locatedIn

t_{db}47 $\mathfrak{D}_b \vdash \mathtt{SREG}(x,y,t) \rightarrow x ::_t \mathtt{idcnt} \wedge \neg(x ::_t \mathtt{sreg}) \wedge y ::_t \mathtt{sreg} \wedge \mathrm{TM}(t)$ (a_b73)

Proof. See proof generated by theorem provers. □

$\mathbf{t_{db}48}$ $\mathfrak{D}_b \vdash \mathtt{LOC}(x,y,t) \rightarrow$
$x ::_t \mathtt{idcnt} \wedge \neg(x ::_t \mathtt{sreg}) \wedge y ::_t \mathtt{idcnt} \wedge \neg(y ::_t \mathtt{sreg}) \wedge \mathtt{TM}(t)$ (a_b74)
Proof. See proof generated by theorem provers. □

$\mathbf{t_{db}49}$ $\mathfrak{D}_b \vdash \mathtt{SREG}(x,y,t) \wedge \mathtt{SREG}(x,z,t) \rightarrow y = z$ (a_b75)
Proof. See proof generated by theorem provers. □

$\mathbf{t_{db}50}$ $\mathfrak{D}_b \nvdash \mathtt{SREG}(x,y,t) \wedge \mathtt{tmP}(t',t) \rightarrow \mathtt{SREG}(x,y,t')$ (a_b76)
Proof. See the counterexample on page 137 where e1 is a material entity such that $\mathtt{SLC}(\mathtt{e1},\mathtt{s12},\mathtt{t12})$, $\mathtt{SLC}(\mathtt{e1},\mathtt{s1},\mathtt{t1})$, and $\mathtt{tmP}(\mathtt{t1},\mathtt{t12})$. □

$\mathbf{t_{db}51}$ $\mathfrak{D}_b \nvdash x ::_t \mathtt{mten} \wedge y ::_t \mathtt{mten} \wedge \mathtt{SREG}(x,r,t) \wedge \mathtt{SREG}(y,r,t) \rightarrow \mathtt{cP}(x,y,t) \wedge \mathtt{cP}(y,x,t)$ (a_b78)
Proof. See the counterexample on page 137 where m1 and e1 are both material entities which are spatiotemporally co-localized but they are not part one of the other. □

$\mathbf{t_{db}52}$ $\mathfrak{D}_b \vdash \mathtt{LOC}(x,y,t) \rightarrow \exists urs(\mathtt{tmP}(u,t) \wedge \mathtt{SREG}(x,r,u) \wedge \mathtt{SREG}(y,s,u) \wedge \mathtt{cP}(r,s,u))$ (a_b76)
Proof. Assume $u = t$. By (a_d4), (d_{db}2), (d_{db}3), $\mathtt{tmP}(t,t)$. From $\mathtt{LOC}(x,y,t)$, by (d_{db}24), we have $\exists sr(\mathtt{SLC}(x,s,t) \wedge \mathtt{SLC}(y,r,t) \wedge \mathtt{P_d}(s,r))$, $x ::_t \mathtt{idcnt} \wedge \neg \mathtt{s}(x)$, $y ::_t \mathtt{idcnt} \wedge \neg \mathtt{s}(y)$. Therefore, by ($d_{db}$21) and ($a_d$76), $\exists sr(\mathtt{SREG}(x,s,t) \wedge \mathtt{SREG}(x,r,t))$. From $\mathtt{SLC}(x,s,t) \wedge \mathtt{SLC}(y,r,t)$, by ($a_d$74), $\mathtt{s}(s) \wedge \mathtt{s}(r)$ then, by (d_{db}13) and (d_{db}1), $s ::_t \mathtt{sreg} \wedge s ::_t \mathtt{sreg}$. By ($d_{db}$20), ($d_{db}$14), and ($d_{db}$15), the thesis follows directly from $s ::_t \mathtt{sreg}$, $s ::_t \mathtt{sreg}$, and $\mathtt{P_d}(s,r)$. □

$\mathbf{t_{db}53}$ $\mathfrak{D}_b \nvdash \mathtt{LOC}(x,y,t) \wedge \mathtt{tmP}(t',t) \rightarrow \mathtt{LOC}(x,y,t')$ (a_b79)
Proof. See the counterexample on page 138 where e1 and e2 are both material entities such that $\mathtt{SLC}(\mathtt{e1},\mathtt{s12},\mathtt{t12})$, $\mathtt{SLC}(\mathtt{e2},\mathtt{s12},\mathtt{t12})$. It follows that $\mathtt{LOC}(\mathtt{e1},\mathtt{s12},\mathtt{t12})$ and $\mathtt{LOC}(\mathtt{e2},\mathtt{s12},\mathtt{t12})$. However we have $\mathtt{SLC}(\mathtt{e1},\mathtt{s1},\mathtt{t1})$, $\mathtt{SLC}(\mathtt{e2},\mathtt{s2},\mathtt{t1})$, $\mathtt{tmP}(\mathtt{t1},\mathtt{t12})$, $\neg\mathtt{P_d}(\mathtt{s1},\mathtt{s2})$, therefore $\neg\mathtt{LOC}(\mathtt{e1},\mathtt{e2},\mathtt{t1})$ and $\neg\mathtt{LOC}(\mathtt{e2},\mathtt{e1},\mathtt{t1})$. □

$\mathbf{t_{db}54}$ $\mathfrak{D}_b \nvdash \mathtt{LOC}(x,y,t) \wedge \mathtt{LOC}(y,z,t') \wedge \mathtt{tmP}(t,t') \rightarrow \mathtt{LOC}(x,z,t)$ (a_b80)
Proof. See the counterexample on page 138 and the proof of (t_{db}53) by observing that $\mathtt{LOC}(\mathtt{e1},\mathtt{e1},\mathtt{t1})$, $\mathtt{LOC}(\mathtt{e1},\mathtt{e2},\mathtt{t12})$, but $\neg\mathtt{LOC}(\mathtt{e1},\mathtt{e2},\mathtt{t1})$. □

$\mathbf{t_{db}55}$ $\mathfrak{D}_b \vdash \mathtt{LOC}(x,y,t) \wedge \mathtt{cP}(y,z,t) \rightarrow \mathtt{LOC}(x,z,t)$ (a_b81)
Proof. From $\mathtt{LOC}(x,y,t)$, by (d_{db}24), $x ::_t \mathtt{idcnt} \wedge \neg \mathtt{s}(x)$ and $y ::_t \mathtt{idcnt} \wedge \neg \mathtt{s}(y)$. From $y ::_t \mathtt{idcnt} \wedge \mathtt{cP}(y,z,t)$, by ($t_{db}$42), $z ::_t \mathtt{idcnt}$ and $\mathtt{tP_d}(y,z,t)$. From $\mathtt{tP_d}(y,z,t) \wedge \neg \mathtt{s}(y)$, by ($a_d$3), $\neg \mathtt{s}(z)$. Then we have that $z ::_t \mathtt{idcnt} \wedge \neg \mathtt{s}(z)$. From $\mathtt{LOC}(x,y,t)$, by (d_{db}24), $\exists sr(\mathtt{SLC}(x,s,t) \wedge \mathtt{SLC}(y,r,t) \wedge \mathtt{P_d}(s,r))$. From $\mathtt{tP_d}(y,z,t) \wedge \mathtt{SLC}(y,r,t)$, by ($a_d$84) and ($a_d$85), $\exists v(\mathtt{SLC}(z,v,t) \wedge \mathtt{P_d}(r,v))$ and, by (a_d6), $\exists sv(\mathtt{SLC}(x,s,t) \wedge \mathtt{SLC}(z,v,t) \wedge \mathtt{P_d}(s,v))$. □

$\mathbf{t_{db}56}$ $\mathfrak{D}_b \nvdash \mathtt{LOC}(x,y,t) \wedge \mathtt{cP}(z,x,t) \rightarrow \mathtt{LOC}(z,y,t)$ (a_b82)
Proof. See the counterexample of the direction ($\leftarrow$) of (t_{db}42) showing that not all the cP-parts of x are independent continuants. □

$\mathbf{t_{db}57}$ $\mathfrak{D}_b \vdash x ::_t \mathtt{idcnt} \wedge \neg(x ::_t \mathtt{sreg}) \wedge y ::_t \mathtt{idcnt} \wedge \neg(y ::_t \mathtt{sreg}) \wedge \mathtt{cP}(x,y,t) \rightarrow$
$\mathtt{LOC}(x,y,t)$ (a_b83)
Proof. If $x ::_t \mathtt{idcnt}$, by (d_{db}11), (t_{db}105), $\exists s(\mathtt{SLC}(x,s,t))$. If $x ::_t \mathtt{site} \cup \mathtt{cfbnd}$ then, by (d_{db}11), (a_d101), and (t_d22), $\exists s(\mathtt{SLC}(x,s,t))$. From $x ::_t \mathtt{idcnt} \wedge \neg \mathtt{s}(x) \wedge$

$\texttt{cP}(x,y,t)$, by (d_{db}20), $\texttt{tP}_\texttt{d}(x,y,t)$. From $\texttt{tP}_\texttt{d}(x,y,t) \wedge \exists s(\texttt{SLC}(x,s,t))$, by ($a_d$84) and ($a_d$85), $\exists sr(\texttt{SLC}(x,s,t) \wedge \texttt{SLC}(y,r,t) \wedge \texttt{P}_\texttt{d}(s,r))$. □

6.2.6 occursIn

t_{db}58 $\mathfrak{D}_\texttt{b} \vdash \texttt{OCCIN}(x,y) \rightarrow \exists t(x ::_t \texttt{proc} \vee x ::_t \texttt{pbnd}) \wedge \exists t(y ::_t \texttt{mten} \vee y ::_t \texttt{site})$ (a_b87)
Proof. We prove

$$\texttt{OCCIN}(x,y) \rightarrow \exists t(x ::_t \texttt{proc} \vee x ::_t \texttt{pbnd}) \wedge \exists t(y ::_t \texttt{mten} \vee y ::_t \texttt{site} \cup \texttt{cfbnd})$$

The thesis follows directly from (d_{db}23), (a_d19), (d_{db}5), and (d_{db}6) by observing that pbnd and proc partition PD. □

t_{db}59 $\mathfrak{D}_\texttt{b} \vdash \texttt{OCCIN}(x,y) \wedge \texttt{EX}(x,t) \rightarrow \texttt{EX}(y,t)$ (a_b95)
Proof. See proof generated by theorem provers. □

t_{db}60 $\mathfrak{D}_\texttt{b} \nvdash \texttt{OCCIN}(x,y) \wedge \texttt{oP}(z,x) \rightarrow \texttt{OCCIN}(z,y)$ (a_b96)
Proof. See the counterexample of

- $\texttt{OCCIN}(x,y) \wedge \texttt{oP}(z,x) \wedge \texttt{PRE}(z,t) \rightarrow \exists r(\texttt{SLC}(z,r,t))$

on page 139 where $\texttt{OCCIN}(\texttt{p12},\texttt{e1})$ (because e1 is a material entity, $\texttt{PD}(\texttt{p12})$, $\texttt{SLC}(\texttt{p12},\texttt{s1},\texttt{t1})$, $\texttt{SLC}(\texttt{e1},\texttt{s1},\texttt{t1})$, $\texttt{SLC}(\texttt{p12},\texttt{s1},\texttt{t2})$, $\texttt{SLC}(\texttt{e1},\texttt{s1},\texttt{t2})$, $\texttt{SLC}(\texttt{p12},\texttt{s1},\texttt{t12})$ and $\texttt{SLC}(\texttt{e1},\texttt{s1},\texttt{t12})$) and in addition $\texttt{oP}(\texttt{p2},\texttt{p12})$, and $\texttt{PRE}(\texttt{p2},\texttt{t1})$ but $\neg\exists s(\texttt{SLC}(\texttt{p2},s,\texttt{t1}))$. □

t_{db}61 $\mathfrak{D}_\texttt{b} \nvdash \texttt{OCCIN}(x,y) \wedge \forall t(\texttt{EX}(x,t) \leftrightarrow \texttt{cP}(y,z,t)) \rightarrow \texttt{OCCIN}(x,z)$ (a_b97)
Proof. See the counterexample on page 135 where, at t1, e23 is a material entity because it is always spatially co-localized with the amount of matter e1. Then, p1 occurs in 23 (because it is always spatially co-localized with e23), e3 is always $\texttt{tP}_\texttt{d}$-part of e23 but e3 is not localized in space therefore p1 cannot occur in e3. □

t_{db}62 $\mathfrak{D}_\texttt{b} \vdash \texttt{OCCIN}(x,y) \wedge \forall t(\texttt{EX}(x,t) \leftrightarrow \texttt{LOC}(y,z,t)) \rightarrow \texttt{OCCIN}(x,z)$ (a_b98)
Proof. From $\texttt{OCCIN}(x,y)$, by (d_{db}23), $\exists u(x ::_u \texttt{mten})$ or $\exists u(x ::_u \texttt{site} \cup \texttt{cfbnd})$ and $\forall t(\texttt{PRE}(x,t) \rightarrow \exists sr(\texttt{SLC}(x,s,t) \wedge \texttt{SLC}(y,r,t) \wedge \texttt{P}_\texttt{d}(s,r)))$. Consider now a given t. From the hypothesis, by (d_{db}1) and (d_{db}24), $z ::_t \texttt{idcnt} \wedge \neg\texttt{S}(z)$ and $\exists uv(\texttt{SLC}(y,u,t) \wedge \texttt{SLC}(z,v,t) \wedge \texttt{P}_\texttt{d}(u,v))$ then, by (a_d6), (a_d76), $\forall t(\texttt{PRE}(x,t) \rightarrow \exists sv(\texttt{SLC}(x,s,t) \wedge \texttt{SLC}(z,v,t) \wedge \texttt{P}_\texttt{d}(s,v)))$. The thesis follows because, by (d_{db}15), (d_{db}13), $z ::_t \texttt{idcnt} \wedge \neg\texttt{S}(z)$ is equivalent to $x ::_t \texttt{mten} \vee x ::_t \texttt{site} \cup \texttt{cfbnd}$. □

6.2.7 specificallyDependsOn

t_{db}63 $\mathfrak{D}_\texttt{b} \vdash \texttt{SDEP}(x,y) \rightarrow \exists t(x ::_t \texttt{sdcnt}) \wedge (\exists t(y ::_t \texttt{sdcnt}) \vee (\exists t(y ::_t \texttt{idcnt}) \wedge \neg\exists t(y ::_t \texttt{sreg})))$ (a_b99)
Proof. By (d_{db}25), $\exists t(x ::_t \texttt{sdcnt})$ and then, by (d_{db}16), $\exists y(y ::_t \texttt{idcnt} \wedge \neg\texttt{S}(y) \wedge \texttt{DQT}(x,y))$. From $\exists y(y ::_t \texttt{idcnt} \wedge \neg\texttt{S}(y) \wedge \texttt{DQT}(x,y))$, by ($a_d$9), ($d_{db}$25), and ($d_{db}$13), the thesis follows directly. □

t_{db}64 $\mathfrak{D}_\texttt{b} \vdash \texttt{SDEP}(x,y) \rightarrow \neg\exists t(\texttt{cO}(x,y,t))$ (a_b100)
Proof. From $\texttt{SDEP}(x,y)$, by (d_{db}25), (d_{db}16) and (a_d8), $\texttt{DQT}(x,y)$, $\texttt{Q}(x)$ and $\texttt{PD}(y) \vee \texttt{PED}(y) \vee \texttt{NPED}(y)$. By contradiction, assume that $\exists tz(\texttt{cP}(z,x,t) \wedge \texttt{cP}(z,y,t))$.

From $\mathtt{cP}(z,x,t) \wedge \mathtt{Q}(x)$ we have $\mathtt{Q}(z)$. From $\mathtt{cP}(z,y,t) \wedge (\mathtt{PD}(y) \vee \mathtt{PED}(y) \vee \mathtt{NPED}(y))$ we have $\neg\mathtt{Q}(z)$. Contradiction. □

t_{db}65 $\mathfrak{D}_b \vdash \mathtt{SDEP}(x,y) \wedge \mathtt{SDEP}(y,z) \wedge x \neq z \rightarrow \mathtt{SDEP}(x,z)$ (a_b101)
Proof. See proof generated by theorem provers. □

t_{db}66 $\mathfrak{D}_b \vdash \mathtt{SDEP}(x,y) \rightarrow \exists t(\mathtt{EX}(x,t) \wedge \mathtt{EX}(y,t)) \wedge \forall t(\mathtt{EX}(x,t) \rightarrow \mathtt{EX}(y,t))$ (a_b102)
Proof. See proof generated by theorem provers. □

t_{db}67 $\mathfrak{D}_b \vdash x ::_t \mathtt{idcnt} \leftrightarrow x ::_t \mathtt{cnt} \wedge \neg\exists yu(\mathtt{SDEP}(x,y) \vee \mathtt{GDEP}(x,y,u))$ (a_b103)
Proof. ($\rightarrow$) From $x ::_t \mathtt{idcnt}$, by (d_{db}19), $x ::_t \mathtt{cnt}$ and, by (d_{db}15), (d_{db}11), (d_{db}12), (d_{db}13), and (d_{db}14), $\mathtt{ED}(x)$, or $\mathtt{F}(x)$, or $\mathtt{S}(x)$. By contradiction, assume that $\exists y(\mathtt{SDEP}(x,y))$, then, by ($d_{db}$25) and ($d_{db}$16), $\mathtt{Q}(x)$. But ED and S are both disjoint from Q. Contradiction. Again by contradiction, assume now that $\exists yt'(\mathtt{GDEP}(x,y,t'))$, then, by ($d_{db}$26), ($d_{db}$17), and ($d_{db}$18), we have that $\mathtt{NPED}(x)$, $\mathtt{PRE}(x,t)$, and $\forall u(\mathtt{PRE}(x,u) \rightarrow \neg\exists s(\mathtt{SLC}(x,s,u))$. NPED is disjoint from both F and S then x can not be an immaterial entity. Furthermore, by (d_{db}11), a material entity needs to have a spatial location at every time at which it is present, then x cannot be a material entity. Contradiction.
($\leftarrow$) First, we prove that $x ::_t \mathtt{sdcnt} \rightarrow \exists y(\mathtt{SDEP}(x,y))$. From the hypothesis, by (d_{db}16), $\exists y(\mathtt{DQT}(x,y))$, and from $x ::_t \mathtt{sdcnt} \wedge \mathtt{DQT}(x,y)$, by ($d_{db}$25), $\exists y(\mathtt{SDEP}(x,y))$. Second, we prove that $x ::_t \mathtt{gdcnt} \rightarrow \exists y(\mathtt{GDEP}(x,y,t))$. From the hypothesis, by (d_{db}18), $\exists z(\mathtt{CONCR}(z,x,t))$ then, by (d_{db}17), $z ::_t \mathtt{sdcnt}$ and then, by (d_{db}16), $\exists y(\mathtt{DQT}(z,y) \wedge y ::_t \mathtt{idcnt} \wedge \neg\mathtt{S}(y))$. Therefore, by ($a_d$4), $\exists zy(\mathtt{P_d}(t,t) \wedge \mathtt{DQT}(z,y) \wedge \mathtt{CONCR}(z,x,t) \wedge y ::_t \mathtt{idcnt} \wedge \neg\mathtt{S}(y))$, i.e., by ($d_{db}$26), $\exists y(\mathtt{GDEP}(x,y,t))$. From these two results, the hypotheses, and the rigidity of sdcnt and gdcnt, we have $x ::_t \mathtt{cnt} \wedge \neg(x ::_t \mathtt{sdcnt}) \wedge \neg(x ::_t \mathtt{gdcnt})$, i.e., by ($d_{db}$19), $x ::_t \mathtt{idcnt}$. □

t_{db}68 $\mathfrak{D}_b \vdash x ::_t \mathtt{sdcnt} \leftrightarrow x ::_t \mathtt{cnt} \wedge \exists y(y ::_t \mathtt{idcnt} \wedge \neg(y ::_t \mathtt{sreg}) \wedge \mathtt{SDEP}(x,y))$ (a_b104)
Proof. See proof generated by theorem provers. □

6.2.8 participatesIn

t_{db}69 $\mathfrak{D}_b \vdash \mathtt{PTC}(x,y,t) \rightarrow x ::_t \mathtt{cnt} \wedge \neg(x ::_t \mathtt{sreg}) \wedge y ::_t \mathtt{proc} \wedge \mathtt{TM}(t)$ (a_b112)
Proof. See proof generated by theorem provers. □

t_{db}70 $\mathfrak{D}_b \nvdash x ::_t \mathtt{proc} \rightarrow \exists y(\mathtt{PTC}(y,x,t))$ (a_b113)
Proof. See the counterexample on page 140 where, at t12, p1 is a process because it is not instantaneous. However no endurants participate to p1 during the whole t12 (only during t1 and t2). □

t_{db}71 $\mathfrak{D}_b \vdash \mathtt{PTC}(x,y,t) \wedge \mathtt{tmP}(u,t) \rightarrow \mathtt{PTC}(x,y,u)$ (a_b114)
Proof. From $\mathtt{PTC}(x,y,t)$, by (d_{db}22), $x ::_t \mathtt{cnt} \wedge \neg\mathtt{S}(x) \wedge y ::_t \mathtt{proc} \wedge \mathtt{PC}(x,y,t)$. From $\mathtt{PC}(x,y,t) \wedge \mathtt{tmP}(u,t)$, by ($a_d$54), ($d_{db}$3), and ($d_{db}$2), $\mathtt{PC}(x,y,u)$. The thesis follows from (t_{db}130), (t_{db}87), (t_{db}134), (t_{db}129), (t_{db}128), and (t_{db}135). □

t_{db}72 $\mathfrak{D}_b \vdash x ::_t \mathtt{sdcnt} \wedge \mathtt{PTC}(x,p,t) \rightarrow$ (a_b115)
$\exists yt'(\mathtt{tmP}(t',t) \wedge y ::_{t'} \mathtt{idcnt} \wedge \neg(y ::_{t'} \mathtt{sreg}) \wedge \mathtt{SDEP}(x,y) \wedge \mathtt{PTC}(y,p,t'))$
Proof. See proof generated by theorem provers. □

6.2.9 concretizes and genericallyDependsOn

$\mathbf{t_{db}73}$ $\mathfrak{D}_b \nvdash \texttt{CONCR}(x,y,t) \rightarrow (x ::_t \texttt{sdcnt} \vee x ::_t \texttt{proc}) \wedge y ::_t \texttt{gdcnt} \wedge \texttt{TM}(t)$ (a_b105)
Proof. See the counterexample on page 140 where $\texttt{CONCR}(\texttt{sl1},\texttt{e2},\texttt{t1})$ but, at t2, e2 is not a generically dependent continuant because at t2 is not concretized. □

$\mathbf{t_{db}74}$ $\mathfrak{D}_b \vdash \texttt{GDEP}(x,y,t) \rightarrow x ::_t \texttt{gdcnt} \wedge y ::_t \texttt{idcnt} \wedge \neg(y ::_t \texttt{sreg}) \wedge \texttt{TM}(t)$ (a_b106)
Proof. Directly from (d_{db}26), (t_{db}105), and (t_{db}102). □

$\mathbf{t_{db}75}$ $\mathfrak{D}_b \vdash x ::_t \texttt{gdcnt} \rightarrow \exists y t'(\texttt{tmP}(t',t) \wedge \texttt{CONCR}(y,x,t'))$ (a_b113)
Proof. See proof generated by theorem provers. □

$\mathbf{t_{db}76}$ $\mathfrak{D}_b \vdash \texttt{CONCR}(x,y,t) \wedge \texttt{tmP}(u,t) \rightarrow \texttt{CONCR}(x,y,u)$ (a_b108)
Proof. From $\texttt{CONCR}(x,y,t)$, by (d_{db}17), $x ::_t \texttt{sdcnt} \wedge \text{NPED}(y) \wedge \texttt{EXD}(y,x,t) \wedge \neg\exists s(\texttt{SLC}(y,s,t))$. From $x ::_t \texttt{sdcnt} \wedge \texttt{tmP}(u,t)$, by ($d_{db}$3), ($d_{db}$2), and ($t_{db}$129), $x ::_u \texttt{sdcnt}$. From $\texttt{EXD}(y,x,t) \wedge \texttt{tmP}(u,t)$, by ($a_d$60), $\texttt{EXD}(y,x,u)$. Finally, from $\neg\exists s(\texttt{SLC}(y,s,t)) \wedge \texttt{tmP}(u,t)$, by ($a_d$78), $\neg\exists s(\texttt{SLC}(y,s,u))$. □

$\mathbf{t_{db}77}$ $\mathfrak{D}_b \vdash \texttt{GDEP}(x,y,t) \rightarrow \exists z u(\texttt{tmP}(u,t) \wedge \texttt{INH}(z,y) \wedge \texttt{CONCR}(z,x,u))$ (a_b109)
Proof. Directly from (d_{db}26) and (d_{db}16) by observing that in this case INH coincides with DQT. □

$\mathbf{t_{db}78}$ $\mathfrak{D}_b \vdash \texttt{CONCR}(x,y,t) \wedge \texttt{INH}(x,z) \rightarrow \texttt{GDEP}(y,z,t)$ (a_b110)
Proof. From $\texttt{INH}(x,z)$, by (d_b10) and (d_{db}25), we have $\texttt{DQT}(x,z)$. By (a_d4), we have that $\texttt{P}_\texttt{d}(t,t)$. Therefore $\texttt{P}_\texttt{d}(t,t) \wedge \texttt{DQT}(x,z) \wedge \texttt{CONCR}(x,y,t)$, i.e., by ($d_{db}$26), $\texttt{GDEP}(y,z,t)$. □

$\mathbf{t_{db}79}$ $\mathfrak{D}_b \nvdash x ::_t \texttt{gdcnt} \wedge \texttt{PTC}(x,p,t) \rightarrow \exists y t'(\texttt{tmP}(t',t) \wedge \texttt{CONCR}(y,x,t') \wedge ((y ::_{t'} \texttt{sdcnt} \wedge \exists z(\texttt{SDEP}(y,z) \wedge \texttt{PTC}(z,p,t'))) \vee (\texttt{oP}(y,p) \wedge \texttt{EX}(y,t'))))$ (a_b111)
Proof. See the counterexample on page 132 where e1 is an independent continuant at t1 and it is not a space region, sl1 is a specifically dependent continuant at t1, and $\texttt{CONCR}(\texttt{sl1},\texttt{e2},\texttt{t1})$. Similarly for t2 and t12. It follows that e2 is a generically dependent continuant at all the times. We also have that p2 is a process (it is not a temporal proper part of other processes). Then $\texttt{e2} ::_{\texttt{t1}} \texttt{gdcnt}$, $\texttt{PTC}(\texttt{e2},\texttt{p2},\texttt{t1})$, and $\texttt{CONCR}(\texttt{sl1},\texttt{e2},\texttt{t1})$ where sl1 is the only entity (a specifically dependent continuant) that concretizes e2 at t1. But sl1 depends only on e1 that never participates in p2 and it is not an occurrent, therefore it cannot be part of p2. □

6.2.10 Taxonomy

With the exception of obj, objagg, and fobj, that however are not included in the mapping, all the universals considered in the BFO taxonomy are rigid through time (see Sect. 2.14.1) and such that the ISA-children of a given category partitionate the category, i.e., they are mutually disjoint and cover the whole category (see Sect. 2.14.2 and Sect.2.14.3).

From the mappings (d_{db}1)-(d_{db}27), it is easy to see that, by construction, all the children-categories cover the parent-category (the parent-category is always defined as the disjunction of the children-categories) except in the case of PAR, the class of particulars (which is defined in terms of EX). (t_{db}80) shows that particulars are not

partitioned by continuants and occurrents, i.e., there are entities of DOLCE that exist in time but are neither continuants nor occurrents. This is the case, for instance, of temporal qualities or abstract regions.

The proof of the mutual disjointness of all the leaves requires more work but it is quite direct. `proc` and `pbnd` are both subclasses of PD but the definition of $x ::_t \texttt{proc}$ requires $\neg(x ::_t \texttt{pbnd})$. `tinst` and `tint` are both subclasses of T (which is disjoint from PD) but the first requires the atomicity of the instances while the second the non-atomicity. `sdcnt` is a subclass of Q (which is disjoint from PD and T). `sreg` is a subclass of S (which is disjoint from PD, T, and Q). `mten` and `site`$\cup$`cfbnd` are both subclasses of ED (which is disjoint from PD, T, Q, and S) but material entities are always (partially) spatially co-localized with an amount of matter while sites and continuant fiat boundaries are never (partially) spatially co-localized with amounts of matter.

Given these facts it remains to prove the rigidity through time of all the leaves. This is done in (t_{db}82)-(t_{db}90).

t_{db}80 $\mathfrak{D}_b \nvdash \text{PAR}(x) \rightarrow (x ::\texttt{cnt} \vee x ::\texttt{occ})$
Proof. Consider an x such that $\text{AR}(x)$ and assume there is a time interval t. By (d_{db}1), $\texttt{EX}(x,t)$, and then, by (d_{db}27), $\text{PAR}(x)$. However x is neither an occurrent (that requires x to be an instance of PD or T) nor a continuant (that requires x to be an instance of ED, S, or Q). □

t_{db}81 $\mathfrak{D}_b \vdash x ::_t \texttt{treg} \wedge \texttt{EX}(x,v) \rightarrow x ::_v \texttt{treg}$
Proof. By (d_{db}7), $\text{T}(x)$, and then, from the hypotheses, $\text{T}(x) \wedge \texttt{EX}(x,v)$, i.e., by ($d_{db}$7), $x ::_v \texttt{treg}$. □

t_{db}82 $\mathfrak{D}_b \vdash x ::_t \texttt{tinst} \wedge \texttt{EX}(x,v) \rightarrow x ::_v \texttt{tinst}$
Proof. Directly from (t_{db}81) by observing that $\texttt{AT}(x)$ does not depend on time. □

t_{db}83 $\mathfrak{D}_b \vdash x ::_t \texttt{tint} \wedge \texttt{EX}(x,v) \rightarrow x ::_v \texttt{tint}$
Proof. Directly from (t_{db}81) by observing that $\texttt{AT}(x)$ does not depend on time. □

t_{db}84 $\mathfrak{D}_b \vdash x ::_t \texttt{pbnd} \wedge \texttt{EX}(x,v) \rightarrow x ::_v \texttt{pbnd}$
Proof. By (d_{db}5) and (a_d20) every process boundary exists at a single time. □

t_{db}85 $\mathfrak{D}_b \vdash x ::_t \texttt{proc} \wedge \texttt{EX}(x,v) \rightarrow x ::_v \texttt{proc}$
Proof. From $x ::_t \texttt{proc}$, by (d_{db}6), $\text{PD}(x) \wedge \neg(x ::_t \texttt{pbnd})$, i.e., $\text{PD}(x)$ and x has a temporal proper part or x does not have an atomic temporal location. All these conditions do not depend on time. The only temporary condition is $\texttt{PRE}(x,t)$ that however is implied by (d_{db}1) and the taxonomical axioms. □

t_{db}86 $\mathfrak{D}_b \vdash x ::_t \texttt{mten} \wedge \texttt{EX}(x,v) \rightarrow x ::_v \texttt{mten}$
Proof. By (d_{db}11) $\text{ED}(x)$. From $\text{ED}(x) \wedge \texttt{EX}(x,v)$, by ($d_{db}$1), ($t_{db}$104) and ($a_d$107), $\texttt{PRE}(x,v)$ and by (d_{db}11) $\exists u(\texttt{PRE}(x,u) \wedge \neg\texttt{AT}(u)) \wedge \forall u(\texttt{PRE}(x,u) \rightarrow \exists y s r(\text{M}(y) \wedge \texttt{SLC}(x,s,u) \wedge \texttt{SLC}(y,r,u) \wedge \text{P}_\text{d}(r,s)))$, thus $x ::_v \texttt{mten}$. □

t_{db}87 $\mathfrak{D}_b \vdash x ::_t \texttt{site}\cup\texttt{cfbnd} \wedge \texttt{EX}(x,v) \rightarrow x ::_v \texttt{site}\cup\texttt{cfbnd}$
Proof. By (d_{db}11) $\text{F}(x)$. From $\text{F}(x) \wedge \texttt{EX}(x,v)$, by ($d_{db}$1), ($t_{db}$104), ($a_d$101), ($a_d$90) and ($a_d$107), $\texttt{PRE}(x,v)$. From $\text{F}(x) \wedge \texttt{PRE}(x,v)$, by ($a_d$101) and ($t_d$22)

$\exists s(\texttt{SLC}(x,s,v))$. By ($d_{db}$12) $\forall u(\texttt{PRE}(x,u) \rightarrow \neg\exists y s r(\texttt{M}(y) \wedge \texttt{SLC}(x,s,u) \wedge \texttt{SLC}(y,r,u) \wedge \texttt{P}_\texttt{d}(r,s)))$, thus $x ::_v \texttt{site} \cup \texttt{cfbnd}$. □

t_{db}88 $\mathfrak{D}_b \vdash x ::_t \texttt{sreg} \wedge \texttt{EX}(x,v) \rightarrow x ::_v \texttt{sreg}$
Proof. By (d_{db}13) $\texttt{S}(x)$. From $\texttt{EX}(x,v)$, by (t_{db}102), $\texttt{T}(v)$. From $\texttt{S}(x) \wedge \texttt{T}(v)$, by ($a_d$103), ($a_d$99), ($a_d$100), and ($d_{db}$1), $\texttt{EX}(x,v)$. From $\texttt{S}(x) \wedge \texttt{EX}(x,v)$, by ($d_{db}$13), $x ::_v \texttt{sreg}$. □

t_{db}89 $\mathfrak{D}_b \vdash x ::_t \texttt{sdcnt} \wedge \texttt{EX}(x,v) \rightarrow x ::_v \texttt{sdcnt}$
Proof. By (d_{db}11) $\texttt{Q}(x)$. From $\texttt{Q}(x) \wedge \texttt{EX}(x,v)$, by ($d_{db}$1), ($t_{db}$104), and ($a_d$109), $\texttt{PRE}(x,v)$. By (d_{db}11) $\exists y(y ::_t \texttt{idcnt} \wedge \neg\texttt{S}(y) \wedge \texttt{DQT}(x,y))$. From $\texttt{DQT}(x,y)$, by (a_d22) the temporal extension of y is included in the one of x. Therefore, $\texttt{EX}(y,v)$. From $y ::_t \texttt{idcnt} \wedge \texttt{EX}(y,v)$, by ($t_{db}$130), ($t_{db}$87), ($t_{db}$134), ($d_{db}$14), and ($d_{db}$15), we have $y ::_v \texttt{idcnt}$. Thus $\texttt{Q}(x) \wedge \texttt{PRE}(x,v) \wedge \exists y(\neg\texttt{S}(y) \wedge y ::_v \texttt{idcnt} \wedge \texttt{DQT}(x,y))$. □

t_{db}90 $\mathfrak{D}_b \vdash x ::_t \texttt{gdcnt} \wedge \texttt{EX}(x,v) \rightarrow x ::_v \texttt{gdcnt}$
Proof. By (d_{db}18) and (d_{db}17), $\textsc{nped}(x)$. From $\textsc{nped}(x) \wedge \texttt{EX}(x,v)$, by ($d_{db}$1), ($t_{db}$104), and ($a_d$115), $\texttt{PRE}(x,v)$. By (d_{db}18) we have $\forall u(\texttt{PRE}(x,u) \rightarrow \exists y(\texttt{CONCR}(y,x,u)))$. □

6.2.11 Additional theorems

t_{db}91 $\mathfrak{D}_b \vdash (\textsc{ed}(x) \vee \textsc{pd}(x) \vee \texttt{Q}(x)) \rightarrow \textsc{par}(x)$
Proof. From $(\textsc{ed}(x) \vee \textsc{pd}(x) \vee \texttt{Q}(x))$, by ($t_d$9), $\exists t(\texttt{PRE}(x,t))$. It follows, by ($d_{db}$1), that $\exists t(\texttt{EX}(x,t))$ and then, by (d_{db}27), $\textsc{par}(x)$. □

t_{db}92 $\mathfrak{D}_b \vdash \exists u(\texttt{T}(u)) \rightarrow \forall x(\textsc{par}(x))$
Proof. By (t_{db}91) and (a_d89) we need to prove the theorem when x is a $\textsc{ab}$. By (d_{db}1) and (a_d4), $\texttt{T}(x) \rightarrow \texttt{EX}(x,x)$ and, by (d_{db}27), $\textsc{par}(x)$. From $\exists u(\texttt{T}(u)) \wedge \textsc{ab}(x) \wedge \neg\texttt{T}(x)$, by ($d_{db}$1), $\exists u(\texttt{EX}(x,u))$ and, by (d_{db}27), $\textsc{par}(x)$. □

t_{db}93 $\mathfrak{D}_b \vdash \neg\exists u(\texttt{T}(u)) \rightarrow \forall x(\textsc{uni}(x))$
Proof. From $\neg\exists u(\texttt{T}(u))$, by ($t_d$9), we have $\neg\exists x(\textsc{ed}(x) \vee \textsc{pd}(x) \vee \texttt{Q}(x))$. Given $\neg\exists u(\texttt{T}(u))$, by ($a_d$89), ($a_d$107), ($a_d$108), ($a_d$109), ($a_d$105), ($a_d$99), and ($a_d$100), it then remains only the case of $\textsc{ab}(x) \wedge \neg\texttt{T}(x)$. But from $\neg\exists u(\texttt{T}(u))$, by ($d_{db}$1), $\neg\exists u(\texttt{EX}(x,u))$ and then, by (d_{db}27) and (d_{db}28), $\textsc{uni}(x)$. □

t_{db}94 $\mathfrak{D}_b \vdash \textsc{pd}(x) \leftrightarrow (\exists t(x ::_t \texttt{proc}) \vee \exists t(x ::_t \texttt{pbnd}))$
Proof. ($\rightarrow$) First note that, by (t_d9), $\exists t(\texttt{PRE}(x,t))$. Assume $\texttt{PRE}(x,t)$. By (d_{db}5) and (d_{db}6), if $\exists y(\texttt{tmPP}(x,y)) \wedge \texttt{TLC}(x,t) \wedge \texttt{AT}(t)$ then $x ::_t \texttt{pbnd}$, otherwise $x ::_t \texttt{proc}$. ($\leftarrow$) Directly from (d_{db}5) and (d_{db}6). □

t_{db}95 $\mathfrak{D}_b \nvdash (\texttt{M}(x) \vee \textsc{pob}(x)) \rightarrow \exists t(x ::_t \texttt{cnt})$
Proof. DOLCE allows instantaneous amount of matters and instantaneous physical objects, See the counterexample of (t_{db}9), (t_{db}24), (t_{db}26). □

t_{db}96 $\mathfrak{D}_b \vdash (\texttt{M}(x) \vee \textsc{pob}(x)) \wedge \exists t(\texttt{TLC}(x,t) \wedge \neg\texttt{AT}(t)) \rightarrow \exists t(x ::_t \texttt{mten})$
Proof. (1) When x is an amount of matter the thesis follows from the fact that $\texttt{M}$ is a subclass of $\textsc{ped}$, (d_{db}11), (a_d4), (a_d12), (a_d24), (a_d29), and (a_d30), i.e., amount of matters, as well as all physical endurants, are always spatially localized. (2)

When x is a non-agentive physical object (NAPO) then, by (a_d73), it is always constituted by an amount of matter. The thesis follows from the case (1) and (a_d88). (3) When x is an agentive physical object (APO) then, by (a_d72), it is always constituted by a NAPO. The thesis follows from the case (2) and (a_d88). □

t_{db}97 $\mathfrak{D}_b \nvdash \exists t(x ::_t \texttt{mten}) \rightarrow \text{PED}(x)$
Proof. In DOLCE NPEDs can be partially spatially co-localized with an amount of matter during their whole temporal extension. See the counterexample of (t_{db}42), (t_{db}45), (t_{db}56), (t_{db}61). □

t_{db}98 $\mathfrak{D}_b \nvdash \text{F}(x) \rightarrow \exists t(x ::_t \texttt{cnt})$
Proof. In DOLCE Fs can be partially spatially co-localized with an amount of matter during (a part of) their temporal extension. See the counterexample on page 141. □

t_{db}99 $\mathfrak{D}_b \nvdash \text{Q}(x) \rightarrow \exists t(x ::_t \texttt{cnt})$
Proof. In DOLCE qualities can inhere in NPEDs and can inhere in instantaneous PEDs. See the counterexample of (t_{db}98). □

t_{db}100 $\mathfrak{D}_b \nvdash \text{NPED}(x) \rightarrow \exists t(x ::_t \texttt{cnt})$
Proof. NPEDs are not necessarily concretized (during their whole temporal extensions) in the sense defined in (d_{db}17), they do not necessarily EXD-depend on qualities. See the non-physical endurant e_2 in the counterexample of (t_{db}97). □

t_{db}101 $\mathfrak{D}_b \vdash x ::_t \texttt{occ} \rightarrow \texttt{EX}(x,t)$
Proof. For processes and process boundaries from (d_{db}1), (d_{db}5), (d_{db}6), (d_d5), and (a_d4). For temporal regions from (d_{db}1) and (a_d4). □

t_{db}102 $\mathfrak{D}_b \vdash \texttt{EX}(x,t) \rightarrow \text{T}(t)$
Proof. If $\text{AB}(x)$ directly from (d_{db}1), (a_d105), (a_d99), and (a_d100). Otherwise, from (d_{db}1) we have $\texttt{PRE}(x,t)$ and the thesis follows from (t_d5). □

t_{db}103 $\mathfrak{D}_b \vdash \texttt{EX}(x,t) \wedge \text{P}_\text{d}(u,t) \rightarrow \texttt{EX}(x,u)$
Proof. If $\texttt{PRE}(x,t)$ then, by (t_d6) $\texttt{PRE}(x,u)$ and then $\texttt{EX}(x,u)$. If $\text{T}(x) \wedge \text{P}_\text{d}(t,x)$, then from $\text{P}_\text{d}(u,t) \wedge \text{P}_\text{d}(t,x)$, by ($a_d$6), $\text{P}_\text{d}(u,x)$, i.e., $\text{T}(x) \wedge \text{P}_\text{d}(u,x)$. If $\text{AB}(x) \wedge \neg\text{T}(x) \wedge \text{T}(t)$ then from $\text{P}_\text{d}(u,t) \wedge \text{T}(t)$, by ($a_d$2), $\text{T}(u)$, i.e., $\text{AB}(x) \wedge \neg\text{T}(x) \wedge \text{T}(u)$. □

t_{db}104 $\mathfrak{D}_b \vdash \neg\text{AB}(x) \rightarrow (\texttt{EX}(x,t) \leftrightarrow \texttt{PRE}(x,t))$
Proof. Directly from (d_{db}1), (a_d105), (a_d99), and (a_d100). □

t_{db}105 $\mathfrak{D}_b \vdash x ::_t \texttt{cnt} \rightarrow \texttt{EX}(x,t)$
Proof. If $x ::_t \texttt{mten}$, $x ::_t \texttt{site} \cup \texttt{cfbnd}$, $x ::_t \texttt{sdcnt}$, or $x ::_t \texttt{sdcnt}$ we have $\texttt{PRE}(x,t)$ and then, by (d_{db}1) $\texttt{EX}(x,t)$. If $x ::_t \texttt{sreg}$ then directly $\texttt{EX}(x,t)$. □

t_{db}106 $\mathfrak{D}_b \vdash x ::_t \texttt{cnt} \wedge \neg(x ::_t \texttt{sreg}) \rightarrow \texttt{PRE}(x,t)$
Proof. See the proof of (t_{db}105). □

t_{db}107 $\mathfrak{D}_b \vdash x ::_t \texttt{cnt} \wedge \text{P}_\text{d}(u,t) \rightarrow x ::_u \texttt{cnt}$
Proof. By (t_{db}86), (t_{db}87), (t_{db}88), (t_{db}89), (t_{db}90), we have that cnt is rigid, i.e., $\forall v(\texttt{EX}(x,v) \rightarrow x ::_t \texttt{cnt})$. By ($t_{db}$105) we have $\texttt{EX}(x,t)$ and from $\texttt{EX}(x,t) \wedge \text{P}_\text{d}(u,t)$, by ($t_{db}$103) $\texttt{EX}(u,t)$ and, by the rigidity of cnt, $\rightarrow x ::_u \texttt{cnt}$. □

6.3 Analysis

We begin this analysis by considering how the entities in the domain of DOLCE are classified from the point of view of BFO. (t_{db}91) shows that endurants, perdurants, and qualities are all BFO particulars (in the sense of the BFO notion of particular, PAR). The case of abstracts is slightly more complex. (t_{db}92) shows that if there exists at least a time interval then all the abstracts are BFO particulars too. This is due to the definition of EX (d_{db}1) that assures that abstracts exist at every time. However, while in DOLCE the existence of endurants, perdurants, and qualities requires them to have a temporal location (and then to be present in time), see (a_d19) and (t_d9), this does not hold for abstracts, i.e., it is possible to have models of DOLCE that contain only abstracts and no time intervals. In this case we would not have BFO particulars, only universals (t_{db}93).

(t_d9) seems to confirm both (M1) and (M2), i.e., the general idea that DOLCE particulars roughly correspond to BFO particulars and that all the particulars of DOLCE are 'imported' into the ones of BFO. However, (t_{db}93) introduces some doubts about the nature of abstracts. The definition of EX (d_{db}1), starting from the fact that temporal and spatial regions are particulars in BFO, extended this idea to all regions (and to all abstracts), i.e., there is a presupposition that all the regions have a uniform nature, they are all particulars in this case (even though, in BFO, temporal and spatiotemporal regions are occurrents while spatial regions are continuants). However, other options are possible. One could consider, for instance, the following two variants of (d_{db}1):

- $\mathtt{EX}(x,t) \coloneqq \mathtt{PRE}(x,t) \vee (\mathtt{T}(x) \wedge \mathtt{P_d}(t,x)) \vee (\mathtt{S}(x) \wedge \mathtt{T}(t))$
- $\mathtt{EX}(x,t) \coloneqq \mathtt{PRE}(x,t)$

In the first case time intervals and space regions become BFO particulars as before (matching the fact that temporal and spatial regions are particulars in BFO) but, according to (d_{db}27) and (d_{db}28), the rest of abstracts would be included as universals. This possibility is interesting because the examples for DOLCE (spaces of) regions are colors, weights, shapes, etc., while, in BFO, *being red*, *being blue*, *being round*, etc., are universals. In this perspective, one can also presuppose that the $\mathtt{P_d}$ relation holding between regions (in a given space) represents a sort of (intensional) ISA relation, while tQL—or, maybe better, the composition of DQT and tQL—represents instantiation (at a time). An option that deserves a detailed analysis even though it introduces differentiation between time intervals and space regions on the one side, and the remaining regions on the other side.

The second option excludes all abstracts, including time intervals and space regions, from particulars. Even in this case one could see $\mathtt{P_d}$ as a sort of ISA relation and TLC and SLC as sorts of instantiation relations (here note that the (a_d23) and (a_d79) go in the direction of considering TLC and SLC as compositions of DQT and tQL as suggested in the previous case also).

In both these cases, the fact that (some) regions become universals is grounded on (d_{db}27) and (d_{db}28). The latter try to match (a_b1) and (a_b2), i.e., the fact that the entities in the domain of BFO are partitioned into universals and particulars. One could, however, decide to split (a_b1) and assume that regions are neither BFO particulars nor BFO universals. Basically, they would form a kind of entity that BFO does not consider. Also this option deserves a deeper analysis.

Moving to finer correspondences, (t_{db}94) shows that perdurants are the union of processes and process boundaries. However, (t_{db}7), (t_{db}9), (t_{db}11), (t_{db}24), and (t_{db}26) make evident that the distinction between processes and processes boundaries behaves quite differently from the original one in BFO. BFO presupposes that processes and process boundaries have a different temporal nature, and constrains their relationship, which is quite complex, via a set of axioms. Vice versa (d_{db}9) and (d_{db}10)—together with (d_{db}6) and (d_{db}7)—reduce this difference to atomicity and to the existence of bigger, with respect to temporal proper part, perdurants. Looking at the counterexamples used in the proof of the previous theorems, this different behavior about processes and process boundaries seems mainly due to the poor approximation of the notions of temporal instant and temporal interval provided by (d_{db}9) and (d_{db}10) together with the low commitment of DOLCE on the existence of perdurants. For instance, in DOLCE it is possible to have three temporally co-localized perdurants p_1, p_2, and p_{12} all with atomic temporal locations and such that the latter is the sum of the firsts, i.e., $\texttt{SUM}_\texttt{d}(p_{12}, p_1, p_2)$. There can also be two additional perdurants p_3 and p_{123} such that the temporal location of p_3 does not overlap the one of p_1 and also $\texttt{SUM}_\texttt{d}(p_{123}, p_{12}, p_3)$ holds. In this scenario, according to (d_{db}6) and (d_{db}7), p_1 is a process because it has an atomic temporal location but it is not a temporal proper part of anything, p_1 is part of p_{12}, but p_{12} is a process boundary (against (t_{db}7) and (t_{db}11)) because it has an atomic temporal location and it is a proper temporal part of p_{123}. For another example, in DOLCE it is possible to have perdurants that are not proper temporal part of any other perdurant and such that all their parts are temporally co-localized with them (possibly with an atomic temporal location) and in their turn are not proper temporal part of any other perdurant (e.g., discard p_3 and p_{123} from the previous example). According to (d_{db}6) and (d_{db}7), these perdurants as well as all their parts are classified as processes against (t_{db}9), (t_{db}24), and (t_{db}26). These examples may look 'exotic', yet it is not easy to rule out them. One could get rid of some of these counterexamples by requiring that processes have a non-atomic temporal location. In this case, we need to remove (t_{db}94) also because it is possible to have perdurants with an atomic temporal location that however are not temporal proper part of any other perdurant.

The different 'temporal behavior' of DOLCE perdurants vs. BFO occurrents is highlighted also by (t_{db}32), which is relative to (a_b32). BFO identifies occurrent parthood between processes or process boundaries with spatiotemporal inclusion, it follows that different processes/process boundaries cannot be spatiotemporally co-localized. DOLCE makes the opposite choice and allows for spatiotemporally co-localized perdurants. A similar situation arises between DOLCE endurants and BFO continuants as highlighted by (t_{db}51) and (t_{db}41). In DOLCE, an amount of matter constituting a statue at a given time t is different from the statue and it is not a $\texttt{tP}_\texttt{d}$-part of the statue at t (and vice versa) even though the two endurants are spatially co-localized at t (and possibly during their whole life), see (t_d15) and (a_d88). Similarly, in DOLCE, a sum of bricks is different from a wall even though they can mereologically coincide at a given time. These differences are among the most important we found. In DOLCE, space and time are not fundamental for the identity of entities. This gives room for a stratification of co-localized entities, which are further distinguished on the basis of their 'modal' properties. In comparison, BFO embraces a 'reductionist' perspective by assuming that a given spatial or spatiotemporal region cannot be the location of different continuants

or occurrents.

While endurants and occurrents, as well as $\mathtt{tP_d}$ and $\mathtt{cP}$, seem intuitively to correspond, (t_{db}41) shows a relevant difference.[9] According to (d_{db}20) and the definitions of the subclasses of $\mathtt{cnt}$, for independent continuants that are not spatial regions, $\mathtt{cP}$ reduces to $\mathtt{tP_d}$. However, for independent continuants that are not object aggregates, the original $\mathtt{cP}$ is antisymmetric, but is not so the relation defined by (d_{db}20), that in this case it is equivalent to $\mathtt{tP_d}$. As we have seen, this fact has an important impact on the endurant/continuant entities accepted by the two theories. One could recover the antisymmetry by changing the definition of $\mathtt{cP}$. This amounts to break the correspondence between $\mathtt{tP_d}$ and $\mathtt{cP}$ for independent continuants by 'injecting' antisymmetry into (d_{db}20), e.g.,

– $\mathtt{cP}(x,y,t) \coloneqq x ::_t \mathtt{cnt} \wedge y ::_t \mathtt{cnt} \wedge ((\mathtt{tP_d}(x,y,t) \wedge (\neg \mathtt{tP_d}(y,x,t) \vee x = y)) \vee \mathtt{P_d}(x,y) \vee \exists zu(\mathtt{DQT}(x,z) \wedge \mathtt{DQT}(y,u) \wedge \mathtt{tP_d}(z,u,t)))$

This new mapping has the advantage of (i) recovering the antisymmetry as in the original BFO; and of (ii) making explicit one difference between $\mathtt{cP}$ and $\mathtt{tP_d}$. However, we can have endurants that in DOLCE are linked by $\mathtt{tP_d}$ that are still imported under $\mathtt{idcnt}$ but are not linked by the newly defined $\mathtt{cP}$. For instance, in the previous example, the (sum of the) bricks and the wall would not be $\mathtt{cP}$-related. In these cases (a_b78) still fails. To solve this problem one could explicitly specialize the definition of $\mathtt{cP}$ to take into account these cases, e.g.,

– $\mathtt{cP}(x,y,t) \coloneqq x ::_t \mathtt{cnt} \wedge y ::_t \mathtt{cnt} \wedge ((\mathtt{tP_d}(x,y,t) \wedge (\neg \mathtt{tP_d}(y,x,t) \vee x = y)) \vee \exists sr(\mathtt{SLC}(x,s,t) \wedge \mathtt{SLC}(y,r,t) \wedge \mathtt{P_d}(s,r)) \vee \mathtt{P_d}(x,y) \vee \exists zu(\mathtt{DQT}(x,z) \wedge \mathtt{DQT}(y,u) \wedge \mathtt{tP_d}(z,u,t)))$

However, in this way the antisymmetry of $\mathtt{cP}$ is lost once more because the bricks and the wall would $\mathtt{cP}$-coincide at t even though they are different. One could then change the definition of $\mathtt{SREG}$ stating that the bricks (or the wall) have not spatial location at t, or that at t they have different spatial locations. Similarly in the case of the statue and the clay. However, this option seems too strong to pursue.[10]

These examples suggest to follow a different approach possibly more respectful of the original commitments of the two ontologies. Let us go back to the classical example of the statue and the clay, or to the one of the brick and the wall. There are two different endurants which are spatially coincident at least at a given time. One can think that (a_b20) and (a_b78) aim at ruling out these kinds of examples: it is not a matter of avoiding spatial coincidence or parthood between the brick and the wall, their goal, one could argue, is to rule out one of these two entities, i.e., to rule out from the domain of quantification, either the brick or the wall.[11] One can then adopt a strategy that 'filters out' some of the DOLCE endurants. Setting this filter is not trivial.

[9] There are other differences: (t_{db}37) shows that the $\mathtt{cP}$ (as defined in (d_{db}20)) does not satisfy the supplementation axiom, and (t_{db}38) that the existence of the product is not guaranteed. Note that the existence of products cannot be inferred for $\mathtt{oP}$ (t_{db}5) either.

[10] We have a similar problem with the class of occurrents as highlighted by (t_{db}32).

[11] Clearly, this is in contrast with (M2) and with the mappings that, vice versa, allow to import all the endurants of DOLCE into BFO.

For the case of the statue and the clay, one can rely on the fact that constitution is a sort of order relation in DOLCE. One can maintain the substratum (the clay) in the domain and discard the statue due to the fact that it is the substratum that constitutes the statue, not vice versa. However, in the case of the bricks and the wall we cannot rely on $\mathtt{tP_d}$ because the sum of the bricks is part of the wall and the wall is part of the sum of the bricks. One needs to find a different relation or some other principle to rule out one of these. But this is not all. Let us assume that we do not import the statue into BFO. Does it mean that BFO cannot talk about statues or that it relies on a different representation? After all, it classifies the clay under amount of matter as well as under statue. One would need to translate the DOLCE claims about the statue into BFO claims about the amount of clay (since it is the only element left in the domain of BFO). The consequences of this approach deserve more investigation. It is however important to note that this first analytical step allows us to understand the differences and to individuate the main problems that a mapping needs to address.

The analysis of more refined correspondences, like (t_{db}95), (t_{db}96), and (t_{db}97), shows that temporally extended amounts of matters/physical objects are material entities. Yet, the other direction does not hold: non-physical endurants could be very well be material entities. However, (t_{db}94), (t_{db}95), together with (t_{db}98), (t_{db}99), and (t_{db}100), show that there are endurants and qualities, all of which are particulars (see (t_{db}91)), that are neither continuants nor occurrents. This means that the partition of particulars into continuants and occurrents assumed by BFO is not preserved by the mapping, hence, some endurants and qualities are in a sort of ontological 'limbo'.

On the other hand, the imported entities do not necessarily cover all the categories of BFO. There are at least two reasons for that, one factual and one structural. (i) The factual reason is that some of the DOLCE categories may be empty. For instance, if there are no features, then `imen` would be empty. Consequently, (a_b6), which states that all universals are non-empty, is not preserved. (ii) The structural reason is that, according to the mappings, some categories (as defined in the mappings) are necessarily empty. This is the case of universals (when there is at least a time in the domain of quantification, see (t_{db}92)). One solution is to 'force' some types of entities in the ontological 'limbo' to be mapped into BFO universals. We already discussed the import of some regions into universals, similarly one could assume that specific subclasses of non-physical endurant (NPED) are concepts behaving like BFO universals. This approach has several consequences on the set of theorems.

One way to choose among the different mapping approaches is to look at what the mapping optimize. For instance, strict and restrictive mappings, carefully set to not allow exchanges of entities which would be in the ontological 'limbo' of the target ontology, might be preferred in the context of data transfer, they are safer in this context. Mappings focusing on ontological significance and the maximization of the number of imported entities are instead arguably better for the goals of comparing ontologies, highlighting core differences, and establishing general interoperability results.

7 The mapping from BFO to DOLCE

7.1 Mappings

In this section we introduce syntactic definitions of DOLCE notions in terms of BFO primitives. Analogously to the DOLCE to BFO direction of Sect. 6 and with similar motivations (see the analysis of Sect. 7.3), we present only a partial mapping covering a subset of the categories and primitives of DOLCE.

As discussed in Sect. 5.2, in DOLCE, the agentive and social dimensions play an important role in characterizing the subcategories of physical object (POB) and non-physical endurant (NPED) but those dimensions are not considered in BFO. Similarly, the subcategories of perdurant (PD) rely on the notions of homeomericity and cumulativity, but these notions are beyond the scope of BFO. We then rule out all these subcategories from the mappings. Furthermore, while it is not clear how the distinction between amounts of matter and physical objects can be made, for features we can rely on the BFO categories of site, continuant fiat boundary and fiat object. Thus, within the category of physical endurants (PED) we consider only the category of features (F).

We will see that physical endurants correspond to independent continuants which are not regions, see (d_{bd}3), while the only entities that can be mapped to non-physical endurants are generically dependent continuants, see (d_{bd}5). In DOLCE, arbitrary sums (AS) have necessarily a physical and a non-physical part, see (a_d41). To match this axiom, in BFO we should find continuants which are not spatial regions and are neither independent nor generically dependent continuants. The only option is that of specifically dependent continuants. However, following our guidelines in Sect. 5, these are mapped to DOLCE physical qualities as stated by (d_{bd}6). As a result, AS would be necessarily empty. For this reason, this category is not covered by the mapping.

In Sect. 6.3 we have seen that one of the main differences between DOLCE and BFO concerns the reduction of the parthood relations to spatial and spatiotemporal inclusions. In particular, in BFO there are no spatially co-localized distinct independent continuants (that are not object aggregates) and there are no spatiotemporally co-localized distinct occurrents. We then lack one of the main basis to build a constitution relation among endurants and perdurants. In principle, one could see the relation between a generic dependent continuant and the mereological sum of its carriers at a given time as a form of constitution. This solution is debatable and would model only a very limited notion of constitution, quite different from the general one in DOLCE. Thus, we refrain from introducing K in the mappings.

Concerning qualities, BFO accepts only qualities inhering in independent continuants. From the point of view of DOLCE (considering (d_{bd}3) below), this restricts the mapping to physical qualities only.

In the case of regions, we have just time intervals and space regions but to locate entities in time and space we need to rely only on TLC and SLC because temporal and spatial locations have no correspondent entities in BFO and the QL and tQL relations do not make sense in this setting. Alternatively, as previously discussed, one could import some BFO universals into DOLCE regions and assume that instance-of corresponds to (DQT composed with) tQL. In this cases, the ISA relation corresponds to the primitive P_d (that in DOLCE is defined on regions). Beside the mapping presented in this section,

this other alternative is also worth development. We leave it for future work.

Finally, the categories of *fact* and *set* that in the original taxonomy of DOLCE appear under abstract (AB) have not been explicitly taken into account in DOLCE-CL, we safely ignore them here.

Summing up, among the DOLCE notions, in the following we introduce:

1. syntactic definitions for the primitive relations: $\mathtt{P_d}$, $\mathtt{tP_d}$, TLC, SLC, PC, DQT, EXD; and
2. syntactic definitions for the categories: ED, PD, Q, AB, F, PED, NPED, PQ, R, TR, PR, T, and S.

Below we list these syntactic definitions together with a short informal description. A deeper analysis about these definitions and their impact on the preservation of the axioms of DOLCE is done in Sect. 7.3.

$\mathbf{d_{bd}1}$ $\mathrm{PD}(x) \coloneqq x :: \mathtt{proc} \vee x :: \mathtt{pbnd}$
Perdurants coincide with the disjunction of processes and process boundaries.

$\mathbf{d_{bd}2}$ $\mathrm{ED}(x) \coloneqq x :: \mathtt{cnt} \wedge \neg(x :: \mathtt{sreg}) \wedge \neg(x :: \mathtt{sdcnt})$
We rule out from endurants temporal and spatial regions that are mapped, respectively, to time intervals and space regions, see ($\mathrm{d_{bd}}8$) and ($\mathrm{d_{bd}}10$).

$\mathbf{d_{bd}3}$ $\mathrm{PED}(x) \coloneqq x :: \mathtt{idcnt} \wedge \neg(x :: \mathtt{sreg})$
Physical endurants are independent continuants that are not spatial regions.

$\mathbf{d_{bd}4}$ $\mathrm{F}(x) \coloneqq x :: \mathtt{site} \vee x :: \mathtt{cfbnd} \vee x :: \mathtt{fobj}$
Features are the union of sites, continuant fiat boundaries, and fiat objects.

$\mathbf{d_{bd}5}$ $\mathrm{NPED}(x) \coloneqq x :: \mathtt{gdcnt}$
Non-physical endurants coincide with generically dependent continuants.

$\mathbf{d_{bd}6}$ $\mathrm{PQ}(x) \coloneqq x :: \mathtt{sdcnt}$
Physical qualities coincide with specifically dependent continuants. Note that relational qualities and realizable entities like roles and dispositions are imported into physical qualities.

$\mathbf{d_{bd}7}$ $\mathrm{Q}(x) \coloneqq \mathrm{PQ}(x)$
Only physical qualities exist (BFO has only qualities of independent continuants)

$\mathbf{d_{bd}8}$ $\mathrm{T}(x) \coloneqq x :: \mathtt{treg}$
Time intervals coincide with temporal regions. Here one could consider a stronger mapping, i.e., assume that T corresponds to `tint`. However, following (M2) we try here to import all the spatial regions.

$\mathbf{d_{bd}9}$ $\mathrm{TR}(x) \coloneqq \mathrm{T}(x)$
Among temporal regions there are only time intervals.

$\mathbf{d_{bd}10}$ $\mathrm{S}(x) \coloneqq x :: \mathtt{sreg}$
Space regions coincide with spatial regions.

$\mathbf{d_{bd}11}$ $\mathrm{PR}(x) \coloneqq \mathrm{S}(x)$
Among physical regions there are only space regions.

$\mathbf{d_{bd}12}$ $\mathrm{R}(x) \coloneqq \mathrm{TR}(x) \vee \mathrm{PR}(x)$

$\mathbf{d_{bd}13}$ $\mathrm{AB}(x) \coloneqq \mathrm{R}(x)$

Among abstracts there are only time intervals and space regions.

$\mathbf{d_{bd}14}$ $\mathtt{TLC}(x,t) \coloneqq (\mathrm{PD}(x) \vee \mathrm{ED}(x) \vee \mathrm{Q}(x)) \wedge \mathrm{T}(t) \wedge \forall u(\mathtt{EX}(x,u) \leftrightarrow \mathtt{tmP}(u,t))$

The temporal location of x is the maximal time at which x exists. In BFO TREG is defined only on processes and process boundaries).

$\mathbf{d_{bd}15}$ $\mathtt{P_d}(x,y) \coloneqq (((\mathrm{PD}(x) \wedge \mathrm{PD}(y)) \vee (\mathrm{T}(x) \wedge \mathrm{T}(y))) \wedge \mathtt{oP}(x,y)) \vee (\mathrm{S}(x) \wedge \mathrm{S}(y) \wedge \forall t(\mathtt{EX}(x,t) \rightarrow \mathtt{cP}(x,y,t)))$

For perdurant and time intervals $\mathtt{P_d}$ coincides with oP while for space regions it reduces to constant parthood, i.e., x is a temporary part of y during its whole existence (note that in BFO, all the spatial regions exist at least at a time).

$\mathbf{d_{bd}16}$ $\mathtt{tP_d}(x,y,t) \coloneqq \mathrm{ED}(x) \wedge \mathrm{ED}(y) \wedge \mathtt{cP}(x,y,t)$

For endurants $\mathtt{tP_d}$ coincides with cP.

$\mathbf{d_{bd}17}$ $\mathtt{PC}(x,y,t) \coloneqq \mathrm{ED}(x) \wedge \mathrm{PD}(y) \wedge \exists z(\mathtt{tmP}(y,z) \wedge \mathtt{PTC}(x,z,t))$

PTC is not defined on process boundaries while DOLCE does not impose any constraint on the kind of the process involved in PC. The existential quantification on z aims to mitigate this difference.

$\mathbf{d_{bd}18}$ $\mathtt{DQT}(x,y) \coloneqq \mathtt{INH}(x,y)$

$\mathbf{d_{bd}19}$ $\mathtt{EXD}(x,y,t) \coloneqq (\mathtt{SDEP}(x,y) \wedge \mathtt{EX}(x,t)) \vee \mathtt{GDEP}(x,y,t) \vee \mathtt{PTC}(x,y,t) \vee \mathtt{PTC}(y,x,t)$

EXD is the generalization of SDEP, GDEP, and PTC (PTC is a sort of mutual dependence).

$\mathbf{d_{bd}20}$ $\mathtt{SLC}(x,s,t) \coloneqq (x ::_t \mathtt{idcnt} \wedge \neg(x ::_t \mathtt{sreg}) \wedge \mathtt{SREG}(x,s,t)) \vee (x ::_t \mathtt{gdcnt} \wedge \exists y(\mathtt{GDEP}(x,y,t) \wedge \forall z(\mathtt{GDEP}(x,z,t) \rightarrow \mathtt{cP}(z,y,t)) \wedge \mathtt{SREG}(y,s,t))) \vee (\mathrm{Q}(x) \wedge \exists y(\mathtt{INH}(x,y) \wedge \forall z(\mathtt{INH}(x,z) \rightarrow \mathtt{cP}(z,y,t)) \wedge \mathtt{SREG}(y,s,t))) \vee (\mathrm{PD}(x) \wedge \exists y(\mathtt{STREG}(x,y) \wedge \mathtt{SPROJ}(y,s,t)))$

For independent continuants that are not spatial regions, SLC coincides with SREG. The spatial location, at t, of a generically dependent continuant x is the spatial region of the maximal entity (if it exist) on which x generically depends on at t. The spatial location of a quality x at t is the spatial region of the maximal (at t) entity (if it exist) in which x inheres. The spatial location of a perdurant is the spatial projection of its spatiotemporal location. (Note that in BFO, no axiom guarantees that a specifically dependent continuant x inheres in a unique continuant, therefore for qualities we define the spatial location only when there is a 'maximal continuant' in which x inheres in. Similarly for generic dependent continuants.)

One could think that these mappings are too restrictive: in DOLCE physical endurants do not include only independent continuants, non-physical endurants do not include only generically dependent continuants (a quite special kind of entities), etc. This may suggest to include $x :: \mathtt{idcnt} \wedge \neg(x :: \mathtt{sreg}) \rightarrow \mathrm{PED}(x)$ as well as $x :: \mathtt{gdcnt} \rightarrow \mathrm{NPED}(x)$, but not the converse formulas. We need however to remember that here we are considering the mapping from BFO to DOLCE. The starting point is the domain

of BFO, i.e., we need first of all to try to classify the entities *in the domain of* BFO in terms of the categories of DOLCE. By looking at the mappings it is easy to see that all the particulars[12] of BFO are classified in terms of DOLCE categories except spatiotemporal regions;[13] (*i*) independent continuants which are not spatial regions are mapped to physical endurants; (*ii*) spatial regions to space regions; (*iii*) generically dependent continuants to non-physical endurants; (*iv*) specifically dependent continuants to DOLCE qualities; (*v*) process and process boundaries to perdurants; and (*vi*) temporal regions to time intervals. The fact that DOLCE intuitively accepts, for instance, additional physical or non-physical endurants shows that, in general, DOLCE has a domain larger than the domain of BFO. Our methodological choice (M2) does not allow the mappings to 'enrich' the domain of BFO with new entities. However, by introducing the mapping at the semantic level, i.e., in this case, as an operator translating BFO-structures into a DOLCE-structure, one could enrich the original domain of BFO by set-theoretically building new entities. For instance, this is the strategy followed in the literature to map theories of time based on points to theories based on intervals.[14] The theory based on intervals does not admit points, but points can be built as set of intervals staying in a given relation that is definable in the theory of intervals. Analogously, one can think of building some entities, e.g., statues as opposed to amount of clay, starting from the amounts of clay and the specific kinds of universals they instantiate. The investigation of 'semantic mappings' is not discussed here.

7.2 Check of the preservation of the original DOLCE axioms

As discussed in Sect. 5.2 (and made explicit in Sect. 7.1) some DOLCE-axioms involve relations or categories that have not been defined in the mappings. In what follows, when possible and relevant we consider approximations of these axioms as expressible in the BFO language, otherwise we set them aside.

Furthermore (a_b6) states that all the universals are non-empty. From an applicative perspective, this is a strong constraint because it forces the user to model even entities which might not be relevant to the application (and perhaps time consuming to analyze). Note that, once (a_b6) is removed, the existential constraints in $\mathfrak{B}$ are compatible with the instantiation of only some categories (see the counterexample used below). From a technical perspective, (a_b6) requires very large models that make the identification and management of counterexamples very difficult even using dedicated software. For these reasons, in this section we do not consider (a_b6).

In the following, $\mathfrak{B}_d$ indicates the 'extension' of $\mathfrak{B} \setminus \{(a_b6)\}$ with the mappings introduced in Sect. 7.1 together with the syntactic definitions introduced in Sect. 3, i.e., $\mathfrak{B}_d = \{\mathfrak{B} \setminus \{(a_b6)\}\} \cup \{(d_{ab}1)\text{-}(d_{ab}20)\} \cup \{(d_a1)\text{-}(d_a9)\}$.

Several proofs has been verified by using theorems provers, they are reported in Sect. 11.5.

[12] We already discussed the possibility to import universals as regions or non-physical endurants, see Sect. 6.3.

[13] We already discussed the fact that spatiotemporal regions seem superfluous, see Sect. 6.1. However, it is easy to modify the mappings to import spatiotemporal regions under DOLCE regions.

[14] See van Benthem, J., *The Logic of Time*, Springer, 2nd ed., 1991.

7.2.1 Mereology

t_{bd}1 $\mathfrak{B}_d \vdash \mathtt{P_d}(x,y) \rightarrow (\mathrm{AB}(x) \wedge \mathrm{AB}(y)) \vee (\mathrm{PD}(x) \wedge \mathrm{PD}(y))$ (a_d1)
Proof. See proof generated by theorem provers. □

t_{bd}2 $\mathfrak{B}_d \vdash \mathtt{P_d}(x,y) \rightarrow (\mathrm{T}(x) \leftrightarrow \mathrm{T}(y))$ (a_d2)
Proof. See proof generated by theorem provers. □

t_{bd}3 $\mathfrak{B}_d \vdash \mathtt{P_d}(x,y) \rightarrow (\mathrm{S}(x) \leftrightarrow \mathrm{S}(y))$ (a_d3)
Proof. See proof generated by theorem provers. □

t_{bd}4 $\mathfrak{B}_d \vdash (\mathrm{AB}(x) \vee \mathrm{PD}(x)) \rightarrow \mathtt{P_d}(x,x)$ (a_d4)
Proof. See proof generated by theorem provers. □

t_{bd}5 $\mathfrak{B}_d \vdash \mathtt{P_d}(x,y) \wedge \mathtt{P_d}(y,x) \rightarrow x = y$ (a_d5)
Proof. For PDs and Ts it follows directly from (d_{bd}1), (d_{bd}8), (a_b157), and (a_b39). For Ss, first observe that, by (a_b7), $\exists t(\mathtt{EX}(x,t) \wedge \mathtt{EX}(y,t))$ and then, by (d_{bd}15), $\exists t(\mathtt{cP}(x,y,t) \wedge \mathtt{cP}(y,x,t))$. The thesis follows from (a_b20), (a_b159), (a_b158), and (a_b145). □

t_{bd}6 $\mathfrak{B}_d \vdash \mathtt{P_d}(x,y) \wedge \mathtt{P_d}(y,z) \rightarrow \mathtt{P_d}(x,z)$ (a_d6)
Proof. See proof generated by theorem provers. □

t_{bd}7 $\mathfrak{B}_d \nvdash (\mathrm{AB}(x) \vee \mathrm{PD}(x)) \wedge \neg \mathtt{P_d}(x,y) \rightarrow \exists z(\mathtt{P_d}(z,x) \wedge \neg \mathtt{O_d}(z,y))$ (a_d7)
Proof. See the counterexample on page 217 where $\mathrm{PD}(\mathtt{pd1})$, $\mathrm{PD}(\mathtt{p1})$, $\mathtt{pd1} \neq \mathtt{p1}$, and $\mathtt{P_d}(\mathtt{pd1},\mathtt{p1})$ but all the parts of p1 (pd1 and p1) overlap pb1. □

7.2.2 Direct Quality

t_{bd}8 $\mathfrak{B}_d \vdash \mathtt{DQT}(x,y) \rightarrow \mathrm{PQ}(x) \wedge \mathrm{PED}(y)$ $\sim$ (a_d8)
(The original axiom requires the distinction between kinds of qualities.)
Proof. See proof generated by theorem provers. □

t_{bd}9 $\mathfrak{B}_d \nvdash \mathtt{DQT}(x,y) \wedge \mathtt{DQT}(x,z) \rightarrow y = z$ (a_d9)
Proof. No axiom in BFO guarantees that a specifically dependent continuant can specifically depend only on a single (independent) continuant. For instance, consider a relational quality x. For the axiom [dbp-1] in BFO-CL (not reported in Sect. 2), $\exists yz(y \neq z \wedge \mathtt{INH}(x,y) \wedge \mathtt{SDEP}(x,z))$. It is enough to suppose $z :: \mathtt{idcnt} \wedge \neg(z :: \mathtt{sreg})$, i.e., $\mathtt{INH}(x,z)$. □

t_{bd}10 $\mathfrak{B}_d \vdash \mathrm{Q}(x) \rightarrow \exists y(\mathtt{DQT}(x,y))$ (a_d10)
Proof. Directly from (a_b104), (d_{bd}7), and (d_b10). □

t_{bd}11 $\mathfrak{B}_d \nvdash \mathrm{PD}(x) \leftrightarrow \exists y(\mathtt{DQT}(y,x) \wedge \mathrm{TL}(y))$ (a_d11)
Proof. $\mathrm{PD}(x)$, by (d_{bd}1), is equivalent to $x :: \mathtt{proc} \vee x :: \mathtt{pbnd}$ while INH requires x to be an indipendent continuant. These categories are however disjoint by (a_b141), (a_b156), and (a_b157). □

t_{bd}12 $\mathfrak{B}_d \nvdash \mathrm{PED}(x) \leftrightarrow \exists y(\mathtt{DQT}(y,x) \wedge \mathrm{PQ}(y))$ $\sim$ (a_d12)
(The original axiom requires $\mathrm{SL}(y)$ that has not been defined.)
Proof. ($\leftarrow$) Directly from (d_b10), (d_{bd}3), (d_{bd}18), and (d_{bd}6). ($\rightarrow$) Does not hold because it is possible to have independent continuants (that are not spatial regions) that do not SDEP-depend. See the counterexample on page 217 where $\mathtt{c1} ::_{\mathtt{t1}} \mathtt{site}$ but SDEP is 'empty'. □

7.2.3 Immediate Quale

QL has not bee defined in terms of BFO.

7.2.4 Temporal Location and Present At

t_{bd}13 $\mathfrak{B}_d \vdash \mathtt{TLC}(x,t) \rightarrow (\mathrm{ED}(x) \vee \mathrm{PD}(x) \vee \mathrm{Q}(x)) \wedge \mathrm{T}(t)$ (a_d18)
Proof. See proof generated by theorem provers. □

t_{bd}14 $\mathfrak{B}_d \nvdash (\mathrm{ED}(x) \vee \mathrm{PD}(x) \vee \mathrm{Q}(x)) \rightarrow \exists t(\mathtt{TLC}(x,t))$ (a_d19)
Proof. See the counterexampleon page 217 where $\mathtt{c2}::_{\mathtt{t1}}\mathtt{site}$ and where we have only three temporal regions, namely, t1, t2, and t12. $\neg\mathtt{TLC}(\mathtt{c2},\mathtt{t1})$ because $\mathtt{EX}(\mathtt{c2},\mathtt{t2})$ and $\neg\mathtt{tmP}(\mathtt{t2},\mathtt{t1})$. $\neg\mathtt{TLC}(\mathtt{c2},\mathtt{t2})$ because $\mathtt{EX}(\mathtt{c2},\mathtt{t1})$ and $\neg\mathtt{tmP}(\mathtt{t1},\mathtt{t2})$. $\neg\mathtt{TLC}(\mathtt{c2},\mathtt{t12})$ because $\mathtt{tmP}(\mathtt{t12},\mathtt{t12})$ and $\neg\mathtt{EX}(\mathtt{c2},\mathtt{t12})$. □

t_{bd}15 $\mathfrak{B}_d \vdash \mathtt{TLC}(x,t) \wedge \mathtt{TLC}(x,u) \rightarrow t = u$ (a_d20)
Proof. See proof generated by theorem provers. □

t_{bd}16 $\mathfrak{B}_d \vdash \mathtt{P}_d(x,y) \wedge \mathtt{TLC}(x,t) \wedge \mathtt{TLC}(y,u) \rightarrow \mathtt{P}_d(t,u)$ (a_d21)
Proof. See proof generated by theorem provers. □

t_{bd}17 $\mathfrak{B}_d \vdash \mathtt{DQT}(x,y) \wedge \mathtt{TLC}(x,t) \wedge \mathtt{TLC}(y,u) \rightarrow \mathtt{P}_d(t,u)$ (a_d22)
Proof. See proof generated by theorem provers. □

7.2.5 Temporary Mereology

t_{bd}18 $\mathfrak{B}_d \vdash \mathtt{tP}_d(x,y,t) \rightarrow \mathrm{ED}(x) \wedge \mathrm{ED}(y) \wedge \mathrm{T}(t)$ (a_d32)
Proof. See proof generated by theorem provers. □

t_{bd}19 $\mathfrak{B}_d \nvdash \mathtt{tP}_d(x,y,t) \rightarrow \mathtt{PRE}(x,t) \wedge \mathtt{PRE}(y,t)$ (a_d33)
Proof. See the counterexample on page 217 where $\mathtt{c2}::_{\mathtt{t1}}\mathtt{site}$, $\mathtt{cP}(\mathtt{c2},\mathtt{c2},\mathtt{t1})$ but c2 does not have a temporal location (in the sense of TLC) on the basis of which PRE is defined. □

t_{bd}20 $\mathfrak{B}_d \vdash \mathtt{tP}_d(x,y,t) \wedge \mathtt{P}_d(u,t) \rightarrow \mathtt{tP}_d(x,y,u)$ (a_d34)
Proof. Directly from (a_b17) (by observing that for temporal regions tmP and oP are equivalent). □

t_{bd}21 $\mathfrak{B}_d \nvdash \mathtt{tP}_d(x,y,t) \wedge \mathtt{tP}_d(x,y,u) \wedge \mathtt{SUM}_d(s,t,u) \rightarrow \mathtt{tP}_d(x,y,s)$ (a_d35)
Proof. See the counterexample on page 217 where $\mathtt{cP}(\mathtt{c2},\mathtt{c2},\mathtt{t1})$, $\mathtt{cP}(\mathtt{c2},\mathtt{c2},\mathtt{t2})$ and $\mathtt{SUM}_d(\mathtt{t12},\mathtt{t1},\mathtt{t2})$ but $\neg\mathtt{cP}(\mathtt{c2},\mathtt{c2},\mathtt{t12})$. □

t_{bd}22 $\mathfrak{B}_d \nvdash \mathrm{ED}(x) \wedge \mathtt{PRE}(x,t) \rightarrow \mathtt{tP}_d(x,x,t)$ (a_d36)
Proof. See the counterexample on page 219 where $\mathrm{ED}(\mathtt{gd1})$ (gd1 is a generically dependent continuant), $\mathtt{PRE}(\mathtt{gd1},\mathtt{t1})$ (because we have $\mathtt{TLC}(\mathtt{gd1},\mathtt{t12})$ and $\mathtt{P}_d(\mathtt{t1},\mathtt{t12})$) but $\neg\mathtt{tP}_d(\mathtt{gd1},\mathtt{gd1},\mathtt{t1})$ (because $\neg\mathtt{cP}(\mathtt{gd1},\mathtt{gd1},\mathtt{t1})$). □

t_{bd}23 $\mathfrak{B}_d \vdash \mathtt{tP}_d(x,y,t) \wedge \mathtt{tP}_d(y,z,t) \rightarrow \mathtt{tP}_d(x,z,t)$ (a_d37)
Proof. See proof generated by theorem provers. □

t_{bd}24 $\mathfrak{B}_d \nvdash \mathrm{ED}(x) \wedge \mathrm{ED}(y) \wedge \mathtt{PRE}(x,t) \wedge \mathtt{PRE}(y,t) \wedge \neg\mathtt{tP}_d(x,y,t) \rightarrow \exists z(\mathtt{tP}_d(z,x,t) \wedge \neg\mathtt{tO}_d(z,y,t))$ (a_d38)
Proof. See the counterexample on page 219 where $\mathrm{ED}(\mathtt{gd1})$ (gd1 is a generically dependent continuant), $\mathtt{PRE}(\mathtt{gd1},\mathtt{t1})$ (because we have $\mathtt{TLC}(\mathtt{gd1},\mathtt{t12})$

and $\mathtt{P_d}(\mathtt{t1},\mathtt{t12})$), $\mathrm{ED}(\mathtt{c1})$ ($\mathtt{c1}$ is a site), $\mathrm{PRE}(\mathtt{c1},\mathtt{t1})$ (because $\mathrm{TLC}(\mathtt{c1},\mathtt{t1})$ and $\mathtt{P_d}(\mathtt{t1},\mathtt{t1})$), and, finally, $\neg\mathtt{tP_d}(\mathtt{gd1},\mathtt{c1},\mathtt{t1})$ (because $\neg\mathtt{cP}(\mathtt{gd1},\mathtt{c1},\mathtt{t1})$) but we have that $\neg\exists z(\mathtt{cP}(z,\mathtt{gd1},\mathtt{t1}))$. □

$\mathbf{t_{bd}}$25 $\mathfrak{B}_\mathtt{d} \vdash \mathrm{PED}(x) \leftrightarrow \exists yt(\mathtt{tP_d}(y,x,t)) \wedge \forall yt(\mathtt{tP_d}(y,x,t) \rightarrow \mathrm{PED}(y))$ ($\mathrm{a_d}$39)
Proof. See proof generated by theorem provers. □

$\mathbf{t_{bd}}$26 $\mathfrak{B}_\mathtt{d} \nvdash \mathrm{NPED}(x) \leftrightarrow \exists yt(\mathtt{tP_d}(y,x,t)) \wedge \forall yt(\mathtt{tP_d}(y,x,t) \rightarrow \mathrm{NPED}(y))$ ($\mathrm{a_d}$40)
Proof. ($\rightarrow$) See the counterexample on page 219 where $\mathrm{NPED}(\mathtt{gd1})$ ($\mathtt{gd1}$ is a generically dependent continuant) but $\neg\exists zt(\mathtt{cP}(z,\mathtt{gd1},t))$.
($\leftarrow$) Suppose $\mathtt{tP_d}(y,x,t) \wedge \mathrm{NPED}(y)$. By ($\mathrm{d_{bd}}$16), ($\mathrm{d_{bd}}$5), and ($\mathrm{d_{bd}}$2), we have $\mathtt{cP}(y,x,t) \wedge y::\mathtt{gdcnt} \wedge x::\mathtt{cnt} \wedge \neg(x::\mathtt{sdcnt})$ and, by ($\mathrm{a_b}$22), $\neg(x::\mathtt{idcnt})$. From $x::\mathtt{cnt} \wedge \neg(x::\mathtt{idcnt}) \wedge \neg(x::\mathtt{sdcnt})$ we have, by the axioms on the taxonomy, $x::\mathtt{gdcnt}$. □

7.2.6 Constitution

K has not been defined in terms of BFO.

7.2.7 Participation

$\mathbf{t_{bd}}$27 $\mathfrak{B}_\mathtt{d} \vdash \mathtt{PC}(x,y,t) \rightarrow \mathrm{ED}(x) \wedge \mathrm{PD}(y) \wedge \mathrm{T}(t)$ ($\mathrm{a_d}$52)
Proof. See proof generated by theorem provers. □

$\mathbf{t_{bd}}$28 $\mathfrak{B}_\mathtt{d} \nvdash \mathtt{PC}(x,y,t) \rightarrow \mathrm{PRE}(x,t) \wedge \mathrm{PRE}(y,t)$ ($\mathrm{a_d}$53)
Proof. See the counterexample on page 217 where $\mathtt{PTC}(\mathtt{c2},\mathtt{p1},\mathtt{t2})$ but $\mathtt{c2}$ does not have a temporal location. □

$\mathbf{t_{bd}}$29 $\mathfrak{B}_\mathtt{d} \vdash \mathtt{PC}(x,y,t) \wedge \mathtt{P_d}(u,t) \rightarrow \mathtt{PC}(x,y,u)$ ($\mathrm{a_d}$54)
Proof. From $\mathtt{PC}(x,y,t)$, by ($\mathrm{d_{bd}}$17), $\mathrm{ED}(x) \wedge \mathrm{PD}(y) \wedge \exists z(\mathtt{tmP}(y,z) \wedge \mathtt{PTC}(x,z,t))$. From $\mathtt{PTC}(x,z,t)$, by ($\mathrm{a_b}$112), $\mathrm{TM}(t)$ and then, by ($\mathrm{d_{bd}}$8), ($\mathrm{d_{bd}}$15), and ($\mathrm{a_b}$58), $\mathtt{P_d}(u,t) \leftrightarrow \mathtt{oP}(u,t)$. From $\mathtt{PTC}(x,z,t) \wedge \mathtt{oP}(u,t)$, by ($\mathrm{a_b}$114), $\mathtt{PTC}(x,z,u)$, then $\mathtt{tmP}(y,z) \wedge \mathtt{PTC}(x,z,t)$. □

$\mathbf{t_{bd}}$30 $\mathfrak{B}_\mathtt{d} \nvdash \mathtt{PC}(x,y,t) \wedge \mathtt{PC}(x,y,u) \wedge \mathtt{SUM_d}(s,t,u) \rightarrow \mathtt{PC}(x,y,s)$ ($\mathrm{a_d}$55)
Proof. See the counterexample on page 217 where $\mathtt{PTC}(\mathtt{c2},\mathtt{p1},\mathtt{t1})$, $\mathtt{PTC}(\mathtt{c2},\mathtt{p1},\mathtt{t2})$, $\mathtt{SUM_d}(\mathtt{t12},\mathtt{t1},\mathtt{t2})$ but $\neg\mathtt{PTC}(\mathtt{c2},\mathtt{p1},\mathtt{t12})$. □

$\mathbf{t_{bd}}$31 $\mathfrak{B}_\mathtt{d} \nvdash \mathrm{PD}(x) \wedge \mathrm{PRE}(x,t) \rightarrow \exists yu(\mathtt{P_d}(u,t) \wedge \mathtt{PC}(y,x,u))$ ($\mathrm{a_d}$56)
Proof. In the counterexample of ($\mathrm{t_{bd}}$14), $\mathtt{pb1} ::_{\mathtt{t1}} \mathtt{pbnd}$ and $\mathtt{pb1}$ EX-exists only at $\mathtt{t1}$. By ($\mathrm{d_{bd}}$1) $\mathtt{pb1}$ is a perdurant, and by ($\mathrm{d_{bd}}$14) and ($\mathrm{d_d}$5), $\mathrm{PRE}(\mathtt{pb1},\mathtt{t1})$ but $\mathtt{pb1}$ does not have participants (at any time). □

$\mathbf{t_{bd}}$32 $\mathfrak{B}_\mathtt{d} \nvdash \mathrm{ED}(x) \wedge \mathrm{PRE}(x,t) \rightarrow \exists yu(\mathtt{P_d}(u,t) \wedge \mathtt{PC}(x,y,u))$ ($\mathrm{a_d}$57)
Proof. See the counterexample on page 218 where $\mathtt{c1}$ is a site (and then an endurant), it has a spatial location ($\mathtt{t12}$), and $\mathrm{PRE}(\mathtt{c1},\mathtt{t1})$ holds but $\mathtt{c1}$ does not participate to any occurrent. □

$\mathbf{t_{bd}}$33 $\mathfrak{B}_\mathtt{d} \nvdash \mathtt{PC}(x,y,t) \wedge \mathtt{P_d}(y,z) \rightarrow \mathtt{PC}(x,z,t)$ ($\mathrm{a_d}$58)
Proof. See the counterexample on page 217 where $\mathtt{PC}(\mathtt{c2},\mathtt{p1},\mathtt{t1})$, $\mathtt{P_d}(\mathtt{pb1},\mathtt{p1})$, but $\neg\mathtt{PC}(\mathtt{c2},\mathtt{pb1},\mathtt{t1})$. □

7.2.8 Existential Dependence

t_{bd}34 $\mathfrak{B}_d \nvdash \mathtt{EXD}(x,y,t) \rightarrow \mathtt{PRE}(x,t) \wedge \mathtt{PRE}(y,t)$ (a_d59)
Proof. See the counterexample on page 217 where $\mathtt{PTC}(\mathtt{c2},\mathtt{p1},\mathtt{t1})$, and then, by ($d_{bd}$19), $\mathtt{EXD}(\mathtt{c2},\mathtt{p1},\mathtt{t1})$, but c2 does not have a temporal location (in the sense of TLC) and then cannot be present. □

t_{bd}35 $\mathfrak{B}_d \nvdash \mathtt{EXD}(x,y,t) \wedge \mathtt{P}_d(u,t) \rightarrow \mathtt{EXD}(x,y,u)$ (a_d60)
Proof. See the counterexample on page 219 where $\mathtt{EXD}(\mathtt{gd1},\mathtt{c2},\mathtt{t12})$ (because $\mathtt{GDEP}(\mathtt{gd1},\mathtt{c2},\mathtt{t12})$) and $\mathtt{oP}(\mathtt{t1},\mathtt{t12})$ but no one of the conditions for EXD is satisfied, i.e., $\neg\mathtt{GDEP}(\mathtt{gd1},\mathtt{c2},\mathtt{t1})$, $\neg\mathtt{PTC}(\mathtt{gd1},\mathtt{c2},\mathtt{t1})$, $\neg\mathtt{PTC}(\mathtt{c2},\mathtt{gd1},\mathtt{t1})$, and $\neg\mathtt{SDEP}(\mathtt{gd1},\mathtt{c2})$. □

t_{bd}36 $\mathfrak{B}_d \nvdash \mathtt{EXD}(x,y,t) \wedge \mathtt{EXD}(x,y,u) \wedge \mathtt{SUM}_d(s,t,u) \rightarrow \mathtt{EXD}(x,y,s)$ (a_d61)
Proof. See the counterexample on page 217 where we have $\mathtt{PTC}(\mathtt{c2},\mathtt{p1},\mathtt{t1})$ and $\mathtt{PTC}(\mathtt{c2},\mathtt{p1},\mathtt{t2})$, then, by ($d_{bd}$19), $\mathtt{EXD}(\mathtt{c2},\mathtt{p1},\mathtt{t1})$ and $\mathtt{EXD}(\mathtt{c2},\mathtt{p1},\mathtt{t2})$. In addition we have $\mathtt{SUM}_d(\mathtt{t12},\mathtt{t1},\mathtt{t2})$ but , but $\neg\mathtt{PTC}(\mathtt{c2},\mathtt{p1},\mathtt{t12})$ (and SDEP and GDEP are 'empty'). □

t_{bd}37 $\mathfrak{B}_d \nvdash \mathtt{EXD}(x,y,u) \wedge \mathtt{PRE}(x,t) \rightarrow \exists z v(\mathtt{P}_d(v,t) \wedge \mathtt{EXD}(x,z,v))$ (a_d62)
Proof. See the counterexample on page 220 where we have $\mathtt{EXD}(\mathtt{c1},\mathtt{p1},\mathtt{t3})$ (because $\mathtt{PTC}(\mathtt{c1},\mathtt{p1},\mathtt{t3})$), $\mathtt{PRE}(\mathtt{c1},\mathtt{i2})$ (because $\mathtt{TLC}(\mathtt{c1},\mathtt{t2})$ and $\mathtt{P}_d(\mathtt{i2},\mathtt{t2})$) but $\neg\exists z(\mathtt{EXD}(\mathtt{c1},z,\mathtt{i2}))$ (i2 is atomic). □

t_{bd}38 $\mathfrak{B}_d \vdash \mathtt{PC}(x,y,t) \rightarrow \mathtt{EXD}(x,y,t) \wedge \mathtt{EXD}(y,x,t)$ (a_d64)
Proof. Directly from (d_{bd}19). □

t_{bd}39 $\mathfrak{B}_d \nvdash \mathtt{DQT}(x,y) \rightarrow \mathtt{SD}(x,y)$ (a_d65)
Proof. See the counterexample on page 219 where $\mathtt{DQT}(\mathtt{sd2},\mathtt{c2})$ (because $\mathtt{SDEP}(\mathtt{sd2},\mathtt{c2})$ and c2 is a site) but sd2 does not have a temporal location (i.e., $\neg\exists t(\mathtt{TLC}(\mathtt{sd2},t))$) therefore, by ($d_d$5), $\neg\exists t(\mathtt{PRE}(\mathtt{sd2},t))$ but the definition of SD (d_d9) requires sd2 to be present at least at one time. □

t_{bd}40 $\mathfrak{B}_d \nvdash \mathrm{NPED}(x) \wedge \mathtt{PRE}(x,t) \rightarrow \exists y u(\mathrm{PED}(y) \wedge \mathtt{P}_d(u,t) \wedge \mathtt{EXD}(x,y,u))$ (a_d66)
Proof. See the counterexample on page 221 where we have $\mathrm{NPED}(\mathtt{gd1})$ (because $\mathtt{gd1} ::_{\mathtt{t1}} \mathtt{gdcnt}$), $\mathtt{PRE}(\mathtt{gd1},\mathtt{t2})$ (because $\mathtt{TLC}(\mathtt{gd1},\mathtt{t12})$ and $\mathtt{P}_d(\mathtt{t2},\mathtt{t12})$ but $\neg\exists z(\mathtt{EXD}(\mathtt{gd1},z,\mathtt{t2}))$. □

7.2.9 Spatial Location

t_{bd}41 $\mathfrak{B}_d \vdash \mathtt{SLC}(x,s,t) \rightarrow (\mathrm{ED}(x) \vee \mathrm{PQ}(x) \vee \mathrm{PD}(x)) \wedge \mathrm{S}(s)$ ~(a_d74)
($\mathrm{ED}(x) \wedge \neg\mathrm{AS}(x)$ has beed substituted by $\mathrm{ED}(x)$ because AS has not been considered in the mappings.)
Proof. See proof generated by theorem provers. □

t_{bd}42 $\mathfrak{B}_d \nvdash \mathtt{SLC}(x,s,t) \rightarrow \mathtt{PRE}(x,t)$ (a_d75)
Proof. See the counterexample on page 217 where $\mathtt{SREG}(\mathtt{c2},\mathtt{s1},\mathtt{t1})$ but c2 does not have a temporal location (and then it is not present at any time). □

t_{bd}43 $\mathfrak{B}_d \vdash \mathtt{SLC}(x,s,t) \wedge \mathtt{SLC}(x,r,t) \rightarrow s = r$ (a_d76)
Proof. According to (d_{bd}20) there are four cases.

(1) $x ::_t \texttt{idcnt} \wedge \neg(x ::_t \texttt{sreg})$. From the hypothesis we have $\texttt{SREG}(x,s,t) \wedge \texttt{SREG}(x,r,t)$ that, by (a_b75), implies $s = r$.
(2) $x ::_t \texttt{gdcnt}$. From the hypothesis and (a_b106) there exist a and b such that $a ::_t \texttt{idcnt} \wedge \neg(a ::_t \texttt{sreg})$, $\texttt{GDEP}(x,a,t) \wedge \forall z(\texttt{GDEP}(x,z,t) \rightarrow \texttt{cP}(z,a,t)) \wedge \texttt{SREG}(a,s,t))$, $b ::_t \texttt{idcnt} \wedge \neg(b ::_t \texttt{sreg})$, $\texttt{GDEP}(x,b,t) \wedge \forall z(\texttt{GDEP}(x,z,t) \rightarrow \texttt{cP}(z,b,t)) \wedge \texttt{SREG}(b,r,t))$ and then $\texttt{cP}(a,b,t) \wedge \texttt{cP}(b,a,t) \wedge \texttt{SREG}(a,s,t) \wedge \texttt{SREG}(b,r,t)$. By ($a_b$83), we have $\texttt{LOC}(a,b,t) \wedge \texttt{LOC}(b,a,t)$. From $\texttt{LOC}(a,b,t)$, by (a_b77), $\exists uvw(\texttt{tmP}(u,t) \wedge \texttt{SREG}(a,v,u) \wedge \texttt{SREG}(b,w,u) \wedge \texttt{cP}(v,w,u))$. From $\texttt{tmP}(u,t) \wedge \texttt{SREG}(a,s,t) \wedge \wedge \texttt{SREG}(a,v,u)$, by ($a_b$75) and ($a_b$76), $s = v$. Similarly for $\texttt{SREG}(b,w,u)$, i.e., we have $r = w$ and then, from $\texttt{cP}(v,w,u)$, we have $\texttt{cP}(s,r,u)$. From $\texttt{tmP}(u,t) \wedge \texttt{LOC}(b,a,t)$, by ($a_b$79), $\texttt{LOC}(b,a,u)$. Then, by (a_b77), $\exists u'v'w'(\texttt{tmP}(u',u) \wedge \texttt{SREG}(b,v',u') \wedge \texttt{SREG}(a,w',u') \wedge \texttt{cP}(v',w',u'))$. Following the previous reasoning we have $\texttt{cP}(r,s,u') \wedge \texttt{tmP}(u',u)$. From $\texttt{cP}(s,r,u) \wedge \texttt{tmP}(u',u)$, by ($a_b$17), $\texttt{cP}(s,r,u')$, therefore we have $\texttt{cP}(s,r,u') \wedge \texttt{cP}(r,s,u')$ that, by (a_b20), implies $s = r$.
(3) $\texttt{Q}(x)$. By (d_b10) there exist a and b such that $a ::_t \texttt{idcnt} \wedge \neg(a ::_t \texttt{sreg})$, $\texttt{INH}(x,a) \wedge \forall z(\texttt{INH}(x,z) \rightarrow \texttt{cP}(z,a,t)) \wedge \texttt{SREG}(a,s,t))$, $b ::_t \texttt{idcnt} \wedge \neg(b ::_t \texttt{sreg})$, $\texttt{INH}(x,b) \wedge \forall z(\texttt{INH}(x,z) \rightarrow \texttt{cP}(z,b,t)) \wedge \texttt{SREG}(b,r,t))$ and then $\texttt{cP}(a,b,t) \wedge \texttt{cP}(b,a,t) \wedge \texttt{SREG}(a,s,t) \wedge \texttt{SREG}(b,r,t)$. We can then follows the proof in (2).
(4) $\texttt{Q}(x)$. From the hypothesis we have $\exists u(\texttt{STREG}(x,u) \wedge \texttt{SPROJ}(u,s,t))$ and $\exists v(\texttt{STREG}(x,v) \wedge \texttt{SPROJ}(v,r,t))$ that, by (a_b89) and (a_b92), imply $s = r$. □

t_{bd}44 $\mathfrak{B}_d \vdash \texttt{SLC}(x,s,u) \wedge \texttt{SLC}(x,r,t) \wedge \texttt{P}_d(u,t) \rightarrow \texttt{P}_d(s,r)$ (a_d77)
Proof. See proof generated by the theorem provers. □

t_{bd}45 $\mathfrak{B}_d \nvdash \texttt{SLC}(x,s,u) \wedge \texttt{PRE}(x,t) \rightarrow \exists r(\texttt{SLC}(x,r,t))$ (a_d78)
Proof. See the counterexample on page 218 where $\texttt{SREG}(\texttt{c1},\texttt{s1},\texttt{t1})$ and also $\texttt{TLC}(\texttt{c1},\texttt{t12})$ (and then $\texttt{PRE}(\texttt{c1},\texttt{t12})$) but $\neg\exists s(\texttt{SREG}(\texttt{c1},s,\texttt{t12})$. □

t_{bd}46 $\mathfrak{B}_d \vdash \text{NPED}(x) \wedge \texttt{SLC}(x,s,t) \rightarrow \exists y(\text{PED}(y) \wedge \texttt{EXD}(x,y,t) \wedge \texttt{SLC}(y,s,t)))$ (a_d81)
Proof. From the hypotheses, by (d_{bd}5), (d_{bd}20), and the rigidity of gdcnt, $\exists y(\texttt{GDEP}(x,y,t) \wedge \forall z(\texttt{GDEP}(x,z,t) \rightarrow \texttt{cP}(z,y,t)) \wedge \texttt{SREG}(y,s,t))$. Consider such y. From $\texttt{GDEP}(x,y,t)$, by (a_b106), $y ::_t \texttt{idcnt} \wedge \neg(y ::_t \texttt{sreg})$ then, by (d_{bd}3), $\text{PED}(y)$. From $\texttt{GDEP}(x,y,t)$, by (d_{bd}19), $\texttt{EXD}(x,y,t)$. From $\texttt{SREG}(y,s,t) \wedge y ::_t \texttt{idcnt} \wedge \neg(y ::_t \texttt{sreg})$, by ($d_{bd}$20), $\texttt{SLC}(y,s,t)$. □

t_{bd}47 $\mathfrak{B}_d \nvdash \texttt{PC}(x,y,t) \wedge \texttt{SLC}(x,s,t) \rightarrow \exists r(\texttt{SLC}(y,r,t))$ (a_d82)
Proof. See the counterexample on page 223 where we have $\texttt{PC}(\texttt{c2},\texttt{p2},\texttt{t12})$ (because $\texttt{PTC}(\texttt{c2},\texttt{p2},\texttt{t12})$, $\texttt{oP}(\texttt{t12},\texttt{t12})$, $\texttt{c2} ::_{\texttt{t12}} \texttt{idcnt}$, $\neg(\texttt{c2} ::_{\texttt{t12}} \texttt{sreg})$, $\texttt{p2} ::_{\texttt{t12}} \texttt{proc}$), $\texttt{SLC}(\texttt{c2},\texttt{s12},\texttt{t12})$ (because $\texttt{c2} ::_{\texttt{t12}} \texttt{idcnt}$, $\neg(\texttt{c2} ::_{\texttt{t12}} \texttt{sreg})$, and $\texttt{SREG}(\texttt{c2},\texttt{s12},\texttt{t12})$), $\texttt{STREG}(\texttt{p2},\texttt{st123})$ but $\neg\exists z(\texttt{SPROJ}(\texttt{st123},z,\texttt{t12}))$. □

t_{bd}48 $\mathfrak{B}_d \vdash \texttt{PC}(x,y,t) \wedge \texttt{SLC}(x,r,t) \wedge \texttt{SLC}(y,s,t) \rightarrow \texttt{P}_d(r,s)$ (a_d83)
Proof. See proof generated by the theorem provers. □

t_{bd}49 $\mathfrak{B}_d \nvdash \texttt{tP}_d(x,y,t) \wedge \texttt{SLC}(x,s,t) \rightarrow \exists r(\texttt{SLC}(y,r,t))$ (a_d84)
Proof. See the counterexample on page 218 where $\texttt{SREG}(\texttt{c1},\texttt{s1},\texttt{t1})$ and also $\texttt{tP}_d(\texttt{c1},\texttt{c2},\texttt{t12})$ but $\neg\exists s(\texttt{SREG}(\texttt{c1},s,\texttt{t12})$. □

t_{bd}50 $\mathfrak{B}_d \vdash \texttt{tP}_d(x,y,t) \wedge \texttt{SLC}(x,r,t) \wedge \texttt{SLC}(y,s,t) \rightarrow \texttt{P}_d(r,s)$ (a_d85)
Proof. See proof generated by the theorem provers. □

$\mathbf{t_{bd}51}$ $\mathfrak{B}_{\mathtt{d}} \nvdash \mathtt{P_d}(x,y) \wedge \mathtt{SLC}(x,s,t) \rightarrow \exists r(\mathtt{SLC}(y,r,t))$ ($\mathrm{a_d}$86)
Proof. See the counterexample on page 223 where $\mathtt{P_d}(\mathtt{p1},\mathtt{p2})$, $\mathtt{SLC}(\mathtt{p1},\mathtt{s1},\mathtt{t12})$ (because $\mathtt{p1}::\mathtt{proc}$, $\mathtt{STREG}(\mathtt{p1},\mathtt{st12})$, and $\mathtt{SPROJ}(\mathtt{st12},\mathtt{s1},\mathtt{t12})$), and finally $\mathtt{STREG}(\mathtt{p2},\mathtt{st123})$ but $\neg\exists z(\mathtt{SPROJ}(\mathtt{st123},z,\mathtt{t12}))$. □

$\mathbf{t_{bd}52}$ $\mathfrak{B}_{\mathtt{d}} \vdash \mathtt{P_d}(x,y) \wedge \mathtt{SLC}(x,s,t) \wedge \mathtt{SLC}(y,r,t) \rightarrow \mathtt{P_d}(s,r)$ ($\mathrm{a_d}$87)
Proof. See proof generated by the theorem provers. □

7.2.10 Taxonomy

From the mappings ($\mathrm{d_{bd}}$1)-($\mathrm{d_{bd}}$20) and the axioms characterizing the taxonomy of BFO, it is easy to see that: (*i*) ED subsumes PED and NPED; (*ii*) PED subsumes F; (*iii*) AB and R are equivalent and they subsume T and TR (that are equivalent) together with S and PR (that are equivalent); and (*iv*) Q and PQ are equivalent. It is also easy to see that: (*i*) ED, PD, Q, and AB are mutually disjoint; (*ii*) T/TR and S/PR are disjoint and cover R/AB; and (*iii*) PED and NPED are disjoint and cover ED. However, note that the spatiotemporal regions of BFO (as well as the universals) are instances of no one of these categories.

7.3 Analysis

The results discussed in Sect. 7.2.10 show that, with the exception of the spatiotemporal regions, all the particulars of BFO find their place in DOLCE. The possibility to avoid spatiotemporal regions, once one has the temporal and the spatial locations of entities, has been analyzed in Sect. 6.2.3. For importing BFO universals into DOLCE we already suggested some possibilities, e.g. mapping universals to regions or to non-physical endurants. Both alternatives would require to modify the mappings ($\mathrm{d_{bd}}$5) and/or ($\mathrm{d_{bd}}$12) (and eventually ($\mathrm{d_{bd}}$9) and ($\mathrm{d_{bd}}$11)). Vice versa, some of the original categories of DOLCE remain necessarily 'empty'. It is the case of AS, AR, TQ, and AQ. Other categories are drastically reduced: physical qualities/regions reduce to spatial locations/space regions and temporal regions reduce to time intervals. It seems that DOLCE accepts a larger variety of entities. This is also confirmed by the possibility to have in DOLCE spatially co-localized, and possibly layered, entities linked by coincidence (mutual $\mathtt{tP_d}$-parthood at a time) and constitution. Whether this lareger variety of entities translates into an higher expressive power needs to be investigated. As shown in the case of spatiotemporal regions, more complex mappings could at least mitigate the gap.

Let us now take into account the results in Sect. 7.2. Even though the proofs of some theorems are still lacking, some observations can already be made. It results quite evident that the majority of the original axioms of DOLCE are not preserved by the mappings. At first sight, one could take this as an evidence of the bad quality of mappings, or of a genuine difference between the two ontologies, or conclude, more drastically, that a comparison of the mappings in the two directions shows that BFO is 'weaker' than DOLCE. A deeper analysis of the reasons underlying the counterexamples of some theorems may suggest a more complex situation with pros and cons on both sides.

Few, and quite technical, differences between the two theories seem to cause several systematic problems for a comprehensive mapping. In DOLCE, PRE is defined only for the entities that have a temporal extension, see (d_d5), while in BFO EX is a primitive relation.[15] The counterexample for (t_{bd}14) shows that in BFO an entity may exist at two non overlapping temporal regions without existing at any bigger temporal region. In these situations the condition $\forall u(\mathtt{EX}(x,u) \leftrightarrow \mathtt{tmP}(u,t))$ in the definition of $\mathtt{TLC}(x,t)$, see (d_{bd}14), is never satisfied and the relation PRE is always 'empty'. This problem has a huge impact in the proofs in Sect. 7.2 because PRE is heavily used in the characterization of the primitives of DOLCE. At the same time, it is a quite subtle problem whose origin seems to have a technical, more than ontological, nature; at the end, in both ontologies the entity exists at both times.

The same problem affects axioms like (t_{bd}21) and (t_{bd}30) where from the holding of a given relation at two times, one wants to infer the holding of the relation at the sum of the times (provided such sum exists). In these cases, the fact that the entity does not exist at the sum of the times rules out (for the majority of relations) the possibility for the relation to hold at the sum. The BFO relation SREG suffers this problem, i.e., a continuant can occupy spatial regions (even the same spatial region) at two given times without having a spatial location at the sum of these times (see the counterexample for (t_{bd}49)). The presence of spatiotemporal regions and TPROJ/SPROJ relations together with the intricate links between SREG, oP, cP, LOC, and OCCIN further complicate the domain of space and spatial location (at least from the formal perspective) impacting all the theorems involving SLC.

This problem suggests a different mapping technique. Rather than encapsulating the relation between EX and TLC into the mapping (d_{bd}14), one could follow a 'two steps' procedure: first introduce a direct mapping between EX and PRE and then define TLC by using PRE (as defined in the mapping), e.g.,

- $\mathtt{PRE}(x,t) \coloneqq (\mathrm{PD}(x) \vee \mathrm{ED}(x) \vee \mathrm{Q}(x)) \wedge \mathtt{EX}(x,t)$
- $\mathtt{TLC}(x,t) \coloneqq \forall u(\mathtt{PRE}(x,u) \leftrightarrow \mathrm{P_d}(u,t))$

In this way, entities that, for instance, exist at two times but not at their mereological sum would still lack a temporal location TLC but this would not prevent them to PRE-exist (just relying on the first mapping). We can then hope to recover some theorems that does not require TLC at the price of possibly losing the original link between PRE and TLC. The study of how this mapping technique can be extended to other primitives and its actual impact are left for future work.

The problems highlighted up to this point have important impacts but are primarily of technical nature. Some genuine ontological differences seem also to exist. First, the defined $\mathrm{P_d}$, like oP, does not satisfy the supplementation axiom, see (t_{bd}7). Second, the relation INH is not equivalent to the original DQT, in particular it does not satisfy the non-migration principle (t_{bd}9). For relational qualities this can be problematic if, for instance, one wants to distinguish "Mary loves John" from "John loves Mary": in both cases we would have a relational quality inhering in both John and Mary. Third, the reflexivity of the defined $\mathtt{tP_d}$, like the one of cP, does not hold for all the endurants/con-

[15] One could think that TREG corresponds to TLC but actually TREG is defined only for occurents and it is only quite minimally linked to EX.

tinuants. It follows that in BFO one can have continuants for which cP is not defined. Fourth, while in DOLCE all the perdurants have participants at every time they exist (t_{bd}31), in BFO, processes need to have at least a participant and this does not hold for process boundaries. Furthermore, in BFO, continuants do not necessarily participate in processes while in DOLCE this is mandatory: all the endurants participate at a process when they exist (t_{bd}32).

8 The mapping from EMMO to DOLCE

In this section we introduce the syntactic definitions of the primitives of DOLCE in terms of the primitives of EMMO introduced in the Mereocausal module (see Sect. 4.1) and in the Semiotic perspective (see Sect. 4.2.2). We heavily exploit also the notions defined in the Concepts extension (see Sect. 4.2.3), the Time extension (see Sect. 4.2.4) and the Existence at a time and Temporal slice extension (see Sect. 4.2.5). Analogously to the DOLCE-BFO mapping, we present only a partial mapping covering a subset of the categories and primitives of DOLCE.

8.1 Mappings

The Time extension defines, in terms of parthood and causation, the temporal individuals (the times) and their structure. We explored the possibility to follow a similar strategy to define the spatial regions and their structure starting only from the Mereocausal module. However, the hypothesis that the causes and their *direct* effects are spatially connected, a debatable assumption in itself given the prefered interpretation of EMMO, is too weak to individuate (by mimicking the construction done in Sect. 4.2.4) spatially equivalent sums of quanta. At the moment, it is still an open question whether space can be built from the sole mereocausal module (possibly by exploiting the naturalistic extension) or introducing a new primitive is necessary to that end. In light of that, space is not considered, and the categories S and SL of DOLCE are excluded from the mapping.

Within endurants, DOLCE distinguishes between physical and non-physical endurants, according to whether they have direct spatial qualities. The exclusion of S and SL makes it impossible to characterize this distinction. We then rule out from the mapping PED and NPED, as well as all their subcategories. This choice is also motivated by the fact that it is not clear whether the notions of agentivity and sociality (crucial to ground the difference between the DOLCE categories of APO, NAPO, ASO, NASO) can be built starting only from mereocausality and semiosis. As a consequence of that, the distinction between PQ and AQ (and then the one between PR and AR) cannot be characterized as well: in DOLCE, physical qualities directly inhere in physical endurants, while abstract qualities directly inhere in non-physical endurants. That said, temporal qualities (TQ) directly inhere in perdurants while physical and abstract qualities directly inheres in endurants. This distinction can be captured in our setting, therefore we consider in the mapping the 'new' class noted $[\text{PQ}\vee\text{AQ}]$ that collects physical and abstract qualities. Similarly, $[\text{PR}\vee\text{AR}]$ collects physical and abstract regions.

As we will see, the times built in EMMO (TME) are mapped to DOLCE's temporal intervals (T). However, in the mapping we do no consider TL, i.e, the temporal qualities associated in DOLCE to the temporal intervals (T). Two reasons underlie this choice. First, the qualities inhering in an object are reduced in EMMO to the mereological sums of the object itself and a top-concept that classifies it (see (d_{ed}15)). Concepts are grounded on semiotic processes (see (d_{e}36)); thus, by having concepts as part, qualities, as well as their qualia (see (d_{ed}16) and (d_{ed}17)), also depend on semiosis. Vice versa, TMEs are built from mereocausation independently of any semiotic process. The introduction of temporal locations would arguably go against this independence

from semiosis. Second, by having TLs, given (t_e51), it would be plausible to assume that the only 'top-concept' for time is the universe. By maintaining (d_{ed}15), this would imply that all the entities have the same temporal location, i.e., the universe. Notably, had we decided to focus on measured temporal duration (on EMMO's side) for the alignment of time, that would have fit well with other semiotic processes; however, pursuing such a route would have meant ignoring DOLCE's ontological stance, and it would have arguably been conductive to a number of issues arising from semiotics' limits concerning objectivity/intersubjectivity.

Finally, the characterization of the subcategories of perdurant (PD) is based on the notions of homeomericity and cumulativity. These notions are 'parametrized' on a finite set of universals explicitly introduced in DOLCE in addition to the categories in Table 4. DOLCE does not commit to specific extensions, the idea is that DOLCE can be extended in different ways and that in all these extensions homeomericity and cumulativity assume a precise definition. The mechanism is quite similar to the one underlying EMMO's extensions (see Sect. 4.2) that, following what had been done in DOLCE, is employed to distinguish endurants from perdurants (see (d_{ed}3) and (d_{ed}4)) among other things. The extensions of DOLCE and the ones of EMMO are however independent one from the other, making it impossible to know a priori if the ϕs added to DOLCE are present in an extension (or basic module) of EMMO. The commitment to the existence of an EMMO extension for each and every DOLCE extension seems, from both a practical and applicative point of view, too strong. For this reason, in the mapping we do not consider the subcategories of PD.

Summing up, among the DOLCE notions, in the following we introduce:

1. syntactic definitions for the primitive relations: $\mathtt{P_d}$, DQT, QL, TLC, tQL, $\mathtt{tP_d}$, K, PC, EXD; and
2. syntactic definitions for the categories (or disjunctions of categories): ED, PD, Q, AB, TQ, [PQ∨AQ], R, TR, [PR∨AR], and T.

Below we list these syntactic definitions together with a short informal description. A deeper analysis about these definitions and their impact on the preservation of the axioms of DOLCE is put forward in Sect. 8.3.

$\mathbf{d_{ed}}$1 $\mathrm{T}(x) \coloneqq \mathtt{TME}(x)$

Time intervals coincide with times.

$\mathbf{d_{ed}}$2 $\mathtt{TLC}(x,t) \coloneqq \mathrm{T}(t) \wedge \mathtt{EQ}(x,t)$

The temporal location of x is the time temporally equivalent to x.

$\mathbf{d_{ed}}$3 $\mathrm{ED}(x) \coloneqq \bigvee_{\phi \in \mathbf{t}(\Delta)} \phi(x) \wedge \bigwedge_{\phi \in \mathbf{t}(\Delta)} (\phi(x) \rightarrow \mathtt{tDSC}\langle\phi\rangle(x))$

An entity x is an endurant when it is classified by at least a type in an extension of EMMO and it is dissective with respect to all the types that classify it.

$\mathbf{d_{ed}}$4 $\mathrm{PD}(x) \coloneqq \bigvee_{\phi \in \mathbf{t}(\Delta)} \phi(x) \wedge \bigwedge_{\phi \in \mathbf{t}(\Delta)} (\phi(x) \rightarrow \mathtt{tnDSC}\langle\phi\rangle(x))$

An entity x is a perdurant when it is classified by at least a type in an extension of EMMO and it is non-dissective with respect to all the types that classify it.

$\mathbf{d_{ed}}$5 $\mathtt{eCPT}(x) \coloneqq \mathtt{CPT}(x) \wedge \forall y(\mathtt{CF}(x,y) \rightarrow \mathrm{ED}(y))$

This is not a primitive of DOLCE but a notion that we will often use in the mappings. It individuates concepts that classify only endurants.

$d_{ed}6$ $\mathtt{pCPT}(x) \coloneqq \mathtt{CPT}(x) \wedge \forall y(\mathtt{CF}(x,y) \rightarrow \mathrm{PD}(y))$

This is not a primitive of DOLCE but a notion that we will often use in the mappings. It individuates concepts that classify only perdurants.

$d_{ed}7$ $\mathtt{epCPT}(x) \coloneqq \mathtt{eCPT}(x) \vee \mathtt{pCPT}(x)$

This is not a primitive of DOLCE but a notion that we will often use in the mappings.

$d_{ed}8$ $\mathrm{TR}(x) \coloneqq \mathrm{T}(x) \vee \mathtt{pCPT}(x)$

Among temporal regions there are times or concepts that classify perdurants, i.e., intuitively, concepts that are part of qualities of perdurants.

$d_{ed}9$ $[\mathrm{PR} \vee \mathrm{AR}](x) \coloneqq \mathtt{eCPT}(x)$

As discussed above, the lack of space regions does not allow to recover the PED/NPED distinction, and then, the PR/AR distinction. We then introduce the union of these categories that contains concepts that classify endurants, i.e., intuitively, concepts that are part of qualities of endurants.

$d_{ed}10$ $\mathrm{R}(x) \coloneqq \mathrm{TR}(x) \vee [\mathrm{PR} \vee \mathrm{AR}](x)$

The regions are the union of temporal, physical, and abstract regions.

$d_{ed}11$ $\mathrm{AB}(x) \coloneqq \mathrm{R}(x)$

The only abstracts that can be built from EMMO are regions.

$d_{ed}12$ $\mathtt{P_d}(x,y) \coloneqq (((\mathrm{PD}(x) \wedge \mathrm{PD}(y)) \vee (\mathrm{T}(x) \wedge \mathrm{T}(y))) \wedge \mathtt{P}(x,y)) \vee$
$(((\mathtt{pCPT}(x) \wedge \mathtt{pCPT}(y)) \vee (\mathtt{eCPT}(x) \wedge \mathtt{eCPT}(y))) \wedge \mathtt{SPC}(x,y))$

For perdurants and times $\mathtt{P_d}$ coincides with $\mathtt{P}$ while for concepts classifying only perdurants or endurants it coincides with the specialization $\mathtt{SPC}$.

$d_{ed}13$ $\mathtt{tP_d}(x,y,t) \coloneqq \mathrm{ED}(x) \wedge \mathrm{ED}(y) \wedge \exists z(\mathtt{TS}(z,x,t) \wedge \mathtt{P}(z,y))$

Temporary parthood is defined on endurants such that the temporal slice (at t) of the first endurant is part of the second endurant.

$d_{ed}14$ $\mathtt{PC}(x,y,t) \coloneqq \mathrm{ED}(x) \wedge \mathrm{PD}(y) \wedge \exists z(\mathtt{TS}(z,x,t) \wedge \mathtt{P}(z,y))$

Participation is the analogous of temporary parthood but now the first relatum is an endurant and the second relatum a perdurant.

$d_{ed}15$ $\mathtt{DQT}(q,x) \coloneqq$
$(\mathrm{ED}(x) \wedge \neg\exists c(\mathtt{eCPT}(c) \wedge \mathtt{O}(x,c)) \wedge \exists r(\mathtt{eCPT}(r) \wedge \mathtt{CF}(r,x) \wedge \mathtt{TOP}(r) \wedge \mathtt{SUM}(q,x,r))) \vee$
$(\mathrm{PD}(x) \wedge \neg\exists c(\mathtt{pCPT}(c) \wedge \mathtt{O}(x,c)) \wedge \exists r(\mathtt{pCPT}(r) \wedge \mathtt{CF}(r,x) \wedge \mathtt{TOP}(r) \wedge \mathtt{SUM}(q,x,r)))$

The direct quality of an endurant or a perdurant (that does not overlap any concept) is the sum of such endurant/perdurant and a top concept (see (d_e40)) that classifies it. The condition $\neg\exists c(\mathtt{CPT}(c) \wedge \mathtt{O}(x,c))$ avoid qualities of concepts and it is necessary because in EMMO concepts are not necessarily disjoint from endurants or perdurants.

$d_{ed}16$ $\mathtt{QL}(r,q) \coloneqq \mathtt{pCPT}(r) \wedge \exists x(\mathrm{PD}(x) \wedge \mathtt{DQT}(q,x) \wedge \mathtt{MIN}(r,x) \wedge \mathtt{SPC}(r,q-x))$

The immediate quale of a perdurant x is the minimal concept (see (d_e41)) for x that classifies only perdurants and specializes the top concept part of a direct quality of x.

d_{ed}17 $\mathtt{tQL}(r,q,t) \coloneqq \mathtt{eCPT}(r) \wedge \exists xy(\mathrm{ED}(x) \wedge \mathtt{DQT}(q,x) \wedge \mathtt{TS}(y,x,t) \wedge \mathtt{MIN}(r,y) \wedge \mathtt{SPC}(r,q-x))$

The temporary quale of an endurant x at t is the minimal concept (see (d_e41)) for the temporal slice of x at t that classifies only endurants and specializes the top concept part of a direct quality of x.

d_{ed}18 $\mathrm{TQ}(x) \coloneqq \exists y(\mathrm{PD}(y) \wedge \mathtt{DQT}(x,y))$

Temporal qualities are direct qualities of perdurants.

d_{ed}19 $[\mathrm{PQ} \vee \mathrm{AQ}](x) \coloneqq \exists y(\mathrm{ED}(y) \wedge \mathtt{DQT}(x,y))$

Physical/Abstract qualities are direct qualities of perdurants. The lack of the PED/NPED distinction does not allow to separate physical from abstract qualities.

d_{ed}20 $\mathrm{Q}(x) \coloneqq \mathrm{TQ}(x) \vee [\mathrm{PQ} \vee \mathrm{AQ}](x)$

The category of qualities is just the union of the the categories defined in (d_{ed}18) and (d_{ed}19).

d_{ed}21 $\mathtt{EXD}(x,y,t) \coloneqq \mathtt{PRE}(x,t) \wedge \mathtt{PRE}(y,t) \wedge \bigvee_{\phi \neq \psi \in \mathbf{t}(\Delta)}(\phi(x) \wedge \psi(y) \wedge \mathtt{EDEP}\langle \phi, \psi \rangle)$

At time t the entity x existentially depends on the entity y when both x and y are present at t and they are classified by at least two types present in the extensions of EMMO in the EDEP relation.

d_{ed}22 $\mathrm{K}(x,y,t) \coloneqq ((\mathrm{ED}(x) \wedge \mathrm{ED}(y)) \vee (\mathrm{PD}(x) \wedge \mathrm{PD}(y))) \wedge \mathtt{CNC}(x,y,t) \wedge \bigvee_{\phi \neq \psi \in \mathbf{t}(\Delta)}(\phi(x) \wedge \psi(y) \wedge \mathtt{CDEP}\langle \psi, \phi \rangle) \wedge \bigwedge_{\phi \neq \psi \in \mathbf{t}(\Delta)}(\phi(x) \wedge \psi(y) \rightarrow \neg \mathtt{CDEP}\langle \phi, \psi \rangle)$

At time t the entity x constitutes the entity y when x and y are both endurants or both perdurants, they coincide at t, and x is classified by a type that is in the CDEP-relation with at least a type classifying y but the vice versa does not hold.

8.2 Check of the preservation of the original DOLCE axioms

As discussed in Sect. 8.1 some DOLCE-axioms involve relations or categories that have not been defined in the mappings. In what follows, when possible and relevant we consider approximations of these axioms as expressible in the EMMO language, otherwise we set them aside.

In the following, $\mathfrak{E}_d$ is the theory $\mathfrak{E}_d = \mathrm{EMMO}_{[\mathrm{MC}]} \cup \mathrm{EMMO}_{[\mathrm{SMS}]} \cup \mathrm{EMMO}_{[\mathrm{CPT}]} \cup \mathrm{EMMO}_{[\mathrm{TM}]} \cup \mathrm{EMMO}_{[\mathrm{TS}]} \cup \{(d_{ed}1)\text{-}(d_{ed}22)\}$, i.e., it includes the mereocausal module, the semiotic, concet, time, and temporal slice extensions, and the mappings introduced in Sect. 8.1.

Differently from thse DOLCE-BFO mapping, here we did not use theorem provers, i.e., all the proofs have been manually produced. There are two main reasons for that: (*i*) the notions introduced in the Time extension and in the Existence at a Time and Temporal Slice extension are heavily used in the following but the complexity of their definitions makes really difficult for authomatic provers to conclude a proof; (*ii*) EMMO

and all its extensions often use the fusion that is extremely powerful but again makes almost useless the theorems provers.

8.2.1 Mereology

(a_d3) is not considered because it uses S (not included in the mappings).

$t_{ed}1$ $\mathfrak{E}_d \vdash P_d(x,y) \rightarrow (PD(x) \wedge PD(y)) \vee (AB(x) \wedge AB(y))$ (a_d1)
Proof. Directly from ($d_{ed}12$), ($d_{ed}11$) and ($d_{ed}10$). □

$t_{ed}2$ $\mathfrak{E}_d \nvdash P_d(x,y) \rightarrow (T(x) \leftrightarrow T(y))$ (a_d2)

$\mathfrak{E}_d \nvdash P_d(x,y) \wedge T(x) \rightarrow T(y)$
Proof. Assume $TME(x) \wedge \neg TME(y) \wedge PD(x) \wedge PD(y) \wedge P(x,y)$. Then, by ($d_{ed}12$), $P_d(x,y) \wedge TME(x)$ but $\neg TME(y)$. □

$\mathfrak{E}_d \nvdash P_d(x,y) \wedge T(y) \rightarrow T(x)$
Proof. Assume $\neg TME(x) \wedge TME(y) \wedge PD(x) \wedge PD(y) \wedge P(x,y)$. Then, by ($d_{ed}12$), $P_d(x,y) \wedge TME(y)$ but $\neg TME(x)$. □

$t_{ed}3$ $\mathfrak{E}_d \vdash (PD(x) \vee AB(x)) \rightarrow P_d(x,x)$ (a_d4)
Proof. From $PD(x) \vee AB(x)$, by ($d_{ed}11$), ($d_{ed}10$), $PD(x) \vee T(x) \vee pCPT(x) \vee eCPT(x)$. For PDs and Ts the conclusion follows from ($d_{ed}12$) and (a_e1). For pCPTs and eCPTs it is enough to observe that SPC is trivially reflexive, see (d_e38). □

$t_{ed}4$ $\mathfrak{E}_d \vdash (PD(x) \vee T(x)) \wedge \neg pCPT(x) \wedge \neg eCPT(x) \wedge \neg pCPT(y) \wedge \neg eCPT(y) \wedge P_d(x,y) \wedge P_d(y,x) \rightarrow x = y$ ~(a_d5)
Proof. From $(PD(x) \vee T(x)) \wedge \neg pCPT(x) \wedge \neg eCPT(x) \wedge P_d(x,y)$, by ($d_{ed}12$), $(PD(y) \vee T(y)) \wedge P(x,y)$. From $(PD(y) \vee T(y)) \wedge \neg pCPT(y) \wedge \neg eCPT(y)$, by ($d_{ed}12$), $P(y,x)$. The thesis follows from (a_e2). □

$t_{ed}5$ $\mathfrak{E}_d \nvdash P_d(x,y) \wedge P_d(y,z) \rightarrow P_d(x,z)$ (a_d6)
Proof. Assume $TME(x) \wedge TME(y) \wedge \neg TME(z) \wedge \neg PD(x) \wedge PD(y) \wedge PD(z) \wedge \neg pCPT(x) \wedge \neg eCPT(x) \wedge P(x,y) \wedge P(y,z)$. From $TME(x) \wedge TME(y) \wedge P(x,y)$, by ($d_{ed}12$), $P_d(x,y)$. From $PD(y) \wedge PD(z) \wedge P(y,z)$, by ($d_{ed}12$), $P_d(y,z)$. However, from $TME(x) \wedge \neg TME(z) \wedge \neg PD(x) \wedge PD(z) \wedge \neg pCPT(x) \wedge \neg eCPT(x)$, by ($d_{ed}12$), $\neg P_d(x,z)$. □

$t_{ed}6$ $\mathfrak{E}_d \nvdash (AB(x) \vee PD(x)) \wedge \neg P_d(x,y) \rightarrow \exists z(P_d(z,x) \wedge \neg O_d(z,y))$ (a_d7)
Proof. Assume $PD(x) \wedge PD(y) \wedge \neg TME(x) \wedge \neg pCPT(x) \wedge \neg eCPT(x)$. From $\neg P_d(x,y)$, by ($d_{ed}12$), $\neg P(x,y)$ ($PD(x) \wedge PD(y) \wedge P(x,y)$ would imply $P_d(x,y)$) and therefore, by (a_e4), $\exists z(P(z,x) \wedge \neg O(z,y))$. However, nothing guarantees that $P(z,x) \wedge PD(x)$ implies $PD(z)$. Then if $\neg PD(z)$, from $\neg TME(x) \wedge \neg pCPT(x) \wedge \neg eCPT(x)$, by ($d_{ed}12$), $\neg P_d(z,x)$. □

8.2.2 Direct Quality

(a_d8) uses PED and NPED that are not included in the mapping; we prove the more general ($t_{ed}7$). (a_d11) uses TL that is not included in the mapping; we prove the more general ($t_{ed}10$). (a_d12) uses PED and SL that are not included in the mapping; we prove the more general ($t_{ed}11$).

t_{ed}7 $\mathfrak{E}_d \vdash \mathtt{DQT}(x,y) \rightarrow (\mathrm{TQ}(x) \wedge \mathrm{PD}(y)) \vee ([\mathrm{PQ} \vee \mathrm{AQ}](x) \wedge \mathrm{ED}(y))$ ~(a_d8)
Proof. Directly from the hypothesis, by (d_{ed}15), (d_{ed}18) and (d_{ed}19). □

t_{ed}8 $\mathfrak{E}_d \vdash \mathtt{DQT}(x,y) \wedge \mathtt{DQT}(x,z) \rightarrow y = z$ (a_d9)
Proof. From the hypotheses, by (d_{ed}15) and the fact that ED e PD are disjoint, $\exists rs(y+r = z+s \wedge \mathtt{eCPT}(r) \wedge \mathtt{eCPT}(s))$ or $\exists rs(y+r = z+s \wedge \mathtt{pCPT}(r) \wedge \mathtt{pCPT}(s))$. By contradiction assume $y \neq z$. Thus, by (a_e2) and (t_e1), [1] $\exists a(\mathtt{qP}(a,y) \wedge \neg\mathtt{qP}(a,z))$ or [2] $\exists a(\mathtt{qP}(a,z) \wedge \neg\mathtt{qP}(a,y))$. Assume [1]. From $y+r = z+s \wedge \mathtt{qP}(a,y) \wedge \neg\mathtt{qP}(a,z)$, by ($d_e$3), ($a_e$2) and ($t_e$1), $\mathtt{qP}(a,s)$ then $\exists a(\mathtt{qP}(a,y) \wedge \mathtt{qP}(a,s))$ and then $\mathtt{O}(y,s)$. If $\mathrm{PD}(y)$, by (d_{ed}15), we have that $\mathtt{pCPT}(s)$, and then $\mathtt{O}(y,s) \wedge \mathtt{pCPT}(s)$ against $\mathtt{DQT}(x,y)$. If $\mathrm{ED}(y)$, by (d_{ed}15), we have that $\mathtt{eCPT}(s)$, and then $\mathtt{O}(y,s) \wedge \mathtt{eCPT}(s)$ against $\mathtt{DQT}(x,y)$. Analogously in the case [2]. □

t_{ed}9 $\mathfrak{E}_d \vdash \mathrm{Q}(x) \rightarrow \exists y(\mathtt{DQT}(x,y))$ (a_d10)
Proof. Directly from the hypothesis, by (d_{ed}20), (d_{ed}18) and (d_{ed}19). □

t_{ed}10 $\mathfrak{E}_d \nvdash \mathrm{PD}(x) \leftrightarrow \exists y(\mathtt{DQT}(y,x) \wedge \mathrm{TQ}(y))$ ~(a_d11)
Proof. $\mathrm{PD}(x)$ does not entail $\exists z sim(\mathtt{SMS}(z,s,x,i,m))$ (to prove this theorem one needs to require that all EMMO's extensions' ϕs should be introduced as strictly connected with the semiotic perspective). □

t_{ed}11 $\mathfrak{E}_d \nvdash \mathrm{ED}(x) \leftrightarrow \exists y(\mathtt{DQT}(y,x) \wedge \mathrm{PQ}(y))$ ~(a_d12)
Proof. $\mathrm{ED}(x)$ does not entail $\exists z sim(\mathtt{SMS}(z,s,x,i,m))$ (to prove this theorem one needs to require that all EMMO's extensions' ϕs should be introduced as strictly connected with the semiotic perspective). □

8.2.3 Immediate Quale

Given the fact that times are not considered as qualia of temporal locations, (a_d15), (a_d16), and (a_d17) cannot be considered in the mapping.

t_{ed}12 $\mathfrak{E}_d \vdash \mathtt{QL}(x,y) \rightarrow \mathrm{TR}(x) \wedge \mathrm{TQ}(y)$ (a_d13)
Proof. From $\mathtt{QL}(x,y)$, by (d_{ed}16), $\mathtt{pCPT}(x) \wedge \exists z(\mathrm{PD}(z) \wedge \mathtt{DQT}(y,z) \wedge \mathtt{MIN}(x,z) \wedge \mathtt{SPC}(x,y-z))$. From $\mathtt{pCPT}(x)$, by (d_{ed}8), $\mathrm{TR}(x)$. From $\mathrm{PD}(z) \wedge \mathtt{DQT}(y,z)$, by ($d_{ed}$18), $\mathrm{TQ}(y)$. □

t_{ed}13 $\mathfrak{E}_d \nvdash \mathrm{TQ}(x) \rightarrow \exists y(\mathtt{QL}(y,x))$ (a_d14)
Proof. The conclusion follows by noticing that, while the hypotheses entail, by (d_{ed}18) and (d_{ed}15), $\exists z(\mathrm{PD}(z) \wedge \mathtt{DQT}(x,z) \wedge \exists r(\mathtt{CF}(r,z) \wedge \mathtt{TOP}(r) \wedge \mathtt{SUM}(x,z,r)))$, nothing ensures that there is no infinite chain $r_i, r_{i+1}, \ldots$ such that $\mathtt{pCPT}(r_i) \wedge \mathtt{pCPT}(r_{i+1}) \wedge \mathtt{CF}(r_i,z) \wedge \mathtt{CF}(r_{i+1},z) \wedge \mathtt{SPC}(r_i,r) \wedge \mathtt{SPC}(r_{i+1},r_i), \ldots$. □

8.2.4 Temporal Location and Present At

(t_{ed}14) and (t_{ed}16) show that, given (d_{ed}2), EX is strictly stronger than PRE (as defined in DOLCE in (d_d5)). (t_{ed}2) and (t_{ed}15) make evident the reason, i.e., the part of a time in not necessarily a time. This problem is mainly due to the fact that times are not necessarily disjoint from endurants, perdurants, or concepts. For this reason, several axioms of DOLCE cannot be preserved in their original form. However, minimal constraints guaranteeing the time-nature of some entities are enough to prove formulas that, from

an ontological perspective, are quite close to the original ones. We will follow this strategy along all the following sections.

Because times are not included among the qualia, also (a_d23) does not make sense and it is not considered. Furthermore, (a_d24) is not considered because it uses SL that is included in the mappings.

t_{ed}14 $\mathfrak{E}_d \vdash \mathtt{EX}(x,t) \rightarrow \mathtt{PRE}(x,t)$
Proof. By (d_e52), $\mathtt{TME}(t) \wedge \exists u(\mathtt{TME}(u) \wedge \mathtt{EQ}(x,u) \wedge \mathtt{P}(t,u))$. From $\exists u(\mathtt{TME}(u) \wedge \mathtt{EQ}(x,u))$, by ($d_{ed}$2), $\exists u(\mathtt{TLC}(x,u))$ and from $\mathtt{T}(u) \wedge \mathtt{T}(t) \wedge \mathtt{P}(t,u)$, by ($d_{ed}$12), $\mathtt{P_d}(t,u)$. The thesis follows directly from (d_d5). □

t_{ed}15 $\mathfrak{E}_d \nvdash \mathtt{PRE}(x,t) \rightarrow \mathtt{TME}(t)$
Proof. By (d_d5), $\exists u(\mathtt{TLC}(x,u) \wedge \mathtt{P_d}(t,u))$ and then, by (d_{ed}2), $\mathtt{T}(u) \wedge \mathtt{EQ}(x,u) \wedge \mathtt{P_d}(t,u)$. Assume, for instance, that $\mathtt{TME}(u) \wedge \mathtt{PD}(u) \wedge \neg\mathtt{TME}(t) \wedge \mathtt{PD}(t) \wedge \mathtt{P}(t,u)$. In this case we have $\mathtt{T}(u) \wedge \mathtt{P_d}(t,u)$ but $\neg\mathtt{TME}(t)$. □

t_{ed}16 $\mathfrak{E}_d \nvdash \mathtt{PRE}(x,t) \rightarrow \mathtt{EX}(x,t)$
Proof. Directly from (t_{ed}15) and (d_e52). □

t_{ed}17 $\mathfrak{E}_d \vdash \mathtt{TME}(t) \rightarrow (\mathtt{EX}(x,t) \leftrightarrow \mathtt{PRE}(x,t))$
Proof. ($\rightarrow$) Directly by (t_{ed}14). ($\leftarrow$) We have to prove that $\mathtt{TME}(t) \wedge \mathtt{PRE}(x,t) \rightarrow \mathtt{EX}(x,t)$. Following the proof in (t_{ed}15), by (d_d5) and (d_{ed}2), $\exists u(\mathtt{T}(u) \wedge \mathtt{EQ}(x,u) \wedge \mathtt{P_d}(t,u))$ and then, given the fact that t is a time, by (d_{ed}12), $\mathtt{TME}(t) \wedge \exists u(\mathtt{T}(u) \wedge \mathtt{EQ}(x,u) \wedge \mathtt{P}(t,u))$. The thesis follows directly from (d_e52). □

t_{ed}18 $\mathfrak{E}_d \nvdash \mathtt{TLC}(x,t) \rightarrow (\mathtt{ED}(x) \vee \mathtt{PD}(x) \vee \mathtt{Q}(x)) \wedge \mathtt{T}(t)$ (a_d18)
Proof. By (d_{ed}2), $\mathtt{T}(t) \wedge \mathtt{EQ}(x,t)$. This assures that t is a time, but there is nothing in EMMO excluding the possibility of having a concept that is temporally equivalent with a time t (and a subclass of concepts can in principle be disjoint from endurants, perdurants, and qualities). □

t_{ed}19 $\mathfrak{E}_d \vdash (\mathtt{ED}(x) \vee \mathtt{PD}(x) \vee \mathtt{Q}(x)) \rightarrow \exists t(\mathtt{TLC}(x,t))$ (a_d19)
Proof. It directly follows from the stronger (t_{ed}20). □.

t_{ed}20 $\mathfrak{E}_d \vdash \forall x \exists t(\mathtt{TLC}(x,t))$
Proof. By (t_e32), $\mathtt{EQ}(x,x)$ and then, by (a_e5), there exists $t = \sigma z\langle \mathtt{EQ}(z,x)\rangle$ that, by (t_e36), means that $\mathtt{EC}(t,x)$, i.e., by (d_e47) and (t_e38), $\mathtt{TME}(t) \wedge \mathtt{EQ}(t,x)$. □

t_{ed}21 $\mathfrak{E}_d \vdash \mathtt{TLC}(x,t) \wedge \mathtt{TLC}(x,u) \rightarrow t = u$ (a_d20)
Proof. Directly by (d_{ed}2) and (t_e66). □

t_{ed}22 $\mathfrak{E}_d \nvdash \mathtt{P_d}(x,y) \wedge \mathtt{TLC}(x,t) \wedge \mathtt{TLC}(y,u) \rightarrow \mathtt{P_d}(t,u)$ (a_d21)
Proof. Consider the model of EMMO in which $\mathtt{SMS}(z,s,o,i,x) \wedge \mathtt{SMS}(z',s',o,i,y) \wedge \neg\mathtt{O}(x,y) \wedge \mathtt{eCPT}(x) \wedge \mathtt{eCPT}(y) \wedge \mathtt{TLC}(x,t) \wedge \mathtt{TLC}(y,u) \wedge \mathtt{ED}(o)$. Since, trivially, $\mathtt{O}(i,i)$, then, by (d_e34), $\neg\mathtt{cEQ}(x,y)$. By assuming that only o enters a semiotic relation with x and y, by (d_e38), $\mathtt{SPC}(x,y)$ (and vice versa) therefore, by (d_{ed}9), (d_{ed}12) and the fact that x and y are eCPTs, $\mathtt{P_d}(x,y)$. However, there is no constraint on t and u (the temporal locations of x and y), actually the two concepts x and y can be temporally disjoint. □

t_{ed}23 $\mathfrak{E}_d \nvdash \mathtt{DQT}(x,y) \wedge \mathtt{TLC}(x,t) \wedge \mathtt{TLC}(y,u) \rightarrow \mathtt{P_d}(t,u)$ (a_d22)
Proof. From $\mathtt{DQT}(x,y)$, by (d_{ed}15) , $\exists r(\mathtt{epCPT}(r) \wedge x = y+r)$. Nothing guarantees that the temporal location of r is included in the one of y. □

8.2.5 Temporary Quale

(a_d25) and (a_d27) use the PR/AR and/or the PQ/AQ distinctions that are not considered in the mapping; we prove the more general (t_{ed}24) and (t_{ed}26). (a_d29), (a_d30), and (a_d31) are not considered because they use SL or S that is not included in the mapping.

t_{ed}24 $\mathfrak{E}_d \vdash \mathtt{tQL}(x,y,t) \rightarrow [\mathrm{PR}\vee\mathrm{AR}](x) \wedge [\mathrm{PQ}\vee\mathrm{AQ}](y) \wedge \mathrm{T}(t)$ $\sim(a_d25)$
Proof. From the hypothesis, by (d_{ed}17), (d_e55) and (d_e52), $\mathrm{T}(t)$. From $\mathtt{tQL}(x,y,t)$, by (d_{ed}17), $\mathtt{eCPT}(x)$ and then, by (d_{ed}9), $[\mathrm{PR}\vee\mathrm{AR}](x)$. From $\mathtt{tQL}(x,y,t)$, by (d_{ed}17) and (d_{ed}19), $[\mathrm{PQ}\vee\mathrm{AQ}](y)$. □

t_{ed}25 $\mathfrak{E}_d \vdash \mathtt{tQL}(r,q,t) \rightarrow \mathtt{PRE}(q,t)$ (a_d26)
Proof. From $\mathtt{tQL}(r,q,t)$, by (d_{ed}17), $\exists xy(\mathtt{DQT}(q,x) \wedge \mathtt{TS}(y,x,t))$. From $\mathtt{TS}(y,x,t)$, by (t_e93), $\mathtt{EX}(x,t)$. From $\mathtt{DQT}(q,x)$, by (d_{ed}15), $\mathtt{P}(x,q)$, thus, by (d_e52) and (a_e3), $\mathtt{EX}(q,t)$, and by (t_{ed}14), $\mathtt{PRE}(q,t)$. □

t_{ed}26 $\mathfrak{E}_d \nvdash [\mathrm{PQ}\vee\mathrm{AQ}](x) \wedge \mathtt{PRE}(x,t) \rightarrow \exists y(\mathtt{tQL}(y,x,t))$ $\sim(a_d27)$
Proof. It is enough to follow what done in the proof of (t_{ed}13). □

t_{ed}27 $\mathfrak{E}_d \nvdash \mathtt{tQL}(r,q,t) \wedge \mathtt{P}_d(u,t) \rightarrow \exists z(\mathtt{tQL}(z,q,u) \wedge \mathtt{P}_d(z,r))$ (a_d28)
Proof. The thesis follows immediately by noticing that it is required to show that $\exists zw(\mathtt{TS}(w,x,u) \wedge \mathtt{MIN}(z,w))$; however, as per the discussion of (t_{ed}10), it needn't be the case that $\exists z sim(\mathtt{SMS}(z,s,w,i,m))$. □

8.2.6 Temporary Mereology

As anticipated in Sect. 8.2.4, given (t_{ed}2), we need sometimes to introduce some additional hypotheses to prove the original axioms of DOLCE. (a_d39), (a_d40), (a_d41), and (a_d42) involve primitives that has not be mapped therefore they are not considered.

t_{ed}28 $\mathfrak{E}_d \vdash \mathtt{tP}_d(x,y,t) \rightarrow \mathrm{ED}(x) \wedge \mathrm{ED}(y) \wedge \mathrm{T}(t)$ (a_d32)
Proof. Directly from (d_{ed}13), (d_e55), and (d_e52). □

t_{ed}29 $\mathfrak{E}_d \vdash \mathtt{tP}_d(x,y,t) \rightarrow \mathtt{PRE}(x,t) \wedge \mathtt{PRE}(y,t)$ (a_d33)
Proof. From the hypothesis, by (d_{ed}13), (d_e55), (t_e93), and (d_e52), $\mathtt{EX}(x,t) \wedge \mathtt{EX}(y,t)$. The thesis follows from (t_{ed}14). □

t_{ed}30 $\mathfrak{E}_d \vdash \mathtt{tP}_d(x,y,t) \wedge \mathtt{P}(u,t) \wedge \mathtt{TME}(u) \rightarrow \mathtt{tP}_d(x,y,u)$ $\sim(a_d34)$
Proof. From the hypotheses, by (d_{ed}13), $\mathrm{ED}(x) \wedge \mathrm{ED}(y) \wedge \exists w(\mathtt{TS}(w,x,t) \wedge \mathtt{P}(w,y))$. From $\mathtt{TS}(w,x,t)$, by (t_e93), $\mathtt{EX}(x,t)$ and then, by (t_e62), $\mathtt{IN}(t,x)$. From $\mathtt{P}(u,t) \wedge \mathtt{IN}(t,x) \wedge \mathtt{TME}(u)$, by ($t_e$25) and ($t_e$24), we have $\mathtt{IN}(u,x) \wedge \mathtt{TME}(u)$ and then, by (t_e63), $\mathtt{EX}(x,u)$. From $\mathtt{EX}(x,u)$, by (t_e77), $\exists z(\mathtt{TS}(z,x,u))$. From $\mathtt{TS}(w,x,t) \wedge \mathtt{P}(u,t) \wedge \mathtt{TS}(z,x,u)$, by ($t_e$82), $\mathtt{P}(z,w)$. From $\mathtt{P}(z,w) \wedge \mathtt{P}(w,y)$, by ($a_e$3), $\mathtt{P}(z,y)$. Thus, $\mathrm{ED}(x) \wedge \mathrm{ED}(y) \wedge \exists z(\mathtt{TS}(z,x,u) \wedge \mathtt{P}(z,y))$, i.e., by ($d_{ed}$13), $\mathtt{tP}_d(x,y,u)$. □

t_{ed}31 $\mathfrak{E}_d \vdash \mathtt{tP}_d(x,y,t) \wedge \mathtt{tP}_d(x,y,u) \wedge \mathtt{TME}(t+u) \rightarrow \mathtt{tP}_d(x,y,t+u)$ $\sim(a_d35)$
Proof. From the hypotheses, by (d_{ed}13), $\mathrm{ED}(x) \wedge \mathrm{ED}(y) \wedge \exists z(\mathtt{TS}(z,x,t) \wedge \mathtt{P}(z,y)) \wedge \exists w(\mathtt{TS}(w,x,u) \wedge \mathtt{P}(w,y)) \wedge \mathtt{TME}(t+u)$ then, by (t_e87) and (d_e3), $\mathrm{ED}(x) \wedge \mathrm{ED}(y) \wedge \exists zw(\mathtt{TS}(z+w,x,t+u) \wedge \mathtt{P}(z+w,y))$. The thesis follows directly by (d_{ed}13). □

t_{ed}32 $\mathfrak{E}_d \vdash \mathrm{ED}(x) \wedge \mathtt{PRE}(x,t) \wedge \mathtt{TME}(t) \rightarrow \mathtt{tP}_d(x,x,t)$ $\sim(a_d36)$

Proof. From $\mathtt{PRE}(x,t) \wedge \mathtt{TME}(t)$, by ($t_{ed}$17), $\mathtt{EX}(x,t)$ and then, by (t_e77) and (t_e80), $\exists y(\mathtt{TS}(y,x,t) \wedge \mathtt{P}(y,x))$. The thesis follows directly by (d_{ed}13). □

t_{ed}33 $\mathfrak{E}_d \vdash \mathtt{tP}_d(x,y,t) \wedge \mathtt{tP}_d(y,z,t) \rightarrow \mathtt{tP}_d(x,z,t)$ (a_d37)

Proof. From $\mathtt{tP}_d(x,y,t) \wedge \mathtt{tP}_d(y,z,t)$, by ($d_{ed}$13), $\mathtt{ED}(x) \wedge \mathtt{ED}(y) \wedge \exists v(\mathtt{TS}(v,x,t) \wedge \mathtt{P}(v,y)) \wedge \mathtt{ED}(y) \wedge \mathtt{ED}(z) \wedge \exists w(\mathtt{TS}(w,y,t) \wedge \mathtt{P}(v,z))$. From $\mathtt{TS}(v,x,t) \wedge \mathtt{TS}(w,y,t) \wedge \mathtt{P}(v,y)$, by ($t_e$86), $\mathtt{P}(v,w)$. From $\mathtt{P}(v,w)$ and $\mathtt{P}(w,z)$, by (a_e3), $\mathtt{P}(v,z)$. Then, $\mathtt{ED}(x) \wedge \mathtt{ED}(z) \wedge \exists v(\mathtt{TS}(v,x,t) \wedge \mathtt{P}(v,z))$, i.e., by ($d_{ed}$13), $\mathtt{tP}_d(x,z,t)$. □

t_{ed}34 $\mathfrak{E}_d \nvdash \mathtt{ED}(x) \wedge \mathtt{ED}(y) \wedge \mathtt{PRE}(x,t) \wedge \mathtt{PRE}(y,t) \wedge \mathtt{TME}(t) \wedge \neg \mathtt{tP}_d(x,y,t) \rightarrow \exists z(\mathtt{tP}_d(z,x,t) \wedge \neg \mathtt{tO}_d(z,y,t))$ ∼(a_d38)

Consider the model of EMMO with just four quanta such that $\mathtt{dC}(1,3) \wedge \mathtt{dC}(1,4) \wedge \mathtt{dC}(2,3) \wedge \mathtt{dC}(2,4)$ and suppose that $t = x = 1{+}2{+}3{+}4$, $y = 1{+}2{+}3$, and $\mathtt{ED}(x) \wedge \mathtt{ED}(y)$. It is easy to see that, if there are no other Q beside 1, 2, 3, 4, then $\mathtt{TME}(t) \wedge \mathtt{EQ}(x,x) \wedge \mathtt{EQ}(y,x) \wedge \mathtt{TS}(x,x,t)$ and then, by (t_{ed}17), (d_e52), and (d_{ed}13), $\mathtt{PRE}(x,t) \wedge \mathtt{PRE}(y,t) \wedge \neg \mathtt{tP}_d(x,y,t)$. Given that all the temporal slices of an endurant are endurants and the fact that the only times in the model are $1{+}2$, $3{+}4$ and $1{+}2{+}3{+}4$, then in addition to x and y, $1{+}2$, $3{+}4$, and 3 are also endurants. Assume that all the other sums of quanta are not endurants. $1{+}2$, $3{+}4$, and 3 do not exists at t, therefore the only potential z s.t., $\mathtt{tP}_d(z,x,t)$ are x and y but we have that $\mathtt{tO}_d(x,y,t) \wedge \mathtt{tO}_d(y,y,t)$ because $\mathtt{tP}_d(y,x,t)$ ($\mathtt{TS}(y,y,t) \wedge \mathtt{P}(y,x)$) and $\mathtt{tP}_d(y,y,t)$. □

8.2.7 Constitution

(a_d43) uses PED and NPED that are not included in the mapping; we prove the more general (t_{ed}35).

t_{ed}35 $\mathfrak{E}_d \vdash \mathtt{K}(x,y,t) \rightarrow ((\mathtt{ED}(x) \wedge \mathtt{ED}(y)) \vee (\mathtt{PD}(x) \wedge \mathtt{PD}(y))) \wedge \mathtt{T}(t)$ ∼(a_d43)

Proof. From the hypothesis, by (d_{ed}22), $((\mathtt{ED}(x) \wedge \mathtt{ED}(y)) \vee (\mathtt{PD}(x) \wedge \mathtt{PD}(y))) \wedge \mathtt{CNC}(x,y,t)$. From $\mathtt{CNC}(x,y,t)$, by (d_e58), $\exists z(\mathtt{TS}(z,x,t))$ and then, by (d_e55) and (d_e52), $\mathtt{TME}(t)$. □

t_{ed}36 $\mathfrak{E}_d \vdash \mathtt{K}(x,y,t) \rightarrow \mathtt{PRE}(x,t) \wedge \mathtt{PRE}(y,t)$ (a_d44)

Proof. From the hypothesis, by (d_{ed}22), (d_e58), (t_e93), $\mathtt{EX}(x,t) \wedge \mathtt{EX}(y,t)$. The thesis follows from (t_{ed}14). □

t_{ed}37 $\mathfrak{E}_d \vdash \mathtt{K}(x,y,t) \wedge \mathtt{P}(u,t) \wedge \mathtt{TME}(u) \rightarrow \mathtt{K}(x,y,u)$ ∼(a_d45)

Proof. Given that $\mathtt{K}(x,y,t)$, by (d_{ed}22), we need to prove that $\mathtt{CNC}(x,y,t) \wedge \bigvee_{\phi \neq \psi \in \mathbf{t}(\Delta)}(\phi(x) \wedge \psi(y) \wedge \mathtt{CDEP}\langle \psi, \phi \rangle) \wedge \bigwedge_{\phi \neq \psi \in \mathbf{t}(\Delta)}(\phi(x) \wedge \psi(y) \rightarrow \neg \mathtt{CDEP}\langle \phi, \psi \rangle) \wedge \mathtt{P}_d(u,t) \rightarrow \mathtt{CNC}(x,y,u) \wedge \bigvee_{\phi \neq \psi \in \mathbf{t}(\Delta)}(\phi(x) \wedge \psi(y) \wedge \mathtt{CDEP}\langle \psi, \phi \rangle) \wedge \bigwedge_{\phi \neq \psi \in \mathbf{t}(\Delta)}(\phi(x) \wedge \psi(y) \rightarrow \neg \mathtt{CDEP}\langle \phi, \psi \rangle)$. We just need to show that $\mathtt{CNC}(x,y,t) \wedge \mathtt{P}(u,t) \wedge \mathtt{TME}(u) \rightarrow \mathtt{CNC}(x,y,u)$. The thesis follows directly from (t_e98). □

t_{ed}38 $\mathfrak{E}_d \vdash \mathtt{K}(x,y,t) \wedge \mathtt{K}(x,y,u) \wedge \mathtt{TME}(t{+}u) \rightarrow \mathtt{K}(x,y,t{+}u)$ ∼(a_d46)

Proof. Given that $\mathtt{K}(x,y,t)$, by (d_{ed}22), we have to prove that $\mathtt{CNC}(x,y,t) \wedge \bigvee_{\phi \neq \psi \in \mathbf{t}(\Delta)}(\phi(x) \wedge \psi(y) \wedge \mathtt{CDEP}\langle \psi, \phi \rangle) \wedge \bigwedge_{\phi \neq \psi \in \mathbf{t}(\Delta)}(\phi(x) \wedge \psi(y) \rightarrow \neg \mathtt{CDEP}\langle \phi, \psi \rangle) \wedge \mathtt{CNC}(x,y,u) \wedge \bigvee_{\phi \neq \psi \in \mathbf{t}(\Delta)}(\phi(x) \wedge \psi(y) \wedge \mathtt{CDEP}\langle \psi, \phi \rangle) \wedge \bigwedge_{\phi \neq \psi \in \mathbf{t}(\Delta)}(\phi(x) \wedge \psi(y) \rightarrow \neg \mathtt{CDEP}\langle \phi, \psi \rangle) \wedge \mathtt{TME}(t{+}u) \rightarrow \mathtt{CNC}(x,y,t{+}u) \wedge \bigvee_{\phi \neq \psi \in \mathbf{t}(\Delta)}(\phi(x) \wedge \psi(y) \wedge \mathtt{CDEP}\langle \psi, \phi \rangle) \wedge$

$\bigwedge_{\phi\neq\psi\in\mathbf{t}(\Delta)}(\phi(x)\wedge\psi(y)\rightarrow\neg\mathtt{CDEP}\langle\phi,\psi\rangle)$. We just need to show that $\mathtt{CNC}(x,y,t)\wedge\mathtt{CNC}(x,y,u)\wedge\mathtt{TME}(t{+}u)\rightarrow\mathtt{CNC}(x,y,t{+}u)$ that follows from (t_e99). □

t_{ed}39 $\mathfrak{E}_d\nvdash\mathtt{K}(x,y,u)\wedge\mathtt{PRE}(y,t)\rightarrow\exists zv(\mathtt{P}_d(v,t)\wedge\mathtt{K}(z,y,v))$ (a_d47)
Proof. Nothing guarantees that z is an endurant or a perdurant, e.g., all the entities (different from y) containing the temporal part of y at v could be instances of no $\phi\in\mathbf{t}(\Delta)$. □

t_{ed}40 $\mathfrak{E}_d\vdash\mathtt{K}(x,y,t)\rightarrow\neg\mathtt{K}(y,x,t)$ (a_d48)
Proof. Directly by (d_{ed}22). □

t_{ed}41 $\mathfrak{E}_d\nvdash\mathtt{K}(x,y,t)\wedge\mathtt{K}(y,z,t)\rightarrow\mathtt{K}(x,z,t)$ (a_d49)
Proof. Consider a model of EMMO s.t. $\mathtt{CDEP}\langle\phi,\psi\rangle\wedge\mathtt{CDEP}\langle\chi,\tau\rangle\wedge\neg\mathtt{CDEP}\langle\psi,\tau\rangle$ and $\phi(a)\wedge\psi(b)\wedge\mathtt{K}(b,a,t)\wedge\chi(b)\wedge\tau(c)\wedge\mathtt{K}(c,b,t)\wedge\chi(d)\wedge\tau(e)\wedge\mathtt{K}(e,d,u)$. d needn't be s.t. $\psi(d)$, and, thus, nothing ensures that there exists a f s.t. $\phi(f)\wedge\mathtt{K}(f,d,u)$. Hence, $\neg\mathtt{CDEP}\langle\phi,\tau\rangle$, and $\neg\mathtt{K}(x,z,t)$. □

t_{ed}42 $\mathfrak{E}_d\nvdash\mathtt{K}(x,y,t)\wedge\mathtt{tPP}_d(z,y,t)\rightarrow\exists w(\mathtt{tPP}_d(w,x,t)\wedge\mathtt{K}(w,z,t))$ (a_d50)
Proof. $\mathtt{K}(x,y,t)\wedge\mathtt{tPP}_d(z,y,t)$ ensures that $\exists s(\mathtt{TS}(s,y,t)\wedge\mathtt{P}(z,s))$ and, thus $\mathtt{EX}(s,t)$. y is an endurant and thus $\phi(y)\rightarrow\phi(s)$. However, it needn't be the case that $\phi(z)$ (that would be ensured only by assuming $z=s$). In fact, there might be no $\psi\in\mathbf{t}(\Delta)$ such that $\psi(z)$. Given that, $\mathtt{K}(w,z,t)$ does not necessarily hold. □

t_{ed}43 $\mathfrak{E}_d\nvdash\mathtt{K}(x,y,t)\wedge\mathtt{K}(z,y,t)\rightarrow(x=z\vee\mathtt{K}(x,z,t)\vee\mathtt{K}(z,x,t))$ (a_d51)
Proof. Consider a model of EMMO in which $\phi(a)\wedge\psi(a)\wedge\chi(a)\wedge\mathtt{EC}(t,a)\wedge\mathtt{EC}(u,b)\wedge\mathtt{EC}(v,c)\wedge\psi(a{+}b)\wedge\psi(b)\wedge\chi(a{+}c)\wedge\chi(c)\wedge\mathtt{pC}(t,u)\wedge\mathtt{pC}(u,v)$ and nothing else is in the extensions of ϕ, ψ and χ. First notice that $a{+}b$, $a{+}c$, as well as all the quanta, are endurants. Second, trivially, $\neg\mathtt{O}(a,b)\wedge\neg\mathtt{O}(a,c)\wedge\neg\mathtt{O}(b,c)$ as they are equivalent to non-overlapping TMEs (i.e., maximal sums of EQ entities), whereas t, u, v cannot overlap given the pC relations among them. It is the case that $\mathtt{K}(a{+}b,a,t)\wedge\mathtt{K}(a{+}c,a,t)$. However, $a{+}b\neq a{+}c$ by hypothesis, and $\neg\mathtt{K}(a{+}c,a{+}b,t)\wedge\neg\mathtt{K}(a{+}b,a{+}c,t)$ because $\chi(a{+}c)\wedge\psi(a{+}b)$, no other type classifies $a{+}c$, no other type classifies $a{+}b$, but $\neg\mathtt{CDEP}\langle\psi,\chi\rangle$ ($\psi(b)\wedge\mathtt{EX}(b,u)$ but all the entities that satisfy χ do not exist at u) and $\neg\mathtt{CDEP}\langle\chi,\psi\rangle$ ($\chi(c)\wedge\mathtt{EX}(c,v)$ but all the entities that satisfy ψ do not exist at v). □

8.2.8 Participation

t_{ed}44 $\mathfrak{E}_d\vdash\mathtt{PC}(x,y,t)\rightarrow\mathrm{ED}(x)\wedge\mathrm{PD}(y)\wedge\mathrm{T}(t)$ (a_d52)
Proof. Directly from (d_{ed}14), (d_e55), and (d_e52). □

t_{ed}45 $\mathfrak{E}_d\vdash\mathtt{PC}(x,y,t)\rightarrow\mathtt{PRE}(x,t)\wedge\mathtt{PRE}(y,t)$ (a_d53)
Proof. From $\mathtt{PC}(x,y,t)$, by (d_{ed}14), $\mathrm{ED}(x)\wedge\mathrm{PD}(y)\wedge\exists z(\mathtt{TS}(z,x,t)\wedge\mathtt{P}(z,y))$. From $\mathtt{TS}(z,x,t)$, by (d_e55) and (t_e93), $\mathtt{EX}(x,t)\wedge\mathtt{EX}(z,t)$. From $\mathtt{EX}(z,t)\wedge\mathtt{P}(z,y)$, by ($t_e$64), $\mathtt{EX}(y,t)$. The thesis follows directly from (t_{ed}14). □

t_{ed}46 $\mathfrak{E}_d\vdash\mathtt{PC}(x,y,t)\wedge\mathtt{P}(u,t)\wedge\mathtt{TME}(u)\rightarrow\mathtt{PC}(x,y,u)$ ∼(a_d54)
Proof. The proof is analogous to that for (t_{ed}30), since there are no relevant dissimilarities between (d_{ed}14) and (d_{ed}13). □

t_{ed}47 $\mathfrak{E}_d\vdash\mathtt{PC}(x,y,t)\wedge\mathtt{PC}(x,y,u)\wedge\mathtt{TME}(t{+}u)\rightarrow\mathtt{PC}(x,y,t{+}u)$ ∼(a_d55)

Proof. From the hypotheses, by (d_{ed}14), $\text{ED}(x) \wedge \text{PD}(y) \wedge \exists z(\text{TS}(z,x,t) \wedge \text{P}(z,y)) \wedge \exists w(\text{TS}(w,x,u) \wedge \text{P}(w,y)) \wedge \text{TME}(t+u)$ then, by (t_e87) and (d_e3), $\text{ED}(x) \wedge \text{PD}(y) \wedge \exists zw(\text{TS}(z+w,x,t+u) \wedge \text{P}(z+w,y))$. The thesis follows directly by (d_{ed}14). □

t_{ed}48 $\mathfrak{E}_d \nvdash \text{PD}(x) \wedge \text{PRE}(x,t) \rightarrow \exists yu(\text{P}_d(u,t) \wedge \text{PC}(y,x,u))$ (a_d56)
Proof. We have to prove, by (d_{ed}14), that $\exists uy(\text{ED}(y) \wedge \exists z(\text{TS}(z,y,u) \wedge \text{P}(z,x)))$. Nothing guarantees the existence of such z; for instance all the temporal slices of x at u could be classified by no $\phi \in \mathbf{t}(\Delta)$. □

t_{ed}49 $\mathfrak{E}_d \nvdash \text{ED}(x) \wedge \text{PRE}(x,t) \rightarrow \exists yu(\text{P}_d(u,t) \wedge \text{PC}(x,y,u))$ (a_d57)
Proof. We have to prove, by (d_{ed}14), that $\exists uy(\text{PD}(y) \wedge \exists z(\text{TS}(z,x,u) \wedge \text{P}(z,y)))$ but nothing guarantees the existence of a perdurant in EMMO. For instance, consider a model with only a quantum. □

t_{ed}50 $\mathfrak{E}_d \vdash \text{PC}(x,y,t) \wedge \text{P}_d(y,z) \wedge \text{PD}(z) \rightarrow \text{PC}(x,z,t)$ $\sim$(a_d58)
Proof. From the hypotheses, by (d_{ed}14) and (d_{ed}12), $\exists a(\text{TS}(a,x,t) \wedge \text{P}(a,y))$ and $\text{PD}(y) \wedge \text{PD}(z) \wedge \text{P}(y,z)$. By ($a_e$3), $\text{P}(a,z) \wedge \text{PD}(z)$. □

8.2.9 Existential Dependence

(a_d66)-(a_d73) are not considered because they involve categories of DOLCE that are excluded from the mapping.

t_{ed}51 $\mathfrak{E}_d \vdash \text{EXD}(x,y,t) \rightarrow \text{PRE}(x,t) \wedge \text{PRE}(y,t)$ (a_d59)
Proof. Directly by (d_{ed}21). □

t_{ed}52 $\mathfrak{E}_d \vdash \text{EXD}(x,y,t) \wedge \text{P}(u,t) \wedge \text{TME}(u) \wedge \text{TME}(t) \rightarrow \text{EXD}(x,y,u)$ $\sim$(a_d60)
Proof. From $\text{EXD}(x,y,t)$, by (d_{ed}21), $\text{PRE}(x,t) \wedge \text{PRE}(y,t)$ and then, by (d_d5) and (d_{ed}2), $\exists vw(\text{TLC}(x,v) \wedge \text{TLC}(y,w) \wedge \text{P}_d(t,v) \wedge \text{P}_d(t,w) \wedge \text{TME}(v) \wedge \text{TME}(w))$. From $\text{P}_d(t,v) \wedge \text{P}_d(t,w) \wedge \text{TME}(t) \wedge \text{TME}(v) \wedge \text{TME}(w)$, by ($d_{ed}$12), $\text{P}(t,v) \wedge \text{P}(t,w)$ that, given that $\text{P}(u,t)$, by (a_e3), implies $\text{P}(u,v) \wedge \text{P}(u,w)$ and then, given that $\text{TME}(u) \wedge \text{TME}(v) \wedge \text{TME}(w)$, by ($d_{ed}$12), $\text{P}_d(u,v) \wedge \text{P}_d(u,w)$. From $\exists vw(\text{TLC}(x,v) \wedge \text{TLC}(y,w) \wedge \text{P}_d(u,v) \wedge \text{P}_d(u,w))$, by ($d_d$5), $\text{PRE}(x,u) \wedge \text{PRE}(y,u)$. The thesis follows by observing that the last condition in (d_{ed}21), i.e., $\bigvee_{\phi \neq \psi \in \mathbf{t}(\Delta)}(\phi(x) \wedge \psi(y) \wedge \text{EDEP}\langle \phi, \psi \rangle)$, still holds because it is independent from the time considered. □

t_{ed}53 $\mathfrak{E}_d \vdash \text{EXD}(x,y,t) \wedge \text{EXD}(x,y,u) \wedge \text{TME}(t) \wedge \text{TME}(u) \wedge \text{TME}(t+u) \rightarrow \text{EXD}(x,y,t+u)$ $\sim$(a_d61)
Proof. From $\text{EXD}(x,y,t)$, by (d_{ed}21), $\text{PRE}(x,t) \wedge \text{PRE}(y,t)$ and then, by (d_d5) and (d_{ed}2), $\exists vw(\text{TLC}(x,v) \wedge \text{TLC}(y,w) \wedge \text{P}_d(t,v) \wedge \text{P}_d(t,w) \wedge \text{TME}(v) \wedge \text{TME}(w))$. From $\text{P}_d(t,v) \wedge \text{P}_d(t,w) \wedge \text{TME}(t) \wedge \text{TME}(v) \wedge \text{TME}(w)$, by ($d_{ed}$12), $\text{P}(t,v) \wedge \text{P}(t,w)$. From $\text{EXD}(x,y,u)$, by (d_{ed}21), $\text{PRE}(x,u) \wedge \text{PRE}(y,u)$ and then, by (d_d5), (t_{ed}21) and (d_{ed}2), $\text{TLC}(x,v) \wedge \text{TLC}(y,w) \wedge \text{P}_d(u,v) \wedge \text{P}_d(u,w)$. From $\text{P}_d(u,v) \wedge \text{P}_d(u,w) \wedge \text{TME}(u) \wedge \text{TME}(v) \wedge \text{TME}(w)$, by ($d_{ed}$12), $\text{P}(u,v) \wedge \text{P}(u,w)$. Thus, from $\text{P}(t,v) \wedge \text{P}(t,w) \wedge \text{P}(u,v) \wedge \text{P}(u,w)$, by ($d_e$3), $\text{P}(t+u,v) \wedge \text{P}(t+u,w)$, and then given that $\text{TME}(t+u) \wedge \text{TME}(v) \wedge \text{TME}(w)$, by ($d_{ed}$12), $\text{P}_d(t+u,v) \wedge \text{P}_d(t+u,w)$. From the fact that $\exists vw(\text{TLC}(x,v) \wedge \text{TLC}(y,w) \wedge \text{P}_d(v,v) \wedge \text{P}_d(v,w))$, by ($d_d$5), $\text{PRE}(x,u) \wedge \text{PRE}(y,u)$. The thesis directly follows by observing that the last condition in (d_{ed}21), namely $\bigvee_{\phi \neq \psi \in \mathbf{t}(\Delta)}(\phi(x) \wedge \psi(y) \wedge \text{EDEP}\langle \phi, \psi \rangle)$, still holds because it is independent from the specific time considered. □

t_{ed}54 $\mathfrak{E}_d \vdash \mathtt{EXD}(x,y,u) \wedge \mathtt{PRE}(x,t) \wedge \mathtt{TME}(t) \rightarrow \exists z v(\mathtt{P_d}(v,t) \wedge \mathtt{EXD}(x,z,v))$ ~(a_d62)
Proof. From $\mathtt{EXD}(x,y,u)$, by (d_{ed}21), $\bigvee_{\phi \neq \psi \in \mathbf{t}(\Delta)}(\phi(x) \wedge \psi(y) \wedge \mathtt{EDEP}\langle\phi,\psi\rangle)$. From $\mathtt{PRE}(x,t) \wedge \mathtt{TME}(t) \wedge \phi(x) \wedge \mathtt{EDEP}\langle\phi,\psi\rangle$, by ($d_e$60) and ($t_{ed}$17), $\exists z(\psi(z) \wedge \mathtt{PRE}(z,t))$ and then, by (d_{ed}21), $\exists z(\mathtt{EXD}(x,z,t))$ (the other conditions to have $\mathtt{EXD}(x,z,t)$ are trivially satisfied). From $\mathtt{TME}(t)$, by (d_{ed}12) and (a_e1), $\mathtt{P_d}(t,t)$. It is enough to consider $v = t$, i.e., $\exists z t(\mathtt{P_d}(t,t) \wedge \mathtt{EXD}(x,z,t))$. □

t_{ed}55 $\mathfrak{E}_d \vdash \mathtt{K}(x,y,t) \rightarrow \mathtt{EXD}(y,x,t)$ (a_d63)
Proof. Directly by observing that CDEP implies EDEP. □

t_{ed}56 $\mathfrak{E}_d \nvdash \mathtt{PC}(x,y,t) \rightarrow \mathtt{EXD}(x,y,t) \wedge \mathtt{EXD}(y,x,t)$ (a_d64)
Proof. Nothing guarantees that the condition $\bigvee_{\phi \neq \psi \in \mathbf{t}(\Delta)}(\phi(x) \wedge \psi(y) \wedge \mathtt{EDEP}\langle\phi,\psi\rangle)$ of (d_{ed}21) is satisfied. □

t_{ed}57 $\mathfrak{E}_d \nvdash \mathtt{DQT}(x,y) \rightarrow \mathtt{SD}(x,y)$ (a_d65)
Proof. From $\mathtt{DQT}(x,y)$, by (d_{ed}15), $\exists r(\mathtt{SUM}(x,r,y))$ and then, by (d_e3), $\mathtt{P}(r,x)$. Nothing guarantees that the temporal location of r coincides with the one of y (for instance r could have as parts some interpretants that do not apply to y and are temporally disjoint from the interpretant that applies to y). Assume then that $\mathtt{TME}(u) \wedge \mathtt{EX}(r,u) \wedge \neg\mathtt{EX}(y,u)$. From $\mathtt{EX}(r,u) \wedge \mathtt{P}(r,x)$, by ($t_e$64), $\mathtt{EX}(x,u)$. Then, by $\mathtt{EX}(x,u) \wedge \neg\mathtt{EX}(y,u) \wedge \mathtt{TME}(u)$, by ($t_{ed}$17), $\mathtt{PRE}(x,u) \wedge \neg\mathtt{PRE}(y,u)$ that, by (d_{ed}21), implies $\neg\mathtt{EXD}(x,y,u) \wedge \mathtt{PRE}(x,u)$ and then, by (d_d9), $\neg\mathtt{SD}(x,y)$. □

8.2.10 Taxonomy

We consider only the constrains on the categories included in the mapping.

t_{ed}58 $\mathfrak{E}_d \nvdash \mathrm{AB}(x) \vee \mathrm{ED}(x) \vee \mathrm{PD}(x) \vee \mathrm{Q}(x)$ (a_d89)
Proof. Consider a sum of quanta that is not a time, has no (proper or improper) part that enters in a semiotic process, and is classified by no $\phi \in \mathbf{t}(\Delta)$. □

t_{ed}59 $\mathfrak{E}_d \vdash (\mathrm{TQ}(x) \vee [\mathrm{PQ} \vee \mathrm{AQ}](x)) \leftrightarrow \mathrm{Q}(x)$ ~(a_d93)
Proof. Directly from (d_{ed}20). □

t_{ed}60 $\mathfrak{E}_d \vdash (\mathrm{TR}(x) \vee [\mathrm{PR} \vee \mathrm{AR}](x)) \leftrightarrow \mathrm{R}(x)$ ~(a_d99)
Proof. Directly from (d_{ed}10). □

t_{ed}61 $\mathfrak{E}_d \vdash \mathrm{R}(x) \rightarrow \mathrm{AB}(x)$ ~(a_d100)
Proof. Directly from (d_{ed}11). □

t_{ed}62 $\mathfrak{E}_d \vdash \mathrm{T}(x) \rightarrow \mathrm{TR}(x)$ ~(a_d105)
Proof. Directly from (d_{ed}8). □

t_{ed}63 $\mathfrak{E}_d \nvdash \neg\exists x(\mathrm{AB}(x) \wedge \mathrm{ED}(x))$ (a_d107)
Proof. Nothing prevent a time to also be an endurant. □

t_{ed}64 $\mathfrak{E}_d \nvdash \neg\exists x(\mathrm{AB}(x) \wedge \mathrm{PD}(x))$ (a_d108)
Proof. Nothing prevent a time to also be an perdurant. □

t_{ed}65 $\mathfrak{E}_d \nvdash \neg\exists x(\mathrm{AB}(x) \wedge \mathrm{Q}(x))$ (a_d109)
Proof. Nothing prevent a time to also be a quality. □

t_{ed}66 $\mathfrak{E}_d \vdash \neg\exists x(\mathrm{ED}(x) \wedge \mathrm{PD}(x))$ (a_d110)
Proof. Directly by (d_{ed}3), (d_{ed}4), (d_e56), and (d_e57). □

t_{ed}67 $\mathfrak{E}_d \nvdash \neg\exists x(\mathrm{PD}(x) \wedge \mathrm{Q}(x))$ (a_d111)
Proof. Perdurants are defined in terms of temporal non-dissectivity while qualities in terms of the semiosis (they are sums of (top) concepts and entities classified by these concepts). The two dimensions are not connected. □

t_{ed}68 $\mathfrak{E}_d \nvdash \neg\exists x(\mathrm{ED}(x) \wedge \mathrm{Q}(x))$ (a_d112)
Proof. Endurants are defined in terms of temporal dissectivity while qualities in terms of the semiosis (they are sums of (top) concepts and entities classified by these concepts). The two dimensions are not connected. □

t_{ed}69 $\mathfrak{E}_d \vdash \neg\exists x(\mathrm{TQ}(x) \wedge [\mathrm{PQ} \vee \mathrm{AQ}](x))$ $\sim$(a_d126), $\sim$(a_d128)
Proof. Directly from (d_{ed}18), (d_{ed}19), (d_{ed}15), and (t_{ed}66). □

t_{ed}70 $\mathfrak{E}_d \nvdash \neg\exists x(\mathrm{TR}(x) \wedge [\mathrm{PR} \vee \mathrm{AR}](x))$ $\sim$(a_d129), $\sim$(a_d131)
Proof. Times in principle could also be concepts that classify only endurants. □

8.3 Analysis

Following the methodology discussed in Sect. 5, we start by 'importing' the whole domain of EMMO into DOLCE. However, (t_{ed}56) shows that, given the mappings introduced in Sect. 8.1, not all the entities of EMMO fall under the categories of DOLCE, i.e., EMMO allows for entities that are unclassifiable in terms of the original DOLCE system of categories when these categories are defined as in the proposed mappings. Clearly, the adoption of less restrictive mappings could mitigate the problem. As discussed in Sect. 5, mappings can be iteratively refined and the widening of the scope of some of the definitions considered in Sect. 8.1 can be the object of further investigations as far as the mappings, in addition to preserve the axioms of DOLCE, remain ontologically plausible. This situation is further complicated by (t_{ed}20) (and (t_{ed}18)) that states that all the entities in EMMO, included the ones mapped to the DOLCE category AB, are present in time. Given the physicalist commitment of EMMO the only choice to build abstract entities is to reduce them to physical sums of quanta, as done for times and concepts.

A second, more vexing, problem concerns the fact that some categories do not result disjoint. While ED and PD are disjoint (see (t_{ed}66)), AB is not disjoint form ED, PD, and Q (see (t_{ed}63), (t_{ed}64), and (t_{ed}65)) and Q is not disjoint from PD and ED (see (t_{ed}67) and (t_{ed}68)). In particular, times (TME-instances) or concepts (see (d_e36)) can also be endurants or perdurants. This is mainly due to the general strategy considered to define these categories: (i) times are defined in terms of mereocausality; (ii) endurants and perdurants are defined in terms of dissectivity or non-dissectivity with respect to the types ϕ introduced in the extensions of EMMO ($\phi \in \mathbf{t}(\Delta)$); (iii) concepts are defined taking the semiotic perspective as a starting point. In EMMO, at the moment, these three dimensions of classification of sum of quanta are independent, i.e., no axiom guarantees the applicability of only one of these dimensions to a given entity. This is a direct result of EMMO's multiperspective approach. While (i) is a purely ontological construction, (ii) and (iii) can be seen as more conceptual or epistemological constructions because they depend on semiotic processes or on the views encapsulated in the extensions of EMMO. In this direction, the possibility to see extensions in terms

of semiotic processes, i.e., to include the $\phi \in \mathbf{t}(\Delta)$ in the class of concepts, is an interesting perspective that, however, requires additional work, and could not be considered in this project.

The possible overlap between the main categories of DOLCE introduces some formal problems when relations defined on several categories are taken into account. For instance, consider $\mathtt{P_d}$ as defined in ($\mathtt{d_{ed}12}$). $\mathtt{P_d}$ is not transitive, see ($\mathtt{t_{ed}5}$), mainly because it can hold between couples of perdurants, times, or concepts but, given that a time can be also a perdurant or a concept and vice versa, there is no guarantee that chains of $\mathtt{P_d}$-relationships apply only to homogeneous entities, i.e., entities belonging to the same category (see the proof of ($\mathtt{t_{ed}5}$) for an example). In particular, a $\mathtt{P_d}$-relationship does not necessarily hold between homogeneous relata: ($\mathtt{t_{ed}2}$) exemplifies this problem in the case of times (but similar theorems hold for perdurants eCPT-concepts, and pCPT-concepts). Consequently, given the fact that PRE is defined in terms of $\mathtt{P_d}$ (see ($\mathtt{d_d5}$)), the second argument of PRE is not necessarily a time, see ($\mathtt{t_{ed}15}$) (and PRE does not imply EX, see ($\mathtt{t_{ed}16}$)). Given ($\mathtt{d_d5}$), when $\mathtt{P_d}$ is defined not only on times, this problem can be removed only by guaranteeing the disjointness of the categories on which $\mathtt{P_d}$ is defined. Similar problems impact also the axioms for constitution (K).

The fact that the relata of $\mathtt{P_d}$ are not necessarily homogeneous and that the second relatum of PRE is not necessarily a time, prevent the preservation of several axioms of DOLCE. However, in some cases, the addition of minimal assumptions allow to prove theorems that, intuitively, are close to the original ones. When possible we considered these modifications (see, for instance, ($\mathtt{t_{ed}4}$), ($\mathtt{t_{ed}30}$), ($\mathtt{t_{ed}31}$), ($\mathtt{t_{ed}32}$)) even though in more complicate cases – e.g., the previously discussed transitivity of $\mathtt{P_d}$ where one should add $(\mathtt{TME}(x) \wedge \mathtt{TME}(y) \wedge \mathtt{TME}(z)) \vee (\mathtt{PD}(x) \wedge \mathtt{PD}(y) \wedge \mathtt{PD}(z)) \vee (\mathtt{eCPT}(x) \wedge \mathtt{eCPT}(y) \wedge \mathtt{eCPT}(z)) \vee (\mathtt{pCPT}(x) \wedge \mathtt{pCPT}(y) \wedge \mathtt{pCPT}(z))$ – we considered only the original axiom.

In principle, it might have been possible to force disjointness by operating on the syntactic definitions, e.g., by deciding on an order of priority between the categories and systematically adding clauses excluding entities falling in a category given more priority (already defined). However, the strategy was deemed inadequate from an ontological point of view (as it would not respect the differences distinguishing the stances of the two ontologies), arbitrary, and conductive to worse alignments overall, even more so in comparison to the alternative outlined above.

Another general problem concerns axioms containing existential conditions (e.g., ($\mathtt{t_{ed}6}$), ($\mathtt{t_{ed}10}$), ($\mathtt{t_{ed}27}$)). The reason why these axioms cannot be preserved has to do with the definitions of endurants, perdurants, and concepts that require the existence of a $\phi \in \mathbf{t}(\Delta)$ or the existence of a semiotic process that in principle have to apply to the entity of which the existence must be proved. However, the principles regulating the semiotic perspective and the extensions are not strong enough to guarantee the existence of such ϕs or semiotic processes. As shown in the proof of ($\mathtt{t_{ed}13}$), the problem is more complex in the case of concepts because EMMO does not exclude infinite chains of concepts one specializing the other. In general, it is not clear how these principles can be strengthened and whether this move is in any case plausible from an ontological and/or applicative point of view. Arguably, this point is related to the fact that the portions of EMMO involved in the mapping have an epistemological

basis, as touched on above.

Finally, note that we defined ED and PD, respectively, in terms of temporal dissectivity (d_e56) and temporal non-dissectivity (d_e57) that, in their turn, are based on the notion of temporal slice (d_e55). This choice is mainly motivated by the idea that the way entities persist must be characterized in terms of their behavior in time. Temporal slices fit quite naturally (and standardly) this idea. However, note that the definition of ED (d_{ed}3) is very close to the one of states (ST) in DOLCE as introduced in the original WonderWeb deliverable D18. Furthrmore, Sider's *Four-Dimensionalism* proposes a clear and simple definition of perdurance through time. Perdurantism is the thesis that objects have a temporal part (temporal slice) at every time at which they exist. Given (t_e77), it is then plausible to conclude that EMMO endorses a form of perdurantism, such that all its entities are perdurants in the sense of Sider. One can then wonder if there is space for endurants within EMMO. We followed here a position considered by David Lewis according to which objects are just boring processes. More precisely, as (d_{ed}3) and (d_{ed}4) make explicit, endurants are "temporally homogeneous" sums of quanta, while perdurants are "temporally non-homegeneous" sums of quanta where homogeneity is defined in terms of the external perspectives provided by the extensions of EMMO. It is then true that the distinction between the defined ED and PD is more close to the one between the commonsense (and epistemological/conceptual) notions of object and process than to the one between the metaphysical notions of endurant and perdurant. In this direction, one could also consider an alternative mapping for ED and PD based on the Persistence perspective of EMMO. In this perspective, dissectivity and non-dissectivity is defined in terms of the *temporal slice item part* defined in (d_e26). Differently from TS, TsiP does not rely on the construction of time and basically identifies the parts of an item that disconnect (in terms of causation) the item itself. (d_e61) and (d_e62) modify (d_e56) and (d_e57), respectively, by substituting TS with TsiP while (d_e63) and (d_e64) are more permissive than their analogues: they require that an entity is dissective (or non-dissective) with respect to just a single $\phi \in \Delta$ (rather than all of them); consequently, wOBJ and wPRC are not necessarily disjoint, and an entity can be both a Weak Object and a Weak Process.

d_e61 $\mathtt{DSC}\langle\phi\rangle(x) \coloneqq \phi(x) \wedge \forall y(\mathtt{TsiP}(y,x) \rightarrow \phi(y)))$ (*x is dissective w.r.t* ϕ)

d_e62 $\mathtt{nDSC}\langle\phi\rangle(x) \coloneqq \phi(x) \wedge \exists y(\mathtt{TsiP}(y,x) \rightarrow \neg\phi(y)))$ (*x is non-dissective w.r.t* ϕ)

d_e63 $\mathtt{wOBJ}\langle\Delta\rangle(x) \coloneqq \bigvee_{\phi\in\mathbf{t}(\Delta)} \phi(x) \wedge \bigvee_{\phi\in\mathbf{t}(\Delta)} \mathtt{DSC}\langle\phi\rangle(x)$ (*Weak Object*)

d_e64 $\mathtt{wPRC}\langle\Delta\rangle(x) \coloneqq \bigvee_{\phi\in\mathbf{t}(\Delta)} \phi(x) \wedge \bigvee_{\phi\in\mathbf{t}(\Delta)} \mathtt{nDSC}\langle\phi\rangle(x)$ (*Weak Process*)

9 The mapping from DOLCE to EMMO

In this section we introduce the syntactic definitions of the primitives of EMMO introduced in the Mereocausal module (see Sect. 4.1) and in the Semiotic perspective (see Sect. 4.2.2) in terms of the primitives of DOLCE. Analogously to the DOLCE-BFO mapping, we present only a partial mapping covering a subset of the categories and primitives of EMMO.

9.1 Mappings

Following the metodological assumption (M2), the idea is to import all the particulars of DOLCE into EMMO. In EMMO, parthood is defined on all the entities in the domain while in DOLCE, $\mathtt{P_d}$ is defined only on perdurants and abstracts, $\mathtt{tP_d}$ only on endurants, and no parthood on qualities exists. As such, P needs to be defined as a disjunction of different relations.

In DOLCE there are no diachronic dependence relations (i.e., dependence relations where the relata have disjoint temporal extensions) therefore we couldn't find an adequate way to define the causation relation of EMMO.

$\mathbf{d_{de}1}$ $\mathtt{P}(x,y) \coloneqq \mathtt{P_d}(x,y) \vee \mathtt{CP}(x,y) \vee \exists zw(\mathtt{DQT}(x,z) \wedge \mathtt{DQT}(y,w) \wedge (\mathtt{P_d}(z,w) \vee \mathtt{CP}(z,w)))$

For PDs and ABs, P coincides with $\mathtt{P_d}$. For EDs, P coincides with constant parthood (CP). For Qs inhering in EDs, P coincides with CP between such bearers while for Qs inhering in PDs, P coincides with $\mathtt{P_d}$ between such bearers.

Providing a full definition (necessary and sufficient conditions) of the semiotic primitive SMS in DOLCE seems quite difficult. The domain of DOLCE contains mental objects (MOB) and agentive physical objects (APO) which can be good candidates to play the role of, respectively, interpretants and interpreters. Furthermore, regions (R) can play the role of signs. The core problem is thus distinguishing true semiotic processes where an agent assigns a region, associated to a mental object, to a given object, from processes that involve these entities without assignments or where there is no correspondence between the involved regions and mental objects. To truly characterize semiotic processes, an extension of DOLCE seems necessary. Since, arguably, said extension would require the introduction of new primitives (not just definitions, as in the case of the extensions put forward in the mapping from EMMO to DOLCE), it is beyond the scope of this work. We can, however, at least introduce some minimal necessary conditions for SMS, as in ($\mathbf{a_{de}1}$), and study whether these conditions are enough to prove or disprove the original axioms of EMMO on SMS.

$\mathbf{a_{de}1}$ $\mathtt{SMS}(z,s,o,i,m) \rightarrow \mathrm{PD}(z) \wedge \mathrm{R}(s) \wedge o \neq z \wedge (\mathrm{APO}(i) \wedge \neg \mathtt{O_d}(i,o)) \wedge \mathrm{MOB}(m) \wedge \forall t(\mathtt{PRE}(z,t) \rightarrow \mathtt{PC}(i,t) \wedge \mathtt{PC}(m,t)) \wedge \mathtt{SD}(m,i)$

A semiotic process z is a perdurant with at least two constant participants: an agentive physical object i (that does not overlap the object o) and a mental object m specifically dependent on i. The sign s is mapped to a region while the object o may be any entity different from the semiotic process z.

9.2 Check of the preservation of the original EMMO axioms

As discussed in Sect. 9.1 some EMMO-axioms involve relations or categories that have not been defined in the mappings.

In the following, $\mathfrak{D}_e$ is the theory $\mathfrak{D}_e = \mathfrak{D} \cup \{(d_{de}1), (a_{de}1)\}$.

9.2.1 Mereology

$t_{de}1$ $\mathfrak{D}_e \vdash \mathtt{P}(x,x)$
Proof. Directly by (a_d89), (a_d4), (t_d11), (a_d9). □

$t_{de}2$ $\mathfrak{D}_e \nvdash \mathtt{P}(x,y) \wedge \mathtt{P}(y,x) \rightarrow x = y$
Proof. Directly by (t_d12). □

$t_{de}3$ $\mathfrak{D}_e \vdash \mathtt{P}(x,y) \wedge \mathtt{P}(y,z) \rightarrow \mathtt{P}(x,z)$
Proof. Directly by (a_d89), (a_d6), (t_d13), (a_d9). □

$t_{de}4$ $\mathfrak{D}_e \nvdash \neg\mathtt{P}(x,y) \rightarrow \exists z(\mathtt{P}(z,x) \wedge \neg\mathtt{O}(z,y))$
Proof. It is enough to consider the model in the proof of (t_d14). In this model we have $\neg\mathtt{P}(x,y)$ (because x and y are endurants but $\neg\mathtt{CP}(x,y)$) and the only parts of x are x and y that however both overlap y (because, by construction, $\mathtt{CP}(y,x)$ and, by (t_d11), $\mathtt{CP}(y,y)$). □

$t_{de}5$ $\mathfrak{D}_e \nvdash \exists x(\phi(x)) \rightarrow \exists y(y = \sigma x\langle\phi(x)\rangle)$
Proof. The fusion is not present in DOLCE. That said, the axiom could in principle be approximated in finite domains given the axioms of DOLCE pertaining to the existence of the (binary) sum for $\mathtt{P}_d$ and $\mathtt{tP}_d$. □

$t_{de}6$ $\mathfrak{D}_e \nvdash \exists y(\mathtt{qP}(y,x))$
Proof. DOLCE is compatible with non atomic domains. □

9.2.2 Causation

Given that causation cannot be defined whitout extending DOLCE, (a_e7)-(a_e13) cannot be considered.

9.2.3 Semiosis

(a_e18) is not considered because it is based on causation.

$t_{de}7$ $\mathfrak{D}_e \nvdash \mathtt{SMS}(z,s,o,i,m) \rightarrow \mathtt{O}(z,s) \wedge \mathtt{O}(z,o) \wedge \mathtt{O}(z,m)$ (a_e17)
Proof. From the hypothesis, by ($a_{de}1$), $\mathrm{PD}(z)$ and $\mathrm{AB}(s)$ and then, by ($d_{de}1$), (a_d1), and (a_d108), $\neg\mathtt{O}(z,s)$. □

$t_{de}8$ $\mathfrak{D}_e \nvdash \mathtt{MR}(m) \rightarrow \exists! i(\mathtt{INT}(i) \wedge \mathtt{P}(m,i))$ (a_e19)
Proof. From the hypothesis, by ($a_{de}1$), $\mathrm{MOB}(m) \wedge \mathrm{APO}(i) \wedge \mathtt{SD}(m,i)$ but in DOLCE m and i can be mereologically disjoint. □

$t_{de}9$ $\mathfrak{D}_e \nvdash \mathtt{SMS}(z,s,o,i,m) \wedge \mathtt{SMS}(z,s',o',i',m') \rightarrow (s = s' \wedge o = o' \wedge i = i' \wedge m = m')$ (a_e20)
Proof. In DOLCE, nothing prevents the possibility to have complex processes involving several APOs, several MOBs, etc. □

t_{de}10 $\mathfrak{D}_e \vdash \mathtt{SMS}(z,s,o,i,m) \rightarrow (z \neq s \wedge z \neq o \wedge z \neq i \wedge \neg \mathtt{O}(s,o) \wedge \neg \mathtt{O}(s,i) \wedge \neg \mathtt{O}(o,i))$ (a_e21)

Proof. Directly from the hypothesis, by (a_{de}1) and the fact that (i) PD, AB, ED, and Q are disjoint cateogires; and (ii) R is a subcategory of AB and APO is a subcategory of ED. □

9.3 Analysis

Given the definition of P provided in (d_{de}1), we obtain a very weak "mereology": the defined P is neither antisymmetric, see (t_{de}2), nor extensional, see (t_{de}4). (t_{de}2) is particularly interesting because it makes explicit the fact that DOLCE allows mereologically coincident endurants (in the sense that it is possible to have $\mathtt{CP}(x,y) \wedge \mathtt{CP}(y,x) \wedge x \neq y$) while the physicalist stance of EMMO rejects such scenarios.

In Sect. 8.1, we defined PC as a form of mereological overlap, see (d_{ed}14). One could exploit this intuition to extend P by including also some cases of participation as done in (d_{de}4). According to (d_{de}4), $\mathtt{P}(x,y)$ also holds between an endurant and a perdurant (when x constantly participates in y) or between a perdurant and an endurant (when y is the unique participant of x). The other disjuncts consider, respectively, (i) the cases of qualities part of EDs or PDs; (ii) the cases of EDs or PDs part of qualities; and (iii) the cases of qualities part of qualities. In (i)-(iii), it is necessary to consider both the case where the quality inheres in an endurant and the case where it inheres in an endurant and use the bearear(s) to establish the link. (t_{de}11)-(t_{de}16), where $\mathfrak{D}'_e = \mathfrak{D} \cup \{(d_{de}4)\}$, show that, in terms of the preservation of the axioms of EMMO, (d_{de}4) worsens the situation because the transitivity of P is lost.

d_{de}2 $\mathtt{cPC}(x,y) \coloneqq \forall t(\mathtt{PRE}(x,t) \rightarrow \mathtt{PC}(x,y,t))$ (*Constant Participation*)

d_{de}3 $\mathtt{ePTC}(x,y) \coloneqq \forall tz(\mathtt{PC}(z,y,t) \rightarrow \mathtt{tP}_\mathtt{d}(z,x,t))$ (*Exclusive Participant*)

d_{de}4 $\mathtt{P}(x,y) \coloneqq \mathtt{P}_\mathtt{d}(x,y) \vee \mathtt{CP}(x,y) \vee \mathtt{cPC}(x,y) \vee \mathtt{ePTC}(y,x) \vee$
$\exists z(\mathtt{DQT}(x,z) \wedge (\mathtt{P}_\mathtt{d}(z,y) \vee \mathtt{CP}(z,y) \vee \mathtt{cPC}(z,y) \vee \mathtt{ePTC}(z,y))) \vee$
$\exists z(\mathtt{DQT}(y,z) \wedge (\mathtt{P}_\mathtt{d}(x,z) \vee \mathtt{CP}(x,z) \vee \mathtt{cPC}(x,z) \vee \mathtt{ePTC}(x,z))) \vee$
$\exists zw(\mathtt{DQT}(x,z) \wedge \mathtt{DQT}(y,w) \wedge (\mathtt{P}_\mathtt{d}(z,w) \vee \mathtt{CP}(z,w) \vee \mathtt{cPC}(z,w) \vee \mathtt{ePTC}(z,w)))$

t_{de}11 $\mathfrak{D}'_e \vdash \mathtt{P}(x,x)$

Proof. Directly from (t_{de}1) by observing that the added disjuncts in (d_{de}4) applies to entities belonging to different categories. □

t_{de}12 $\mathfrak{D}'_e \nvdash \mathtt{P}(x,y) \wedge \mathtt{P}(y,x) \rightarrow x = y$

Proof. Directly by (t_d12). □

t_{de}13 $\mathfrak{D}'_e \nvdash \mathtt{P}(x,y) \wedge \mathtt{P}(y,z) \rightarrow \mathtt{P}(x,z)$

Proof. Assume a model with an endurant y and two perdurants x and z, such that x and y are present only at t, $\mathtt{PC}(y,x,t) \wedge \mathtt{PC}(y,z,t) \wedge \neg \mathtt{O}_\mathtt{d}(x,z)$, and y is the only participant to x. From $\mathrm{ED}(y) \wedge \mathtt{PRE}(y,t)$, by ($a_d$36), $\mathtt{tP}_\mathtt{d}(y,y,t)$. Then, given that y is the only particant to x, we have $\mathtt{ePTC}(y,x)$ and then $\mathtt{P}(x,y)$. Given that y exists only at t and $\mathtt{PC}(y,z,t)$, then $\mathtt{cPC}(y,z)$ and then $\mathtt{P}(y,z)$. But, x and z are both perdurants, therefore to have $\mathtt{P}(x,z)$ we need to have $\mathtt{P}_\mathtt{d}(x,z)$ that contradicts $\neg \mathtt{O}_\mathtt{d}(x,z)$. □

t_{de}14 $\mathfrak{D}'_e \nvdash \neg \mathtt{P}(x,y) \rightarrow \exists z(\mathtt{P}(z,x) \wedge \neg \mathtt{O}(z,y))$
Proof. It is enough to consider the model in the proof of (t_d14). In this model we have $\neg \mathtt{P}(x,y)$ (because x and y are endurants but $\neg \mathtt{CP}(x,y)$) and the only parts of x are x and y that however both overlap y (because, by construction, $\mathtt{CP}(y,x)$ and, by (t_d11), $\mathtt{CP}(y,y)$). □

t_{de}15 $\mathfrak{D}'_e \nvdash \exists x(\phi(x)) \rightarrow \exists y(y = \sigma x\langle \phi(x)\rangle)$
Proof. See the comment for (t_{de}5). □

t_{de}16 $\mathfrak{D}'_e \nvdash \exists y(\mathtt{qP}(y,x))$
Proof. DOLCE is compatible with non atomic domains. □

Concerning the semiotic primitive SMS, first observe that (a_{de}1) can be easily (and, arguably, reasonably) strengthened to preserve (t_{de}9). On the other hand, both (t_{de}7) and (t_{de}8) seem to highlight genuine ontological disagreements.

According to (t_{de}8), $\mathtt{SMS}(z,s,o,i,m)$ implies $\mathtt{P}(m,i)$, i.e., given (a_{de}1), that mental object m should be part of the agentive physical object i. In DOLCE, both MOB and APO are subcategories of ED, therefore, given (d_{de}1) or (d_{de}4), we need to have $\mathtt{P}_\mathtt{d}(m,i)$ to infer $\mathtt{P}(m,i)$. However, in DOLCE, MOB and APO are linked by means of existential dependence instead of parthood. A possible way out could be to further extend the relation P including some cases of existential dependence (EXD); still, as it has been said, the core difference concerns the different stance of the two ontologies with respect to the ontological treatment of MRs/APOs.

According to (t_{de}7), $\mathtt{SMS}(z,s,o,i,m)$ implies $\mathtt{O}(z,s)$, i.e., given (a_{de}1), the perdurant z should overlap the region s. However, all the disjuncts in both (d_{de}1) and (d_{de}4) do not hold between PDs and ABs. In this case, it is not easy to extend the definition of P to include these cases because a link between PDs and RS can be established only via the qualia of the qualities of the perdurants that, however, do not seem significant in the case of semiosis. Notice that, according to (t_{de}7), $\mathtt{SMS}(z,s,o,i,m)$ also implies $\mathtt{O}(z,o)$. Here again we have the problem that o can also be an abstract. Additionally, this constraint raises interesting questions about the nature of z and the possibility of having semiotic processes in which signs are assigned to past objects. They appear to be allowed in EMMO, given its generally "perdurantist" stance, prima facie contra an intuitive reading of semiotic processes; however, it ought to be remembered that, while EMMO's semiotic perspective takes inspiration from Peirce's theory of signs, SMS is ultimately to be understood as a shortcut for specific causal connections.

10 Top Reference Ontology OWL 2 DL Framework

The bi-directional alignments between DOLCE and BFO, and between DOLCE and EMMO, are part of the logical framework providing the foundation for comparison and interoperability between available state of the art TLOs, as stated in the DoA of OntoCommons. The FOL mappings introduced in the previous sections, allow to establish semantic correspondences between entities belonging to different ontologies (so-called semantic alignments). The alignment links, expressed in the form of syntactic definitions (as well as annotations and examples) constitute the so-called Meta ontology that is part of the more comprehensive Top Reference Ontology (TRO), as represented in Fig. 8.

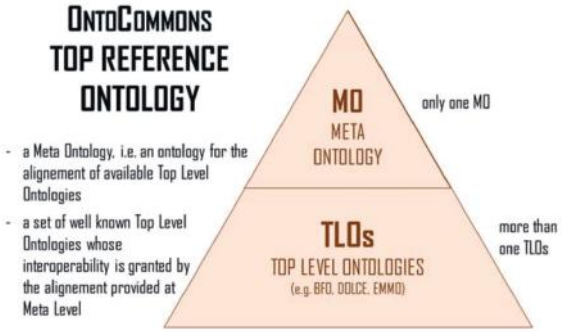

Figure 8: TRO structure.

In order to facilitate the exploitation of said alignments via the variety of tools that support semantic web languages, the TLOs and the alignments have been used as a basis to create a framework for the TRO in OWL 2 (in short TRO-OWL). In this framework, the OWL 2 version of the TLOs are integrated with the set of preliminary alignments discussed in Sect. 10.1 and 10.2, with each TLO and set of alignments being integrated in a modular fashion. The mapping themselves are divided into two parts: a first one, OWL 2 DL compliant, containing simple taxonomical relations, and second one, SWRL-based, including more complex connections, even concerning relations. The reduced expressiveness of OWL 2 DL does not allow to fully represent the conceptualization behind the TLOs and their alignments; yet it enforces computability, and massively increases the usability (computational costs-wise) of the ontologies, as shown in Fig.9. Without sacrificing computability, the RL extension preserves some of the advantages provided by expressive formal systems; as such, the SWRL mappings can be exploited to exchange data on demand, or for local applications, if not as the standard.

The TRO-OWL framework has been published publicly in the GitHub repository at `https://github.com/OntoCommons/OntologyFramework`, see Fig. 10, and will be continuously updated during the OntoCommons project duration.

The structure of the TRO-OWL ontology modules is summarized in Fig. 11, also

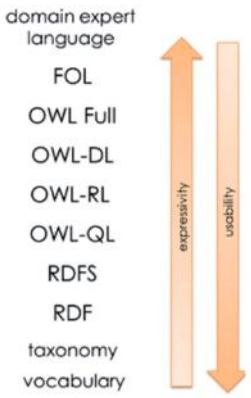

Figure 9: Expressiveness vs. usability.

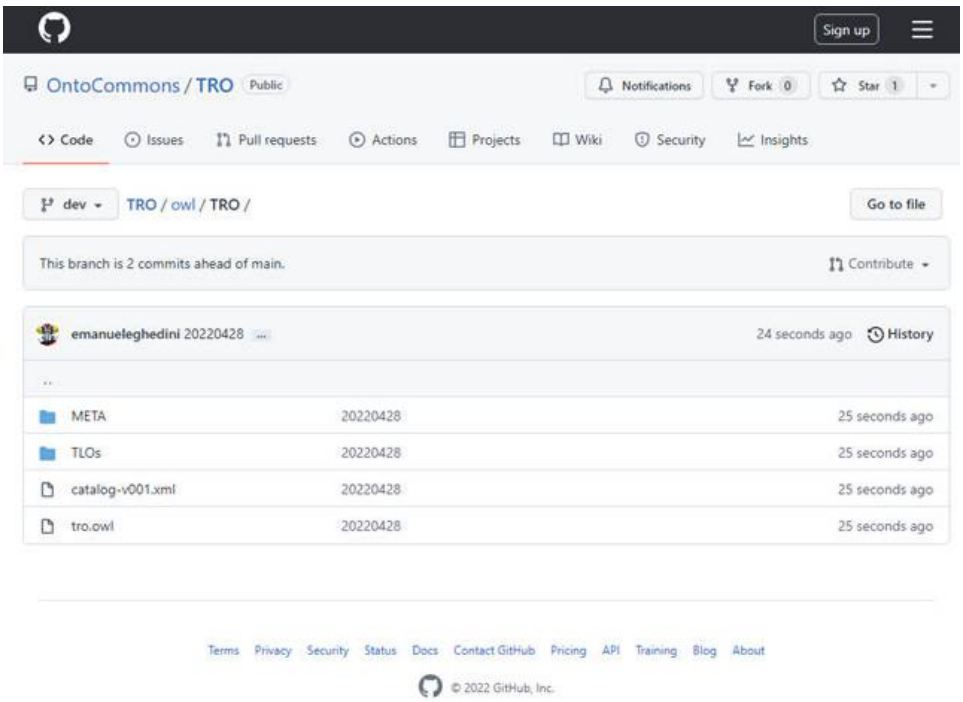

Figure 10: GitHub repository for the TRO OWL 2 DL framework.

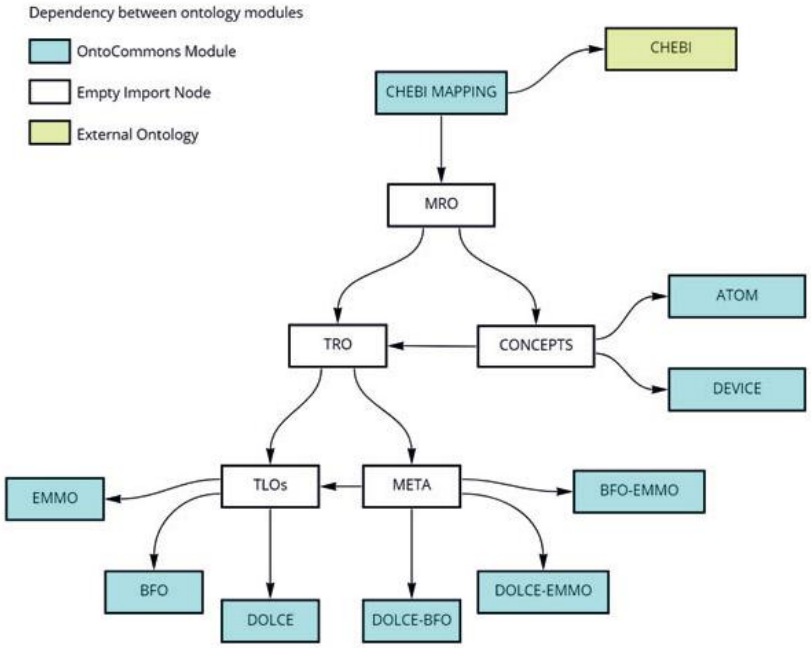

Figure 11: TRO-OWL ontology modules and their dependencies.

including the Middle Reference Ontology (MRO), that hosts the alignments of a selected set of MLOs with the OCES via bridge concepts. As a remainder, the Meta ontology has to be understood as the combination of the mappings between the three selected TLOs.

The framework is fully Protégé compatible and will offer to the end user the possibility to navigate through the TLOs, their mappings and lower-level ontologies connected to them, as shown in Fig. 12. The OWL mappings, derived from rigorous FOL alignments, ground partial links between the TLOs' respective taxonomic trees as shown in Fig. 13. In our intention, the TRO-OWL will also come to constitute a development framework for semantic web ontologists to build and test OntoCommons compliant ontologies.

10.1 OWL alignments: BFO-DOLCE

The analysis provided in Sect. 6 and Sect. 7 can be used to suggest some preliminary `subClassOf` mappings between the classes of the OWL versions of BFO and DOLCE. In this context, the following three observations are particularly important:

- in DOLCE, but not in BFO, it is possible to have different spatially co-localized endurants (including POBs and FS, see (t_{ab} 41) and (t_{ab} 51)) as well as differernt

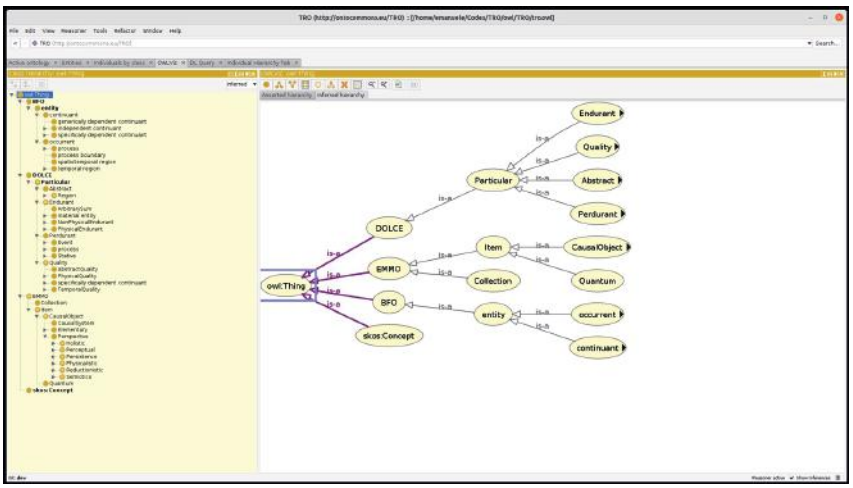

Figure 12: Protégé screenshot of the TRO-OWL framework.

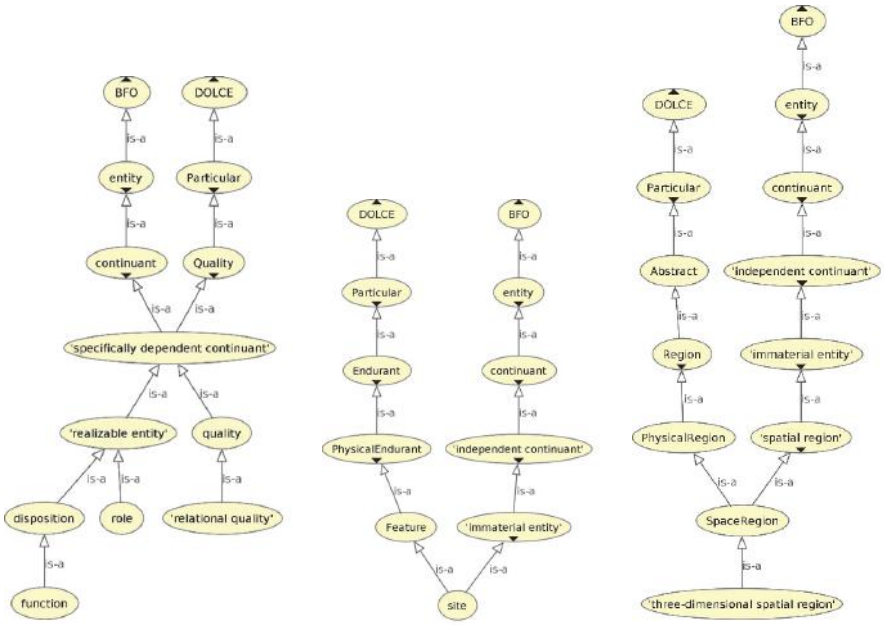

Figure 13: OWLViz representation of the taxonomical alignments between DOLCE and BFO for the `BFO:'specifically dependent continuant'`, `BFO:site` and `DOLCE:SpaceRegion` concepts.

spatiotemporally co-localized perdurants (see (t_{db}32)). Some EDs (PDs) could then not be classified under `cnt` (`occ`). Consequently, the qualities of these endurants would not find a place under `sdcnt` just because the endurants they inhere in would not be continuants as required by (a_b99).

- The generically dependent continuants of BFO were found out to be a quite peculiar case of non-spatially located NPEDs (remember that in DOLCE non-physical endurants can have a spatial location). Furthermore, according to (d_{db}11), some non-physical endurants can be classified under `mten`.
- In BFO, the identification of the subcategories of spatial and temporal regions, as well as the distinctions between processes and process boundaries, and between sites and continuant fiat boundaries, are grounded on dimensional factors. Sect. 6 show that the attempt to approximate BFO temporal instants (intervals) as DOLCE (non-)atomic time intervals prevented us to prove several axioms of BFO concerning the temporal extension and the nature of both processes and process boundaries. Thus, it is still unclear whether DOLCE's categories T and S have uniform instances—more specifically, whether all the instances of T are one-dimensional time intervals and all the instances of S are three-dimensional space regions—or if lower dimensional entities can are allowed. This doubt also impacts the possibility of having perdurants corresponding to process boundaries (that have a zero-dimensional temporal extension) and endurants corresponding to continuant fiat boundaries (that have a zero-, one-, or two-dimensional spatial extension) in DOLCE.

Taking into account these observations and following a "cautious" mapping strategy, we preliminarily identify the following inclusions of BFO classes into DOLCE classes

```
subClassOf(BFO:'material entity' DOLCE:Endurant)
subClassOf(BFO:site DOLCE:Feature)
subClassOf(BFO:'generically dependent continuant'
    DOLCE:NonPhysicalEndurant)
subClassOf(BFO:'specifically dependent continuant' DOLCE:Quality)
subClassOf(BFO:'process' DOLCE:Perdurant)
subClassOf(BFO:'one-dimensional temporal region' DOLCE:TimeInterval)
subClassOf(BFO:'three-dimensional spatial region' DOLCE:SpaceRegion)
```

and the following inclusions of DOLCE classes into BFO classes

```
subClassOf(DOLCE:AmountOfMatter BFO:'material entity')
subClassOf(DOLCE:TimeInterval BFO:'temporal region')
subClassOf(DOLCE:SpaceRegion BFO:'spatial region')
```

In order to better exploit the alignment, one option is making use of Rule Language, moving to OWL 2 SWRL. The following links from BFO to DOLCE become then expressible:

```
BFO:'exists at'(?x,?y), BFO:site(?x),
    BFO:'one-dimensional temporal region'(?y)
    → DOLCE:presentAt(?x,?y)
```

```
BFO:‘specifically depends on’(?x,?y),
     BFO:‘specifically dependent continuant’(?x), BFO:site(?y)
     → DOLCE:directQualityOf(?x,?y)

BFO:‘occupies temporal region’(?x,?y), BFO:‘process’(?x),
     BFO:‘one-dimensional temporal region’(?y)
     → DOLCE:temporallyLocatedAt(?x,?y)

BFO:‘occupies spatial region at all times’(?x,?y),
     BFO:‘material entity’(?x),
     BFO:‘three-dimensional spatial region’(?y)
     → DOLCE:constantlySpatiallyLocatedAt(?x,?y)

BFO:‘exists at’(?x,?y), BFO:‘process’(?x),
     BFO:‘one-dimensional temporal region’(?y)
     → DOLCE:presentAt(?x,?y)

BFO:‘continuant part of at all times’(?x,?y),
     BFO:site(?x), BFO:site(?y)
     → DOLCE:constantPartOf(?x,?y)

BFO:‘specifically depends on’(?x,?y),
     BFO:‘specifically dependent continuant’(?x),
     BFO:‘material entity’(?y)
     → DOLCE:directQualityOf(?x,?y)

BFO:‘participates in at all times’(?x,?y),
     BFO:‘generically dependent continuant’(?x),
     BFO:‘process’(?y)
     → DOLCE:constantParticipantOf(?x,?y)

BFO:‘occupies spatial region at all times’(?x,?y), BFO:site(?x),
     BFO:‘three-dimensional spatial region’(?y)
     → DOLCE:constantlySpatiallyLocatedAt(?x,?y)

BFO:‘participates in at all times’(?x,?y), BFO:‘material entity’(?x),
     BFO:‘process’(?y)
     → DOLCE:constantParticipantOf(?x,?y)

BFO:‘occurrent part of’(?x,?y),
     BFO:‘one-dimensional temporal region’(?x),
     BFO:‘one-dimensional temporal region’(?y)
     → DOLCE:partOf(?x,?y)

BFO:‘participates in at all times’(?x,?y), BFO:site(?x),
     BFO:‘process’(?y)
     → DOLCE:constantParticipantOf(?x,?y)

BFO:‘continuant part of at all times’(?x,?y),
     BFO:‘material entity’(?x), BFO:‘material entity’(?y)
     → DOLCE:constantPartOf(?x,?y)

BFO:‘occurrent part of’(?x,?y),
     BFO:‘process’(?x), BFO:‘process’(?y)
     → DOLCE:partOf(?x,?y)
```

```
BFO:‘exists at’(?x,?y), BFO:‘specifically dependent continuant’(?x),
     BFO:‘one-dimensional temporal region’(?y)
     → DOLCE:presentAt(?x,?y)

BFO:‘exists at’(?x,?y), BFO:‘generically dependent continuant’(?x),
     BFO:‘one-dimensional temporal region’(?y)
     → DOLCE:presentAt(?x,?y)

BFO:‘exists at’(?x,?y), BFO:‘material entity’(?x),
     BFO:‘one-dimensional temporal region’(?y)
    → DOLCE:presentAt(?x,?y)

BFO:‘continuant part of at all times’(?x,?y),
     BFO:‘generically dependent continuant’(?x),
     BFO:‘generically dependent continuant’(?y)
     → DOLCE:constantPartOf(?x,?y)
```

Likewise, the following links from DOLCE to BFO become expressible.

```
DOLCE:presentAt(?x,?y),
     DOLCE:AmountOfMatter(?x), DOLCE:TimeInterval(?y)
     → BFO:‘exists at’(?x,?y)

DOLCE:constantPartOf(?x,?y),
     DOLCE:AmountOfMatter(?x), DOLCE:AmountOfMatter(?y)
     → BFO:‘has continuant part of at all times’(?x,?y)

DOLCE:constantlySpatiallyLocatedAt(?x,?y),
     DOLCE:AmountOfMatter(?x), DOLCE:SpaceRegion(?y)
    → BFO:‘occupies spatial region at all times’(?x,?y)

DOLCE:partOf(?x,?y), DOLCE:TimeInterval(?x), DOLCE:TimeInterval(?y)
     → BFO:‘has temporal part’(?x,?y)

DOLCE:directQualityOf(?x,?y),
     DOLCE:PhysicalQuality(?x), DOLCE:AmountOfMatter(?y)
     → BFO:‘inheres in’(?x,?y)
```

10.2 OWL alignments: DOLCE-EMMO

Mirroring the situation regarding the FOL-alignments, even in OWL the mappings between DOLCE and EMMO are more complex, and further machinery has to be employed. The need for constructive extensions to bridge the gap between the two ontologies has analogues in the OWL-version of the mappings, where only fairly uninformative links can be established if the following hidden classes are not introduced:

```
META:StrictObject

META:StrictProcess

META:SSOSum

META:NoSTQuality

META:NoSTRegion
```

The introduced classes are characterized as follows:

```
subClassOf(META:StrictObject EMMO:Object)
Disjoint(META:StrictObject EMMO:Process)
subClassOf(META:StrictProcess EMMO:Process)
Disjoint(META:StrictProcess EMMO:Object)
subClassOf(META:SSOSum EMMO:CausalStructure)
subClassOf(META:SSOSum EMMO:hasProperPart exactly 1 EMMO:Sign)
subClassOf(META:SSOSum EMMO:hasProperPart exactly 1 EMMO:SemioticObject)
subClassOf(META:NoSTQuality DOLCE:Quality)
Disjoint(META:NoSTQuality DOLCE:TemporalLocation)
Disjoint(META:NoSTQuality DOLCE:SpatialLocation)
subClassOf(META:NoSTRegion DOLCE:Region)
Disjoint(META:NoSTRegion DOLCE:TimeRegion)
Disjoint(META:NoSTRegion DOLCE:SpaceRegion)
```

The characterization of META:SSOSum can be improved by making use of SWRL rules; similar rules are not strictly part of EMMO (or DOLCE), and that also goes for the META classes, though they are operationally considered as belonging to be part of the ontology employed to characterize them. The introduction of these rules requires particular care, in order not to avoid unintended changes to the original ontologies. It might be possible to make use of rules involving classes and relations from a plurality of ontology to achieve more specific links, however, such an endeavor requires particular methodological care, and it was decided not to pursue an experimental line, without proper prior investigation, in the context of the project.

```
META:SSOSum(?y), EMMO:SemioticObject(?x), EMMO:Sign(?z),
    EMMO:hasProperPart(?y,?x), EMMO:hasProperPart(?y,?z)
    → EMMO:hasSign(?x,?z)
```

Notably, given OWL's open-world assumption, it is not possible to force Objects which are not Process to be categorized as Strict Objects (and vice versa); that said, it is possible to build an extremely simple external tool to do so when exchanging data. The tool can take a more complex form if the conditions in FOL are replicated, operating check on all the classes introduced by extensions in OWL.

As in the case of BFO and DOLCE, we preliminary identify inclusions of DOLCE classes (and DOLCE-based hidden META classes) into EMMO classes (and EMMO-based hidden META classes), by taking into account the observations put forward in Sections 8.3 and 9.3, and accepting the following approximations to strengthen the connections between the two ontologies:

- the connections between Perdurants and Endurants and Objects and Processes will disregard the differences between TS and TsiP in FOL. The two notions are arguably fairly similar and satisfy the same roles; nevertheless, one takes into account all the information provided by the models, while the other focuses on an entity's internal "topological" features; moreover, while TS has three arguments

and allows for entities to be temporal slices of themselves, TsiP is a binary relation not depending on a specific time, and it is irreflexive, specializing proper parthood. A formal investigation would be required to situate the two relations one with respect to the other; it will suffice to say that, given the differences outlined above, they tend to diverge when limiting cases are involved.

- in DOLCE, self-connectedness is not a central notion, while EMMO chiefly focuses on causally self-connected entities. Since EMMO's causality cannot be defined in DOLCE, it is also not possible to define the involved notion of self-connectedness. The mapping between EMMO and DOLCE will ignore this point on the grounds that doing so greatly improves the strength of the resulting alignment, that the unity conditions for the individuation of entities often informally involve self-connectedness (though not always), and that there shouldn't be practical consequences given the other categories and relations involved in the mapping.
- CPTs cannot be re-defined in OWL; as such, the related mappings will focus on Signs and MRs instead; given this forced choice, it was decided to adopt an especially cautious stance regarding the related mappings.

The class-inclusions are reported in what follows:

```
subClassOf(DOLCE:Endurant META:StrictObject)

subClassOf(DOLCE:Perdurant META:StrictProcess)

subClassOf(META:NoSTRegion EMMO:Sign)

subClassOf(META:NoSTQuality EMMO:SSOSum)
```

and the following inclusions of EMMO classes (and EMMO-based hidden META classes) into DOLCE classes (and DOLCE-based hidden META classes):

```
subClassOf(META:StrictObject DOLCE:Endurant)

subClassOf(META:StrictProcess DOLCE:Perdurant)
```

Once again, in order to better exploit the alignment, we recommend employing OWL SWRL and Rules. The following links from DOLCE to EMMO become then expressible:

```
DOLCE:Endurant(?x), DOLCE:Quality(?y),
     DOLCE:directQualityOf(?y,?x)
     → EMMO:SemioticObject(?x)

DOLCE:Perdurant(?x), DOLCE:Quality(?y),
     DOLCE:directQualityOf(?y,?x)
     → EMMO:SemioticObject(?x)

DOLCE:overlaps(?x,?y),
     DOLCE:Perdurant(?x), DOLCE:Perdurant(?y)
     → EMMO:overlaps(?x,?y)

DOLCE:partOf(?x,?y),
     DOLCE:Perdurant(?x), DOLCE:Perdurant(?y)
     → EMMO:hasPart(?y,?x)
```

```
DOLCE:properPartOf(?x,?y),
     DOLCE:Perdurant(?x), DOLCE:Perdurant(?y)
     → EMMO:hasProperPart(?y,?x)

DOLCE:constantlyOverlaps(?x,?y),
     DOLCE:Endurant(?x), DOLCE:Endurant(?y)
     → EMMO:overlaps(?x,?y)

DOLCE:constantProperPartOf(?x,?y),
     DOLCE:Endurant(?x), DOLCE:Endurant(?y)
     → EMMO:hasProperPart(?y,?x)

DOLCE:constantParticipantOf(?x,?y),
     DOLCE:Endurant(?x), DOLCE:Perdurant(?y)
     → EMMO:overlaps(?x,?y)

DOLCE:constantConstituentOf(?x,?y),
     DOLCE:PhysicalEndurant(?x), DOLCE:PhysicalEndurant(?y)
     → EMMO:overlaps(?x,?y)

DOLCE:constantConstituentOf(?x,?y),
     DOLCE:NonPhysicalEndurant(?x), DOLCE:NonPhysicalEndurant(?y)
     → EMMO:overlaps(?x,?y)

DOLCE:constantConstituentOf(?x,?y),
     DOLCE:Perdurant(?x), DOLCE:Perdurant(?y)
     → EMMO:overlaps(?x,?y)
```

Likewise, the following links from EMMO to DOLCE become expressible:

```
EMMO:overlaps(?x,?y),
     META:StrictProcess(?x), META:StrictProcess(?y)
     → DOLCE:overlaps(?x,?y)

EMMO:hasPart(?x,?y),
     META:StrictProcess(?x), META:StrictProcess(?y)
     → DOLCE:partOf(?y,?x)

EMMO:hasProperPart(?x,?y),
     META:StrictProcess(?x), META:StrictProcess(?y)
     → DOLCE:properPartOf(?y,?x)

EMMO:overlaps(?x,?y),
     META:StrictObject(?x), META:StrictObject(?y)
     → DOLCE:constantlyOverlaps(?x,?y)

EMMO:hasPart(?x,?y),
     META:StrictObject(?x), META:StrictObject(?y)
     → DOLCE:constantPartOf(?y,?x)

EMMO:hasProperPart(?x,?y),
     META:StrictObject(?x), META:StrictObject(?y)
     → DOLCE:constantProperPartOf(?y,?x)
```

It is worth pointing out that some classes and relations from different ontologies employed in the mapping share the same label. Some caution is advised in the mapping

in SWRL since some of the most common editing tools (such as Protégé) are label-based.

For the sake of completeness, we mention a relevant alternative to the OWL connections pertaining to DOLCE:Quality (and, consequently, META:NoSTQuality), from DOLCE to EMMO:

```
subClassOf(META:NoSTQuality EMMO:Sign)

DOLCE:Endurant(?x), DOLCE:Quality(?y), DOLCE:directQualityOf(?y,?x)
     → EMMO:hasSign(?x,?y)

DOLCE:Perdurant(?x), DOLCE:Quality(?y), DOLCE:directQualityOf(?y,?x),
     → EMMO:hasSign(?x,?y)
```

Given this alternative proposal, both DOLCE:Quality and DOLCE:Region are mapped into EMMO:Sign, regardless of the ontological distinctions in DOLCE, and contra the proposed FOL alignment. Nevertheless, it ought to be pointed out that EMMO's categories do not seem to be sensitive to the distinction given EMMO's overall nominalistic stance, and the alternative mapping is not completely unreasonable from a model-theoretic point of view. That said, the chosen mapping has the further benefit of preserving the link with trope-theory and of avoiding possible issues related to parthood, which might be relevant in the context multiple mappings, as it should be the standard in an ecosystem of ontologies.

Related to this last point, as in the case of the FOL mappings, even in OWL EMMO acts as a mediator, connecting EMMO and BFO, While it was not possible to consider direct alignments between EMMO and BFO due to time constraints, it should be mentioned that BFO includes a relation of precedence, and is more (mereological) extensionalism-friendly; as such, such an alignment might provide significant means to strengthen the ecosystem overall.

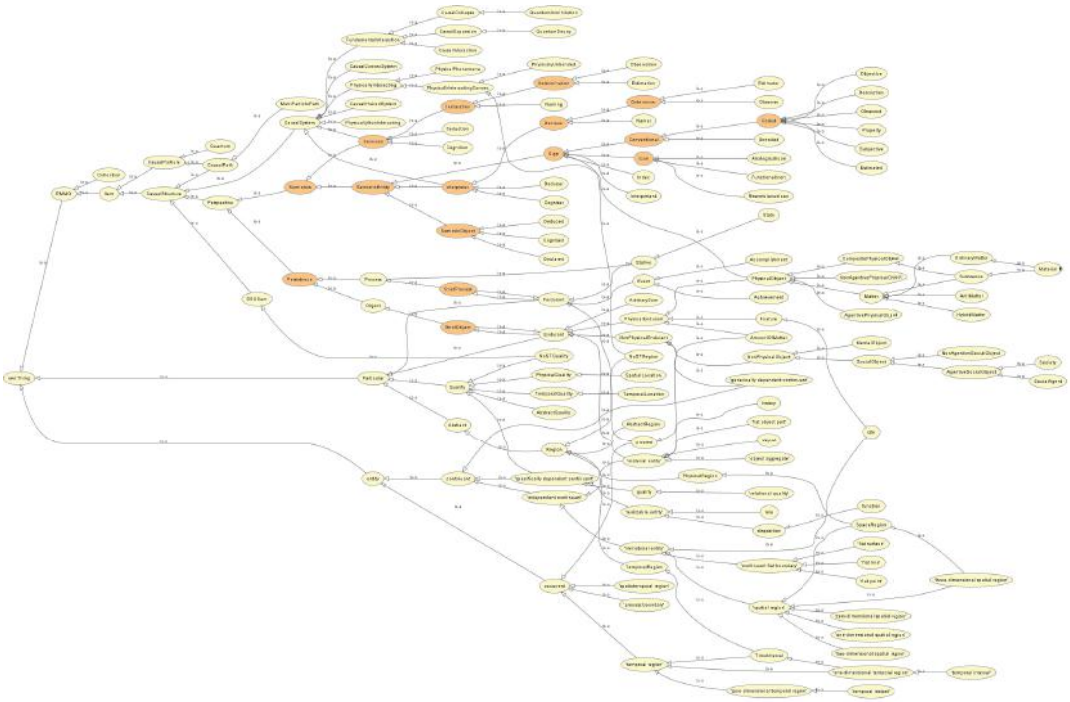

Figure 14: Fully connected TRO

11 Appendix: automated validation of mappings

In the following, we attach some automated proofs that validate the theorems stated in this work. To do this, we used two automated theorem provers for first-order logic: Prover9[16] and Vampire[17], as well as the automated model builder Mace4[18]. The provers are used in order to infer a logical consequence from a set of axioms. Since first-order logic is semi-decidable the provers are not guaranteed to prove or disprove the inference of an axiom from a set of premises so the automated provers cannot be expected to prove or disprove all theorems. Our methodology was as follows: to prove a theorem of DOLCE we translated all axioms and definitions of DOLCE written in Section 3 in the correct syntax for the provers (Vampire uses tptp, while Prover9 has its own syntax) and we submitted this translation together with the translation of the theorem to the theorem prover. We repeated this procedure for all theorems and the provers returned the proofs listed in Section 11.1. We used both Vampire and Prover9, since Vampire is generally faster, but was sometimes outperformed by Prover9 in our testing. For the theorems that both provers were able to prove, within a comparable timeframe, we show only the Prover9 proof, to avoid redundancy, and also because the proof appears shorter since Prover9 syntax does not require explicit quantification of variables.

To prove a theorem of the mapping from DOLCE to BFO, we did the same, but we also added the definitions of BFO and the definitions of the (DOLCE to BFO) mapping in the input of the provers. In this case, the proofs were more difficult, and the provers were not able to find all of them. For all remaining theorems, we produced manual proofs, and we verified some of them, using the provers to verify intermediate lemmas until the thesis was proved. All verified proofs are reported in Section 11.3.

To formally verify counterexamples for the non-theorems of the mapping from DOLCE to BFO, we produced counterexamples by hand that we wrote using the syntax of the provers in the following way: we wrote axioms enforcing the existence of exactly the desired constants required by the counterexample, as well as the existence of exactly the desired relations between them. To these axioms we added the axioms of DOLCE taxonomy, to simplify our work, since the inclusions between the classes of the taxonomy allow us to only work with taxonomical leaves, while the disjunction axioms between the classes allow us to skip some axioms stating that individual constants with different names are different. In this way we obtained, for each counterexample, a theory that exactly describes the counterexample syntactically.

Afterward, we ran Mace4 using all these axioms as inputs, to verify that the set of axioms was consistent and obtain explicit semantic descriptions of all counterexamples. Then, for each axiom of DOLCE, we successfully verified that the axiom can be proved from the theory, thus proving that the (unique) model of the theory is a model of DOLCE. Lastly, we checked that in the model the theorem that we wanted to disprove is actually false, but, at this point, this is trivial.

Unfortunately, the obvious way of obtaining counterexamples with Mace4, that is, asking Mace4 to directly find a model of all the axioms of DOLCE with the addition of

[16] Can be downloaded from https://www.cs.unm.edu/~mccune/prover9/
[17] Can be downloaded from the pertinent GitHub page https://vprover.github.io/
[18] Accessible from the same address as Prover9: https://www.cs.unm.edu/~mccune/prover9/

the negation of the theorem to be disproven, cannot be implemented, as it requires an excessive computational effort.

In Section 11.2 we report the syntactic description of the models and not the semantic one, as it is too long and can be generated from the syntactic description using Mace4. Some models are counterexamples of more than one theorem, in those cases we list all the theorems they are counterexamples of.

In order to verify the mappings and the counterexamples corresponding to the opposite direction, that is, from BFO to DOLCE, we proceeded in the same way, and the results are shown in sections 11.5 and 11.4. The only difference is that we used the formal axiomatisation of BFO developed by its authors[19].

Notice that the proofs use prefixed notation for the ternary (or binary, when the time argument is absent) predicate of instantiation, in place of the infixed syntax $(::(x,y,t)$ in place of $x::_t y)$.

[19] https://github.com/BFO-ontology/BFO-2020

11.1 Validation of DOLCE theorems

Proof of Theorem (t_d1)

```
1 DQT(A,B) -> TQ(A) & PD(B) | PQ(A) & PED(B) | AQ(A) & NPED(B). [assumption].
2 AS(A) | NPED(A) | PED(A) <-> ED(A). [assumption].
3 AQ(A) | PQ(A) | TQ(A) <-> Q(A). [assumption].
4 -(exists A (PD(A) & Q(A))). [assumption].
5 -(exists A (ED(A) & Q(A))). [assumption].
6 -(exists A (PED(A) & NPED(A))). [assumption].
7 DQT(A,B) -> A != B. [goal].
8 -DQT(A,B) | TQ(A) | PQ(A) | AQ(A). [clausify(1)].
9 -DQT(A,B) | TQ(A) | PQ(A) | NPED(B). [clausify(1)].
10 -TQ(A) | Q(A). [clausify(3)].
11 -PQ(A) | Q(A). [clausify(3)].
12 Q(A) | -DQT(A,B) | PQ(A) | AQ(A). [resolve(10,a,8,b)].
13 Q(A) | -DQT(A,B) | PQ(A) | NPED(B). [resolve(10,a,9,b)].
14 -DQT(A,B) | PD(B) | PED(B) | AQ(A). [clausify(1)].
15 -DQT(A,B) | PD(B) | PED(B) | NPED(B). [clausify(1)].
16 -NPED(A) | ED(A). [clausify(2)].
17 -PED(A) | ED(A). [clausify(2)].
18 -AQ(A) | Q(A). [clausify(3)].
19 -PD(A) | -Q(A). [clausify(4)].
20 -ED(A) | -Q(A). [clausify(5)].
21 -PED(A) | -NPED(A). [clausify(6)].
22 DQT(c1,c2). [deny(7)].
23 c2 = c1. [deny(7)].
24 Q(A) | -DQT(A,B) | AQ(A) | Q(A). [resolve(12,c,11,a)].
25 Q(A) | -DQT(A,B) | AQ(A). [copy(24),merge(d)].
26 Q(A) | -DQT(A,B) | NPED(B) | Q(A). [resolve(13,c,11,a)].
27 Q(A) | -DQT(A,B) | NPED(B). [copy(26),merge(d)].
28 DQT(c1,c1). [back_rewrite(22),rewrite([23(2)])].
29 Q(c1) | NPED(c1). [resolve(28,a,27,b)].
30 Q(c1) | AQ(c1). [resolve(28,a,25,b)].
31 PD(c1) | PED(c1) | NPED(c1). [resolve(28,a,15,a)].
32 PD(c1) | PED(c1) | AQ(c1). [resolve(28,a,14,a)].
33 AQ(c1) | -PD(c1). [resolve(30,a,19,b)].
34 PD(c1) | NPED(c1) | ED(c1). [resolve(31,b,17,a)].
35 PD(c1) | AQ(c1) | -NPED(c1). [resolve(32,b,21,a)].
36 PD(c1) | NPED(c1) | -Q(c1). [resolve(34,c,20,a)].
37 PD(c1) | NPED(c1). [resolve(36,c,29,a),merge(c)].
38 PD(c1) | AQ(c1). [resolve(37,b,35,c),merge(b)].
39 PD(c1) | ED(c1). [resolve(37,b,16,a)].
40 PD(c1) | Q(c1). [resolve(38,b,18,a)].
41 PD(c1) | -Q(c1). [resolve(39,b,20,a)].
42 PD(c1). [resolve(41,b,40,b),merge(b)].
43 AQ(c1). [back_unit_del(33),unit_del(b,42)].
44 Q(c1). [resolve(43,a,18,a)].
45 $F. [resolve(44,a,19,b),unit_del(a,42)].
```

Proof of Theorem (t_d2)

```
1 DQT(A,B) -> TQ(A) & PD(B) | PQ(A) & PED(B) | AQ(A) & NPED(B). [assumption].
2 -(exists A (PQ(A) & AQ(A))). [assumption].
3 -(exists A (TQ(A) & AQ(A))). [assumption].
4 DQT(A,B) & AQ(A) -> NPED(B). [goal].
5 -DQT(A,B) | TQ(A) | PQ(A) | NPED(B). [clausify(1)].
6 -TQ(A) | -AQ(A). [clausify(3)].
7 -PQ(A) | -AQ(A). [clausify(2)].
8 -AQ(A) | -DQT(A,B) | PQ(A) | NPED(B). [resolve(6,a,5,b)].
9 DQT(c1,c2). [deny(4)].
10 AQ(c1). [deny(4)].
11 -NPED(c2). [deny(4)].
12 -AQ(A) | -DQT(A,B) | NPED(B) | -AQ(A). [resolve(8,c,7,a)].
13 -AQ(A) | -DQT(A,B) | NPED(B). [copy(12),merge(d)].
14 $F. [resolve(13,b,9,a),unit_del(a,10),unit_del(b,11)].
```

Proof of Theorem (t_d3)

```
1 DQT(A,B) -> TQ(A) & PD(B) | PQ(A) & PED(B) | AQ(A) & NPED(B). [assumption].
2 QL(A,B) -> TR(A) & TQ(B). [assumption].
3 TQ(A) -> (exists B QL(B,A)). [assumption].
4 -(exists A (TQ(A) & PQ(A))). [assumption].
5 -(exists A (PQ(A) & AQ(A))). [assumption].
6 DQT(A,B) & PQ(A) -> PED(B). [goal].
7 -TQ(A) | QL(f7(A),A). [clausify(3)].
8 -DQT(A,B) | TQ(A) | PED(B) | AQ(A). [clausify(1)].
9 -QL(A,B) | TQ(B). [clausify(2)].
10 -TQ(A) | -PQ(A). [clausify(4)].
11 -PQ(A) | -AQ(A). [clausify(5)].
12 PQ(c1). [deny(6)].
13 -PQ(A) | -QL(B,A). [resolve(10,a,9,b)].
14 DQT(c1,c2). [deny(6)].
15 -PED(c2). [deny(6)].
16 QL(f7(A),A) | -DQT(A,B) | PED(B) | AQ(A). [resolve(7,a,8,b)].
17 -AQ(c1). [resolve(12,a,11,a)].
18 -QL(A,c1). [resolve(13,a,12,a)].
19 $F. [resolve(16,b,14,a),unit_del(a,18),unit_del(b,15),unit_del(c,17)].
```

Proof of Theorem (t_d4)

```
1 DQT(A,B) -> TQ(A) & PD(B) | PQ(A) & PED(B) | AQ(A) & NPED(B). [assumption].
2 -(exists A (TQ(A) & PQ(A))). [assumption].
3 -(exists A (TQ(A) & AQ(A))). [assumption].
4 DQT(A,B) & TQ(A) -> PD(B). [goal].
5 -TQ(A) | -PQ(A). [clausify(2)].
6 -TQ(A) | -AQ(A). [clausify(3)].
7 TQ(c1). [deny(4)].
8 -DQT(A,B) | PD(B) | PQ(A) | AQ(A). [clausify(1)].
9 DQT(c1,c2). [deny(4)].
10 -PD(c2). [deny(4)].
11 -PQ(c1). [resolve(7,a,5,a)].
12 -AQ(c1). [resolve(7,a,6,a)].
13 $F. [resolve(9,a,8,a),unit_del(a,10),unit_del(b,11),unit_del(c,12)].
```

Proof of Theorem (t_d5)

```
1 P(A,B) -> (T(A) <-> T(B)). [assumption].
2 PRE(A,B) <-> (exists C (TLC(A,C) & P(B,C))). [assumption].
3 TLC(A,B) -> (ED(A) | PD(A) | Q(A)) & T(B). [assumption].
4 PRE(A,B) -> (ED(A) | PD(A) | Q(A)) & T(B). [goal].
5 -P(A,B) | T(A) | -T(B). [clausify(1)].
6 -PRE(A,B) | TLC(A,f9(A,B)). [clausify(2)].
7 -PRE(A,B) | P(B,f9(A,B)). [clausify(2)].
8 -TLC(A,B) | ED(A) | PD(A) | Q(A). [clausify(3)].
9 -TLC(A,B) | T(B). [clausify(3)].
10 PRE(c1,c2). [deny(4)].
11 -ED(c1) | -T(c2). [deny(4)].
12 -PD(c1) | -T(c2). [deny(4)].
13 -Q(c1) | -T(c2). [deny(4)].
14 P(c2,f9(c1,c2)). [resolve(10,a,7,a)].
15 TLC(c1,f9(c1,c2)). [resolve(10,a,6,a)].
16 T(c2) | -T(f9(c1,c2)). [resolve(14,a,5,a)].
17 T(f9(c1,c2)). [resolve(15,a,9,a)].
18 ED(c1) | PD(c1) | Q(c1). [resolve(15,a,8,a)].
19 T(c2). [back_unit_del(16),unit_del(b,17)].
20 -Q(c1). [back_unit_del(13),unit_del(b,19)].
21 -PD(c1). [back_unit_del(12),unit_del(b,19)].
22 -ED(c1). [back_unit_del(11),unit_del(b,19)].
23 $F. [back_unit_del(18),unit_del(a,22),unit_del(b,21),unit_del(c,20)].
```

Proof of Theorem (t_d6)

```
1 P(A,B) & P(B,C) -> P(A,C). [assumption].
2 PRE(A,B) <-> (exists C (TLC(A,C) & P(B,C))). [assumption].
3 PRE(A,B) & P(C,B) -> PRE(A,C). [goal].
4 -P(A,B) | -P(B,C) | P(A,C). [clausify(1)].
5 -PRE(A,B) | TLC(A,f9(A,B)). [clausify(2)].
6 -PRE(A,B) | P(B,f9(A,B)). [clausify(2)].
7 PRE(A,B) | -TLC(A,C) | -P(B,C). [clausify(2)].
8 PRE(c1,c2). [deny(3)].
9 P(c3,c2). [deny(3)].
10 -PRE(c1,c3). [deny(3)].
11 P(c2,f9(c1,c2)). [resolve(8,a,6,a)].
12 TLC(c1,f9(c1,c2)). [resolve(8,a,5,a)].
13 -P(c2,A) | P(c3,A). [resolve(9,a,4,a)].
14 PRE(c1,A) | -P(A,f9(c1,c2)). [resolve(12,a,7,b)].
15 P(c3,f9(c1,c2)). [resolve(13,a,11,a)].
16 $F. [resolve(14,b,15,a),unit_del(a,10)].
```

Proof of Theorem (t_d7)

```
1 P(A,B) -> AB(A) & AB(B) | PD(A) & PD(B). [assumption].
2 P(A,B) & P(B,C) -> P(A,C). [assumption].
3 PRE(A,B) <-> (exists C (TLC(A,C) & P(B,C))). [assumption].
4 TLC(A,B) -> (ED(A) | PD(A) | Q(A)) & T(B). [assumption].
5 ED(A) | PD(A) | Q(A) -> (exists B TLC(A,B)). [assumption].
6 P(A,B) & TLC(A,C) & TLC(B,D) -> P(C,D). [assumption].
7 -(exists A (AB(A) & ED(A))). [assumption].
8 -(exists A (AB(A) & PD(A))). [assumption].
9 -(exists A (AB(A) & Q(A))). [assumption].
10 PRE(A,B) & P(A,C) -> PRE(C,B). [goal].
11 -P(A,B) | AB(A) | PD(A). [clausify(1)].
12 -P(A,B) | AB(A) | PD(B). [clausify(1)].
13 -AB(A) | -ED(A). [clausify(7)].
14 -AB(A) | -PD(A). [clausify(8)].
15 -AB(A) | -Q(A). [clausify(9)].
16 -P(A,B) | -P(B,C) | P(A,C). [clausify(2)].
17 -PRE(A,B) | TLC(A,f9(A,B)). [clausify(3)].
18 -PRE(A,B) | P(B,f9(A,B)). [clausify(3)].
19 PRE(A,B) | -TLC(A,C) | -P(B,C). [clausify(3)].
20 -TLC(A,B) | ED(A) | PD(A) | Q(A). [clausify(4)].
21 -PD(A) | TLC(A,f10(A)). [clausify(5)].
22 -P(A,B) | -TLC(A,C) | -TLC(B,D) | P(C,D). [clausify(6)].
23 PRE(c1,c2). [deny(10)].
24 P(c1,c3). [deny(10)].
25 -PRE(c3,c2). [deny(10)].
```

```
26 -ED(A) | -P(A,B) | PD(A). [resolve(13,a,11,b)].
27 -PD(A) | -P(A,B) | PD(B). [resolve(14,a,12,b)].
28 -Q(A) | -P(A,B) | PD(A). [resolve(15,a,11,b)].
29 P(c2,f9(c1,c2)). [resolve(23,a,18,a)].
30 TLC(c1,f9(c1,c2)). [resolve(23,a,17,a)].
31 -ED(c1) | PD(c1). [resolve(26,b,24,a)].
32 -PD(c1) | PD(c3). [resolve(27,b,24,a)].
33 -Q(c1) | PD(c1). [resolve(28,b,24,a)].
34 -P(f9(c1,c2),A) | P(c2,A). [resolve(29,a,16,a)].
35 -P(c1,A) | -TLC(A,B) | P(f9(c1,c2),B). [resolve(30,a,22,b)].
36 ED(c1) | PD(c1) | Q(c1). [resolve(30,a,20,a)].
37 PD(c1) | Q(c1). [resolve(36,a,31,a),merge(c)].
38 PD(c1). [resolve(37,b,33,a),merge(b)].
39 PD(c3). [back_unit_del(32),unit_del(a,38)].
40 TLC(c3,f10(c3)). [resolve(39,a,21,a)].
41 PRE(c3,A) | -P(A,f10(c3)). [resolve(40,a,19,b)].
42 P(f9(c1,c2),f10(c3)). [resolve(35,b,40,a),unit_del(a,24)].
43 P(c2,f10(c3)). [resolve(42,a,34,a)].
44 $F. [resolve(43,a,41,b),unit_del(a,25)].
```

Proof of Theorem (t_d8)

```
1 SD(A,B) <-> (exists C PRE(A,C)) & (all C (PRE(A,C) -> EXD(A,B,C))). [assumption].
2 EXD(A,B,C) -> PRE(A,C) & PRE(B,C). [assumption].
3 DQT(A,B) -> SD(A,B). [assumption].
4 DQT(A,B) & PRE(A,C) -> PRE(B,C). [goal].
5 -SD(A,B) | -PRE(A,C) | EXD(A,B,C). [clausify(1)].
6 -DQT(A,B) | SD(A,B). [clausify(3)].
7 -EXD(A,B,C) | PRE(B,C). [clausify(2)].
8 DQT(c1,c2). [deny(4)].
9 PRE(c1,c3). [deny(4)].
10 -PRE(c2,c3). [deny(4)].
11 -DQT(A,B) | -PRE(A,C) | EXD(A,B,C). [resolve(6,b,5,a)].
12 -DQT(c1,A) | EXD(c1,A,c3). [resolve(11,b,9,a)].
13 EXD(c1,c2,c3). [resolve(12,a,8,a)].
14 $F. [resolve(13,a,7,a),unit_del(a,10)].
```

Proof of Theorem (t_d10)

```
1 -(exists A (PED(A) & NPED(A))). [assumption].
2 (exists A exists B exists C exists D (tP(A,E,D) & PED(A) & tP(B,E,C) & NPED(A))) -> AS(E). [goal].
3 -PED(A) | -NPED(A). [clausify(1)].
4 PED(c2). [deny(2)].
5 NPED(c2). [deny(2)].
6 $F. [resolve(4,a,3,a),unit_del(a,5)].
```

Proof of Theorem (t_d15)

```
1 K(A,B,C) -> -K(B,A,C). [assumption].
2 K(A,B,C) -> A != B. [goal].
3 -K(A,B,C) | -K(B,A,C). [clausify(1)].
4 K(c1,c2,c3). [deny(2)].
5 c2 = c1. [deny(2)].
6 -K(A,A,B). [factor(3,a,b)].
7 $F. [back_rewrite(4),rewrite([5(2)]),unit_del(a,6)].
```

Proof of Theorem (t_d16)

```
1 P(A,B) & P(B,C) -> P(A,C). [assumption].
2 PRE(A,B) <-> (exists C (TLC(A,C) & P(B,C))). [assumption].
3 TLC(A,B) & TLC(A,C) -> B = C. [assumption].
4 K(A,B,C) -> PRE(A,C) & PRE(B,C). [assumption].
5 K(A,B,C) & PRE(B,D) -> (exists E exists F (P(F,D) & K(E,B,F))). [assumption].
6 K(A,B,C) & PRE(B,D) & P(E,D) -> (exists F exists V6 (P(F,E) & K(V6,B,F))). [goal].
7 -P(A,B) | -P(B,C) | P(A,C). [clausify(1)].
8 -PRE(A,B) | TLC(A,f9(A,B)). [clausify(2)].
9 -PRE(A,B) | P(B,f9(A,B)). [clausify(2)].
10 PRE(A,B) | -TLC(A,C) | -P(B,C). [clausify(2)].
11 -TLC(A,B) | -TLC(A,C) | C = B. [clausify(3)].
12 -K(A,B,C) | PRE(B,C). [clausify(4)].
13 -K(A,B,C) | -PRE(B,D) | P(f26(A,C,D,B),D). [clausify(5)].
14 -K(A,B,C) | -PRE(B,D) | K(f25(A,C,D,B),B,f26(A,C,D,B)). [clausify(5)].
15 K(c1,c5,c2). [deny(6)].
16 PRE(c5,c3). [deny(6)].
17 P(c4,c3). [deny(6)].
18 -P(A,c4) | -K(B,c5,A). [deny(6)].
19 -PRE(c5,A) | K(f25(c1,c2,A,c5),c5,f26(c1,c2,A,c5)). [resolve(15,a,14,a)].
20 -PRE(c5,A) | P(f26(c1,c2,A,c5),A). [resolve(15,a,13,a)].
21 PRE(c5,c2). [resolve(15,a,12,a)].
22 P(c3,f9(c5,c3)). [resolve(16,a,9,a)].
23 TLC(c5,f9(c5,c3)). [resolve(16,a,8,a)].
24 -P(c3,A) | P(c4,A). [resolve(17,a,7,a)].
25 TLC(c5,f9(c5,c2)). [resolve(21,a,8,a)].
26 -TLC(c5,A) | f9(c5,c3) = A. [resolve(23,a,11,b)].
27 PRE(c5,A) | -P(A,f9(c5,c2)). [resolve(25,a,10,b)].
28 P(c4,f9(c5,c3)). [resolve(24,a,22,a)].
29 f9(c5,c3) = f9(c5,c2). [resolve(26,a,25,a)].
```

```
30 P(c4,f9(c5,c2)).  [back_rewrite(28),rewrite([29(4)])].
31 PRE(c5,c4).  [resolve(27,b,30,a)].
32 P(f26(c1,c2,c4,c5),c4).  [resolve(31,a,20,a)].
33 K(f25(c1,c2,c4,c5),c5,f26(c1,c2,c4,c5)).  [resolve(31,a,19,a)].
34 $F.  [resolve(33,a,18,b),unit_del(a,32)].
```

Proof of Theorem (t_d17)

```
1 PD(A) & PRE(A,B) & P(C,B) -> (exists D exists E (P(E,C) & PC(D,A,E))).  [assumption].
2 PD(A) & PRE(A,B) & P(C,B) -> (exists D exists E (P(E,C) & PC(D,A,E))).  [goal].
3 -PD(A) | -PRE(A,B) | -P(C,B) | P(f33(B,C,A),C).  [clausify(1)].
4 -PD(A) | -PRE(A,B) | -P(C,B) | PC(f32(B,C,A),A,f33(B,C,A)).  [clausify(1)].
5 PD(c3).  [deny(2)].
6 PRE(c3,c1).  [deny(2)].
7 P(c2,c1).  [deny(2)].
8 -P(A,c2) | -PC(B,c3,A).  [deny(2)].
9 -P(A,c1) | PC(f32(c1,A,c3),c3,f33(c1,A,c3)).  [resolve(6,a,4,b),unit_del(a,5)].
10 -P(A,c1) | P(f33(c1,A,c3),A).  [resolve(6,a,3,b),unit_del(a,5)].
11 PC(f32(c1,c2,c3),c3,f33(c1,c2,c3)).  [resolve(9,a,7,a)].
12 P(f33(c1,c2,c3),c2).  [resolve(10,a,7,a)].
13 $F.  [resolve(11,a,8,b),unit_del(a,12)].
```

Proof of Theorem (t_d18)

```
1 tP(A,B,C) -> PRE(A,C) & PRE(B,C).  [assumption].
2 tP(A,B,C) & P(D,C) -> tP(A,B,D).  [assumption].
3 ED(A) & PRE(A,B) -> tP(A,A,B).  [assumption].
4 ED(A) & PRE(A,B) -> (exists C exists D (P(D,B) & PC(A,C,D))).  [assumption].
5 ED(A) & PRE(A,B) & P(C,B) -> (exists D exists E (P(E,C) & PC(A,D,E))).  [goal].
6 -tP(A,B,C) | PRE(B,C).  [clausify(1)].
7 -tP(A,B,C) | -P(D,C) | tP(A,B,D).  [clausify(2)].
8 -ED(A) | -PRE(A,B) | tP(A,A,B).  [clausify(3)].
9 -ED(A) | -PRE(A,B) | P(f31(B,A),B).  [clausify(4)].
10 -ED(A) | -PRE(A,B) | PC(A,f30(B,A),f31(B,A)).  [clausify(4)].
11 ED(c3).  [deny(5)].
12 PRE(c3,c1).  [deny(5)].
13 P(c2,c1).  [deny(5)].
14 -P(A,c2) | -PC(c3,B,A).  [deny(5)].
15 tP(c3,c3,c1).  [resolve(12,a,8,b),unit_del(a,11)].
16 -P(A,c1) | tP(c3,c3,A).  [resolve(15,a,7,a)].
17 tP(c3,c3,c2).  [resolve(16,a,13,a)].
18 PRE(c3,c2).  [resolve(17,a,6,a)].
19 PC(c3,f30(c2,c3),f31(c2,c3)).  [resolve(18,a,10,b),unit_del(a,11)].
20 P(f31(c2,c3),c2).  [resolve(18,a,9,b),unit_del(a,11)].
21 $F.  [resolve(19,a,14,b),unit_del(a,20)].
```

Proof of Theorem (t_d19)

```
1 PC(A,B,C) -> ED(A) & PD(B) & T(C).  [assumption].
2 PD(A) & PRE(A,B) -> (exists C exists D (P(D,B) & PC(C,A,D))).  [assumption].
3 PC(A,B,C) & PRE(B,D) -> (exists E exists C (P(C,D) & PC(E,B,C))).  [goal].
4 -PC(A,B,C) | PD(B).  [clausify(1)].
5 -PD(A) | -PRE(A,B) | P(f29(B,A),B).  [clausify(2)].
6 -PD(A) | -PRE(A,B) | PC(f28(B,A),A,f29(B,A)).  [clausify(2)].
7 PC(c1,c4,c2).  [deny(3)].
8 PRE(c4,c3).  [deny(3)].
9 -P(A,c3) | -PC(B,c4,A).  [deny(3)].
10 PD(c4).  [resolve(7,a,4,a)].
11 PC(f28(c3,c4),c4,f29(c3,c4)).  [resolve(8,a,6,b),unit_del(a,10)].
12 P(f29(c3,c4),c3).  [resolve(8,a,5,b),unit_del(a,10)].
13 $F.  [resolve(11,a,9,b),unit_del(a,12)].
```

Proof of Theorem (t_d20)

```
1 PC(A,B,C) -> ED(A) & PD(B) & T(C).  [assumption].
2 ED(A) & PRE(A,B) -> (exists C exists D (P(D,B) & PC(A,C,D))).  [assumption].
3 PC(A,B,C) & PRE(A,D) -> (exists E exists C (P(C,D) & PC(A,E,C))).  [goal].
4 -PC(A,B,C) | ED(A).  [clausify(1)].
5 -ED(A) | -PRE(A,B) | P(f31(B,A),B).  [clausify(2)].
6 -ED(A) | -PRE(A,B) | PC(A,f30(B,A),f31(B,A)).  [clausify(2)].
7 PC(c4,c1,c2).  [deny(3)].
8 PRE(c4,c3).  [deny(3)].
9 -P(A,c3) | -PC(c4,B,A).  [deny(3)].
10 ED(c4).  [resolve(7,a,4,a)].
11 PC(c4,f30(c3,c4),f31(c3,c4)).  [resolve(8,a,6,b),unit_del(a,10)].
12 P(f31(c3,c4),c3).  [resolve(8,a,5,b),unit_del(a,10)].
13 $F.  [resolve(11,a,9,b),unit_del(a,12)].
```

Proof of Theorem (t_d21)

```
1 PC(A,B,C) -> ED(A) & PD(B) & T(C).  [assumption].
2 -(exists A (ED(A) & PD(A))).  [assumption].
3 PC(A,B,C) -> A != B.  [goal].
4 -PC(A,B,C) | ED(A).  [clausify(1)].
5 -PC(A,B,C) | PD(B).  [clausify(1)].
6 -ED(A) | -PD(A).  [clausify(2)].
```

```
7 PC(c1,c2,c3).  [deny(3)].
8 c2 = c1.  [deny(3)].
9 PC(c1,c1,c3).  [back_rewrite(7),rewrite([8(2)])].
10 PD(c1).  [resolve(9,a,5,a)].
11 ED(c1).  [resolve(9,a,4,a)].
12 $F.  [resolve(11,a,6,a),unit_del(a,10)].
```

Proof of Theorem (t_d22)

```
154. ! [X0,X4] : ((PRE(X0,X4) & PED(X0)) => ? [X3] : SLC(X0,X3,X4)) [input]
196. ! [X0,X4] : ((PRE(X0,X4) & PED(X0)) => ? [X3] : SLC(X0,X3,X4)) [input]
197. ~! [X0,X4] : ((PRE(X0,X4) & PED(X0)) => ? [X3] : SLC(X0,X3,X4)) [negated conjecture 196]
305. ! [X0,X1] : ((PRE(X0,X1) & PED(X0)) => ? [X2] : SLC(X0,X2,X1)) [rectify 154]
344. ~! [X0,X1] : ((PRE(X0,X1) & PED(X0)) => ? [X2] : SLC(X0,X2,X1)) [rectify 197]
566. ! [X0,X1] : (? [X2] : SLC(X0,X2,X1) | (~PRE(X0,X1) | ~PED(X0))) [ennf transformation 305]
567. ! [X0,X1] : (? [X2] : SLC(X0,X2,X1) | ~PRE(X0,X1) | ~PED(X0)) [flattening 566]
575. ? [X0,X1] : (! [X2] : ~SLC(X0,X2,X1) & (PRE(X0,X1) & PED(X0))) [ennf transformation 344]
576. ? [X0,X1] : (! [X2] : ~SLC(X0,X2,X1) & PRE(X0,X1) & PED(X0)) [flattening 575]
705. ! [X1,X0] : (? [X2] : SLC(X0,X2,X1) => SLC(X0,sK67(X0,X1),X1)) [choice axiom]
706. ! [X0,X1] : (SLC(X0,sK67(X0,X1),X1) | ~PRE(X0,X1) | ~PED(X0)) [skolemisation 567,705]
779. ? [X0,X1] : (! [X2] : ~SLC(X0,X2,X1) & PRE(X0,X1) & PED(X0)) => (! [X2] : ~SLC(sK85,X2,sK86) & PRE(sK85,sK86)
     & PED(sK85)) [choice axiom]
780. ! [X2] : ~SLC(sK85,X2,sK86) & PRE(sK85,sK86) & PED(sK85) [skolemisation 576,779]
1029. SLC(X0,sK67(X0,X1),X1) | ~PRE(X0,X1) | ~PED(X0) [cnf transformation 706]
1136. PED(sK85) [cnf transformation 780]
1137. PRE(sK85,sK86) [cnf transformation 780]
1138. ~SLC(sK85,X2,sK86) [cnf transformation 780]
1993. ~PRE(sK85,sK86) | ~PED(sK85) [resolution 1029,1138]
1999. ~PED(sK85) [subsumption resolution 1993,1137]
2000. $false [subsumption resolution 1999,1136]
```

11.2 Validation of DOLCE-BFO mapping (non-theorems)

Below we lists all the axioms characterizing the DOLCE taxonomy. These axioms must be included in each counterexample in order to obtain its full syntactical description. Since the taxonomy is constant among the counterexamples, we write it only once. Notice that we do not write formulas with defined predicates. This is because such predicates were removed from each axioms to be proved through automated substitution with the corresponding definient formula.

DOLCE taxonomy

```
AB(X) | ED(X) | PD(X) | Q(X).
AS(X) | NPED(X) | PED(X) <-> ED(X).
PRO(X) | ST(X) -> STV(X).
EV(X) | STV(X) -> PD(X).
AQ(X) | PQ(X) | TQ(X) <-> Q(X).
ACC(X) | ACH(X) -> EV(X).
APO(X) | NAPO(X) <-> POB(X).
SAG(X) | SC(X) -> ASO(X).
ASO(X) | NASO(X) <-> SOB(X).
SOB(X) | MOB(X) -> NPOB(X).
AR(X) | PR(X) | TR(X) <-> R(X).
R(X) -> AB(X).
F(X) | M(X) | POB(X) -> PED(X).
NPOB(X) -> NPED(X).
S(X) -> PR(X).
SL(X) -> PQ(X).
T(X) -> TR(X).
TL(X) -> TQ(X).
-(exists X (AB(X) & ED(X))).
-(exists X (AB(X) & PD(X))).
-(exists X (AB(X) & Q(X))).
-(exists X (ED(X) & PD(X))).
-(exists X (PD(X) & Q(X))).
-(exists X (ED(X) & Q(X))).
-(exists X (PED(X) & NPED(X))).
-(exists X (PED(X) & AS(X))).
-(exists X (NPED(X) & AS(X))).
-(exists X (M(X) & F(X))).
-(exists X (F(X) & POB(X))).
-(exists X (M(X) & POB(X))).
-(exists X (MOB(X) & SOB(X))).
-(exists X (ASO(X) & NASO(X))).
-(exists X (SAG(X) & SC(X))).
-(exists X (APO(X) & NAPO(X))).
-(exists X (EV(X) & STV(X))).
-(exists X (ACH(X) & ACC(X))).
-(exists X (ST(X) & PRO(X))).
-(exists X (TQ(X) & PQ(X))).
-(exists X (PQ(X) & AQ(X))).
-(exists X (TQ(X) & AQ(X))).
-(exists X (TR(X) & PR(X))).
-(exists X (PR(X) & AR(X))).
-(exists X (TR(X) & AR(X))).
```

Counter model to (t_{db}5)

```
(T(t1) & T(t2) & T(t3) & T(t4) & T(t123) & T(t234)).

all X (-(ED(X))).
all X (-(PD(X))).
all X (-(Q(X))).

all X ((X = t1) | (X = t2) | (X = t3) | (X = t4) | (X = t123) | (X = t234)).

(t1 != t2) & (t1 != t3) & (t1 != t4) & (t1 != t123) & (t1 != t234) & (t2 != t3) & (t2 != t4) & (t2 != t123) & (t2
      != t234) & (t3 != t4) & (t3 != t123) & (t3 != t234) & (t4 != t123) & (t4 != t234) & (t123 != t234).

all X all Y (P(X,Y) <-> ((X = t1 & Y = t1) | (X = t2 & Y = t2) | (X = t3 & Y = t3) | (X = t4 & Y = t4) | (X =
      t123 & Y = t123) | (X = t234 & Y = t234) | (X = t1 & Y = t123) | (X = t2 & Y = t123) | (X = t3 & Y = t123)
      | (X = t2 & Y = t234) | (X = t3 & Y = t234) | (X = t4 & Y = t234))).
all X all Y (-(DQT(X,Y))).
all X all Y (-(QL(X,Y))).
all X all Y (-(TLC(X,Y))).
all X all Y all Z (- (TQL(X,Y,Z))).
all X all Y all Z (- (tP(X,Y,Z))).
all X all Y all Z (- (PC(X,Y,Z))).
all X all Y all Z (- (K(X,Y,Z))).
all X all Y all Z (- (EXD(X,Y,Z))).
all X all Y all Z (- (SLC(X,Y,Z))).
```

Counter model to (t_{db}7), (t_{db}11)

```
(M(m1) & PD(p1) & PD(p2) & PD(p3) & PD(p12) & PD(p123) & T(t1) & T(t2) & T(t12) & S(s1) & TL(tl1) & TL(t12) & TL(
      t13) & TL(tl12) & TL(tl123) & SL(sl1)).

all X ((X=m1) | (X=p1) | (X=p2) | (X=p3) | (X=p12) | (X=p123) | (X=t1) | (X=t2) | (X=t12) | (X=s1) | (X=tl1) | (X=
      t12) | (X=t13) | (X=tl12) | (X=tl123) | (X=sl1)).

((p1 != p2) & (p1 != p12) & (p2 != p12) & (p1 != p123) & (p2 != p123) & (p12 != p123) & (t1 != t2) & (t1 != t12) &
      (t2 != t12)).
(p1 != p3) & (p2 != p3) & (p12 != p3) & (p123 != p3).

((tl1 != tl2) & (tl1 != tl3) & (tl1 != tl12) & (tl1 != tl123)).
((tl2 != tl3) & (tl2 != tl12) & (tl2 != tl123)).
((tl3 != tl12) & (tl3 != tl123)).
((tl12 != tl123)).

all X (- (EV(X))).
all X (- (STV(X))).

all X all Y (P(X,Y) <-> ((X = p1 & Y = p1) | (X = p2 & Y = p2) | (X = p3 & Y = p3) | (X = p12 & Y = p12) | (X =
      p123 & Y = p123) | (X = p1 & Y = p12) | (X = p2 & Y = p12) | (X = p1 & Y = p123) | (X = p2 & Y = p123) | (
      X = p3 & Y = p123) | (X = p12 & Y = p123) | (X = t1 & Y = t1) | (X = t2 & Y = t2) | (X = t12 & Y = t12) | (
      X = t1 & Y = t12) | (X = t2 & Y = t12) | (X = s1 & Y = s1))).
all X all Y (DQT(X,Y) <-> ((X = tl1 & Y = p1) | (X = tl2 & Y = p2) | (X = tl3 & Y = p3) | (X = tl12 & Y = p12) | (
      X = tl123 & Y = p123) | (X = sl1 & Y = m1))).
all X all Y (QL(X,Y) <-> ((X = t1 & Y = tl1) | (X = t1 & Y = t12) | (X = t2 & Y = t13) | (X = t1 & Y = tl12) | (
      X = t12 & Y = tl123))).
all X all Y (TLC(X,Y) <-> ((X = m1 & Y = t12) | (X = p1 & Y = t1) | (X = p2 & Y = t1) | (X = p3 & Y = t2) | (X =
      p12 & Y = t1) | (X = p123 & Y = t12) | (X = tl1 & Y = t1) | (X = tl2 & Y = t1) | (X = tl3 & Y = t2) | (X =
      tl12 & Y = t1) | (X = tl123 & Y = t12) | (X = sl1 & Y = t12))).
all X all Y all Z (TQL(X,Y,Z) <-> ((X = s1 & Y = sl1 & Z = t1) | (X = s1 & Y = sl1 & Z = t2) | (X = s1 & Y = sl1 &
      Z = t12))).
all X all Y all Z (tP(X,Y,Z) <-> ((X = m1 & Y = m1 & Z = t1) | (X = m1 & Y = m1 & Z = t2) | (X = m1 & Y = m1 & Z =
      t12))).
all X all Y all Z (PC(X,Y,Z) <-> ((X = m1 & Y = p1 & Z = t1) | (X = m1 & Y = p2 & Z = t1) | (X = m1 & Y = p3 & Z =
      t2) | (X = m1 & Y = p12 & Z = t1) | (X = m1 & Y = p123 & Z = t1) | (X = m1 & Y = p123 & Z = t2) | (X = m1
      & Y = p123 & Z = t12))).
all X all Y all Z (- (K(X,Y,Z))).
all X all Y all Z (EXD(X,Y,Z) <-> ((X = m1 & Y = p1 & Z = t1) | (X = m1 & Y = p2 & Z = t1) | (X = m1 & Y = p3 & Z
      = t2) | (X = m1 & Y = p12 & Z = t1) | (X = m1 & Y = p123 & Z = t1) | (X = m1 & Y = p123 & Z = t2) | (X = m1
      & Y = p123 & Z = t12) | (X = p1 & Y = m1 & Z = t1) | (X = p2 & Y = m1 & Z = t1) | (X = p3 & Y = m1 & Z = t2
      ) | (X = p12 & Y = m1 & Z = t1) | (X = p123 & Y = m1 & Z = t1) | (X = p123 & Y = m1 & Z = t2) | (X = p123 &
      Y = m1 & Z = t12) | (X = tl1 & Y = p1 & Z = t1) | (X = tl2 & Y = p2 & Z = t1) | (X = tl3 & Y = p3 & Z = t2)
      | (X = tl12 & Y = p12 & Z = t1) | (X = tl123 & Y = p123 & Z = t1) | (X = tl123 & Y = p123 & Z = t2) | (X =
      tl123 & Y = p123 & Z = t12) | (X = sl1 & Y = m1 & Z = t1) | (X = sl1 & Y = m1 & Z = t2) | (X = sl1 & Y = m1
      & Z = t12))).
all X all Y all Z (SLC(X,Y,Z) <-> ((X = m1 & Y = s1 & Z = t1) | (X = m1 & Y = s1 & Z = t2) | (X = m1 & Y = s1 & Z
      = t12) | (X = p1 & Y = s1 & Z = t1) | (X = p2 & Y = s1 & Z = t1) | (X = p3 & Y = s1 & Z = t2) | (X = p12 &
      Y = s1 & Z = t1) | (X = p123 & Y = s1 & Z = t1) | (X = p123 & Y = s1 & Z = t2) | (X = p123 & Y = s1 & Z =
      t12) | X = sl1 & Y = s1 & Z = t1) | (X = sl1 & Y = s1 & Z = t2) | (X = sl1 & Y = s1 & Z = t12)).
```

Counter model to (t_{db}9), (t_{db}24), (t_{db}26), (t_{db}95)

```
M(m1) & PD(p1) & T(t1) & S(s1) & TL(tl1) & SL(sl1).

all X ((X=m1) | (X=p1) | (X=t1) | (X=s1) | (X=tl1) | (X=sl1)).

all X (- (EV(X))).
all X (- (STV(X))).

all X all Y (P(X,Y) <-> ((X = p1 & Y = p1) | (X = t1 & Y = t1) | (X = s1 & Y = s1))).
all X all Y (DQT(X,Y) <-> ((X = tl1 & Y = p1) | (X = sl1 & Y = m1))).
all X all Y (QL(X,Y) <-> ((X = t1 & Y = tl1))).
all X all Y (TLC(X,Y) <-> ((X = m1 & Y = t1) | (X = p1 & Y = t1) | (X = tl1 & Y = t1) | (X = sl1 & Y = t1))).
all X all Y all Z (TQL(X,Y,Z) <-> ((X = s1 & Y = sl1 & Z = t1))).
all X all Y all Z (tP(X,Y,Z) <-> ((X = m1 & Y = m1 & Z = t1))).
all X all Y all Z (PC(X,Y,Z) <-> ((X = m1 & Y = p1 & Z = t1))).
all X all Y all Z (- (K(X,Y,Z))).
all X all Y all Z (EXD(X,Y,Z) <-> ((X = m1 & Y = p1 & Z = t1) | (X = p1 & Y = m1 & Z = t1) | (X = tl1 & Y = p1 &
      Z = t1) | (X = s1l & Y = m1 & Z = t1))).
all X all Y all Z (SLC(X,Y,Z) <-> ((X = m1 & Y = s1 & Z = t1) | (X = p1 & Y = s1 & Z = t1) | (X = sl1 & Y = s1 & Z
      = t1))).
```

Counter model to (t_{db}29), (t_{db}79)

```
(M(e1) & NPED(e2) & PD(p1) & PD(p2) & TL(tl1) & TL(tl2) & SL(sl1) & T(t1) & T(t2) &T(t12) & S(s1)).

all X ((X = e1) | (X = e2) | (X = p1) | (X = p2) | (X = tl1) | (X = tl2) | (X = sl1) | (X = t1) | (X = t2) | (X =
      t12) | (X = s1)).

(p1 != p2) & (tl1 != tl2) & (t1 != t2) & (t1 != t12) & (t2 != t12).

all X (- (EV(X))).
all X (- (STV(X))).
all X (- (NPOB(X))).

all X all Y (P(X,Y) <-> ((X = p1 & Y = p1) | (X = p2 & Y = p2) | (X = t1 & Y = t1) | (X = t2 & Y = t2) | (X = t12
      & Y = t12) | (X = s1 & Y = s1) | (X = t1 & Y = t12) | (X = t2 & Y = t12))).
all X all Y (DQT(X,Y) <-> ((X = tl1 & Y = p1) | (X = tl2 & Y = p2) | (X = sl1 & Y = e1))).
all X all Y (QL(X,Y) <-> ((X = t12 & Y = tl1) | (X = t12 & Y = tl2))).
all X all Y (TLC(X,Y) <-> ((X = e1 & Y = t12) | (X = e2 & Y = t12) | (X = p1 & Y = t12) | (X = p2 & Y = t12) | (X
      = tl1 & Y = t12) | (X = t12 & Y = t12) | (X = sl1 & Y = t12))).
all X all Y all Z (TQL(X,Y,Z) <-> ((X = s1 & Y = sl1 & Z = t1) | (X = s1 & Y = sl1 & Z = t2) | (X = s1 & Y = sl1 &
      Z = t12))).
all X all Y all Z (tP(X,Y,Z) <-> ((X = e1 & Y = e1 & Z = t1) | (X = e1 & Y = e1 & Z = t2) | (X = e1 & Y = e1 & Z =
      t12) | (X = e2 & Y = e2 & Z = t1) | (X = e2 & Y = e2 & Z = t2) | (X = e2 & Y = e2 & Z = t12))).
all X all Y all Z (PC(X,Y,Z) <-> ((X = e1 & Y = p1 & Z = t1) | (X = e1 & Y = p1 & Z = t2) | (X = e1 & Y = p1 & Z =
      t12) | (X = e2 & Y = p2 & Z = t1) | (X = e2 & Y = p2 & Z = t2) | (X = e2 & Y = p2 & Z = t12))).
all X all Y all Z (-(K(X,Y,Z))).
all X all Y all Z (EXD(X,Y,Z) <-> ((X = e2 & Y = e1 & Z = t1) | (X = e2 & Y = e1 & Z = t2) | (X = e1 & Y = p1 & Z
      = t1) | (X = e1 & Y = p1 & Z = t2) | (X = e1 & Y = p1 & Z = t12) | (X = p1 & Y = e1 & Z = t1) | (X = p1 & Y
      = e1 & Z = t2) | (X = p1 & Y = e1 & Z = t12) | (X = e2 & Y = p2 & Z = t1) | (X = e2 & Y = p2 & Z = t2) | (X
      = e2 & Y = p2 & Z = t12) |(X = p2 & Y = e2 & Z = t1) | (X = p2 & Y = e2 & Z = t2) | (X = p2 & Y = e2 & Z =
      t12) | (X = tl1 & Y = p1 & Z = t1) | (X = tl1 & Y = p1 & Z = t2) | (X = tl1 & Y = p1 & Z = t12) | (X = t12 &
       Y = p2 & Z = t1) | (X = t12 & Y = p2 & Z = t2) | (X = t12 & Y = p2 & Z = t12) | (X = sl1 & Y = e1 & Z = t1)
       | (X = sl1 & Y = e1 & Z = t2) | (X = sl1 & Y = e1 & Z = t12) | (X = e2 & Y = sl1 & Z = t1) | (X = e2 & Y =
      sl1 & Z = t2) | (X = e2 & Y = sl1 & Z = t12))).
all X all Y all Z (SLC(X,Y,Z) <-> ((X = e1 & Y = s1 & Z = t1) | (X = e1 & Y = s1 & Z = t2) | (X = e1 & Y = s1 & Z
      = t12) | (X = p1 & Y = s1 & Z = t1) | (X = p1 & Y = s1 & Z = t2) | (X = p1 & Y = s1 & Z = t12) | (X = sl1 &
      Y = s1 & Z = t1) | (X = sl1 & Y = s1 & Z = t2) | (X = sl1 & Y = s1 & Z = t12))).
```

Counter model to (t_{db}32) (left to right implication)

```
NPED(e1) & M(m1) & PD(p1) & PD(p2) & TL(tl1) & TL(tl2) & SL(slm1) & T(t1) & S(s1).

all X ((X=e1) | (X=m1) | (X=p1) | (X=p2) | (X=tl1) | (X=tl2) | (X=slm1) | (X=t1) | (X=s1)).

(p1 != p2) & (tl1 != tl2).

all X (~ (EV(X))).
all X (~ (STV(X))).
all X (~ (NPOB(X))).

all X all Y (P(X,Y) <-> ((X = p1 & Y = p1) | (X = p2 & Y = p2) | (X = t1 & Y = t1) | (X = s1 & Y = s1))).
all X all Y (DQT(X,Y) <-> ((X = tl1 & Y = p1) | (X = tl2 & Y = p2) | (X = slm1 & Y = m1))).
all X all Y (QL(X,Y) <-> ((X = t1 & Y = tl1) | (X = t1 & Y = tl2))).
all X all Y (TLC(X,Y) <-> ((X = m1 & Y = t1) | (X = e1 & Y = t1) | (X = p1 & Y = t1) | (X = p2 & Y = t1) | (X =
      tl1 & Y = t1) | (X = tl2 & Y = t1) | (X = slm1 & Y = t1))).
all X all Y all Z (TQL(X,Y,Z) <-> (X = s1 & Y = slm1 & Z = t1)).
all X all Y all Z (tP(X,Y,Z) <-> ((X = m1 & Y = m1 & Z = t1) | (X = e1 & Y = e1 & Z = t1))).
all X all Y all Z (PC(X,Y,Z) <-> ((X = m1 & Y = p2 & Z = t1) | (X = e1 & Y = p1 & Z = t1))).
all X all Y all Z (~ (K(X,Y,Z))).
all X all Y all Z (EXD(X,Y,Z) <-> ((X = m1 & Y = p2 & Z = t1) | (X = p2 & Y = m1 & Z = t1) | (X = e1 & Y = p1 & Z
      = t1) | (X = p1 & Y = e1 & Z = t1) | (X = tl1 & Y = p1 & Z = t1) | (X = tl2 & Y = p2 & Z = t1) | (X = slm1 &
       Y = m1 & Z = t1) | (X = e1 & Y = m1 & Z = t1))).
all X all Y all Z ((SLC(X,Y,Z) <-> (X = m1 & Y = s1 & Z = t1) | (X = p2 & Y = s1 & Z = t1) | (X = slm1 & Y = s1 &
      Z = t1))).
```

Counter model to (t_{db}32) (right to left implication)

```
M(m1) & PD(p1) & PD(p2) & T(t1) & S(s1) & TL(tl1) & TL(tl2) & SL(sl1).

all X ((X=m1) | (X=p1) | (X=p2) | (X=t1) | (X=s1) | (X=tl1) | (X=tl2) | (X=sl1)).

(p1 != p2) & (tl1 != tl2).

all X (~ (EV(X))).
all X (~ (STV(X))).

all X all Y (P(X,Y) <-> ((X = p1 & Y = p1) | (X = p2 & Y = p2) | (X = t1 & Y = t1) | (X = s1 & Y = s1))).
all X all Y (DQT(X,Y) <-> ((X = tl1 & Y = p1) | (X = tl2 & Y = p2) | (X = sl1 & Y = m1))).
all X all Y (QL(X,Y) <-> ((X = t1 & Y = tl1) | (X = t1 & Y = tl2))).
all X all Y (TLC(X,Y) <-> ((X = m1 & Y = t1) | (X = p1 & Y = t1) | (X = p2 & Y = t1) | (X = tl1 & Y = t1) | (X =
      tl2 & Y = t1) | (X = sl1 & Y = t1))).
all X all Y all Z (TQL(X,Y,Z) <-> ((X = s1 & Y = sl1 & Z = t1))).
all X all Y all Z (tP(X,Y,Z) <-> ((X = m1 & Y = m1 & Z = t1))).
all X all Y all Z (PC(X,Y,Z) <-> ((X = m1 & Y = p1 & Z = t1) | (X = m1 & Y = p2 & Z = t1))).
all X all Y all Z (~ (K(X,Y,Z))).
all X all Y all Z (EXD(X,Y,Z) <-> ((X = m1 & Y = p1 & Z = t1) | (X = p1 & Y = m1 & Z = t1) | (X = m1 & Y = p2 & Z
      = t1) | (X = p2 & Y = m1 & Z = t1) | (X = tl1 & Y = p1 & Z = t1) | (X = tl2 & Y = p2 & Z = t1) | (X = sl1 &
      Y = m1 & Z = t1))).
all X all Y all Z (SLC(X,Y,Z) <-> ((X = m1 & Y = s1 & Z = t1) | (X = p1 & Y = s1 & Z = t1) | (X = p2 & Y = s1 & Z
      = t1) | (X = sl1 & Y = s1 & Z = t1))).
```

Counter model to (t_{db}37), (t_{db}41)

```
(M(m1) & F(e1) & F(e2) & NAPO(e3) & PD(p1) & PD(p2) & TL(tl1) & TL(tl2) & SL(slm1) & SL(sl1) & SL(sl2) & SL(sl3) &
        T(t1) & S(s1) & S(s2)).

all X ((X = m1) | (X = e1) | (X = e2) | (X = e3) | (X = p1) | (X = p2) | (X = tl1) | (X = tl2) | (X = slm1) | (X =
        sl1) | (X = sl2) | (X = sl3) | (X = t1) | (X = s1) | (X = s2)).

(e1 != e2) & (p1 != p2) & (tl1 != tl2) & (sl1 != sl2) & (sl1 != sl3) & (sl1 != slm1) & (sl2 != sl3) & (sl2 != slm1
      ) & (sl3 != slm1) & (s1 != s2).

all X (- (EV(X))).
all X (- (STV(X))).

all X all Y (P(X,Y) <-> ((X = p1 & Y = p1) | (X = p2 & Y = p2) | (X = t1 & Y = t1) | (X = s1 & Y = s1) | (X = s2 &
        Y = s2))).
all X all Y (DQT(X,Y) <-> ((X = tl1 & Y = p1) | (X = tl2 & Y = p2) | (X = slm1 & Y = m1) | (X = sl1 & Y = e1) | (X
        = sl2 & Y = e2) | (X = sl3 & Y = e3))).
all X all Y (QL(X,Y) <-> ((X = t1 & Y = tl1) | (X = t1 & Y = tl2))).
all X all Y (TLC(X,Y) <-> ((X = m1 & Y = t1) | (X = e1 & Y = t1) | (X = e2 & Y = t1) | (X = e3 & Y = t1) | (X = p1
        & Y = t1) | (X = p2 & Y = t1) | (X = tl1 & Y = t1) | (X = tl2 & Y = t1) | (X = slm1 & Y = t1) | (X = sl1 &
      Y = t1) | (X = sl2 & Y = t1) | (X = sl3 & Y = t1))).
all X all Y all Z (TQL(X,Y,Z) <-> ((X = s2 & Y = slm1 & Z = t1) | (X = s1 & Y = sl1 & Z = t1) | (X = s1 & Y = sl2
        & Z = t1) | (X = s2 & Y = sl3 & Z = t1))).
all X all Y all Z (tP(X,Y,Z) <-> ((X = m1 & Y = m1 & Z = t1) | (X = e1 & Y = e1 & Z = t1) | (X = e2 & Y = e2 & Z =
        t1) | (X = e3 & Y = e3 & Z = t1) | (X = e1 & Y = e2 & Z = t1) | (X = e2 & Y = e1 & Z = t1))).
all X all Y all Z (PC(X,Y,Z) <-> ((X = m1 & Y = p2 & Z = t1) | (X = e1 & Y = p1 & Z = t1) | (X = e2 & Y = p1 & Z =
        t1) | (X = e3 & Y = p2 & Z = t1))).
all X all Y all Z (K(X,Y,Z) <-> ((X = m1 & Y = e3 & Z = t1))).
all X all Y all Z (EXD(X,Y,Z) <-> ((X = m1 & Y = p2 & Z = t1) | (X = p2 & Y = m1 & Z = t1) | (X = e1 & Y = p1 & Z
      = t1) | (X = p1 & Y = e1 & Z = t1) | (X = e2 & Y = p1 & Z = t1) | (X = p1 & Y = e2 & Z = t1) | (X = e3 & Y =
        p2 & Z = t1) | (X = p2 & Y = e3 & Z = t1) | (X = tl1 & Y = p1 & Z = t1) | (X = tl2 & Y = p2 & Z = t1) | (X
        = slm1 & Y = m1 & Z = t1) | (X = sl1 & Y = e1 & Z = t1) | (X = sl2 & Y = e2 & Z = t1) | (X = sl3 & Y = e3 &
        Z = t1) | (X = e1 & Y = e3 & Z = t1) | (X = e2 & Y = e3 & Z = t1) | (X = e3 & Y = m1 & Z = t1))).
all X all Y all Z (SLC(X,Y,Z) <-> ((X = m1 & Y = s2 & Z = t1) | (X = e1 & Y = s1 & Z = t1) | (X = e2 & Y = s1 & Z
      = t1) | (X = e3 & Y = s2 & Z = t1) | (X = p1 & Y = s1 & Z = t1) | (X = p2 & Y = s2 & Z = t1) | (X = slm1 &
        Y = s2 & Z = t1) | (X = sl1 & Y = s1 & Z = t1) | (X = sl2 & Y = s1 & Z = t1) | (X = sl3 & Y = s2 & Z = t1
        ))).
```

Counter model to (t_{db}38)

```
(T(t1) & S(s1) & S(s2) & S(s3) & S(s4) & S(s123) & S(s234)).

all X ((X = t1) | (X = s1) | (X = s2) | (X = s3) | (X = s4) | (X = s123) | (X = s234)).

(s1 != s2) & (s1 != s3) & (s1 != s4) & (s1 != s123) & (s1 != s234) &
(s2 != s3) & (s2 != s4) & (s2 != s123) & (s2 != s234) &
(s3 != s4) & (s3 != s123) & (s3 != s234) &
(s4 != s123) & (s4 != s234) &
(s123 != s234).

all X (-(ED(X))).
all X (-(PD(X))).
all X (-(Q(X))).

all X all Y (P(X,Y) <-> ((X = t1 & Y = t1) | (X = s1 & Y = s1) | (X = s2 & Y = s2) | (X = s3 & Y = s3) | (X = s4 &
        Y = s4) | (X = s123 & Y = s123) | (X = s234 & Y = s234) | (X = s1 & Y = s123) | (X = s2 & Y = s123) | (X
      = s3 & Y = s123) | (X = s2 & Y = s234) | (X = s3 & Y = s234) | (X = s4 & Y = s234))).
all X all Y (-(DQT(X,Y))).
all X all Y (-(QL(X,Y))).
all X all Y (-(TLC(X,Y))).
all X all Y all Z (- (TQL(X,Y,Z))).
all X all Y all Z (- (tP(X,Y,Z))).
all X all Y all Z  (- (PC(X,Y,Z))).
all X all Y all Z (- (K(X,Y,Z))).
all X all Y all Z (- (EXD(X,Y,Z))).
all X all Y all Z (- (SLC(X,Y,Z))).
```

Counter model to (t_{db}42), (t_{db}45), (t_{db}56), (t_{db}61), (t_{db}97)

```
(M(e1) & NPED(e2) & NPED(e3) & NPED(e23) & PD(p1) & TL(tl1) & SL(sl1) & SL(sl2) & SL(sl23) & T(t1) & T(t2) & T(t12
    ) & S(s1)).

all X ((X = e1) | (X = e2) | (X = e3) | (X = e23) | (X = p1) | (X = tl1) | (X = sl1) | (X = sl2) | (X = sl23) | (X
    = t1) | (X = t2) | (X = t12) | (X = s1)).

(e2 != e3) & (e2 != e23) & (e3 != e23) & (sl1 != sl2) & (sl1 != sl23) & (sl2 != sl23) & (t1 != t2) & (t1 != t12) &
    (t2 != t12).

all X (- (EV(X))).
all X (- (STV(X))).
all X (- (NPOB(X))).

all X all Y (P(X,Y) <-> ((X = p1 & Y = p1) | (X = t1 & Y = t1) | (X = t2 & Y = t2) | (X = t12 & Y = t12) | (X = s1
    & Y = s1))).
all X all Y (DQT(X,Y) <-> ((X = tl1 & Y = p1) | (X = sl1 & Y = e1) | (X = sl2 & Y = e2) | (X = sl23 & Y = e23))).
all X all Y (QL(X,Y) <-> ((X = t12 & Y = tl1))).
all X all Y (TLC(X,Y) <-> ((X = e1 & Y = t12) | (X = e2 & Y = t12) | (X = e3 & Y = t12) | (X = e23 & Y = t12) | (X
    = p1 & Y = t12) | (X = tl1 & Y = t12) | (X = sl1 & Y = t12) | (X = sl2 & Y = t12) | (X = sl23 & Y = t12))).
all X all Y all Z (TQL(X,Y,Z) <-> ( (X = s1 & Y = sl1 & Z = t1) | (X = s1 & Y = sl1 & Z = t2) | (X = s1 & Y = sl1
    & Z = t12) | (X = s1 & Y = sl2 & Z = t1) | (X = s1 & Y = sl2 & Z = t2) | (X = s1 & Y = sl2 & Z = t12) | (X =
    s1 & Y = sl23 & Z = t1) | (X = s1 & Y = sl23 & Z = t2) | (X = s1 & Y = sl23 & Z = t12))).
all X all Y all Z (tP(X,Y,Z) <-> ( (X = e1 & Y = e1 & Z = t1) | (X = e1 & Y = e1 & Z = t2) | (X = e1 & Y = e1 & Z
    = t12) | (X = e2 & Y = e2 & Z = t1) | (X = e2 & Y = e2 & Z = t2) | (X = e2 & Y = e2 & Z = t12) | (X = e3 & Y
    = e3 & Z = t1) | (X = e3 & Y = e3 & Z = t2) | (X = e3 & Y = e3 & Z = t12) | (X = e23 & Y = e23 & Z = t1) |
    (X = e23 & Y = e23 & Z = t2) | (X = e23 & Y = e23 & Z = t12) | (X = e2 & Y = e23 & Z = t1) | (X = e2 & Y =
    e23 & Z = t2) | (X = e2 & Y = e23 & Z = t12) | (X = e3 & Y = e23 & Z = t1) | (X = e3 & Y = e23 & Z = t2) | (
    X = e3 & Y = e23 & Z = t12))).
all X all Y all Z (PC(X,Y,Z) <-> ( (X = e1 & Y = p1 & Z = t1) | (X = e1 & Y = p1 & Z = t2) | (X = e1 & Y = p1 & Z
    = t12) | (X = e2 & Y = p1 & Z = t1) | (X = e2 & Y = p1 & Z = t2) | (X = e2 & Y = p1 & Z = t12) | (X = e3 & Y
    = p1 & Z = t1) | (X = e3 & Y = p1 & Z = t2) | (X = e3 & Y = p1 & Z = t12) | (X = e23 & Y = p1 & Z = t1) | (
    X = e23 & Y = p1 & Z = t2) | (X = e23 & Y = p1 & Z = t12))).
all X all Y all Z (-(K(X,Y,Z))).
all X all Y all Z (EXD(X,Y,Z) <-> ( (X = e1 & Y = p1 & Z = t1) | (X = e1 & Y = p1 & Z = t2) | (X = e1 & Y = p1 & Z
    = t12) | (X = p1 & Y = e1 & Z = t1) | (X = p1 & Y = e1 & Z = t2) | (X = p1 & Y = e1 & Z = t12) | (X = e2 &
    Y = p1 & Z = t1) | (X = e2 & Y = p1 & Z = t2) | (X = e2 & Y = p1 & Z = t12) | (X = p1 & Y = e2 & Z = t1) | (
    X = p1 & Y = e2 & Z = t2) | (X = p1 & Y = e2 & Z = t12) | (X = e3 & Y = p1 & Z = t1) | (X = e3 & Y = p1 & Z
    = t2) | (X = e3 & Y = p1 & Z = t12) | (X = p1 & Y = e3 & Z = t1) | (X = p1 & Y = e3 & Z = t2) | (X = p1 & Y
    = e3 & Z = t12) | (X = e23 & Y = p1 & Z = t1) | (X = e23 & Y = p1 & Z = t2) | (X = e23 & Y = p1 & Z = t12) |
    (X = p1 & Y = e23 & Z = t1) | (X = p1 & Y = e23 & Z = t2) | (X = p1 & Y = e23 & Z = t12) | (X = tl1 & Y =
    p1 & Z = t1) | (X = tl1 & Y = p1 & Z = t2) | (X = tl1 & Y = p1 & Z = t12) | (X = sl1 & Y = e1 & Z = t1) | (X
    = sl1 & Y = e1 & Z = t2) | (X = sl1 & Y = e1 & Z = t12) | (X = sl2 & Y = e2 & Z = t1) | (X = sl2 & Y = e2 &
    Z = t2) | (X = sl2 & Y = e2 & Z = t12) | (X = sl23 & Y = e23 & Z = t1) | (X = sl23 & Y = e23 & Z = t2) | (X
    = sl23 & Y = e23 & Z = t12) | (X = e2 & Y = e1 & Z = t1) | (X = e2 & Y = e1 & Z = t2) | (X = e2 & Y = e1 &
    Z = t12) | (X = e3 & Y = e1 & Z = t1) | (X = e3 & Y = e1 & Z = t2) | (X = e3 & Y = e1 & Z = t12) | (X = e3 &
    Y = sl1 & Z = t1) | (X = e3 & Y = sl1 & Z = t2) | (X = e3 & Y = sl1 & Z = t12) | (X = e23 & Y = e1 & Z = t1
    ) | (X = e23 & Y = e1 & Z = t2) | (X = e23 & Y = e1 & Z = t12))).
all X all Y all Z (SLC(X,Y,Z) <-> ((X = e1 & Y = s1 & Z = t1) | (X = e1 & Y = s1 & Z = t2) | (X = e1 & Y = s1 & Z
    = t12) | (X = e2 & Y = s1 & Z = t1) | (X = e2 & Y = s1 & Z = t2) | (X = e2 & Y = s1 & Z = t12) | (X = e23 &
    Y = s1 & Z = t1) | (X = e23 & Y = s1 & Z = t2) | (X = e23 & Y = s1 & Z = t12) | (X = p1 & Y = s1 & Z = t1) |
    (X = p1 & Y = s1 & Z = t2) | (X = p1 & Y = s1 & Z = t12) | (X = sl1 & Y = s1 & Z = t1) | (X = sl1 & Y = s1
    & Z = t2) | (X = sl1 & Y = s1 & Z = t12) | (X = sl2 & Y = s1 & Z = t1) | (X = sl2 & Y = s1 & Z = t2) | (X =
    sl2 & Y = s1 & Z = t12) | (X = sl3 & Y = s1 & Z = t1) | (X = sl3 & Y = s1 & Z = t2) | (X = sl3 & Y = s1 & Z
    = t12))).
```

Counter model to (t_{ab}43)

```
(M(m1) & F(e1) & F(e2) & F(e12) & NAPO(e3) & PD(p1) & TL(t11) & SL(slm1) & SL(s11) & SL(s12) & SL(s13) & SL(s112)
      & T(t1) & T(t2) & T(t12) & S(s1) & S(s2) & S(s3) & S(s4) & S(s12) & S(s24) & S(s34) & S(s1234)).

all X ((X = m1) | (X = e1) | (X = e2) | (X = e3) | (X = e12) | (X = p1) | (X = t11) | (X = slm1) | (X = s11) | (X
      = s12) | (X = s13) | (X = s112) | (X = t1) |  (X = t2) | (X = t12) | (X = s1) | (X = s2)  | (X = s3) | (X =
      s4)  | (X = s12) | (X = s24)  | (X = s34) | (X = s1234)).

(e1 != e2) & (e1 != e12) & (e2 != e12) & (s11 != s12) & (s11 != s13) & (s11 != slm1) & (s11 != s112) & (s12 != s13
      ) & (s12 != slm1) & (s12 != s112) & (s13 != slm1) & (s13 != s112) & (slm1 != s112) & (t1 != t2) & (t1 != t12
      ) & (t2 != t12) & (s1 != s2) & (s1 != s3) & (s1 != s4) & (s1 != s12) & (s1 != s24) & (s1 != s34) & (s1 !=
      s1234) & (s2 != s3) & (s2 != s4) & (s2 != s12) & (s2 != s24) & (s2 != s34) & (s2 != s1234) & (s3 != s4) & (
      s3 != s12) & (s3 != s24) & (s3 != s34) & (s3 != s1234) & (s4 != s12) & (s4 != s24) & (s4 != s34) & (s4 !=
      s1234) & (s12 != s24) & (s12 != s34) & (s12 != s1234) & (s24 != s34) & (s24 != s1234) & (s34 != s1234).

all X (~ (EV(X))).
all X (~ (STV(X))).

all X all Y (P(X,Y) <-> ((X = p1 & Y = p1) | (X = t1 & Y = t1) | (X = t2 & Y = t2) | (X = t12 & Y = t12) | (X = t1
      & Y = t12) | (X = t2 & Y = t12) | (X = s1 & Y = s1) | (X = s2 & Y = s2) | (X = s3 & Y = s3) | (X = s4 & Y =
      s4)  | (X = s12 & Y = s12)  | (X = s24 & Y = s24) | (X = s34 & Y = s34) | (X = s1234 & Y = s1234) | (X = s1
      & Y = s12) | (X = s2 & Y = s12) | (X = s2 & Y = s24) | (X = s4 & Y = s24)  | (X = s3 & Y = s34)  | (X = s4
      & Y = s34) | (X = s1 & Y = s1234) | (X = s2 & Y = s1234) | (X = s3 & Y = s1234) | (X = s4 & Y = s1234) | (X
      = s12 & Y = s1234) | (X = s24 & Y = s1234) | (X = s34 & Y = s1234))).

all X all Y (DQT(X,Y) <-> ((X = t11 & Y = p1) | (X = slm1 & Y = m1) | (X = s11 & Y = e1) | (X = s12 & Y = e2) | (X
      = s13 & Y = e3) | (X = s112 & Y = e12))).

all X all Y (QL(X,Y) <-> ((X = t12 & Y = t11))).

all X all Y (TLC(X,Y) <-> ((X = m1 & Y = t12) | (X = e1 & Y = t12) | (X = e2 & Y = t12) | (X = e3 & Y = t12) | (X
      = e12 & Y = t12) | (X = p1 & Y = t12) | (X = t11 & Y = t12) | (X = slm1 & Y = t12) | (X = s11 & Y = t12) | (
      X = s12 & Y = t12) | (X = s13 & Y = t12) | (X = s112 & Y = t12))).

all X all Y all Z (TQL(X,Y,Z) <-> ((X = s24 & Y = slm1 & Z = t1) | (X = s24 & Y = slm1 & Z = t2) | (X = s24 & Y =
      slm1 & Z = t12) | (X = s12 & Y = s11 & Z = t1) | (X = s12 & Y = s11 & Z = t2) | (X = s12 & Y = s11 & Z = t12
      ) | (X = s34 & Y = s12 & Z = t1) | (X = s34 & Y = s12 & Z = t2) | (X = s34 & Y = s12 & Z = t12) | (X = s24 &
      Y = s13 & Z = t1) | (X = s24 & Y = s13 & Z = t2) | (X = s24 & Y = s13 & Z = t12) | (X = s1234 & Y = s112 &
      Z = t1) | (X = s1234 & Y = s112 & Z = t2) | (X = s1234 & Y = s112 & Z = t12))).

all X all Y all Z (tP(X,Y,Z) <-> ((X = m1 & Y = m1 & Z = t1) | (X = m1 & Y = m1 & Z = t2) | (X = m1 & Y = m1 & Z =
      t12) | (X = e1 & Y = e1 & Z = t1) | (X = e1 & Y = e1 & Z = t2) | (X = e1 & Y = e1 & Z = t12) | (X = e2 & Y
      = e2 & Z = t1) | (X = e2 & Y = e2 & Z = t2) | (X = e2 & Y = e2 & Z = t12) | (X = e3 & Y = e3 & Z = t1) | (X
      = e3 & Y = e3 & Z = t2) | (X = e3 & Y = e3 & Z = t12) | (X = e12 & Y = e12 & Z = t1) | (X = e12 & Y = e12 &
      Z = t2) | (X = e12 & Y = e12 & Z = t12) | (X = e1 & Y = e12 & Z = t1) | (X = e1 & Y = e12 & Z = t2) | (X =
      e1 & Y = e12 & Z = t12) | (X = e2 & Y = e12 & Z = t1) | (X = e2 & Y = e12 & Z = t2) | (X = e2 & Y = e12 & Z
      = t12))).

all X all Y all Z (K(X,Y,Z) <-> ((X = m1 & Y = e3 & Z = t1) | (X = m1 & Y = e3 & Z = t2) | (X = m1 & Y = e3 & Z =
      t12))).

all X all Y all Z (PC(X,Y,Z) <-> ((X = m1 & Y = p1 & Z = t1) | (X = m1 & Y = p1 & Z = t2) | (X = m1 & Y = p1 & Z =
      t12) | (X = e1 & Y = p1 & Z = t1) | (X = e1 & Y = p1 & Z = t2) | (X = e1 & Y = p1 & Z = t12) | (X = e2 & Y
      = p1 & Z = t1) | (X = e2 & Y = p1 & Z = t2) | (X = e2 & Y = p1 & Z = t12) | (X = e3 & Y = p1 & Z = t1) | (X
      = e3 & Y = p1 & Z = t2) | (X = e3 & Y = p1 & Z = t12) | (X = e12 & Y = p1 & Z = t1) | (X = e12 & Y = p1 & Z
      = t2) | (X = e12 & Y = p1 & Z = t12))).

all X all Y all Z (EXD(X,Y,Z) <-> ((X = m1 & Y = p1 & Z = t1) | (X = m1 & Y = p1 & Z = t2) | (X = m1 & Y = p1 & Z
      = t12) | (X = p1 & Y = m1 & Z = t1) | (X = p1 & Y = m1 & Z = t2) | (X = p1 & Y = m1 & Z = t12) | (X = e1 & Y
      = p1 & Z = t1) | (X = e1 & Y = p1 & Z = t2) | (X = e1 & Y = p1 & Z = t12) | (X = p1 & Y = e1 & Z = t1) | (X
      = p1 & Y = e1 & Z = t2) | (X = p1 & Y = e1 & Z = t12) | (X = e2 & Y = p1 & Z = t1) | (X = e2 & Y = p1 & Z =
      t2) | (X = e2 & Y = p1 & Z = t12) | (X = p1 & Y = e2 & Z = t1) | (X = p1 & Y = e2 & Z = t2) | (X = p1 & Y =
      e2 & Z = t12) | (X = e3 & Y = p1 & Z = t1) | (X = e3 & Y = p1 & Z = t2) | (X = e3 & Y = p1 & Z = t12) | (X
      = p1 & Y = e3 & Z = t1) | (X = p1 & Y = e3 & Z = t2) | (X = p1 & Y = e3 & Z = t12) | (X = e12 & Y = p1 & Z =
      t1) | (X = e12 & Y = p1 & Z = t2) | (X = e12 & Y = p1 & Z = t12) | (X = p1 & Y = e12 & Z = t1) | (X = p1 &
      Y = e12 & Z = t2) | (X = p1 & Y = e12 & Z = t12) | (X = t11 & Y = p1 & Z = t1) |  (X = t11 & Y = p1 & Z = t2
      ) |  (X = t11 & Y = p1 & Z = t12) | (X = slm1 & Y = m1 & Z = t1) | (X = slm1 & Y = m1 & Z = t2) |  (X = slm1
      & Y = m1 & Z = t12) | (X = s11 & Y = e1 & Z = t1) | (X = s11 & Y = e1 & Z = t2) | (X = s11 & Y = e1 & Z =
      t12) | (X = s12 & Y = e2 & Z = t1) | (X = s12 & Y = e2 & Z = t2) | (X = s12 & Y = e2 & Z = t12) | (X = s13 &
      Y = e3 & Z = t1) | (X = s13 & Y = e3 & Z = t2) | (X = s13 & Y = e3 & Z = t12) | (X = s112 & Y = e12 & Z =
      t1) | (X = s112 & Y = e12 & Z = t2) | (X = s112 & Y = e12 & Z = t12) | (X = e1 & Y = e3 & Z = t1) | (X = e1
      & Y = e3 & Z = t2) | (X = e1 & Y = e3 & Z = t12) | (X = e2 & Y = e3 & Z = t1) | (X = e2 & Y = e3 & Z = t2) |
      (X = e2 & Y = e3 & Z = t12) | (X = e12 & Y = e3 & Z = t1) | (X = e12 & Y = e3 & Z = t2) | (X = e12 & Y = e3
      & Z = t12) | (X = e3 & Y = m1 & Z = t1) | (X = e3 & Y = m1 & Z = t2) | (X = e3 & Y = m1 & Z = t12))).

all X all Y all Z (SLC(X,Y,Z) <-> ((X = m1 & Y = s24 & Z = t1) | (X = m1 & Y = s24 & Z = t2) | (X = m1 & Y = s24 &
      Z = t12) | (X = e1 & Y = s12 & Z = t1) | (X = e1 & Y = s12 & Z = t2) | (X = e1 & Y = s12 & Z = t12) | (X =
      e2 & Y = s34 & Z = t1) | (X = e2 & Y = s34 & Z = t2) | (X = e2 & Y = s34 & Z = t12) | (X = e3 & Y = s24 & Z
      = t1) | (X = e3 & Y = s24 & Z = t2) | (X = e3 & Y = s24 & Z = t12) | (X = e12 & Y = s1234 & Z = t1) | (X =
      e12 & Y = s1234 & Z = t2) | (X = e12 & Y = s1234 & Z = t12) | (X = slm1 & Y = s24 & Z = t1) | (X = slm1 & Y
      = s24 & Z = t2) | (X = slm1 & Y = s24 & Z = t12) | (X = s11 & Y = s12 & Z = t1) | (X = s11 & Y = s12 & Z =
      t2) | (X = s11 & Y = s12 & Z = t12) | (X = s12 & Y = s34 & Z = t1) | (X = s12 & Y = s34 & Z = t2) | (X = s12
      & Y = s34 & Z = t12) | (X = s13 & Y = s24 & Z = t1) | (X = s13 & Y = s24 & Z = t2) | (X = s13 & Y = s24 & Z
      = t12) | (X = s112 & Y = s1234 & Z = t1) | (X = s112 & Y = s1234 & Z = t2) | (X = s112 & Y = s1234 & Z =
      t12) | (X = p1 & Y = s1234 & Z = t1) | (X = p1 & Y = s1234 & Z = t2) | (X = p1 & Y = s1234 & Z = t12))).
```

Counter model to (t_{db}50)

```
(M(e1) & PD(p1) & TL(tl1) & SL(sl1) & T(t1) & T(t2) & T(t12) & S(s1) & S(s2) & S(s12)).

all X ((X = e1) | (X = p1) | (X = tl1) | (X = sl1) | (X = t1) | (X = t2) | (X = t12) | (X = s1) | (X = s2) | (X =
      s12)).

(t1 != t2) & (t1 != t12) & (t2 != t12) & (s1 != s2) & (s1 != s12) & (s2 != s12).

all X (- (EV(X))).
all X (- (STV(X))).

all X all Y (P(X,Y) <-> ((X = p1 & Y = p1) | (X = t1 & Y = t1) | (X = t2 & Y = t2) | (X = t12 & Y = t12) | (X = t1
      & Y = t12) | (X = t2 & Y = t12) | (X = s1 & Y = s1) | (X = s2 & Y = s2) | (X = s12 & Y = s12) | (X = s1 & Y
      = s12) | (X = s2 & Y = s12))).
all X all Y (DQT(X,Y) <-> ((X = tl1 & Y = p1) | (X = sl1 & Y = e1))).
all X all Y (QL(X,Y) <-> ((X = t12 & Y = tl1))).
all X all Y (TLC(X,Y) <-> ((X = e1 & Y = t12) | (X = p1 & Y = t12) | (X = tl1 & Y = t12) | (X = sl1 & Y = t12))).
all X all Y all Z (TQL(X,Y,Z) <-> ( (X = s1 & Y = sl1 & Z = t1) | (X = s2 & Y = sl1 & Z = t2) | (X = s12 & Y = sl1
      & Z = t12))).
all X all Y all Z (tP(X,Y,Z) <-> ( (X = e1 & Y = e1 & Z = t1) | (X = e1 & Y = e1 & Z = t2) | (X = e1 & Y = e1 & Z
      = t12))).
all X all Y all Z (PC(X,Y,Z) <-> ( (X = e1 & Y = p1 & Z = t1) | (X = e1 & Y = p1 & Z = t2) | (X = e1 & Y = p1 & Z
      = t12))).
all X all Y all Z (- (K(X,Y,Z))).
all X all Y all Z (EXD(X,Y,Z) <-> ( (X = e1 & Y = p1 & Z = t1) | (X = e1 & Y = p1 & Z = t2) | (X = e1 & Y = p1 & Z
      = t12) | (X = p1 & Y = e1 & Z = t1) | (X = p1 & Y = e1 & Z = t2) | (X = p1 & Y = e1 & Z = t12) | (X = tl1 &
      Y = p1 & Z = t1) | (X = tl1 & Y = p1 & Z = t2) | (X = tl1 & Y = p1 & Z = t12) | (X = sl1 & Y = e1 & Z = t1
      ) | (X = sl1 & Y = e1 & Z = t2) | (X = sl1 & Y = e1 & Z = t12))).
all X all Y all Z (SLC(X,Y,Z) <-> ( (X = e1 & Y = s1 & Z = t1) | (X = e1 & Y = s2 & Z = t2) | (X = e1 & Y = s12 &
      Z = t12) | (X = p1 & Y = s1 & Z = t1) | (X = p1 & Y = s2 & Z = t2) | (X = p1 & Y = s12 & Z = t12) | (X = sl1
      & Y = s1 & Z = t1) | (X = sl1 & Y = s2 & Z = t2) | (X = sl1 & Y = s12 & Z = t12))).
```

Counter model to (t_{db}51)

```
(M(e1) & PED(e2) & PD(p1) & TL(tl1) & SL(sl1) & SL(sl2) & T(t1) & T(t2) & T(t12) & S(s1)).

all X ((X = e1) | (X = e2) | (X = p1) | (X = tl1) | (X = sl1) | (X = sl2) | (X = t1) | (X = t2) | (X = t12) | (X =
      s1)).

(e1 != e2) & (sl1 != sl2) & (t1 != t2) & (t1 != t12) & (t2 != t12).

- (M(e2)).
all X (- (EV(X))).
all X (- (STV(X))).
all X (- (F(X))).
all X (- (POB(X))).

all X all Y (P(X,Y) <-> ((X = p1 & Y = p1) | (X = t1 & Y = t1) | (X = t2 & Y = t2) | (X = t12 & Y = t12) | (X = t1
      & Y = t12) | (X = t2 & Y = t12) | (X = s1 & Y = s1))).
all X all Y (DQT(X,Y) <-> ((X = tl1 & Y = p1) | (X = sl1 & Y = e1) | (X = sl2 & Y = e2))).
all X all Y (QL(X,Y) <-> ((X = t12 & Y = tl1))).
all X all Y (TLC(X,Y) <-> ((X = e1 & Y = t12) | (X = e2 & Y = t12) | (X = p1 & Y = t12) | (X = tl1 & Y = t12) | (X
      = sl1 & Y = t12) | (X = sl2 & Y = t12))).
all X all Y all Z (TQL(X,Y,Z) <-> ((X = s1 & Y = sl1 & Z = t1) | (X = s1 & Y = sl1 & Z = t2) | (X = s1 & Y = sl1 &
      Z = t12) | (X = s1 & Y = sl2 & Z = t1) | (X = s1 & Y = sl2 & Z = t2) | (X = s1 & Y = sl2 & Z = t12))).
all X all Y all Z (tP(X,Y,Z) <-> ((X = e1 & Y = e1 & Z = t1) | (X = e1 & Y = e1 & Z = t2) | (X = e1 & Y = e1 & Z =
      t12) | (X = e2 & Y = e2 & Z = t1) | (X = e2 & Y = e2 & Z = t2) | (X = e2 & Y = e2 & Z = t12))).
all X all Y all Z (PC(X,Y,Z) <-> ((X = e1 & Y = p1 & Z = t1) | (X = e1 & Y = p1 & Z = t2) | (X = e1 & Y = p1 & Z =
      t12) | (X = e2 & Y = p1 & Z = t1) | (X = e2 & Y = p1 & Z = t2) | (X = e2 & Y = p1 & Z = t12))).
all X all Y all Z (- (K(X,Y,Z))).
all X all Y all Z (EXD(X,Y,Z) <-> ((X = e1 & Y = p1 & Z = t1) | (X = e1 & Y = p1 & Z = t2) | (X = e1 & Y = p1 & Z
      = t12) | (X = p1 & Y = e1 & Z = t1) | (X = p1 & Y = e1 & Z = t2) | (X = p1 & Y = e1 & Z = t12) | (X = e2 &
      Y = p1 & Z = t1) | (X = e2 & Y = p1 & Z = t2) | (X = e2 & Y = p1 & Z = t12) | (X = p1 & Y = e2 & Z = t1) | (
      X = p1 & Y = e2 & Z = t2) | (X = p1 & Y = e2 & Z = t12) | (X = tl1 & Y = p1 & Z = t1) | (X = tl1 & Y = p1 &
      Z = t2) | (X = tl1 & Y = p1 & Z = t12) | (X = sl1 & Y = e1 & Z = t1) | (X = sl1 & Y = e1 & Z = t2) | (X =
      sl1 & Y = e1 & Z = t12) | (X = sl2 & Y = e2 & Z = t1) | (X = sl2 & Y = e2 & Z = t2) | (X = sl2 & Y = e2 & Z
      = t12))).
all X all Y all Z (SLC(X,Y,Z) <-> ((X = e1 & Y = s1 & Z = t1) | (X = e1 & Y = s1 & Z = t2) | (X = e1 & Y = s1 & Z
      = t12) | (X = e2 & Y = s1 & Z = t1) | (X = e2 & Y = s1 & Z = t2) | (X = e2 & Y = s1 & Z = t12) | (X = p1 & Y
      = s1 & Z = t1) | (X = p1 & Y = s1 & Z = t2) | (X = p1 & Y = s1 & Z = t12) | (X = sl1 & Y = s1 & Z = t1) | (
      X = sl1 & Y = s1 & Z = t2) | (X = sl1 & Y = s1 & Z = t12) | (X = sl2 & Y = s1 & Z = t1) | (X = sl2 & Y = s1
      & Z = t2) | (X = sl2 & Y = s1 & Z = t12))).
```

Counter model to (t_{db}53), (t_{db}54)

```
(M(e1) & M(e2) & PD(p1) & TL(t11) & SL(s11) & SL(s12) & T(t1) & T(t2) & T(t12) & S(s1) & S(s2) & S(s12)).

all X ((X = e1) | (X = e2) | (X = p1) | (X = t11) | (X = s11) | (X = s12) | (X = t1) | (X = t2) | (X = t12) | (X =
      s1) | (X = s2) | (X = s12)).

(e1 != e2) & (s11 != s12) & (t1 != t2) & (t1 != t12) & (t2 != t12) & (s1 != s2) & (s1 != s12) & (s2 != s12).

all X (- (EV(X))).
all X (- (STV(X))).

all X all Y (P(X,Y) <-> ((X = p1 & Y = p1) | (X = t1 & Y = t1) | (X = t2 & Y = t2) | (X = t12 & Y = t12) | (X = t1
      & Y = t12) | (X = t2 & Y = t12) | (X = s1 & Y = s1) | (X = s2 & Y = s2) | (X = s12 & Y = s12) | (X = s1 & Y
      = s12) | (X = s2 & Y = s12))).
all X all Y (DQT(X,Y) <-> ((X = t11 & Y = p1) | (X = s11 & Y = e1) | (X = s12 & Y = e2))).
all X all Y (QL(X,Y) <-> ((X = t12 & Y = t11))).
all X all Y (TLC(X,Y) <-> ((X = e1 & Y = t12) | (X = e2 & Y = t12) | (X = p1 & Y = t12) | (X = t11 & Y = t12) | (X
      = s11 & Y = t12) | (X = s12 & Y = t12))).
all X all Y all Z (TQL(X,Y,Z) <-> ( (X = s1 & Y = s11 & Z = t1) | (X = s2 & Y = s11 & Z = t2) | (X = s12 & Y = s11
      & Z = t12) | (X = s2 & Y = s12 & Z = t1) | (X = s1 & Y = s12 & Z = t2) | (X = s12 & Y = s12 & Z = t12))).
all X all Y all Z (tP(X,Y,Z) <-> ( (X = e1 & Y = e1 & Z = t1) | (X = e1 & Y = e1 & Z = t2) | (X = e1 & Y = e1 & Z
      = t12) | (X = e2 & Y = e2 & Z = t1)  | (X = e2 & Y = e2 & Z = t2) | (X = e2 & Y = e2 & Z = t12))).
all X all Y all Z (PC(X,Y,Z) <-> ( (X = e1 & Y = p1 & Z = t1) | (X = e1 & Y = p1 & Z = t2) | (X = e1 & Y = p1 & Z
      = t12) | (X = e2 & Y = p1 & Z = t1) | (X = e2 & Y = p1 & Z = t2) | (X = e2 & Y = p1 & Z = t12))).
all X all Y all Z (- (K(X,Y,Z))).
all X all Y all Z (EXD(X,Y,Z) <-> ( (X = e1 & Y = p1 & Z = t1) | (X = e1 & Y = p1 & Z = t2) | (X = e1 & Y = p1 & Z
      = t12) | (X = p1 & Y = e1 & Z = t1) | (X = p1 & Y = e1 & Z = t2) | (X = p1 & Y = e1 & Z = t12) | (X = e2 &
      Y = p1 & Z = t1) | (X = e2 & Y = p1 & Z = t2) | (X = e2 & Y = p1 & Z = t12) | (X = p1 & Y = e2 & Z = t1) | (
      X = p1 & Y = e2 & Z = t2) | (X = p1 & Y = e2 & Z = t12) | (X = t11 & Y = p1 & Z = t1) | (X = t11 & Y = p1 &
      Z = t2) |  (X = t11 & Y = p1 & Z = t12) | (X = s11 & Y = e1 & Z = t1) | (X = s11 & Y = e1 & Z = t2) | (X =
      s11 & Y = e1 & Z = t12) | (X = s12 & Y = e2 & Z = t1) | (X = s12 & Y = e2 & Z = t2) | (X = s12 & Y = e2 & Z
      = t12))).
all X all Y all Z (SLC(X,Y,Z) <-> ( (X = e1 & Y = s1 & Z = t1) | (X = e1 & Y = s2 & Z = t2) | (X = e1 & Y = s12 &
      Z = t12) | (X = e2 & Y = s2 & Z = t1) | (X = e2 & Y = s1 & Z = t2) | (X = e2 & Y = s12 & Z = t12) | (X = p1
      & Y = s12 & Z = t1) | (X = p1 & Y = s12 & Z = t2) | (X = p1 & Y = s12 & Z = t12) | (X = s11 & Y = s1 & Z =
      t1) | (X = s11 & Y = s2 & Z = t2) | (X = s11 & Y = s12 & Z = t12) | (X = s12 & Y = s2 & Z = t1) | (X = s12 &
      Y = s1 & Z = t2) | (X = s12 & Y = s12 & Z = t12))).
```

Counter model to ($t_{дв}$60)

```
(M(e1) & NPED(e2) & PD(p1) & PD(p2) & PD(p12) & TL(tl1) & TL(tl2) & TL(tl12) & SL(sl1) & T(t1) & T(t2) & T(t12) &
      S(s1)).

all X ((X = e1) | (X = e2) | (X = p1) | (X = p2) | (X = p12) | (X = tl1) | (X = tl2) | (X = tl12) | (X = sl1) | (X
      = t1) | (X = t2) | (X = t12) | (X = s1)).

(p1 != p2) & (p1 != p12) & (p2 != p12) & (tl1 != tl2) & (tl1 != tl12) & (tl2 != tl12) & (t1 != t2) & (t1 != t12) &
      (t2 != t12).

all X (- (EV(X))).
all X (- (STV(X))).
all X (- (NPOB(X))).

all X all Y (P(X,Y) <-> ((X = p1 & Y = p1) | (X = p2 & Y = p2) | (X = p12 & Y = p12) | (X = p1 & Y = p12) | (X =
      p2 & Y = p12) | (X = t1 & Y = t1) | (X = t2 & Y = t2) | (X = t12 & Y = t12) | (X = t1 & Y = t12) | (X = t2 &
      Y = t12) | (X = s1 & Y = s1))).
all X all Y (DQT(X,Y) <-> ((X = tl1 & Y = p1) | (X = tl2 & Y = p2) | (X = tl12 & Y = p12) | (X = sl1 & Y = e1))).
all X all Y (QL(X,Y) <-> ((X = t12 & Y = tl1) | (X = t12 & Y = tl2) | (X = t12 & Y = tl12))).
all X all Y (TLC(X,Y) <-> ((X = e1 & Y = t12) | (X = e2 & Y = t12) | (X = p1 & Y = t12) | (X = p2 & Y = t12) | (X
      = p12 & Y = t12) | (X = tl1 & Y = t12) | (X = tl2 & Y = t12) | (X = tl12 & Y = t12) | (X = sl1 & Y = t12))).
all X all Y all Z (TQL(X,Y,Z) <-> ( (X = s1 & Y = sl1 & Z = t1) | (X = s1 & Y = sl1 & Z = t2) | (X = s1 & Y = sl1
      & Z = t12))).
all X all Y all Z (tP(X,Y,Z) <-> ( (X = e1 & Y = e1 & Z = t1) | (X = e1 & Y = e1 & Z = t2) | (X = e1 & Y = e1 & Z
      = t12) | (X = e2 & Y = e2 & Z = t1) | (X = e2 & Y = e2 & Z = t2) | (X = e2 & Y = e2 & Z = t12))).
all X all Y all Z (PC(X,Y,Z) <-> ( (X = e1 & Y = p1 & Z = t1) | (X = e1 & Y = p1 & Z = t2) | (X = e1 & Y = p1 & Z
      = t12) | (X = e2 & Y = p2 & Z = t1) | (X = e2 & Y = p2 & Z = t2) | (X = e2 & Y = p2 & Z = t12) | (X = e1 & Y
      = p12 & Z = t1) | (X = e1 & Y = p12 & Z = t2) | (X = e1 & Y = p12 & Z = t12) | (X = e2 & Y = p12 & Z = t1)
      | (X = e2 & Y = p12 & Z = t2) | (X = e2 & Y = p12 & Z = t12))).
all X all Y all Z (-(K(X,Y,Z))).
all X all Y all Z (EXD(X,Y,Z) <-> ( (X = e1 & Y = p1 & Z = t1) | (X = e1 & Y = p1 & Z = t2) | (X = e1 & Y = p1 & Z
      = t12) | (X = p1 & Y = e1 & Z = t1) | (X = p1 & Y = e1 & Z = t2) | (X = p1 & Y = e1 & Z = t12) | (X = e2 &
      Y = p2 & Z = t1) | (X = e2 & Y = p2 & Z = t2) | (X = e2 & Y = p2 & Z = t12) | (X = p2 & Y = e2 & Z = t1) | (
      X = p2 & Y = e2 & Z = t2) | (X = p2 & Y = e2 & Z = t12) | (X = e1 & Y = p12 & Z = t1) | (X = e1 & Y = p12 &
      Z = t2) | (X = e1 & Y = p12 & Z = t12) | (X = p12 & Y = e1 & Z = t1) | (X = p12 & Y = e1 & Z = t2) | (X =
      p12 & Y = e1 & Z = t12) | (X = e2 & Y = p12 & Z = t1) | (X = e2 & Y = p12 & Z = t2) | (X = e2 & Y = p12 & Z
      = t12) | (X = p12 & Y = e2 & Z = t1) | (X = p12 & Y = e2 & Z = t2) | (X = p12 & Y = e2 & Z = t12) | (X = tl1
      & Y = p1 & Z = t1) | (X = tl1 & Y = p1 & Z = t2) | (X = tl1 & Y = p1 & Z = t12) | (X = tl2 & Y = p2 & Z =
      t1) | (X = tl2 & Y = p2 & Z = t2) | (X = tl2 & Y = p2 & Z = t12) | (X = tl12 & Y = p12 & Z = t1) | (X = tl12
      & Y = p12 & Z = t2) | (X = tl12 & Y = p12 & Z = t12) | (X = e2 & Y = e1 & Z = t1) | (X = e2 & Y = e1 & Z =
      t2) | (X = e2 & Y = e1 & Z = t12) | (X = sl1 & Y = e1 & Z = t1) | (X = sl1 & Y = e1 & Z = t2) | (X = sl1 & Y
      = e1 & Z = t12))).
all X all Y all Z (SLC(X,Y,Z) <-> ( (X = e1 & Y = s1 & Z = t1) | (X = e1 & Y = s1 & Z = t2) | (X = e1 & Y = s1 & Z
      = t12) | (X = p1 & Y = s1 & Z = t1) | (X = p1 & Y = s1 & Z = t2) | (X = p1 & Y = s1 & Z = t12) | (X = p12 &
      Y = s1 & Z = t1) | (X = p12 & Y = s1 & Z = t2) | (X = p12 & Y = s1 & Z = t12) | (X = sl1 & Y = s1 & Z = t1)
      | (X = sl1 & Y = s1 & Z = t2) | (X = sl1 & Y = s1 & Z = t12))).
```

Counter model to (t_{db}70)

```
(M(e1) & M(e2) & PD(p1) & TL(t11) & SL(s11) & SL(s12) & T(t1) & T(t2) & T(t12) & S(s1)).

all X ((X = e1) | (X = e2) | (X = p1) | (X = t11) | (X = s11) | (X = s12) | (X = t1) | (X = t2) | (X = t12) | (X =
       s1)).

(e1 != e2) & (s11 != s12) & (t1 != t2) & (t1 != t12) & (t2 != t12).

all X (- (EV(X))).
all X (- (STV(X))).

all X all Y (P(X,Y) <-> ((X = p1 & Y = p1) | (X = t1 & Y = t1) | (X = t2 & Y = t2) | (X = t12 & Y = t12) | (X = t1
       & Y = t12) | (X = t2 & Y = t12) | (X = s1 & Y = s1))).
all X all Y (DQT(X,Y) <-> ((X = t11 & Y = p1) | (X = s11 & Y = e1) | (X = s12 & Y = e2))).
all X all Y (QL(X,Y) <-> ((X = t12 & Y = t11))).
all X all Y (TLC(X,Y) <-> ((X = e1 & Y = t1) | (X = e2 & Y = t2) | (X = p1 & Y = t12) | (X = t11 & Y = t12) | (X =
       s11 & Y = t1) | (X = s12 & Y = t2))).
all X all Y all Z (TQL(X,Y,Z) <-> ((X = s1 & Y = s11 & Z = t1) | (X = s2 & Y = s12 & Z = t2))).
all X all Y all Z (tP(X,Y,Z) <-> ((X = e1 & Y = e1 & Z = t1) | (X = e2 & Y = e2 & Z = t2))).
all X all Y all Z (PC(X,Y,Z) <-> ((X = e1 & Y = p1 & Z = t1) | (X = e2 & Y = p1 & Z = t2))).
all X all Y all Z (- (K(X,Y,Z))).
all X all Y all Z (EXD(X,Y,Z) <-> ( (X = e1 & Y = p1 & Z = t1) | (X = p1 & Y = e1 & Z = t1) | (X = e2 & Y = p1 & Z
       = t2) | (X = p1 & Y = e2 & Z = t2) | (X = t11 & Y = p1 & Z = t1) | (X = t11 & Y = p1 & Z = t2) | (X = t11
      & Y = p1 & Z = t12) | (X = s11 & Y = e1 & Z = t1) | (X = s12 & Y = e2 & Z = t2))).
all X all Y all Z (SLC(X,Y,Z) <-> ( (X = e1 & Y = s1 & Z = t1) | (X = e2 & Y = s1 & Z = t2) | (X = p1 & Y = s1 & Z
       = t1) | (X = p1 & Y = s1 & Z = t2) | (X = s11 & Y = s1 & Z = t1) | (X = s12 & Y = s1 & Z = t2))).
```

Counter model to (t_{db}73)

```
(M(e1) & NPED(e2) & PD(p1) & TL(t11) & SL(s11) & T(t1) & T(t2) &T(t12) & S(s1)).

all X ((X = e1) | (X = e2) | (X = p1) | (X = t11) | (X = s11) | (X = t1) | (X = t2) | (X = t12) | (X = s1)).

(t1 != t2) & (t1 != t12) & (t2 != t12).

all X (- (EV(X))).
all X (- (STV(X))).
all X (- (NPOB(X))).

all X all Y (P(X,Y) <-> ((X = p1 & Y = p1) | (X = t1 & Y = t1) | (X = t2 & Y = t2) | (X = t12 & Y = t12) | (X = s1
       & Y = s1) | (X = t1 & Y = t12) | (X = t2 & Y = t12))).
all X all Y (DQT(X,Y) <-> ((X = t11 & Y = p1) | (X = s11 & Y = e1))).
all X all Y (QL(X,Y) <-> ((X = t12 & Y = t11))).
all X all Y (TLC(X,Y) <-> ((X = e1 & Y = t12) | (X = e2 & Y = t12) | (X = p1 & Y = t12) | (X = t11 & Y = t12) | (X
       = s11 & Y = t12))).
all X all Y all Z (TQL(X,Y,Z) <-> ( (X = s1 & Y = s11 & Z = t1) | (X = s1 & Y = s11 & Z = t2) | (X = s1 & Y = s11
      & Z = t12))).
all X all Y all Z (tP(X,Y,Z) <-> ( (X = e1 & Y = e1 & Z = t1) | (X = e1 & Y = e1 & Z = t2) | (X = e1 & Y = e1 & Z
       = t12) | (X = e2 & Y = e2 & Z = t1) | (X = e2 & Y = e2 & Z = t2) | (X = e2 & Y = e2 & Z = t12))).
all X all Y all Z (PC(X,Y,Z) <-> ( (X = e1 & Y = p1 & Z = t1) | (X = e1 & Y = p1 & Z = t2) | (X = e1 & Y = p1 & Z
       = t12) | (X = e2 & Y = p1 & Z = t1) | (X = e2 & Y = p1 & Z = t2) | (X = e2 & Y = p1 & Z = t12))).
all X all Y all Z (-(K(X,Y,Z))).
all X all Y all Z (EXD(X,Y,Z) <-> ( (X = e1 & Y = p1 & Z = t1) | (X = e1 & Y = p1 & Z = t2) | (X = e1 & Y = p1 & Z
       = t12) | (X = p1 & Y = e1 & Z = t1) | (X = p1 & Y = e1 & Z = t2) | (X = p1 & Y = e1 & Z = t12) | (X = e2 &
      Y = p1 & Z = t1) | (X = e2 & Y = p1 & Z = t2) | (X = e2 & Y = p1 & Z = t12) | (X = p1 & Y = e2 & Z = t1) | (
      X = p1 & Y = e2 & Z = t2) | (X = p1 & Y = e2 & Z = t12) | (X = t11 & Y = p1 & Z = t1) | (X = t11 & Y = p1 &
      Z = t2) | (X = t11 & Y = p1 & Z = t12) | (X = s11 & Y = e1 & Z = t1) | (X = s11 & Y = e1 & Z = t2) | (X =
       s11 & Y = e1 & Z = t12) | (X = e2 & Y = e1 & Z = t1) | (X = e2 & Y = e1 & Z = t2) | (X = e2 & Y = e1 & Z =
       t12) | (X = e2 & Y = s11 & Z = t1))).
all X all Y all Z (SLC(X,Y,Z) <-> ( (X = e1 & Y = s1 & Z = t1) | (X = e1 & Y = s1 & Z = t2) | (X = e1 & Y = s1 & Z
       = t12) | (X = p1 & Y = s1 & Z = t1) | (X = p1 & Y = s1 & Z = t2) | (X = p1 & Y = s1 & Z = t12) | (X = s11 &
       Y = s1 & Z = t1) | (X = s11 & Y = s1 & Z = t2) | (X = s11 & Y = s1 & Z = t12))).
```

Counter model to ($t_{db}98$)

```
(Mdcat(e1) & Fdcat(e2) & NAPOdcat(e3) & PDdcat(p1) & TLdcat(t11) & SLdcat(s11) & SLdcat(s12) & SLdcat(s13) & Tdcat
      (t1) & Sdcat(s1)).

all X ((X = e1) | (X = e2) | (X = e3) | (X = p1) | (X = t11) | (X = s11) | (X = s12) | (X = s13) | (X = t1) | (X =
      s1)).

(e1 != e2) & (s11 != s12) & (s11 != s13) & (s12 != s13).

all X (- (EVdcat(X))).
all X (- (STVdcat(X))).

all X all Y (Pd(X,Y) <-> ((X = p1 & Y = p1) | (X = t1 & Y = t1) | (X = s1 & Y = s1) )).
all X all Y (DQTd(X,Y) <-> ((X = t11 & Y = p1) | (X = s11 & Y = e1) | (X = s12 & Y = e2) | (X = s13 & Y = e3))).
all X all Y (QLd(X,Y) <-> ((X = t1 & Y = t11))).
all X all Y (TLCd(X,Y) <-> ((X = e1 & Y = t1) | (X = e2 & Y = t1) | (X = e3 & Y = t1) | (X = p1 & Y = t1) | (X =
      t11 & Y = t1) | (X = s11 & Y = t1) | (X = s12 & Y = t1) | (X = s13 & Y = t1))).
all X all Y all Z (TQLd(X,Y,Z) <-> ((X = s1 & Y = s11 & Z = t1) | (X = s1 & Y = s12 & Z = t1) | (X = s1 & Y = s13
      & Z = t1))).
all X all Y all Z (TPd(X,Y,Z) <-> ((X = e1 & Y = e1 & Z = t1) | (X = e2 & Y = e2 & Z = t1) | (X = e3 & Y = e3 & Z
      = t1))).
all X all Y all Z (PCd(X,Y,Z) <-> ((X = e1 & Y = p1 & Z = t1) | (X = e2 & Y = p1 & Z = t1) | (X = e3 & Y = p1 & Z
      = t1))).
all X all Y all Z ((Kd(X,Y,Z)) <-> (X=e1 & Y=e3 & Z= t1)).
all X all Y all Z (EXDd(X,Y,Z) <-> ((X = e1 & Y = p1 & Z = t1) | (X = p1 & Y = e1 & Z = t1) | (X = e2 & Y = p1 & Z
      = t1) | (X = p1 & Y = e2 & Z = t1) | (X = e3 & Y = p1 & Z = t1) | (X = p1 & Y = e3 & Z = t1) | (X = t11 & Y
      = p1 & Z = t1) | (X = s11 & Y = e1 & Z = t1) | (X = s12 & Y = e2 & Z = t1) | (X = s13 & Y = e3 & Z = t1) |
      (X = e2 & Y = e3 & Z = t1) | (X = e3 & Y = e1 & Z = t1) | (X = e2 & Y = e1 & Z = t1))).
all X all Y all Z (SLCd(X,Y,Z) <-> ((X = e1 & Y = s1 & Z = t1) | (X = e2 & Y = s1 & Z = t1) | (X = e3 & Y = s1 & Z
      = t1) | (X = p1 & Y = s1 & Z = t1) | (X = s11 & Y = s1 & Z = t1) | (X = s12 & Y = s1 & Z = t1) | (X = s13 &
      Y = s1 & Z = t1))).
```

11.3 Validation of DOLCE-BFO mapping (theorems)

Proof of Theorem (t_{db}1)

```
5. ! [X0,X1] : (P(X0,X1) => ((PD(X1) & PD(X0)) | (AB(X1) & AB(X0)))) [input]
8. ! [X0] : ((PD(X0) | AB(X0)) => P(X0,X0)) [input]
13. ! [X0,X1,X2] : ((DQT(X0,X2) & DQT(X0,X1)) => X1 = X2) [input]
14. ! [X0] : (Q(X0) => ? [X1] : DQT(X0,X1)) [input]
15. ! [X0] : (PD(X0) <=> ? [X1] : (TL(X1) & DQT(X1,X0))) [input]
26. ! [X0,X4] : (PRE(X0,X4) <=> ? [X5] : (P(X4,X5) & TLC(X0,X5))) [input]
27. ! [X0,X4] : (TLC(X0,X4) => (T(X4) & (Q(X0) | PD(X0) | ED(X0)))) [input]
28. ! [X0] : ((Q(X0) | PD(X0) | ED(X0)) => ? [X4] : TLC(X0,X4)) [input]
82. ! [X0,X1] : (SD(X0,X1) <=> (! [X4] : (PRE(X0,X4) => exD(X0,X1,X4)) & ? [X4] : PRE(X0,X4))) [input]
83. ! [X0,X1,X4] : (exD(X0,X1,X4) => (PRE(X1,X4) & PRE(X0,X4))) [input]
89. ! [X0,X1] : (DQT(X0,X1) => SD(X0,X1)) [input]
118. ! [X0] : ((tQ(X0) | PQ(X0) | AQ(X0)) <=> Q(X0)) [input]
124. ! [X0] : ((TR(X0) | PR(X0) | AR(X0)) <=> R(X0)) [input]
125. ! [X0] : (R(X0) => AB(X0)) [input]
130. ! [X0] : (T(X0) => TR(X0)) [input]
131. ! [X0] : (TL(X0) => tQ(X0)) [input]
133. ~? [X0] : (PD(X0) & AB(X0)) [input]
162. ! [X0,X4] : (EX(X0,X4) <=> ((T(X4) & ~T(X0) & AB(X0)) | (P(X4,X0) & T(X0)) | PRE(X0,X4))) [input]
163. ! [X0,X1] : (oP(X0,X1) <=> (((T(X1) & T(X0)) | (PD(X1) & PD(X0))) & P(X0,X1))) [input]
167. ! [X0,X4] : (::(X0,proc,X4) <=> (~::(X0,pbnd,X4) & PRE(X0,X4) & PD(X0))) [input]
168. ! [X0,X4] : (::(X0,treg,X4) <=> (EX(X0,X4) & T(X0))) [input]
169. ! [X0,X4] : (::(X0,occ,X4) <=> (::(X0,treg,X4) | ::(X0,proc,X4) | ::(X0,pbnd,X4))) [input]
189. ! [X0,X1] : (oP(X0,X1) => (? [X4] : ::(X1,occ,X4) & ? [X4] : ::(X0,occ,X4))) [input]
190. ~! [X0,X1] : (oP(X0,X1) => (? [X4] : ::(X1,occ,X4) & ? [X4] : ::(X0,occ,X4))) [negated conjecture 189]
192. ! [X0,X1] : (PRE(X0,X1) <=> ? [X2] : (P(X1,X2) & TLC(X0,X2))) [rectify 26]
193. ! [X0,X1] : (TLC(X0,X1) => (T(X1) & (Q(X0) | PD(X0) | ED(X0)))) [rectify 27]
194. ! [X0] : ((Q(X0) | PD(X0) | ED(X0)) => ? [X1] : TLC(X0,X1)) [rectify 28]
260. ! [X0,X1] : (SD(X0,X1) <=> (! [X2] : (PRE(X0,X2) => exD(X0,X1,X2)) & ? [X3] : PRE(X0,X3))) [rectify 82]
261. ! [X0,X1,X2] : (exD(X0,X1,X2) => (PRE(X1,X2) & PRE(X0,X2))) [rectify 83]
304. ! [X0,X1] : (EX(X0,X1) <=> ((T(X1) & ~T(X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1))) [rectify 162]
308. ! [X0,X1] : (::(X0,proc,X1) <=> (~::(X0,pbnd,X1) & PRE(X0,X1) & PD(X0))) [rectify 167]
309. ! [X0,X1] : (::(X0,treg,X1) <=> (EX(X0,X1) & T(X0))) [rectify 168]
310. ! [X0,X1] : (::(X0,occ,X1) <=> (::(X0,treg,X1) | ::(X0,proc,X1) | ::(X0,pbnd,X1))) [rectify 169]
335. ~! [X0,X1] : (oP(X0,X1) => (? [X2] : ::(X1,occ,X2) & ? [X3] : ::(X0,occ,X3))) [rectify 190]
336. ! [X0] : ((TR(X0) | PR(X0) | AR(X0)) => R(X0)) [unused predicate definition removal 124]
339. ! [X0,X1] : (SD(X0,X1) => (! [X2] : (PRE(X0,X2) => exD(X0,X1,X2)) & ? [X3] : PRE(X0,X3))) [unused predicate
     definition removal 260]
370. ! [X0,X1] : (((PD(X1) & PD(X0)) | (AB(X1) & AB(X0))) | ~P(X0,X1)) [ennf transformation 5]
371. ! [X0,X1] : ((PD(X1) & PD(X0)) | (AB(X1) & AB(X0)) | ~P(X0,X1)) [flattening 370]
374. ! [X0] : (P(X0,X0) | (~PD(X0) & ~AB(X0))) [ennf transformation 8]
383. ! [X0,X1,X2] : (X1 = X2 | (~DQT(X0,X2) | ~DQT(X0,X1))) [ennf transformation 13]
384. ! [X0,X1,X2] : (X1 = X2 | ~DQT(X0,X2) | ~DQT(X0,X1)) [flattening 383]
385. ! [X0] : (? [X1] : DQT(X0,X1) | ~Q(X0)) [ennf transformation 14]
400. ! [X0,X1] : ((T(X1) & (Q(X0) | PD(X0) | ED(X0))) | ~TLC(X0,X1)) [ennf transformation 193]
401. ! [X0] : (? [X1] : TLC(X0,X1) | (~Q(X0) & ~PD(X0) & ~ED(X0))) [ennf transformation 194]
488. ! [X0,X1] : ((! [X2] : (exD(X0,X1,X2) | ~PRE(X0,X2)) & ? [X3] : PRE(X0,X3)) | ~SD(X0,X1)) [ennf
     transformation 339]
489. ! [X0,X1,X2] : ((PRE(X1,X2) & PRE(X0,X2)) | ~exD(X0,X1,X2)) [ennf transformation 261]
498. ! [X0,X1] : (SD(X0,X1) | ~DQT(X0,X1)) [ennf transformation 89]
533. ! [X0] : (R(X0) | (~TR(X0) & ~PR(X0) & ~AR(X0))) [ennf transformation 336]
534. ! [X0] : (AB(X0) | ~R(X0)) [ennf transformation 125]
538. ! [X0] : (TR(X0) | ~T(X0)) [ennf transformation 130]
539. ! [X0] : (tQ(X0) | ~TL(X0)) [ennf transformation 131]
541. ! [X0] : (~PD(X0) | ~AB(X0)) [ennf transformation 133]
564. ? [X0,X1] : ((! [X2] : ~::(X1,occ,X2) | ! [X3] : ~::(X0,occ,X3)) & oP(X0,X1)) [ennf transformation 335]
567. ! [X1,X0] : (sP1(X1,X0) <=> ((T(X1) & ~T(X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1))) [predicate
     definition introduction]
568. ! [X0,X1] : (EX(X0,X1) <=> sP1(X1,X0)) [definition folding 304,567]
592. ! [X0] : (? [X1] : DQT(X0,X1) => DQT(X0,sK8(X0))) [choice axiom]
593. ! [X0] : (DQT(X0,sK8(X0)) | ~Q(X0)) [skolemisation 385,592]
594. ! [X0] : ((PD(X0) | ! [X1] : (~TL(X1) | ~DQT(X1,X0))) & (? [X1] : (TL(X1) & DQT(X1,X0)) | ~PD(X0))) [nnf
     transformation 15]
595. ! [X0] : ((PD(X0) | ! [X1] : (~TL(X1) | ~DQT(X1,X0))) & (? [X2] : (TL(X2) & DQT(X2,X0)) | ~PD(X0))) [rectify
     594]
596. ! [X0] : (? [X2] : (TL(X2) & DQT(X2,X0)) => (TL(sK9(X0)) & DQT(sK9(X0),X0))) [choice axiom]
597. ! [X0] : ((PD(X0) | ! [X1] : (~TL(X1) | ~DQT(X1,X0))) & ((TL(sK9(X0)) & DQT(sK9(X0),X0)) | ~PD(X0))) [
     skolemisation 595,596]
606. ! [X0,X1] : ((PRE(X0,X1) | ! [X2] : (~P(X1,X2) | ~TLC(X0,X2))) & (? [X2] : (P(X1,X2) & TLC(X0,X2)) | ~PRE(X0,
     X1))) [nnf transformation 192]
607. ! [X0,X1] : ((PRE(X0,X1) | ! [X2] : (~P(X1,X2) | ~TLC(X0,X2))) & (? [X3] : (P(X1,X3) & TLC(X0,X3)) | ~PRE(X0,
     X1))) [rectify 606]
608. ! [X1,X0] : (? [X3] : (P(X1,X3) & TLC(X0,X3)) => (P(X1,sK13(X0,X1)) & TLC(X0,sK13(X0,X1)))) [choice axiom]
609. ! [X0,X1] : ((PRE(X0,X1) | ! [X2] : (~P(X1,X2) | ~TLC(X0,X2))) & ((P(X1,sK13(X0,X1)) & TLC(X0,sK13(X0,X1))) |
     ~PRE(X0,X1))) [skolemisation 607,608]
610. ! [X0] : (? [X1] : TLC(X0,X1) => TLC(X0,sK14(X0))) [choice axiom]
611. ! [X0] : (TLC(X0,sK14(X0)) | (~Q(X0) & ~PD(X0) & ~ED(X0))) [skolemisation 401,610]
660. ! [X0] : (? [X3] : PRE(X0,X3) => PRE(X0,sK49(X0))) [choice axiom]
661. ! [X0,X1] : ((! [X2] : (exD(X0,X1,X2) | ~PRE(X0,X2)) & PRE(X0,sK49(X0))) | ~SD(X0,X1)) [skolemisation
     488,660]
693. ! [X0] : (((tQ(X0) | PQ(X0) | AQ(X0)) | ~Q(X0)) & (Q(X0) | (~tQ(X0) & ~PQ(X0) & ~AQ(X0)))) [nnf
     transformation 118]
694. ! [X0] : ((tQ(X0) | PQ(X0) | AQ(X0) | ~Q(X0)) & (Q(X0) | (~tQ(X0) & ~PQ(X0) & ~AQ(X0)))) [flattening 693]
698. ! [X1,X0] : ((sP1(X1,X0) | ((~T(X1) | T(X0) | ~AB(X0)) & (~P(X1,X0) | ~T(X0)) & ~PRE(X0,X1))) & (((T(X1) & ~T
     (X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1)) | ~sP1(X1,X0))) [nnf transformation 567]
699. ! [X1,X0] : ((sP1(X1,X0) | ((~T(X1) | T(X0) | ~AB(X0)) & (~P(X1,X0) | ~T(X0)) & ~PRE(X0,X1))) & ((T(X1) & ~T(
     X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1) | ~sP1(X1,X0))) [flattening 698]
```

```
700. ! [X0,X1] : ((sP1(X0,X1) | ((~T(X0) | T(X1) | ~AB(X1)) & (~P(X0,X1) | ~T(X1)) & ~PRE(X1,X0))) & ((T(X0) & ~T(
       X1) & AB(X1)) | (P(X0,X1) & T(X1)) | PRE(X1,X0) | ~sP1(X0,X1))) [rectify 699]
701. ! [X0,X1] : ((EX(X0,X1) | ~sP1(X1,X0)) & (sP1(X1,X0) | ~EX(X0,X1))) [nnf transformation 568]
702. ! [X0,X1] : ((oP(X0,X1) | (((~T(X1) | ~T(X0)) & (~PD(X1) | ~PD(X0))) | ~P(X0,X1))) & ((((T(X1) & T(X0)) | (PD
       (X1) & PD(X0))) & P(X0,X1)) | ~oP(X0,X1))) [nnf transformation 163]
703. ! [X0,X1] : ((oP(X0,X1) | ((~T(X1) | ~T(X0)) & (~PD(X1) | ~PD(X0))) | ~P(X0,X1)) & ((((T(X1) & T(X0)) | (PD(
       X1) & PD(X0))) & P(X0,X1)) | ~oP(X0,X1))) [flattening 702]
715. ! [X0,X1] : ((::(X0,proc,X1) | (::(X0,pbnd,X1) | ~PRE(X0,X1) | ~PD(X0))) & ((~::(X0,pbnd,X1) & PRE(X0,X1) &
       PD(X0)) | ~::(X0,proc,X1))) [nnf transformation 308]
716. ! [X0,X1] : ((::(X0,proc,X1) | ::(X0,pbnd,X1) | ~PRE(X0,X1) | ~PD(X0)) & ((~::(X0,pbnd,X1) & PRE(X0,X1) & PD(
       X0)) | ~::(X0,proc,X1))) [flattening 715]
717. ! [X0,X1] : ((::(X0,treg,X1) | (~EX(X0,X1) | ~T(X0))) & ((EX(X0,X1) & T(X0)) | ~::(X0,treg,X1))) [nnf
       transformation 309]
718. ! [X0,X1] : ((::(X0,treg,X1) | ~EX(X0,X1) | ~T(X0)) & ((EX(X0,X1) & T(X0)) | ~::(X0,treg,X1))) [flattening
       717]
719. ! [X0,X1] : ((::(X0,occ,X1) | (~::(X0,treg,X1) & ~::(X0,proc,X1) & ~::(X0,pbnd,X1))) & ((::(X0,treg,X1) | ::(
       X0,proc,X1) | ::(X0,pbnd,X1)) | ~::(X0,occ,X1))) [nnf transformation 310]
720. ! [X0,X1] : ((::(X0,occ,X1) | (~::(X0,treg,X1) & ~::(X0,proc,X1) & ~::(X0,pbnd,X1))) & (::(X0,treg,X1) | ::(
       X0,proc,X1) | ::(X0,pbnd,X1) | ~::(X0,occ,X1))) [flattening 719]
766. ? [X0,X1] : ((! [X2] : ~::(X1,occ,X2) | ! [X3] : ~::(X0,occ,X3)) & oP(X0,X1)) => ((! [X2] : ~::(sK85,occ,X2)
       | ! [X3] : ~::(sK84,occ,X3)) & oP(sK84,sK85)) [choice axiom]
767. (! [X2] : ~::(sK85,occ,X2) | ! [X3] : ~::(sK84,occ,X3)) & oP(sK84,sK85) [skolemisation 564,766]
781. ~P(X0,X1) | AB(X0) | PD(X0) [cnf transformation 371]
782. ~P(X0,X1) | AB(X1) | PD(X0) [cnf transformation 371]
783. ~P(X0,X1) | AB(X0) | PD(X1) [cnf transformation 371]
789. P(X0,X0) | ~AB(X0) [cnf transformation 374]
805. ~DQT(X0,X2) | X1 = X2 | ~DQT(X0,X1) [cnf transformation 384]
806. DQT(X0,sK8(X0)) | ~Q(X0) [cnf transformation 593]
807. DQT(sK9(X0),X0) | ~PD(X0) [cnf transformation 597]
808. TL(sK9(X0)) | ~PD(X0) [cnf transformation 597]
826. ~TLC(X0,X2) | ~P(X1,X2) | PRE(X0,X1) [cnf transformation 609]
828. ~TLC(X0,X1) | T(X1) [cnf transformation 400]
830. TLC(X0,sK14(X0)) | ~PD(X0) [cnf transformation 611]
933. ~SD(X0,X1) | PRE(X0,sK49(X0)) [cnf transformation 661]
934. exD(X0,X1,X2) | ~PRE(X0,X2) | ~SD(X0,X1) [cnf transformation 661]
936. ~exD(X0,X1,X2) | PRE(X1,X2) [cnf transformation 489]
944. ~DQT(X0,X1) | SD(X0,X1) [cnf transformation 498]
987. ~tQ(X0) | Q(X0) [cnf transformation 694]
992. ~TR(X0) | R(X0) [cnf transformation 533]
993. ~R(X0) | AB(X0) [cnf transformation 534]
999. ~T(X0) | TR(X0) [cnf transformation 538]
1000. ~TL(X0) | tQ(X0) [cnf transformation 539]
1002. ~AB(X0) | ~PD(X0) [cnf transformation 541]
1029. ~P(X0,X1) | sP1(X0,X1) | ~T(X1) [cnf transformation 700]
1032. ~sP1(X1,X0) | EX(X0,X1) [cnf transformation 701]
1033. ~oP(X0,X1) | P(X0,X1) [cnf transformation 703]
1034. ~oP(X0,X1) | PD(X0) | T(X0) [cnf transformation 703]
1036. ~oP(X0,X1) | PD(X0) | T(X1) [cnf transformation 703]
1054. ::(X0,proc,X1) | ::(X0,pbnd,X1) | ~PRE(X0,X1) | ~PD(X0) [cnf transformation 716]
1057. ::(X0,treg,X1) | ~EX(X0,X1) | ~T(X0) [cnf transformation 718]
1059. ~::(X0,pbnd,X1) | ::(X0,occ,X1) [cnf transformation 720]
1060. ~::(X0,proc,X1) | ::(X0,occ,X1) [cnf transformation 720]
1061. ~::(X0,treg,X1) | ::(X0,occ,X1) [cnf transformation 720]
1121. oP(sK84,sK85) [cnf transformation 767]
1122. ~::(sK85,occ,X2) | ~::(sK84,occ,X3) [cnf transformation 767]
1138. 1 <=> ! [X3] : ~::(sK84,occ,X3) [avatar definition]
1139. ~::(sK84,occ,X3) <- (1) [avatar component clause 1138]
1141. 2 <=> ! [X2] : ~::(sK85,occ,X2) [avatar definition]
1142. ~::(sK85,occ,X2) <- (2) [avatar component clause 1141]
1143. 1 | 2 [avatar split clause 1122,1141,1138]
1144. tQ(sK9(X0)) | ~PD(X0) [resolution 808,1000]
1150. Q(sK9(X0)) | ~PD(X0) [resolution 1144,987]
1169. T(sK14(X0)) | ~PD(X0) [resolution 830,828]
1170. TR(sK14(X0)) | ~PD(X0) [resolution 1169,999]
1178. SD(X0,sK8(X0)) | ~Q(X0) [resolution 944,806]
1183. R(sK14(X0)) | ~PD(X0) [resolution 1170,992]
1185. P(sK84,sK85) [resolution 1033,1121]
1190. AB(sK14(X0)) | ~PD(X0) [resolution 1183,993]
1195. AB(sK84) | PD(sK84) [resolution 781,1185]
1197. 3 <=> PD(sK84) [avatar definition]
1199. PD(sK84) <- (3) [avatar component clause 1197]
1201. 4 <=> AB(sK84) [avatar definition]
1203. AB(sK84) <- (4) [avatar component clause 1201]
1204. 3 | 4 [avatar split clause 1195,1201,1197]
1207. AB(sK85) | PD(sK84) [resolution 782,1185]
1211. AB(sK84) | PD(sK85) [resolution 783,1185]
1213. 5 <=> PD(sK85) [avatar definition]
1215. PD(sK85) <- (5) [avatar component clause 1213]
1216. 5 | 4 [avatar split clause 1211,1201,1213]
1217. ~PD(sK84) <- (4) [resolution 1203,1002]
1218. $false <- (3, 4) [subsumption resolution 1217,1199]
1219. ~3 | ~4 [avatar contradiction clause 1218]
1227. 6 <=> T(sK85) [avatar definition]
1231. 7 <=> T(sK84) [avatar definition]
1232. T(sK84) <- (7) [avatar component clause 1231]
1291. PD(sK84) | T(sK84) [resolution 1034,1121]
1295. PD(sK84) | T(sK85) [resolution 1036,1121]
1403. sP1(X1,X1) | ~T(X1) | ~AB(X1) [resolution 1029,789]
1526. ~DQT(X1,X0) | sK8(X1) = X0 | ~Q(X1) [resolution 805,806]
1531. ~P(X2,sK14(X3)) | PRE(X3,X2) | ~PD(X3) [resolution 826,830]
1627. EX(X2,X2) | ~AB(X2) | ~T(X2) [resolution 1403,1032]
```

```
1664. ::(X0,occ,X1) | ~T(X0) | ~EX(X0,X1) [resolution 1057,1061]
1796. ~SD(X0,X2) | ~PRE(X0,X1) | PRE(X2,X1) [resolution 934,936]
1973. 17 <=> AB(sK85) [avatar definition]
1976. 3 | 17 [avatar split clause 1207,1973,1197]
2482. ::(X0,pbnd,X1) | ~PRE(X0,X1) | ~PD(X0) | ::(X0,occ,X1) [resolution 1054,1060]
2486. ::(X0,occ,X1) | ~PD(X0) | ~PRE(X0,X1) [subsumption resolution 2482,1059]
3054. sK8(sK9(X6)) = X6 | ~Q(sK9(X6)) | ~PD(X6) [resolution 1526,807]
3061. ~PD(X6) | sK8(sK9(X6)) = X6 [subsumption resolution 3054,1150]
3071. sK85 = sK8(sK9(sK85)) <- (5) [resolution 3061,1215]
3183. 6 | 3 [avatar split clause 1295,1197,1227]
3190. 7 | 3 [avatar split clause 1291,1197,1231]
3210. PRE(X1,sK14(X1)) | ~PD(X1) | ~AB(sK14(X1)) [resolution 1531,789]
3222. PRE(X1,sK14(X1)) | ~PD(X1) [subsumption resolution 3210,1190]
3285. ~T(sK85) | ~EX(sK85,X0) <- (2) [resolution 1664,1142]
3335. SD(sK9(sK85),sK85) | ~Q(sK9(sK85)) <- (5) [superposition 1178,3071]
3360. 37 <=> Q(sK9(sK85)) [avatar definition]
3362. ~Q(sK9(sK85)) <- (~37) [avatar component clause 3360]
3369. 39 <=> SD(sK9(sK85),sK85) [avatar definition]
3371. SD(sK9(sK85),sK85) <- (39) [avatar component clause 3369]
3372. ~37 | 39 | ~5 [avatar split clause 3335,1213,3369,3360]
3421. ~PD(sK85) <- (~37) [resolution 3362,1150]
3445. ~5 | 37 [avatar split clause 3421,3360,1213]
3455. 49 <=> ! [X0] : ~EX(sK85,X0) [avatar definition]
3456. ~EX(sK85,X0) <- (49) [avatar component clause 3455]
3457. 49 | ~6 | ~2 [avatar split clause 3285,1141,1227,3455]
3484. ~T(sK84) | ~EX(sK84,X0) <- (1) [resolution 1139,1664]
3485. ~EX(sK84,X0) <- (1, 7) [subsumption resolution 3484,1232]
3515. ~PD(sK84) | ~PRE(sK84,X2) <- (1) [resolution 2486,1139]
3564. ~AB(sK84) | ~T(sK84) <- (1, 7) [resolution 3485,1627]
3582. 50 <=> ! [X2] : ~PRE(sK84,X2) [avatar definition]
3583. ~PRE(sK84,X2) <- (50) [avatar component clause 3582]
3584. 50 | ~3 | ~1 [avatar split clause 3515,1138,1197,3582]
3585. ~AB(sK84) <- (1, 7) [subsumption resolution 3564,1232]
3586. ~4 | ~1 | ~7 [avatar split clause 3585,1231,1138,1201]
3589. ~PD(sK85) | ~PRE(sK85,X0) <- (2) [resolution 1142,2486]
3629. ~AB(sK85) | ~T(sK85) <- (49) [resolution 3456,1627]
3651. 51 <=> ! [X0] : ~PRE(sK85,X0) [avatar definition]
3652. ~PRE(sK85,X0) <- (51) [avatar component clause 3651]
3653. 51 | ~5 | ~2 [avatar split clause 3589,1141,1213,3651]
3664. ~6 | ~17 | ~49 [avatar split clause 3629,3455,1973,1227]
3813. ~PRE(sK9(sK85),X0) | PRE(sK85,X0) <- (39) [resolution 3371,1796]
3814. PRE(sK9(sK85),sK49(sK9(sK85))) <- (39) [resolution 3371,933]
3815. ~PRE(sK9(sK85),X0) <- (39, 51) [subsumption resolution 3813,3652]
3816. $false <- (39, 51) [subsumption resolution 3814,3815]
3817. ~39 | ~51 [avatar contradiction clause 3816]
4000. ~PD(sK84) <- (50) [resolution 3222,3583]
4003. $false <- (3, 50) [subsumption resolution 4000,1199]
4004. ~3 | ~50 [avatar contradiction clause 4003]
4005. $false [avatar sat refutation
      1143,1204,1216,1219,1976,3183,3190,3372,3445,3457,3584,3586,3653,3664,3817,4004]
```

Proof of Theorem (t_{db}2)

```
6. ! [X0,X1] : (P(X0,X1) => (T(X0) <=> T(X1))) [input]
8. ! [X0] : ((PD(X0) | AB(X0)) => P(X0,X0)) [input]
124. ! [X0] : ((TR(X0) | PR(X0) | AR(X0)) <=> R(X0)) [input]
125. ! [X0] : (R(X0) => AB(X0)) [input]
130. ! [X0] : (T(X0) => TR(X0)) [input]
163. ! [X0,X1] : (oP(X0,X1) <=> (((T(X1) & T(X0)) | (PD(X1) & PD(X0))) & P(X0,X1))) [input]
166. ! [X0,X4] : (::(X0,pbnd,X4) <=> (AT(X4) & TLC(X0,X4) & ? [X1] : tmPP(X0,X1) & PD(X0))) [input]
167. ! [X0,X4] : (::(X0,proc,X4) <=> (~::(X0,pbnd,X4) & PRE(X0,X4) & PD(X0))) [input]
168. ! [X0,X4] : (::(X0,treg,X4) <=> (EX(X0,X4) & T(X0))) [input]
169. ! [X0,X4] : (::(X0,occ,X4) <=> (::(X0,treg,X4) | ::(X0,proc,X4) | ::(X0,pbnd,X4))) [input]
189. ! [X0] : (? [X4] : ::(X0,occ,X4) => oP(X0,X0)) [input]
190. ~! [X0] : (? [X4] : ::(X0,occ,X4) => oP(X0,X0)) [negated conjecture 189]
307. ! [X0,X1] : (::(X0,pbnd,X1) <=> (AT(X1) & TLC(X0,X1) & ? [X2] : tmPP(X0,X2) & PD(X0))) [rectify 166]
308. ! [X0,X1] : (::(X0,proc,X1) <=> (~::(X0,pbnd,X1) & PRE(X0,X1) & PD(X0))) [rectify 167]
309. ! [X0,X1] : (::(X0,treg,X1) <=> (EX(X0,X1) & T(X0))) [rectify 168]
310. ! [X0,X1] : (::(X0,occ,X1) <=> (::(X0,treg,X1) | ::(X0,proc,X1) | ::(X0,pbnd,X1))) [rectify 169]
335. ~! [X0] : (? [X1] : ::(X0,occ,X1) => oP(X0,X0)) [rectify 190]
336. ! [X0] : ((TR(X0) | PR(X0) | AR(X0)) => R(X0)) [unused predicate definition removal 124]
372. ! [X0,X1] : ((T(X0) <=> T(X1)) | ~P(X0,X1)) [ennf transformation 6]
374. ! [X0] : (P(X0,X0) | (~PD(X0) & ~AB(X0))) [ennf transformation 8]
533. ! [X0] : (R(X0) | (~TR(X0) & ~PR(X0) & ~AR(X0))) [ennf transformation 336]
534. ! [X0] : (AB(X0) | ~R(X0)) [ennf transformation 125]
538. ! [X0] : (TR(X0) | ~T(X0)) [ennf transformation 130]
564. ? [X0] : (~oP(X0,X0) & ? [X1] : ::(X0,occ,X1)) [ennf transformation 335]
588. ! [X0,X1] : (((T(X0) | ~T(X1)) & (T(X1) | ~T(X0))) | ~P(X0,X1)) [nnf transformation 372]
702. ! [X0,X1] : ((oP(X0,X1) | (((~T(X1) | ~T(X0)) & (~PD(X1) | ~PD(X0))) | ~P(X0,X1))) & ((((T(X1) & T(X0)) | (PD
     (X1) & PD(X0))) & P(X0,X1)) | ~oP(X0,X1))) [nnf transformation 163]
703. ! [X0,X1] : ((oP(X0,X1) | ((~T(X1) | ~T(X0)) & (~PD(X1) | ~PD(X0))) | ~P(X0,X1)) & ((((T(X1) & T(X0)) | (PD(
     X1) & PD(X0))) & P(X0,X1)) | ~oP(X0,X1))) [flattening 702]
710. ! [X0,X1] : ((::(X0,pbnd,X1) | (~AT(X1) | ~TLC(X0,X1) | ! [X2] : ~tmPP(X0,X2) | ~PD(X0))) & ((AT(X1) & TLC(X0
     ,X1) & ? [X2] : tmPP(X0,X2) & PD(X0)) | ~::(X0,pbnd,X1))) [nnf transformation 307]
711. ! [X0,X1] : ((::(X0,pbnd,X1) | ~AT(X1) | ~TLC(X0,X1) | ! [X2] : ~tmPP(X0,X2) | ~PD(X0)) & ((AT(X1) & TLC(X0,
     X1) & ? [X2] : tmPP(X0,X2) & PD(X0)) | ~::(X0,pbnd,X1))) [flattening 710]
712. ! [X0,X1] : ((::(X0,pbnd,X1) | ~AT(X1) | ~TLC(X0,X1) | ! [X2] : ~tmPP(X0,X2) | ~PD(X0)) & ((AT(X1) & TLC(X0,
     X1) & ? [X3] : tmPP(X0,X3) & PD(X0)) | ~::(X0,pbnd,X1))) [rectify 711]
713. ! [X0] : (? [X3] : tmPP(X0,X3) => tmPP(X0,sK69(X0))) [choice axiom]
714. ! [X0,X1] : ((::(X0,pbnd,X1) | ~AT(X1) | ~TLC(X0,X1) | ! [X2] : ~tmPP(X0,X2) | ~PD(X0)) & ((AT(X1) & TLC(X0,
     X1) & tmPP(X0,sK69(X0)) & PD(X0)) | ~::(X0,pbnd,X1))) [skolemisation 712,713]
715. ! [X0,X1] : ((::(X0,proc,X1) | (::(X0,pbnd,X1) | ~PRE(X0,X1) | ~PD(X0))) & ((~::(X0,pbnd,X1) & PRE(X0,X1) &
     PD(X0)) | ~::(X0,proc,X1))) [nnf transformation 308]
716. ! [X0,X1] : ((::(X0,proc,X1) | ::(X0,pbnd,X1) | ~PRE(X0,X1) | ~PD(X0)) & ((~::(X0,pbnd,X1) & PRE(X0,X1) & PD(
     X0)) | ~::(X0,proc,X1))) [flattening 715]
717. ! [X0,X1] : ((::(X0,treg,X1) | (~EX(X0,X1) | ~T(X0))) & ((EX(X0,X1) & T(X0)) | ~::(X0,treg,X1))) [nnf
     transformation 309]
718. ! [X0,X1] : ((::(X0,treg,X1) | ~EX(X0,X1) | ~T(X0)) & ((EX(X0,X1) & T(X0)) | ~::(X0,treg,X1))) [flattening
     717]
719. ! [X0,X1] : ((::(X0,occ,X1) | (~::(X0,treg,X1) & ~::(X0,proc,X1) & ~::(X0,pbnd,X1))) & ((::(X0,treg,X1) | ::(
     X0,proc,X1) | ::(X0,pbnd,X1)) | ~::(X0,occ,X1))) [nnf transformation 310]
720. ! [X0,X1] : ((::(X0,occ,X1) | (~::(X0,treg,X1) & ~::(X0,proc,X1) & ~::(X0,pbnd,X1))) & (::(X0,treg,X1) | ::(
     X0,proc,X1) | ::(X0,pbnd,X1) | ~::(X0,occ,X1))) [flattening 719]
766. ? [X0] : (~oP(X0,X0) & ? [X1] : ::(X0,occ,X1)) => (~oP(sK84,sK84) & ? [X1] : ::(sK84,occ,X1)) [choice axiom]
767. ? [X1] : ::(sK84,occ,X1) => ::(sK84,occ,sK85) [choice axiom]
768. ~oP(sK84,sK84) & ::(sK84,occ,sK85) [skolemisation 564,767,766]
787. ~P(X0,X1) | ~T(X1) | T(X0) [cnf transformation 588]
790. P(X0,X0) | ~AB(X0) [cnf transformation 374]
791. P(X0,X0) | ~PD(X0) [cnf transformation 374]
993. ~TR(X0) | R(X0) [cnf transformation 533]
994. ~R(X0) | AB(X0) [cnf transformation 534]
1000. ~T(X0) | TR(X0) [cnf transformation 538]
1039. ~P(X0,X1) | ~PD(X1) | ~PD(X0) | oP(X0,X1) [cnf transformation 703]
1040. oP(X0,X1) | ~T(X1) | ~T(X0) | ~P(X0,X1) [cnf transformation 703]
1047. ~::(X0,pbnd,X1) | PD(X0) [cnf transformation 714]
1052. ~::(X0,proc,X1) | PD(X0) [cnf transformation 716]
1056. ~::(X0,treg,X1) | T(X0) [cnf transformation 718]
1059. ~::(X0,occ,X1) | ::(X0,proc,X1) | ::(X0,pbnd,X1) | ::(X0,treg,X1) [cnf transformation 720]
1122. ::(sK84,occ,sK85) [cnf transformation 768]
1123. ~oP(sK84,sK84) [cnf transformation 768]
1131. ~P(X0,X1) | ~T(X1) | oP(X0,X1) [subsumption resolution 1040,787]
1367. oP(X1,X1) | ~T(X1) | ~AB(X1) [resolution 1131,790]
1512. 1 <=> PD(sK84) [avatar definition]
1514. ~PD(sK84) <- (~1) [avatar component clause 1512]
1516. 2 <=> T(sK84) [avatar definition]
1517. T(sK84) <- (2) [avatar component clause 1516]
1518. ~T(sK84) <- (~2) [avatar component clause 1516]
1530. ~T(sK84) | ~AB(sK84) [resolution 1367,1123]
1537. 3 <=> AB(sK84) [avatar definition]
1539. ~AB(sK84) <- (~3) [avatar component clause 1537]
1540. ~3 | ~2 [avatar split clause 1530,1516,1537]
1792. ~PD(X0) | ~PD(X0) | oP(X0,X0) | ~PD(X0) [resolution 1039,791]
1806. oP(X0,X0) | ~PD(X0) [duplicate literal removal 1792]
1808. ~PD(sK84) [resolution 1806,1123]
2661. ::(sK84,proc,sK85) | ::(sK84,pbnd,sK85) | ::(sK84,treg,sK85) [resolution 1059,1122]
2663. 4 <=> ::(sK84,treg,sK85) [avatar definition]
2665. ::(sK84,treg,sK85) <- (4) [avatar component clause 2663]
2667. 5 <=> ::(sK84,pbnd,sK85) [avatar definition]
2669. ::(sK84,pbnd,sK85) <- (5) [avatar component clause 2667]
2671. 6 <=> ::(sK84,proc,sK85) [avatar definition]
2673. ::(sK84,proc,sK85) <- (6) [avatar component clause 2671]
2674. 4 | 5 | 6 [avatar split clause 2661,2671,2667,2663]
```

```
2679. T(sK84) <- (4) [resolution 2665,1056]
2694. $false <- (~2, 4) [subsumption resolution 2679,1518]
2695. 2 | ~4 [avatar contradiction clause 2694]
2704. PD(sK84) <- (5) [resolution 2669,1047]
2705. $false <- (~1, 5) [subsumption resolution 2704,1514]
2706. 1 | ~5 [avatar contradiction clause 2705]
2716. PD(sK84) <- (6) [resolution 2673,1052]
2719. 1 | ~6 [avatar split clause 2716,2671,1512]
2720. ~1 [avatar split clause 1808,1512]
2758. TR(sK84) <- (2) [resolution 1517,1000]
2791. R(sK84) <- (2) [resolution 2758,993]
2792. AB(sK84) <- (2) [resolution 2791,994]
2793. $false <- (2, ~3) [subsumption resolution 2792,1539]
2794. ~2 | 3 [avatar contradiction clause 2793]
2795. $false [avatar sat refutation 1540,2674,2695,2706,2719,2720,2794]
```

Proof of Theorem ($t_{db}3$)

```
9. ! [X0,X1] : ((P(X1,X0) & P(X0,X1)) => X0 = X1) [input]
163. ! [X0,X1] : (oP(X0,X1) <=> (((T(X1) & T(X0)) | (PD(X1) & PD(X0))) & P(X0,X1))) [input]
189. ! [X0,X1] : ((oP(X1,X0) & oP(X0,X1)) => X0 = X1) [input]
190. ~! [X0,X1] : ((oP(X1,X0) & oP(X0,X1)) => X0 = X1) [negated conjecture 189]
374. ! [X0,X1] : (X0 = X1 | (~P(X1,X0) | ~P(X0,X1))) [ennf transformation 9]
375. ! [X0,X1] : (X0 = X1 | ~P(X1,X0) | ~P(X0,X1)) [flattening 374]
563. ? [X0,X1] : (X0 != X1 & (oP(X1,X0) & oP(X0,X1))) [ennf transformation 190]
564. ? [X0,X1] : (X0 != X1 & oP(X1,X0) & oP(X0,X1)) [flattening 563]
702. ! [X0,X1] : ((oP(X0,X1) | (((~T(X1) | ~T(X0)) & (~PD(X1) | ~PD(X0))) | ~P(X0,X1))) & ((((T(X1) & T(X0)) | (PD
     (X1) & PD(X0))) & P(X0,X1)) | ~oP(X0,X1))) [nnf transformation 163]
703. ! [X0,X1] : ((oP(X0,X1) | ((~T(X1) | ~T(X0)) & (~PD(X1) | ~PD(X0))) | ~P(X0,X1)) & ((((T(X1) & T(X0)) | (PD(
     X1) & PD(X0))) & P(X0,X1)) | ~oP(X0,X1))) [flattening 702]
766. ? [X0,X1] : (X0 != X1 & oP(X1,X0) & oP(X0,X1)) => (sK84 != sK85 & oP(sK85,sK84) & oP(sK84,sK85)) [choice
     axiom]
767. sK84 != sK85 & oP(sK85,sK84) & oP(sK84,sK85) [skolemisation 564,766]
791. ~P(X1,X0) | X0 = X1 | ~P(X0,X1) [cnf transformation 375]
1033. ~oP(X0,X1) | P(X0,X1) [cnf transformation 703]
1121. oP(sK84,sK85) [cnf transformation 767]
1122. oP(sK85,sK84) [cnf transformation 767]
1123. sK84 != sK85 [cnf transformation 767]
1178. P(sK84,sK85) [resolution 1033,1121]
1179. P(sK85,sK84) [resolution 1033,1122]
1475. sK84 = sK85 | ~P(sK85,sK84) [resolution 791,1178]
1477. ~P(sK85,sK84) [subsumption resolution 1475,1123]
1478. $false [subsumption resolution 1477,1179]
```

Proof of Theorem ($t_{db}4$)

```
5. ! [X0,X1] : (P(X0,X1) => ((PD(X1) & PD(X0)) | (AB(X1) & AB(X0)))) [input]
6. ! [X0,X1] : (P(X0,X1) => (T(X0) <=> T(X1))) [input]
10. ! [X0,X1,X2] : ((P(X1,X2) & P(X0,X1)) => P(X0,X2)) [input]
133. ~? [X0] : (PD(X0) & AB(X0)) [input]
163. ! [X0,X1] : (oP(X0,X1) <=> (((T(X1) & T(X0)) | (PD(X1) & PD(X0))) & P(X0,X1))) [input]
189. ! [X0,X1,X2] : ((oP(X1,X2) & oP(X0,X1)) => oP(X0,X2)) [input]
190. ~! [X0,X1,X2] : ((oP(X1,X2) & oP(X0,X1)) => oP(X0,X2)) [negated conjecture 189]
369. ! [X0,X1] : (((PD(X1) & PD(X0)) | (AB(X1) & AB(X0))) | ~P(X0,X1)) [ennf transformation 5]
370. ! [X0,X1] : ((PD(X1) & PD(X0)) | (AB(X1) & AB(X0)) | ~P(X0,X1)) [flattening 369]
371. ! [X0,X1] : ((T(X0) <=> T(X1)) | ~P(X0,X1)) [ennf transformation 6]
376. ! [X0,X1,X2] : (P(X0,X2) | (~P(X1,X2) | ~P(X0,X1))) [ennf transformation 10]
377. ! [X0,X1,X2] : (P(X0,X2) | ~P(X1,X2) | ~P(X0,X1)) [flattening 376]
540. ! [X0] : (~PD(X0) | ~AB(X0)) [ennf transformation 133]
563. ? [X0,X1,X2] : (~oP(X0,X2) & (oP(X1,X2) & oP(X0,X1))) [ennf transformation 190]
564. ? [X0,X1,X2] : (~oP(X0,X2) & oP(X1,X2) & oP(X0,X1)) [flattening 563]
588. ! [X0,X1] : (((T(X0) | ~T(X1)) & (T(X1) | ~T(X0))) | ~P(X0,X1)) [nnf transformation 371]
702. ! [X0,X1] : ((oP(X0,X1) | (((~T(X1) | ~T(X0)) & (~PD(X1) | ~PD(X0))) | ~P(X0,X1))) & ((((T(X1) & T(X0)) | (PD
     (X1) & PD(X0))) & P(X0,X1)) | ~oP(X0,X1))) [nnf transformation 163]
703. ! [X0,X1] : ((oP(X0,X1) | ((~T(X1) | ~T(X0)) & (~PD(X1) | ~PD(X0))) | ~P(X0,X1)) & ((((T(X1) & T(X0)) | (PD(
     X1) & PD(X0))) & P(X0,X1)) | ~oP(X0,X1))) [flattening 702]
766. ? [X0,X1,X2] : (~oP(X0,X2) & oP(X1,X2) & oP(X0,X1)) => (~oP(sK84,sK86) & oP(sK85,sK86) & oP(sK84,sK85)) [
     choice axiom]
767. ~oP(sK84,sK86) & oP(sK85,sK86) & oP(sK84,sK85) [skolemisation 564,766]
783. ~P(X0,X1) | AB(X0) | PD(X1) [cnf transformation 370]
785. ~P(X0,X1) | ~T(X0) | T(X1) [cnf transformation 588]
786. ~P(X0,X1) | ~T(X1) | T(X0) [cnf transformation 588]
792. ~P(X1,X2) | P(X0,X2) | ~P(X0,X1) [cnf transformation 377]
1002. ~AB(X0) | ~PD(X0) [cnf transformation 540]
1033. ~oP(X0,X1) | P(X0,X1) [cnf transformation 703]
1034. ~oP(X0,X1) | PD(X0) | T(X0) [cnf transformation 703]
1036. ~oP(X0,X1) | PD(X0) | T(X1) [cnf transformation 703]
1038. ~P(X0,X1) | ~PD(X1) | ~PD(X0) | oP(X0,X1) [cnf transformation 703]
1039. oP(X0,X1) | ~T(X1) | ~T(X0) | ~P(X0,X1) [cnf transformation 703]
1121. oP(sK84,sK85) [cnf transformation 767]
1122. oP(sK85,sK86) [cnf transformation 767]
1123. ~oP(sK84,sK86) [cnf transformation 767]
1131. ~P(X0,X1) | ~T(X1) | oP(X0,X1) [subsumption resolution 1039,786]
1178. P(sK84,sK85) [resolution 1033,1121]
1179. P(sK85,sK86) [resolution 1033,1122]
1193. 1 <=> PD(sK84) [avatar definition]
1194. ~PD(sK84) <- (~1) [avatar component clause 1193]
1202. 3 <=> PD(sK85) [avatar definition]
1203. ~PD(sK85) <- (~3) [avatar component clause 1202]
```

```
1204. PD(sK85) <- (3) [avatar component clause 1202]
1206. 4 <=> AB(sK85) [avatar definition]
1208. AB(sK85) <- (4) [avatar component clause 1206]
1222. AB(sK85) | PD(sK86) [resolution 783,1179]
1224. 5 <=> PD(sK86) [avatar definition]
1226. PD(sK86) <- (5) [avatar component clause 1224]
1227. 5 | 4 [avatar split clause 1222,1206,1224]
1246. 8 <=> T(sK84) [avatar definition]
1251. 9 <=> T(sK86) [avatar definition]
1255. ~PD(sK85) <- (4) [resolution 1208,1002]
1256. $false <- (3, 4) [subsumption resolution 1255,1204]
1257. ~3 | ~4 [avatar contradiction clause 1256]
1321. PD(sK84) | T(sK84) [resolution 1034,1121]
1326. PD(sK85) | T(sK86) [resolution 1036,1122]
1531. ~P(X14,sK85) | P(X14,sK86) [resolution 792,1179]
1589. P(sK84,sK86) [resolution 1531,1178]
1602. ~T(sK84) | T(sK86) [resolution 1589,785]
1609. ~T(sK86) | oP(sK84,sK86) [resolution 1589,1131]
1999. ~PD(sK86) | ~PD(sK84) | oP(sK84,sK86) [resolution 1038,1589]
2005. ~PD(sK84) | oP(sK84,sK86) <- (5) [subsumption resolution 1999,1226]
2009. ~PD(sK84) <- (5) [subsumption resolution 2005,1123]
2010. ~1 | ~5 [avatar split clause 2009,1224,1193]
2073. T(sK84) <- (~1) [subsumption resolution 1321,1194]
2074. 8 | 1 [avatar split clause 2073,1193,1246]
2075. 9 | ~8 [avatar split clause 1602,1246,1251]
2076. T(sK86) <- (~3) [subsumption resolution 1326,1203]
2077. 9 | 3 [avatar split clause 2076,1202,1251]
2078. ~T(sK86) [subsumption resolution 1609,1123]
2079. ~9 [avatar split clause 2078,1251]
2091. $false [avatar sat refutation 1227,1257,2010,2074,2075,2077,2079]
```

Proof of Theorem (t_{db}6)

```
5. ! [X0,X1] : (P(X0,X1) => ((PD(X1) & PD(X0)) | (AB(X1) & AB(X0)))) [input]
6. ! [X0,X1] : (P(X0,X1) => (T(X0) <=> T(X1))) [input]
10. ! [X0,X1,X2] : ((P(X1,X2) & P(X0,X1)) => P(X0,X2)) [input]
34. ! [X0,X4] : (PRE(X0,X4) => (T(X4) & (Q(X0) | PD(X0) | ED(X0)))) [input]
36. ! [X0,X4,X1] : ((P(X0,X1) & PRE(X0,X4)) => PRE(X1,X4)) [input]
124. ! [X0] : ((TR(X0) | PR(X0) | AR(X0)) <=> R(X0)) [input]
125. ! [X0] : (R(X0) => AB(X0)) [input]
130. ! [X0] : (T(X0) => TR(X0)) [input]
132. ~? [X0] : (ED(X0) & AB(X0)) [input]
133. ~? [X0] : (PD(X0) & AB(X0)) [input]
134. ~? [X0] : (Q(X0) & AB(X0)) [input]
162. ! [X0,X4] : (EX(X0,X4) <=> ((T(X4) & ~T(X0) & AB(X0)) | (P(X4,X0) & T(X0)) | PRE(X0,X4))) [input]
163. ! [X0,X1] : (oP(X0,X1) <=> (((T(X1) & T(X0)) | (PD(X1) & PD(X0))) & P(X0,X1))) [input]
189. ! [X0,X1,X4] : ((EX(X0,X4) & oP(X0,X1)) => EX(X1,X4)) [input]
190. ~! [X0,X1,X4] : ((EX(X0,X4) & oP(X0,X1)) => EX(X1,X4)) [negated conjecture 189]
200. ! [X0,X1] : (PRE(X0,X1) => (T(X1) & (Q(X0) | PD(X0) | ED(X0)))) [rectify 34]
202. ! [X0,X1,X2] : ((P(X0,X2) & PRE(X0,X1)) => PRE(X2,X1)) [rectify 36]
304. ! [X0,X1] : (EX(X0,X1) <=> ((T(X1) & ~T(X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1))) [rectify 162]
335. ~! [X0,X1,X2] : ((EX(X0,X2) & oP(X0,X1)) => EX(X1,X2)) [rectify 190]
336. ! [X0] : ((TR(X0) | PR(X0) | AR(X0)) => R(X0)) [unused predicate definition removal 124]
370. ! [X0,X1] : (((PD(X1) & PD(X0)) | (AB(X1) & AB(X0))) | ~P(X0,X1)) [ennf transformation 5]
371. ! [X0,X1] : ((PD(X1) & PD(X0)) | (AB(X1) & AB(X0)) | ~P(X0,X1)) [flattening 370]
372. ! [X0,X1] : ((T(X0) <=> T(X1)) | ~P(X0,X1)) [ennf transformation 6]
377. ! [X0,X1,X2] : (P(X0,X2) | (~P(X1,X2) | ~P(X0,X1))) [ennf transformation 10]
378. ! [X0,X1,X2] : (P(X0,X2) | ~P(X1,X2) | ~P(X0,X1)) [flattening 377]
411. ! [X0,X1] : ((T(X1) & (Q(X0) | PD(X0) | ED(X0))) | ~PRE(X0,X1)) [ennf transformation 200]
414. ! [X0,X1,X2] : (PRE(X2,X1) | (~P(X0,X2) | ~PRE(X0,X1))) [ennf transformation 202]
415. ! [X0,X1,X2] : (PRE(X2,X1) | ~P(X0,X2) | ~PRE(X0,X1)) [flattening 414]
533. ! [X0] : (R(X0) | (~TR(X0) & ~PR(X0) & ~AR(X0))) [ennf transformation 336]
534. ! [X0] : (AB(X0) | ~R(X0)) [ennf transformation 125]
538. ! [X0] : (TR(X0) | ~T(X0)) [ennf transformation 130]
540. ! [X0] : (~ED(X0) | ~AB(X0)) [ennf transformation 132]
541. ! [X0] : (~PD(X0) | ~AB(X0)) [ennf transformation 133]
542. ! [X0] : (~Q(X0) | ~AB(X0)) [ennf transformation 134]
564. ? [X0,X1,X2] : (~EX(X1,X2) & (EX(X0,X2) & oP(X0,X1))) [ennf transformation 335]
565. ? [X0,X1,X2] : (~EX(X1,X2) & EX(X0,X2) & oP(X0,X1)) [flattening 564]
568. ! [X1,X0] : (sP1(X1,X0) <=> ((T(X1) & ~T(X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1))) [predicate
      definition introduction]
569. ! [X0,X1] : (EX(X0,X1) <=> sP1(X1,X0)) [definition folding 304,568]
589. ! [X0,X1] : (((T(X0) | ~T(X1)) & (T(X1) | ~T(X0))) | ~P(X0,X1)) [nnf transformation 372]
699. ! [X1,X0] : ((sP1(X1,X0) | ((~T(X1) | T(X0) | ~AB(X0)) & (~P(X1,X0) | ~T(X0)) & ~PRE(X0,X1))) & (((T(X1) & ~T
      (X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1)) | ~sP1(X1,X0))) [nnf transformation 568]
700. ! [X1,X0] : ((sP1(X1,X0) | ((~T(X1) | T(X0) | ~AB(X0)) & (~P(X1,X0) | ~T(X0)) & ~PRE(X0,X1))) & ((T(X1) & ~T(
      X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1) | ~sP1(X1,X0))) [flattening 699]
701. ! [X0,X1] : ((sP1(X0,X1) | ((~T(X0) | T(X1) | ~AB(X1)) & (~P(X0,X1) | ~T(X1)) & ~PRE(X1,X0))) & ((T(X0) & ~T(
      X1) & AB(X1)) | (P(X0,X1) & T(X1)) | PRE(X1,X0) | ~sP1(X0,X1))) [rectify 700]
702. ! [X0,X1] : ((EX(X0,X1) | ~sP1(X1,X0)) & (sP1(X1,X0) | ~EX(X0,X1))) [nnf transformation 569]
703. ! [X0,X1] : ((oP(X0,X1) | (((~T(X1) | ~T(X0)) & (~PD(X1) | ~PD(X0))) | ~P(X0,X1))) & ((((T(X1) & T(X0)) | (PD
      (X1) & PD(X0))) & P(X0,X1)) | ~oP(X0,X1))) [nnf transformation 163]
704. ! [X0,X1] : ((oP(X0,X1) | ((~T(X1) | ~T(X0)) & (~PD(X1) | ~PD(X0))) | ~P(X0,X1)) & ((((T(X1) & T(X0)) | (PD(
      X1) & PD(X0))) & P(X0,X1)) | ~oP(X0,X1))) [flattening 703]
767. ? [X0,X1,X2] : (~EX(X1,X2) & EX(X0,X2) & oP(X0,X1)) => (~EX(sK85,sK86) & EX(sK84,sK86) & oP(sK84,sK85)) [
      choice axiom]
768. ~EX(sK85,sK86) & EX(sK84,sK86) & oP(sK84,sK85) [skolemisation 565,767]
784. ~P(X0,X1) | AB(X0) | PD(X1) [cnf transformation 371]
786. ~P(X0,X1) | ~T(X0) | T(X1) [cnf transformation 589]
793. ~P(X1,X2) | P(X0,X2) | ~P(X0,X1) [cnf transformation 378]
```

```
841. ~PRE(X0,X1) | PD(X0) | ED(X0) | Q(X0) [cnf transformation 411]
844. ~PRE(X0,X1) | ~P(X0,X2) | PRE(X2,X1) [cnf transformation 415]
993. ~TR(X0) | R(X0) [cnf transformation 533]
994. ~R(X0) | AB(X0) [cnf transformation 534]
1000. ~T(X0) | TR(X0) [cnf transformation 538]
1002. ~ED(X0) | ~AB(X0) [cnf transformation 540]
1003. ~AB(X0) | ~PD(X0) [cnf transformation 541]
1004. ~Q(X0) | ~AB(X0) [cnf transformation 542]
1023. ~sP1(X0,X1) | T(X1) | PRE(X1,X0) | AB(X1) [cnf transformation 701]
1026. ~sP1(X0,X1) | P(X0,X1) | PRE(X1,X0) | ~T(X1) [cnf transformation 701]
1029. ~PRE(X1,X0) | sP1(X0,X1) [cnf transformation 701]
1030. ~P(X0,X1) | sP1(X0,X1) | ~T(X1) [cnf transformation 701]
1032. ~EX(X0,X1) | sP1(X1,X0) [cnf transformation 702]
1033. ~sP1(X1,X0) | EX(X0,X1) [cnf transformation 702]
1034. ~oP(X0,X1) | P(X0,X1) [cnf transformation 704]
1035. ~oP(X0,X1) | PD(X0) | T(X0) [cnf transformation 704]
1122. oP(sK84,sK85) [cnf transformation 768]
1123. EX(sK84,sK86) [cnf transformation 768]
1124. ~EX(sK85,sK86) [cnf transformation 768]
1177. sP1(sK86,sK84) [resolution 1032,1123]
1180. P(sK84,sK85) [resolution 1034,1122]
1193. 1 <=> PD(sK84) [avatar definition]
1194. ~PD(sK84) <- (~1) [avatar component clause 1193]
1197. 2 <=> AB(sK84) [avatar definition]
1198. ~AB(sK84) <- (~2) [avatar component clause 1197]
1199. AB(sK84) <- (2) [avatar component clause 1197]
1201. ~PD(sK84) <- (2) [resolution 1199,1003]
1202. ~1 | ~2 [avatar split clause 1201,1197,1193]
1209. AB(sK84) | PD(sK85) [resolution 784,1180]
1216. ~T(sK84) | T(sK85) [resolution 786,1180]
1218. 3 <=> T(sK85) [avatar definition]
1220. T(sK85) <- (3) [avatar component clause 1218]
1222. 4 <=> T(sK84) [avatar definition]
1223. T(sK84) <- (4) [avatar component clause 1222]
1224. ~T(sK84) <- (~4) [avatar component clause 1222]
1225. 3 | ~4 [avatar split clause 1216,1222,1218]
1284. PD(sK84) | T(sK84) [resolution 1035,1122]
1288. PD(sK84) <- (~4) [subsumption resolution 1284,1224]
1289. 1 | 4 [avatar split clause 1288,1222,1193]
1291. 7 <=> PD(sK85) [avatar definition]
1295. 8 <=> AB(sK85) [avatar definition]
1297. AB(sK85) <- (8) [avatar component clause 1295]
1300. 7 | 2 [avatar split clause 1209,1197,1291]
1488. ~P(X7,sK84) | P(X7,sK85) [resolution 793,1180]
1885. T(sK84) | PRE(sK84,sK86) | AB(sK84) [resolution 1023,1177]
1887. PRE(sK84,sK86) | AB(sK84) <- (~4) [subsumption resolution 1885,1224]
1888. PRE(sK84,sK86) <- (~2, ~4) [subsumption resolution 1887,1198]
1892. ~P(sK84,X1) | PRE(X1,sK86) <- (~2, ~4) [resolution 1888,844]
1932. PRE(sK85,sK86) <- (~2, ~4) [resolution 1892,1180]
1941. sP1(sK86,sK85) <- (~2, ~4) [resolution 1932,1029]
1947. EX(sK85,sK86) <- (~2, ~4) [resolution 1941,1033]
1948. $false <- (~2, ~4) [subsumption resolution 1947,1124]
1949. 2 | 4 [avatar contradiction clause 1948]
1957. 13 <=> PRE(sK84,sK86) [avatar definition]
1959. PRE(sK84,sK86) <- (13) [avatar component clause 1957]
1966. 15 <=> P(sK86,sK84) [avatar definition]
1968. P(sK86,sK84) <- (15) [avatar component clause 1966]
1980. TR(sK85) <- (3) [resolution 1220,1000]
2004. R(sK85) <- (3) [resolution 1980,993]
2007. AB(sK85) <- (3) [resolution 2004,994]
2008. 8 | ~3 [avatar split clause 2007,1218,1295]
2010. ~PD(sK85) <- (8) [resolution 1297,1003]
2024. ~7 | ~8 [avatar split clause 2010,1295,1291]
2058. P(sK86,sK84) | PRE(sK84,sK86) | ~T(sK84) [resolution 1026,1177]
2061. P(sK86,sK84) | PRE(sK84,sK86) <- (4) [subsumption resolution 2058,1223]
2062. 13 | 15 | ~4 [avatar split clause 2061,1222,1966,1957]
2090. PD(sK84) | ED(sK84) | Q(sK84) <- (13) [resolution 1959,841]
2097. ED(sK84) | Q(sK84) <- (~1, 13) [subsumption resolution 2090,1194]
2099. 18 <=> Q(sK84) [avatar definition]
2101. Q(sK84) <- (18) [avatar component clause 2099]
2103. 19 <=> ED(sK84) [avatar definition]
2105. ED(sK84) <- (19) [avatar component clause 2103]
2106. 18 | 19 | 1 | ~13 [avatar split clause 2097,1957,1193,2103,2099]
2157. ~AB(sK84) <- (18) [resolution 2101,1004]
2171. $false <- (2, 18) [subsumption resolution 2157,1199]
2172. ~2 | ~18 [avatar contradiction clause 2171]
2186. ~AB(sK84) <- (19) [resolution 2105,1002]
2200. $false <- (2, 19) [subsumption resolution 2186,1199]
2201. ~2 | ~19 [avatar contradiction clause 2200]
2216. P(sK86,sK85) <- (15) [resolution 1968,1488]
2288. sP1(sK86,sK85) | ~T(sK85) <- (15) [resolution 2216,1030]
2305. sP1(sK86,sK85) <- (3, 15) [subsumption resolution 2288,1220]
2324. EX(sK85,sK86) <- (3, 15) [resolution 2305,1033]
2325. $false <- (3, 15) [subsumption resolution 2324,1124]
2326. ~3 | ~15 [avatar contradiction clause 2325]
2327. $false [avatar sat refutation 1202,1225,1289,1300,1949,2008,2024,2062,2106,2172,2201,2326]
```

Proof of Theorem (t_{db}8)

```
5. ! [X0,X1] : (P(X0,X1) => ((PD(X1) & PD(X0)) | (AB(X1) & AB(X0)))) [input]
8. ! [X0] : ((PD(X0) | AB(X0)) => P(X0,X0)) [input]
26. ! [X0,X4] : (PRE(X0,X4) <=> ? [X5] : (P(X4,X5) & TLC(X0,X5))) [input]
27. ! [X0,X4] : (TLC(X0,X4) => (T(X4) & (Q(X0) | PD(X0) | ED(X0)))) [input]
28. ! [X0] : ((Q(X0) | PD(X0) | ED(X0)) => ? [X4] : TLC(X0,X4)) [input]
124. ! [X0] : ((TR(X0) | PR(X0) | AR(X0)) <=> R(X0)) [input]
125. ! [X0] : (R(X0) => AB(X0)) [input]
130. ! [X0] : (T(X0) => TR(X0)) [input]
133. ~? [X0] : (PD(X0) & AB(X0)) [input]
163. ! [X0,X1] : (oP(X0,X1) <=> (((T(X1) & T(X0)) | (PD(X1) & PD(X0))) & P(X0,X1))) [input]
167. ! [X0,X4] : (::(X0,proc,X4) <=> (~::(X0,pbnd,X4) & PRE(X0,X4) & PD(X0)))) [input]
189. ! [X0,X4,X1] : ((oP(X1,X0) & ::(X0,proc,X4)) => ? [X9] : (::(X1,pbnd,X9) | ::(X1,proc,X9))) [input]
190. ~! [X0,X4,X1] : ((oP(X1,X0) & ::(X0,proc,X4)) => ? [X9] : (::(X1,pbnd,X9) | ::(X1,proc,X9))) [negated
     conjecture 189]
192. ! [X0,X1] : (PRE(X0,X1) <=> ? [X2] : (P(X1,X2) & TLC(X0,X2))) [rectify 26]
193. ! [X0,X1] : (TLC(X0,X1) => (T(X1) & (Q(X0) | PD(X0) | ED(X0)))) [rectify 27]
194. ! [X0] : ((Q(X0) | PD(X0) | ED(X0)) => ? [X1] : TLC(X0,X1)) [rectify 28]
308. ! [X0,X1] : (::(X0,proc,X1) <=> (~::(X0,pbnd,X1) & PRE(X0,X1) & PD(X0)))) [rectify 167]
335. ~! [X0,X1,X2] : ((oP(X2,X0) & ::(X0,proc,X1)) => ? [X3] : (::(X2,pbnd,X3) | ::(X2,proc,X3))) [rectify 190]
336. ! [X0] : ((TR(X0) | PR(X0) | AR(X0)) => R(X0)) [unused predicate definition removal 124]
370. ! [X0,X1] : (((PD(X1) & PD(X0)) | (AB(X1) & AB(X0))) | ~P(X0,X1)) [ennf transformation 5]
371. ! [X0,X1] : ((PD(X1) & PD(X0)) | (AB(X1) & AB(X0)) | ~P(X0,X1)) [flattening 370]
374. ! [X0] : (P(X0,X0) | (~PD(X0) & ~AB(X0))) [ennf transformation 8]
400. ! [X0,X1] : ((T(X1) & (Q(X0) | PD(X0) | ED(X0))) | ~TLC(X0,X1)) [ennf transformation 193]
401. ! [X0] : (? [X1] : TLC(X0,X1) | (~Q(X0) & ~PD(X0) & ~ED(X0))) [ennf transformation 194]
533. ! [X0] : (R(X0) | (~TR(X0) & ~PR(X0) & ~AR(X0))) [ennf transformation 336]
534. ! [X0] : (AB(X0) | ~R(X0)) [ennf transformation 125]
538. ! [X0] : (TR(X0) | ~T(X0)) [ennf transformation 130]
541. ! [X0] : (~PD(X0) | ~AB(X0)) [ennf transformation 133]
564. ? [X0,X1,X2] : (! [X3] : (~::(X2,pbnd,X3) & ~::(X2,proc,X3)) & (oP(X2,X0) & ::(X0,proc,X1))) [ennf
     transformation 335]
565. ? [X0,X1,X2] : (! [X3] : (~::(X2,pbnd,X3) & ~::(X2,proc,X3)) & oP(X2,X0) & ::(X0,proc,X1)) [flattening 564]
607. ! [X0,X1] : ((PRE(X0,X1) | ! [X2] : (~P(X1,X2) | ~TLC(X0,X2))) & (? [X2] : (P(X1,X2) & TLC(X0,X2)) | ~PRE(X0,
     X1))) [nnf transformation 192]
608. ! [X0,X1] : ((PRE(X0,X1) | ! [X2] : (~P(X1,X2) | ~TLC(X0,X2))) & (? [X3] : (P(X1,X3) & TLC(X0,X3)) | ~PRE(X0,
     X1))) [rectify 607]
609. ! [X1,X0] : (? [X3] : (P(X1,X3) & TLC(X0,X3)) => (P(X1,sK13(X0,X1)) & TLC(X0,sK13(X0,X1)))) [choice axiom]
610. ! [X0,X1] : ((PRE(X0,X1) | ! [X2] : (~P(X1,X2) | ~TLC(X0,X2))) & ((P(X1,sK13(X0,X1)) & TLC(X0,sK13(X0,X1))) |
      ~PRE(X0,X1))) [skolemisation 608,609]
611. ! [X0] : (? [X1] : TLC(X0,X1) => TLC(X0,sK14(X0))) [choice axiom]
612. ! [X0] : (TLC(X0,sK14(X0)) | (~Q(X0) & ~PD(X0) & ~ED(X0))) [skolemisation 401,611]
703. ! [X0,X1] : ((oP(X0,X1) | (((~T(X1) | ~T(X0)) & (~PD(X1) | ~PD(X0))) | ~P(X0,X1))) & ((((T(X1) & T(X0)) | (PD
     (X1) & PD(X0))) & P(X0,X1)) | ~oP(X0,X1))) [nnf transformation 163]
704. ! [X0,X1] : ((oP(X0,X1) | ((~T(X1) | ~T(X0)) & (~PD(X1) | ~PD(X0))) | ~P(X0,X1)) & ((((T(X1) & T(X0)) | (PD(
     X1) & PD(X0))) & P(X0,X1)) | ~oP(X0,X1))) [flattening 703]
716. ! [X0,X1] : ((::(X0,proc,X1) | (::(X0,pbnd,X1) | ~PRE(X0,X1) | ~PD(X0))) & ((~::(X0,pbnd,X1) & PRE(X0,X1) &
     PD(X0)) | ~::(X0,proc,X1))) [nnf transformation 308]
717. ! [X0,X1] : ((::(X0,proc,X1) | ::(X0,pbnd,X1) | ~PRE(X0,X1) | ~PD(X0)) & ((~::(X0,pbnd,X1) & PRE(X0,X1) & PD(
     X0)) | ~::(X0,proc,X1))) [flattening 716]
767. ? [X0,X1,X2] : (! [X3] : (~::(X2,pbnd,X3) & ~::(X2,proc,X3)) & oP(X2,X0) & ::(X0,proc,X1)) => (! [X3] : (~::(
     sK86,pbnd,X3) & ~::(sK86,proc,X3)) & oP(sK86,sK84) & ::(sK84,proc,sK85)) [choice axiom]
768. ! [X3] : (~::(sK86,pbnd,X3) & ~::(sK86,proc,X3)) & oP(sK86,sK84) & ::(sK84,proc,sK85) [skolemisation 565,767]
783. ~P(X0,X1) | AB(X1) | PD(X0) [cnf transformation 371]
790. P(X0,X0) | ~AB(X0) [cnf transformation 374]
827. ~TLC(X0,X2) | ~P(X1,X2) | PRE(X0,X1) [cnf transformation 610]
829. ~TLC(X0,X1) | T(X1) [cnf transformation 400]
831. TLC(X0,sK14(X0)) | ~PD(X0) [cnf transformation 612]
993. ~TR(X0) | R(X0) [cnf transformation 533]
994. ~R(X0) | AB(X0) [cnf transformation 534]
1000. ~T(X0) | TR(X0) [cnf transformation 538]
1003. ~AB(X0) | ~PD(X0) [cnf transformation 541]
1034. ~oP(X0,X1) | P(X0,X1) [cnf transformation 704]
1052. ~::(X0,proc,X1) | PD(X0) [cnf transformation 717]
1055. ::(X0,proc,X1) | ::(X0,pbnd,X1) | ~PRE(X0,X1) | ~PD(X0) [cnf transformation 717]
1122. ::(sK84,proc,sK85) [cnf transformation 768]
1123. oP(sK86,sK84) [cnf transformation 768]
1124. ~::(sK86,proc,X3) [cnf transformation 768]
1125. ~::(sK86,pbnd,X3) [cnf transformation 768]
1165. T(sK14(X0)) | ~PD(X0) [resolution 831,829]
1166. TR(sK14(X0)) | ~PD(X0) [resolution 1165,1000]
1179. R(sK14(X0)) | ~PD(X0) [resolution 1166,993]
1181. P(sK86,sK84) [resolution 1034,1123]
1183. PD(sK84) [resolution 1052,1122]
1187. AB(sK14(X0)) | ~PD(X0) [resolution 1179,994]
1193. 1 <=> PD(sK86) [avatar definition]
1194. ~PD(sK86) <- (~1) [avatar component clause 1193]
1195. PD(sK86) <- (1) [avatar component clause 1193]
1205. AB(sK84) | PD(sK86) [resolution 783,1181]
1206. AB(sK84) <- (~1) [subsumption resolution 1205,1194]
1210. ~PD(sK84) <- (~1) [resolution 1206,1003]
1211. $false <- (~1) [subsumption resolution 1210,1183]
1212. 1 [avatar contradiction clause 1211]
1536. ~P(X2,sK14(X3)) | PRE(X3,X2) | ~PD(X3) [resolution 827,831]
2502. ::(sK86,pbnd,X0) | ~PRE(sK86,X0) | ~PD(sK86) [resolution 1055,1124]
2507. ~PRE(sK86,X0) | ~PD(sK86) [subsumption resolution 2502,1125]
2508. ~PRE(sK86,X0) <- (1) [subsumption resolution 2507,1195]
3216. PRE(X1,sK14(X1)) | ~PD(X1) | ~AB(sK14(X1)) [resolution 1536,790]
3229. PRE(X1,sK14(X1)) | ~PD(X1) [subsumption resolution 3216,1187]
3276. ~PD(sK86) <- (1) [resolution 3229,2508]
```

```
3279. $false <- (1) [subsumption resolution 3276,1195]
3280. ~1 [avatar contradiction clause 3279]
3281. $false [avatar sat refutation 1212,3280]
```

Proof of Theorem (t_{db}10)

```
8. ! [X0] : ((PD(X0) | AB(X0)) => P(X0,X0)) [input]
26. ! [X0,X4] : (PRE(X0,X4) <=> ? [X5] : (P(X4,X5) & TLC(X0,X5))) [input]
27. ! [X0,X4] : (TLC(X0,X4) => (T(X4) & (Q(X0) | PD(X0) | ED(X0)))) [input]
36. ! [X0,X4,X1] : ((P(X0,X1) & PRE(X0,X4)) => PRE(X1,X4)) [input]
124. ! [X0] : ((TR(X0) | PR(X0) | AR(X0)) <=> R(X0)) [input]
125. ! [X0] : (R(X0) => AB(X0)) [input]
130. ! [X0] : (T(X0) => TR(X0)) [input]
133. ~? [X0] : (PD(X0) & AB(X0)) [input]
163. ! [X0,X1] : (oP(X0,X1) <=> (((T(X1) & T(X0)) | (PD(X1) & PD(X0))) & P(X0,X1))) [input]
166. ! [X0,X4] : (::(X0,pbnd,X4) <=> (AT(X4) & TLC(X0,X4) & ? [X1] : tmPP(X0,X1) & PD(X0))) [input]
167. ! [X0,X4] : (::(X0,proc,X4) <=> (~::(X0,pbnd,X4) & PRE(X0,X4) & PD(X0))) [input]
189. ! [X0,X4,X1] : ((oP(X0,X1) & ::(X0,pbnd,X4)) => ? [X9] : (::(X1,proc,X9) | ::(X1,pbnd,X9))) [input]
190. ~! [X0,X4,X1] : ((oP(X0,X1) & ::(X0,pbnd,X4)) => ? [X9] : (::(X1,proc,X9) | ::(X1,pbnd,X9))) [negated
       conjecture 189]
192. ! [X0,X1] : (PRE(X0,X1) <=> ? [X2] : (P(X1,X2) & TLC(X0,X2))) [rectify 26]
193. ! [X0,X1] : (TLC(X0,X1) => (T(X1) & (Q(X0) | PD(X0) | ED(X0)))) [rectify 27]
202. ! [X0,X1,X2] : ((P(X0,X2) & PRE(X0,X1)) => PRE(X2,X1)) [rectify 36]
307. ! [X0,X1] : (::(X0,pbnd,X1) <=> (AT(X1) & TLC(X0,X1) & ? [X2] : tmPP(X0,X2) & PD(X0))) [rectify 166]
308. ! [X0,X1] : (::(X0,proc,X1) <=> (~::(X0,pbnd,X1) & PRE(X0,X1) & PD(X0))) [rectify 167]
335. ~! [X0,X1,X2] : ((oP(X0,X2) & ::(X0,pbnd,X1)) => ? [X3] : (::(X2,proc,X3) | ::(X2,pbnd,X3))) [rectify 190]
336. ! [X0] : ((TR(X0) | PR(X0) | AR(X0)) => R(X0)) [unused predicate definition removal 124]
374. ! [X0] : (P(X0,X0) | (~PD(X0) & ~AB(X0))) [ennf transformation 8]
400. ! [X0,X1] : ((T(X1) & (Q(X0) | PD(X0) | ED(X0))) | ~TLC(X0,X1)) [ennf transformation 193]
414. ! [X0,X1,X2] : (PRE(X2,X1) | (~P(X0,X2) | ~PRE(X0,X1))) [ennf transformation 202]
415. ! [X0,X1,X2] : (PRE(X2,X1) | ~P(X0,X2) | ~PRE(X0,X1)) [flattening 414]
533. ! [X0] : (R(X0) | (~TR(X0) & ~PR(X0) & ~AR(X0))) [ennf transformation 336]
534. ! [X0] : (AB(X0) | ~R(X0)) [ennf transformation 125]
538. ! [X0] : (TR(X0) | ~T(X0)) [ennf transformation 130]
541. ! [X0] : (~PD(X0) | ~AB(X0)) [ennf transformation 133]
564. ? [X0,X1,X2] : (! [X3] : (~::(X2,proc,X3) & ~::(X2,pbnd,X3)) & (oP(X0,X2) & ::(X0,pbnd,X1))) [ennf
       transformation 335]
565. ? [X0,X1,X2] : (! [X3] : (~::(X2,proc,X3) & ~::(X2,pbnd,X3)) & oP(X0,X2) & ::(X0,pbnd,X1)) [flattening 564]
607. ! [X0,X1] : ((PRE(X0,X1) | ! [X2] : (~P(X1,X2) | ~TLC(X0,X2))) & (? [X2] : (P(X1,X2) & TLC(X0,X2)) | ~PRE(X0,
       X1))) [nnf transformation 192]
608. ! [X0,X1] : ((PRE(X0,X1) | ! [X2] : (~P(X1,X2) | ~TLC(X0,X2))) & (? [X3] : (P(X1,X3) & TLC(X0,X3)) | ~PRE(X0,
       X1))) [rectify 607]
609. ! [X1,X0] : (? [X3] : (P(X1,X3) & TLC(X0,X3)) => (P(X1,sK13(X0,X1)) & TLC(X0,sK13(X0,X1)))) [choice axiom]
610. ! [X0,X1] : ((PRE(X0,X1) | ! [X2] : (~P(X1,X2) | ~TLC(X0,X2))) & ((P(X1,sK13(X0,X1)) & TLC(X0,sK13(X0,X1))) |
       ~PRE(X0,X1))) [skolemisation 608,609]
703. ! [X0,X1] : ((oP(X0,X1) | (((~T(X1) | ~T(X0)) & (~PD(X1) | ~PD(X0))) | ~P(X0,X1))) & ((((T(X1) & T(X0)) | (PD
       (X1) & PD(X0))) & P(X0,X1)) | ~oP(X0,X1))) [nnf transformation 163]
704. ! [X0,X1] : ((oP(X0,X1) | ((~T(X1) | ~T(X0)) & (~PD(X1) | ~PD(X0))) | ~P(X0,X1)) & ((((T(X1) & T(X0)) | (PD(
       X1) & PD(X0))) & P(X0,X1)) | ~oP(X0,X1))) [flattening 703]
711. ! [X0,X1] : ((::(X0,pbnd,X1) | (~AT(X1) | ~TLC(X0,X1) | ! [X2] : ~tmPP(X0,X2) | ~PD(X0))) & ((AT(X1) & TLC(X0
       ,X1) & ? [X2] : tmPP(X0,X2) & PD(X0)) | ~::(X0,pbnd,X1))) [nnf transformation 307]
712. ! [X0,X1] : ((::(X0,pbnd,X1) | ~AT(X1) | ~TLC(X0,X1) | ! [X2] : ~tmPP(X0,X2) | ~PD(X0)) & ((AT(X1) & TLC(X0,
       X1) & ? [X2] : tmPP(X0,X2) & PD(X0)) | ~::(X0,pbnd,X1))) [flattening 711]
713. ! [X0,X1] : ((::(X0,pbnd,X1) | ~AT(X1) | ~TLC(X0,X1) | ! [X2] : ~tmPP(X0,X2) | ~PD(X0)) & ((AT(X1) & TLC(X0,
       X1) & ? [X3] : tmPP(X0,X3) & PD(X0)) | ~::(X0,pbnd,X1))) [rectify 712]
714. ! [X0] : (? [X3] : tmPP(X0,X3) => tmPP(X0,sK69(X0))) [choice axiom]
715. ! [X0,X1] : ((::(X0,pbnd,X1) | ~AT(X1) | ~TLC(X0,X1) | ! [X2] : ~tmPP(X0,X2) | ~PD(X0)) & ((AT(X1) & TLC(X0,
       X1) & tmPP(X0,sK69(X0)) & PD(X0)) | ~::(X0,pbnd,X1))) [skolemisation 713,714]
716. ! [X0,X1] : ((::(X0,proc,X1) | (::(X0,pbnd,X1) | ~PRE(X0,X1) | ~PD(X0))) & ((~::(X0,pbnd,X1) & PRE(X0,X1) &
       PD(X0)) | ~::(X0,proc,X1))) [nnf transformation 308]
717. ! [X0,X1] : ((::(X0,proc,X1) | ::(X0,pbnd,X1) | ~PRE(X0,X1) | ~PD(X0)) & ((~::(X0,pbnd,X1) & PRE(X0,X1) & PD(
       X0)) | ~::(X0,proc,X1))) [flattening 716]
767. ? [X0,X1,X2] : (! [X3] : (~::(X2,proc,X3) & ~::(X2,pbnd,X3)) & oP(X0,X2) & ::(X0,pbnd,X1)) => (! [X3] : (~::(
       sK86,proc,X3) & ~::(sK86,pbnd,X3)) & oP(sK84,sK86) & ::(sK84,pbnd,sK85)) [choice axiom]
768. ! [X3] : (~::(sK86,proc,X3) & ~::(sK86,pbnd,X3)) & oP(sK84,sK86) & ::(sK84,pbnd,sK85) [skolemisation 565,767]
790. P(X0,X0) | ~AB(X0) [cnf transformation 374]
827. ~TLC(X0,X2) | ~P(X1,X2) | PRE(X0,X1) [cnf transformation 610]
829. ~TLC(X0,X1) | T(X1) [cnf transformation 400]
844. ~PRE(X0,X1) | ~P(X0,X2) | PRE(X2,X1) [cnf transformation 415]
993. ~TR(X0) | R(X0) [cnf transformation 533]
994. ~R(X0) | AB(X0) [cnf transformation 534]
1000. ~T(X0) | TR(X0) [cnf transformation 538]
1003. ~AB(X0) | ~PD(X0) [cnf transformation 541]
1034. ~oP(X0,X1) | P(X0,X1) [cnf transformation 704]
1036. ~oP(X0,X1) | PD(X1) | T(X0) [cnf transformation 704]
1047. ~::(X0,pbnd,X1) | PD(X0) [cnf transformation 715]
1049. ~::(X0,pbnd,X1) | TLC(X0,X1) [cnf transformation 715]
1055. ::(X0,proc,X1) | ::(X0,pbnd,X1) | ~PRE(X0,X1) | ~PD(X0) [cnf transformation 717]
1122. ::(sK84,pbnd,sK85) [cnf transformation 768]
1123. oP(sK84,sK86) [cnf transformation 768]
1124. ~::(sK86,pbnd,X3) [cnf transformation 768]
1125. ~::(sK86,proc,X3) [cnf transformation 768]
1181. P(sK84,sK86) [resolution 1034,1123]
1183. PD(sK84) [resolution 1047,1122]
1187. 1 <=> AB(sK85) [avatar definition]
1189. AB(sK85) <- (1) [avatar component clause 1187]
1210. 3 <=> PD(sK86) [avatar definition]
1211. ~PD(sK86) <- (~3) [avatar component clause 1210]
1212. PD(sK86) <- (3) [avatar component clause 1210]
1214. 4 <=> AB(sK84) [avatar definition]
```

```
1216. AB(sK84) <- (4) [avatar component clause 1214]
1230. 6 <=> T(sK84) [avatar definition]
1231. T(sK84) <- (6) [avatar component clause 1230]
1293. PD(sK86) | T(sK84) [resolution 1036,1123]
1300. TLC(sK84,sK85) [resolution 1049,1122]
1301. T(sK85) [resolution 1300,829]
1302. TR(sK85) [resolution 1301,1000]
1303. R(sK85) [resolution 1302,993]
1304. AB(sK85) [resolution 1303,994]
1305. 1 [avatar split clause 1304,1187]
1456. TR(sK84) <- (6) [resolution 1231,1000]
1458. R(sK84) <- (6) [resolution 1456,993]
1460. AB(sK84) <- (6) [resolution 1458,994]
1461. 4 | ~6 [avatar split clause 1460,1230,1214]
1579. ~P(X9,sK85) | PRE(sK84,X9) [resolution 827,1300]
1733. PRE(sK84,sK85) | ~AB(sK85) [resolution 1579,790]
1738. PRE(sK84,sK85) <- (1) [subsumption resolution 1733,1189]
1741. ~P(sK84,X1) | PRE(X1,sK85) <- (1) [resolution 1738,844]
1969. PRE(sK86,sK85) <- (1) [resolution 1741,1181]
2484. ::(sK86,pbnd,X0) | ~PRE(sK86,X0) | ~PD(sK86) [resolution 1055,1125]
2489. ~PRE(sK86,X0) | ~PD(sK86) [subsumption resolution 2484,1124]
2490. ~PRE(sK86,X0) <- (3) [subsumption resolution 2489,1212]
2492. $false <- (1, 3) [resolution 2490,1969]
2499. ~1 | ~3 [avatar contradiction clause 2492]
2526. T(sK84) <- (~3) [subsumption resolution 1293,1211]
2527. 6 | 3 [avatar split clause 2526,1210,1230]
2558. ~PD(sK84) <- (4) [resolution 1216,1003]
2559. $false <- (4) [subsumption resolution 2558,1183]
2560. ~4 [avatar contradiction clause 2559]
2561. $false [avatar sat refutation 1305,1461,2499,2527,2560]
```

Proof of Theorem (t_{db}12)

```
6. ! [X0,X1] : (P(X0,X1) => (T(X0) <=> T(X1))) [input]
8. ! [X0] : ((PD(X0) | AB(X0)) => P(X0,X0)) [input]
124. ! [X0] : ((TR(X0) | PR(X0) | AR(X0)) <=> R(X0)) [input]
125. ! [X0] : (R(X0) => AB(X0)) [input]
130. ! [X0] : (T(X0) => TR(X0)) [input]
162. ! [X0,X4] : (EX(X0,X4) <=> ((T(X4) & ~T(X0) & AB(X0)) | (P(X4,X0) & T(X0)) | PRE(X0,X4))) [input]
163. ! [X0,X1] : (oP(X0,X1) <=> (((T(X1) & T(X0)) | (PD(X1) & PD(X0))) & P(X0,X1))) [input]
168. ! [X0,X4] : (::(X0,treg,X4) <=> (EX(X0,X4) & T(X0))) [input]
189. ! [X0,X1] : (oP(X0,X1) => (? [X4] : ::(X0,treg,X4) <=> ? [X4] : ::(X1,treg,X4))) [input]
190. ~! [X0,X1] : (oP(X0,X1) => (? [X4] : ::(X0,treg,X4) <=> ? [X4] : ::(X1,treg,X4))) [negated conjecture 189]
304. ! [X0,X1] : (EX(X0,X1) <=> ((T(X1) & ~T(X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1))) [rectify 162]
309. ! [X0,X1] : (::(X0,treg,X1) <=> (EX(X0,X1) & T(X0))) [rectify 168]
335. ~! [X0,X1] : (oP(X0,X1) => (? [X2] : ::(X0,treg,X2) <=> ? [X3] : ::(X1,treg,X3))) [rectify 190]
336. ! [X0] : ((TR(X0) | PR(X0) | AR(X0)) => R(X0)) [unused predicate definition removal 124]
372. ! [X0,X1] : ((T(X0) <=> T(X1)) | ~P(X0,X1)) [ennf transformation 6]
374. ! [X0] : (P(X0,X0) | (~PD(X0) & ~AB(X0))) [ennf transformation 8]
533. ! [X0] : (R(X0) | (~TR(X0) & ~PR(X0) & ~AR(X0))) [ennf transformation 336]
534. ! [X0] : (AB(X0) | ~R(X0)) [ennf transformation 125]
538. ! [X0] : (TR(X0) | ~T(X0)) [ennf transformation 130]
564. ? [X0,X1] : ((? [X2] : ::(X0,treg,X2) <~> ? [X3] : ::(X1,treg,X3)) & oP(X0,X1)) [ennf transformation 335]
567. ! [X1,X0] : (sP1(X1,X0) <=> ((T(X1) & ~T(X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1))) [predicate
      definition introduction]
568. ! [X0,X1] : (EX(X0,X1) <=> sP1(X1,X0)) [definition folding 304,567]
588. ! [X0,X1] : (((T(X0) | ~T(X1)) & (T(X1) | ~T(X0))) | ~P(X0,X1)) [nnf transformation 372]
698. ! [X1,X0] : ((sP1(X1,X0) | ((~T(X1) | T(X0) | ~AB(X0)) & (~P(X1,X0) | ~T(X0)) & ~PRE(X0,X1))) & (((T(X1) & ~T
      (X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1)) | ~sP1(X1,X0))) [nnf transformation 567]
699. ! [X1,X0] : ((sP1(X1,X0) | ((~T(X1) | T(X0) | ~AB(X0)) & (~P(X1,X0) | ~T(X0)) & ~PRE(X0,X1))) & ((T(X1) & ~T(
      X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1) | ~sP1(X1,X0))) [flattening 698]
700. ! [X0,X1] : ((sP1(X0,X1) | ((~T(X0) | T(X1) | ~AB(X1)) & (~P(X0,X1) | ~T(X1)) & ~PRE(X1,X0))) & ((T(X0) & ~T(
      X1) & AB(X1)) | (P(X0,X1) & T(X1)) | PRE(X1,X0) | ~sP1(X0,X1))) [rectify 699]
701. ! [X0,X1] : ((EX(X0,X1) | ~sP1(X1,X0)) & (sP1(X1,X0) | ~EX(X0,X1))) [nnf transformation 568]
702. ! [X0,X1] : ((oP(X0,X1) | (((~T(X1) | ~T(X0)) & (~PD(X1) | ~PD(X0))) | ~P(X0,X1))) & ((((T(X1) & T(X0)) | (PD
      (X1) & PD(X0))) & P(X0,X1)) | ~oP(X0,X1))) [nnf transformation 163]
703. ! [X0,X1] : ((oP(X0,X1) | ((~T(X1) | ~T(X0)) & (~PD(X1) | ~PD(X0))) | ~P(X0,X1)) & ((((T(X1) & T(X0)) | (PD(
      X1) & PD(X0))) & P(X0,X1)) | ~oP(X0,X1))) [flattening 702]
717. ! [X0,X1] : (((::(X0,treg,X1) | (~EX(X0,X1) | ~T(X0))) & ((EX(X0,X1) & T(X0)) | ~::(X0,treg,X1))) [nnf
      transformation 309]
718. ! [X0,X1] : (((::(X0,treg,X1) | ~EX(X0,X1) | ~T(X0)) & ((EX(X0,X1) & T(X0)) | ~::(X0,treg,X1))) [flattening
      717]
766. ? [X0,X1] : (((! [X3] : ~::(X1,treg,X3) | ! [X2] : ~::(X0,treg,X2)) & (? [X3] : ::(X1,treg,X3) | ? [X2] : ::(
      X0,treg,X2))) & oP(X0,X1)) [nnf transformation 564]
767. ? [X0,X1] : (((! [X3] : ~::(X1,treg,X3) | ! [X2] : ~::(X0,treg,X2)) & (? [X3] : ::(X1,treg,X3) | ? [X2] : ::(
      X0,treg,X2)) & oP(X0,X1)) [flattening 766]
768. ? [X0,X1] : (((! [X2] : ~::(X1,treg,X2) | ! [X3] : ~::(X0,treg,X3)) & (? [X4] : ::(X1,treg,X4) | ? [X5] : ::(
      X0,treg,X5)) & oP(X0,X1)) [rectify 767]
769. ? [X0,X1] : (((! [X2] : ~::(X1,treg,X2) | ! [X3] : ~::(X0,treg,X3)) & (? [X4] : ::(X1,treg,X4) | ? [X5] : ::(
      X0,treg,X5)) & oP(X0,X1)) => ((! [X2] : ~::(sK85,treg,X2) | ! [X3] : ~::(sK84,treg,X3)) & (? [X4] : ::(sK85,
      treg,X4) | ? [X5] : ::(sK84,treg,X5)) & oP(sK84,sK85)) [choice axiom]
770. ? [X4] : ::(sK85,treg,X4) => ::(sK85,treg,sK86) [choice axiom]
771. ? [X5] : ::(sK84,treg,X5) => ::(sK84,treg,sK87) [choice axiom]
772. (! [X2] : ~::(sK85,treg,X2) | ! [X3] : ~::(sK84,treg,X3)) & (::(sK85,treg,sK86) | ::(sK84,treg,sK87)) & oP(
      sK84,sK85) [skolemisation 768,771,770,769]
790. ~P(X0,X1) | ~T(X0) | T(X1) [cnf transformation 588]
791. ~P(X0,X1) | ~T(X1) | T(X0) [cnf transformation 588]
794. P(X0,X0) | ~AB(X0) [cnf transformation 374]
997. ~TR(X0) | R(X0) [cnf transformation 533]
998. ~R(X0) | AB(X0) [cnf transformation 534]
```

```
1004. ~T(X0) | TR(X0) [cnf transformation 538]
1034. ~P(X0,X1) | sP1(X0,X1) | ~T(X1) [cnf transformation 700]
1037. ~sP1(X1,X0) | EX(X0,X1) [cnf transformation 701]
1038. ~oP(X0,X1) | P(X0,X1) [cnf transformation 703]
1060. ~::(X0,treg,X1) | T(X0) [cnf transformation 718]
1062. ::(X0,treg,X1) | ~EX(X0,X1) | ~T(X0) [cnf transformation 718]
1126. oP(sK84,sK85) [cnf transformation 772]
1127. ::(sK85,treg,sK86) | ::(sK84,treg,sK87) [cnf transformation 772]
1128. ~::(sK85,treg,X2) | ~::(sK84,treg,X3) [cnf transformation 772]
1144. 1 <=> ! [X3] : ~::(sK84,treg,X3) [avatar definition]
1145. ~::(sK84,treg,X3) <- (1) [avatar component clause 1144]
1147. 2 <=> ! [X2] : ~::(sK85,treg,X2) [avatar definition]
1148. ~::(sK85,treg,X2) <- (2) [avatar component clause 1147]
1149. 1 | 2 [avatar split clause 1128,1147,1144]
1151. 3 <=> ::(sK84,treg,sK87) [avatar definition]
1153. ::(sK84,treg,sK87) <- (3) [avatar component clause 1151]
1155. 4 <=> ::(sK85,treg,sK86) [avatar definition]
1157. ::(sK85,treg,sK86) <- (4) [avatar component clause 1155]
1158. 3 | 4 [avatar split clause 1127,1155,1151]
1199. P(sK84,sK85) [resolution 1038,1126]
1227. ~T(sK84) | T(sK85) [resolution 790,1199]
1232. ~T(sK85) | T(sK84) [resolution 791,1199]
1405. sP1(X1,X1) | ~T(X1) | ~AB(X1) [resolution 1034,794]
1407. sP1(sK84,sK85) | ~T(sK85) [resolution 1034,1199]
1441. 9 <=> T(sK85) [avatar definition]
1442. T(sK85) <- (9) [avatar component clause 1441]
1443. ~T(sK85) <- (~9) [avatar component clause 1441]
1445. 10 <=> sP1(sK84,sK85) [avatar definition]
1447. sP1(sK84,sK85) <- (10) [avatar component clause 1445]
1448. ~9 | 10 [avatar split clause 1407,1445,1441]
1460. 13 <=> T(sK84) [avatar definition]
1462. T(sK84) <- (13) [avatar component clause 1460]
1465. 13 | ~9 [avatar split clause 1232,1441,1460]
1466. 9 | ~13 [avatar split clause 1227,1460,1441]
1473. 15 <=> AB(sK84) [avatar definition]
1475. AB(sK84) <- (15) [avatar component clause 1473]
1479. $false <- (2, 4) [subsumption resolution 1157,1148]
1480. ~2 | ~4 [avatar contradiction clause 1479]
1483. T(sK84) <- (3) [resolution 1153,1060]
1486. 13 | ~3 [avatar split clause 1483,1151,1460]
1494. TR(sK84) <- (13) [resolution 1462,1004]
1517. R(sK84) <- (13) [resolution 1494,997]
1540. AB(sK84) <- (13) [resolution 1517,998]
1541. 15 | ~13 [avatar split clause 1540,1460,1473]
1544. T(sK85) <- (4) [resolution 1157,1060]
1545. $false <- (4, ~9) [subsumption resolution 1544,1443]
1546. ~4 | 9 [avatar contradiction clause 1545]
1547. $false <- (1, 3) [subsumption resolution 1153,1145]
1548. ~1 | ~3 [avatar contradiction clause 1547]
1609. EX(sK85,sK84) <- (10) [resolution 1447,1037]
1684. ~EX(sK85,X6) | ~T(sK85) <- (2) [resolution 1062,1148]
1686. ~EX(sK85,X6) <- (2, 9) [subsumption resolution 1684,1442]
1687. $false <- (2, 9, 10) [resolution 1686,1609]
1689. ~2 | ~9 | ~10 [avatar contradiction clause 1687]
1690. ~EX(sK84,X0) | ~T(sK84) <- (1) [resolution 1145,1062]
1691. ~EX(sK84,X0) <- (1, 13) [subsumption resolution 1690,1462]
2022. EX(X3,X3) | ~AB(X3) | ~T(X3) [resolution 1405,1037]
2301. ~AB(sK84) | ~T(sK84) <- (1, 13) [resolution 2022,1691]
2302. ~T(sK84) <- (1, 13, 15) [subsumption resolution 2301,1475]
2303. $false <- (1, 13, 15) [subsumption resolution 2302,1462]
2304. ~1 | ~13 | ~15 [avatar contradiction clause 2303]
2305. $false [avatar sat refutation 1149,1158,1448,1465,1466,1480,1486,1541,1546,1548,1689,2304]
```

Proof of Theorem (t_{db} 13)

```
1 ED(A) | PD(A) | Q(A) -> (exists B PRE(A,B)). [assumption].
2 P(A,B) -> AB(A) & AB(B) | PD(A) & PD(B). [assumption].
3 AB(A) | PD(A) -> P(A,A). [assumption].
4 -(exists A (AB(A) & PD(A))). [assumption].
5 ::(A,B) <-> (exists C ::(A,B,C)). [assumption].
6 EX(A,B) <-> PRE(A,B) | T(A) & P(B,A) | AB(A) & -T(A) & T(B). [assumption].
7 oP(A,B) <-> P(A,B) & (PD(A) & PD(B) | T(A) & T(B)). [assumption].
8 tmP(A,B) <-> oP(A,B) & (all C (oP(C,B) & (all D (EX(C,D) -> EX(A,D))) -> oP(C,A))). [assumption].
9 ::(A,proc,B) <-> PD(A) & PRE(A,B) & -::(A,pbnd,B). [assumption].
10 ::(A,treg,B) <-> T(A) & EX(A,B). [assumption].
11 ::(A,occ,B) <-> ::(A,pbnd,B) | ::(A,proc,B) | ::(A,treg,B). [assumption].
12 tmP(A,B) -> ::(A,occ) & ::(B,occ). [goal].
13 -AB(A) | P(A,A). [clausify(3)].
14 -P(A,B) | AB(A) | PD(B). [clausify(2)].
15 -P(A,B) | AB(B) | PD(A). [clausify(2)].
16 -AB(A) | -PD(A). [clausify(4)].
17 -PD(A) | PRE(A,f2(A)). [clausify(1)].
18 -::(A,B) | ::(A,B,f79(A,B)). [clausify(5)].
19 ::(A,B) | -::(A,B,C). [clausify(5)].
20 EX(A,B) | -T(A) | -P(B,A). [clausify(6)].
21 -oP(A,B) | P(A,B). [clausify(7)].
22 -oP(A,B) | PD(B) | T(A). [clausify(7)].
23 -oP(A,B) | PD(B) | T(B). [clausify(7)].
24 -tmP(A,B) | oP(A,B). [clausify(8)].
25 ::(A,proc,B) | -PD(A) | -PRE(A,B) | ::(A,pbnd,B). [clausify(9)].
```

```
26 ::(A,treg,B) | -T(A) | -EX(A,B). [clausify(10)].
27 ::(A,occ,B) | -::(A,pbnd,B). [clausify(11)].
28 ::(A,occ,B) | -::(A,proc,B). [clausify(11)].
29 ::(A,occ,B) | -::(A,treg,B). [clausify(11)].
30 tmP(c1,c2). [deny(12)].
31 -::(c1,occ) | -::(c2,occ). [deny(12)].
32 P(A,A) | -P(A,B) | PD(B). [resolve(13,a,14,b)].
33 -PD(A) | -P(B,A) | PD(B). [resolve(16,a,15,b)].
34 oP(c1,c2). [resolve(30,a,24,a)].
35 PD(c2) | T(c2). [resolve(34,a,23,a)].
36 PD(c2) | T(c1). [resolve(34,a,22,a)].
37 P(c1,c2). [resolve(34,a,21,a)].
38 -PD(c2) | PD(c1). [resolve(37,a,33,b)].
39 P(c1,c1) | PD(c2). [resolve(37,a,32,b)].
40 EX(c2,c1) | -T(c2). [resolve(37,a,20,c)].
41 PD(c2) | EX(c1,c1) | -T(c1). [resolve(39,a,20,c)].
42 EX(c2,c1) | PD(c2). [resolve(40,b,35,b)].
43 PD(c2) | ::(c2,treg,c1) | -T(c2). [resolve(42,a,26,c)].
44 PD(c2) | EX(c1,c1). [resolve(41,c,36,b),merge(c)].
45 PD(c2) | ::(c1,treg,c1) | -T(c1). [resolve(44,b,26,c)].
46 PD(c2) | ::(c2,treg,c1). [resolve(43,c,35,b),merge(c)].
47 PD(c2) | ::(c2,occ,c1). [resolve(46,b,29,b)].
48 PD(c2) | ::(c2,occ). [resolve(47,b,19,b)].
49 PD(c2) | -::(c1,occ). [resolve(48,b,31,b)].
50 PD(c2) | ::(c1,treg,c1). [resolve(45,c,36,b),merge(c)].
51 PD(c2) | ::(c1,occ,c1). [resolve(50,b,29,b)].
52 PD(c2) | ::(c1,occ). [resolve(51,b,19,b)].
53 PD(c2). [resolve(52,b,49,b),merge(b)].
54 PD(c1). [back_unit_del(38),unit_del(a,53)].
55 PRE(c2,f2(c2)). [resolve(53,a,17,a)].
56 PRE(c1,f2(c1)). [resolve(54,a,17,a)].
57 ::(c2,proc,f2(c2)) | ::(c2,pbnd,f2(c2)). [resolve(55,a,25,c),unit_del(b,53)].
58 ::(c1,proc,f2(c1)) | ::(c1,pbnd,f2(c1)). [resolve(56,a,25,c),unit_del(b,54)].
59 ::(c2,pbnd,f2(c2)) | ::(c2,proc). [resolve(57,a,19,b)].
60 ::(c2,proc) | ::(c2,pbnd). [resolve(59,a,19,b)].
61 ::(c2,pbnd) | ::(c2,proc,f79(c2,proc)). [resolve(60,a,18,a)].
62 ::(c2,pbnd) | ::(c2,occ,f79(c2,proc)). [resolve(61,b,28,b)].
63 ::(c2,pbnd) | ::(c2,occ). [resolve(62,b,19,b)].
64 ::(c2,pbnd) | -::(c1,occ). [resolve(63,b,31,b)].
65 ::(c1,pbnd,f2(c1)) | ::(c1,proc). [resolve(58,a,19,b)].
66 ::(c1,proc) | ::(c1,occ,f2(c1)). [resolve(65,a,27,b)].
67 ::(c1,proc) | ::(c1,occ). [resolve(66,b,19,b)].
68 ::(c1,proc) | ::(c2,pbnd). [resolve(67,b,64,b)].
69 ::(c1,proc) | ::(c2,pbnd,f79(c2,pbnd)). [resolve(68,b,18,a)].
70 ::(c1,proc) | ::(c2,occ,f79(c2,pbnd)). [resolve(69,b,27,b)].
71 ::(c1,proc) | ::(c2,occ). [resolve(70,b,19,b)].
72 ::(c1,proc) | -::(c1,occ). [resolve(71,b,31,b)].
73 ::(c1,proc). [resolve(72,b,67,b),merge(b)].
74 ::(c1,proc,f79(c1,proc)). [resolve(73,a,18,a)].
75 ::(c1,occ,f79(c1,proc)). [resolve(74,a,28,b)].
76 ::(c1,occ). [resolve(75,a,19,b)].
77 ::(c2,pbnd). [back_unit_del(64),unit_del(b,76)].
78 -::(c2,occ). [back_unit_del(31),unit_del(a,76)].
79 ::(c2,pbnd,f79(c2,pbnd)). [resolve(77,a,18,a)].
80 ::(c2,occ,f79(c2,pbnd)). [resolve(79,a,27,b)].
81 $F. [resolve(80,a,19,b),unit_del(a,78)].
```

Proof of Theorem (t_{ab} 14)

```
6. ! [X0,X1] : (P(X0,X1) => (T(X0) <=> T(X1))) [input]
8. ! [X0] : ((PD(X0) | AB(X0)) => P(X0,X0)) [input]
124. ! [X0] : ((TR(X0) | PR(X0) | AR(X0)) <=> R(X0)) [input]
125. ! [X0] : (R(X0) => AB(X0)) [input]
130. ! [X0] : (T(X0) => TR(X0)) [input]
163. ! [X0,X1] : (oP(X0,X1) <=> (((T(X1) & T(X0)) | (PD(X1) & PD(X0))) & P(X0,X1))) [input]
164. ! [X0,X1] : (tmP(X0,X1) <=> (! [X2] : ((! [X4] : (EX(X2,X4) => EX(X0,X4)) & oP(X2,X1)) => oP(X2,X0)) & oP(X0,
     X1))) [input]
166. ! [X0,X4] : (::(X0,pbnd,X4) <=> (AT(X4) & TLC(X0,X4) & ? [X1] : tmPP(X0,X1) & PD(X0))) [input]
167. ! [X0,X4] : (::(X0,proc,X4) <=> (~::(X0,pbnd,X4) & PRE(X0,X4) & PD(X0))) [input]
168. ! [X0,X4] : (::(X0,treg,X4) <=> (EX(X0,X4) & T(X0))) [input]
169. ! [X0,X4] : (::(X0,occ,X4) <=> (::(X0,treg,X4) | ::(X0,proc,X4) | ::(X0,pbnd,X4))) [input]
189. ! [X0] : (? [X4] : ::(X0,occ,X4) => tmP(X0,X0)) [input]
190. ~! [X0] : (? [X4] : ::(X0,occ,X4) => tmP(X0,X0)) [negated conjecture 189]
305. ! [X0,X1] : (tmP(X0,X1) <=> (! [X2] : ((! [X3] : (EX(X2,X3) => EX(X0,X3)) & oP(X2,X1)) => oP(X2,X0)) & oP(X0,
     X1))) [rectify 164]
307. ! [X0,X1] : (::(X0,pbnd,X1) <=> (AT(X1) & TLC(X0,X1) & ? [X2] : tmPP(X0,X2) & PD(X0))) [rectify 166]
308. ! [X0,X1] : (::(X0,proc,X1) <=> (~::(X0,pbnd,X1) & PRE(X0,X1) & PD(X0))) [rectify 167]
309. ! [X0,X1] : (::(X0,treg,X1) <=> (EX(X0,X1) & T(X0))) [rectify 168]
310. ! [X0,X1] : (::(X0,occ,X1) <=> (::(X0,treg,X1) | ::(X0,proc,X1) | ::(X0,pbnd,X1))) [rectify 169]
335. ~! [X0] : (? [X1] : ::(X0,occ,X1) => tmP(X0,X0)) [rectify 190]
336. ! [X0] : ((TR(X0) | PR(X0) | AR(X0)) => R(X0)) [unused predicate definition removal 124]
372. ! [X0,X1] : ((T(X0) <=> T(X1)) | ~P(X0,X1)) [ennf transformation 6]
374. ! [X0] : (P(X0,X0) | (~PD(X0) & ~AB(X0))) [ennf transformation 8]
533. ! [X0] : (R(X0) | (~TR(X0) & ~PR(X0) & ~AR(X0))) [ennf transformation 336]
534. ! [X0] : (AB(X0) | ~R(X0)) [ennf transformation 125]
538. ! [X0] : (TR(X0) | ~T(X0)) [ennf transformation 130]
558. ! [X0,X1] : (tmP(X0,X1) <=> (! [X2] : (oP(X2,X0) | (? [X3] : (~EX(X0,X3) & EX(X2,X3)) | ~oP(X2,X1))) & oP(X0,
     X1))) [ennf transformation 305]
559. ! [X0,X1] : (tmP(X0,X1) <=> (! [X2] : (oP(X2,X0) | ? [X3] : (~EX(X0,X3) & EX(X2,X3)) | ~oP(X2,X1)) & oP(X0,X1
     ))) [flattening 558]
```

```
564. ? [X0] : (~tmP(X0,X0) & ? [X1] : ::(X0,occ,X1)) [ennf transformation 335]
588. ! [X0,X1] : (((T(X0) | ~T(X1)) & (T(X1) | ~T(X0))) | ~P(X0,X1)) [nnf transformation 372]
702. ! [X0,X1] : ((oP(X0,X1) | (((~T(X1) | ~T(X0)) & (~PD(X1) | ~PD(X0))) | ~P(X0,X1))) & ((((T(X1) & T(X0)) | (PD
     (X1) & PD(X0))) & P(X0,X1)) | ~oP(X0,X1))) [nnf transformation 163]
703. ! [X0,X1] : ((oP(X0,X1) | ((~T(X1) | ~T(X0)) & (~PD(X1) | ~PD(X0)))) | ~P(X0,X1)) & ((((T(X1) & T(X0)) | (PD(
     X1) & PD(X0))) & P(X0,X1)) | ~oP(X0,X1))) [flattening 702]
704. ! [X0,X1] : ((tmP(X0,X1) | (? [X2] : (~oP(X2,X0) & ! [X3] : (EX(X0,X3) | ~EX(X2,X3)) & oP(X2,X1)) | ~oP(X0,X1
     ))) & ((! [X2] : (oP(X2,X0) | ? [X3] : (~EX(X0,X3) & EX(X2,X3)) | ~oP(X2,X1)) & oP(X0,X1)) | ~tmP(X0,X1))) [
     nnf transformation 559]
705. ! [X0,X1] : ((tmP(X0,X1) | ? [X2] : (~oP(X2,X0) & ! [X3] : (EX(X0,X3) | ~EX(X2,X3)) & oP(X2,X1)) | ~oP(X0,X1)
     ) & ((! [X2] : (oP(X2,X0) | ? [X3] : (~EX(X0,X3) & EX(X2,X3)) | ~oP(X2,X1)) & oP(X0,X1)) | ~tmP(X0,X1))) [
     flattening 704]
706. ! [X0,X1] : ((tmP(X0,X1) | ? [X2] : (~oP(X2,X0) & ! [X3] : (EX(X0,X3) | ~EX(X2,X3)) & oP(X2,X1)) | ~oP(X0,X1)
     ) & ((! [X4] : (oP(X4,X0) | ? [X5] : (~EX(X0,X5) & EX(X4,X5)) | ~oP(X4,X1)) & oP(X0,X1)) | ~tmP(X0,X1))) [
     rectify 705]
707. ! [X1,X0] : (? [X2] : (~oP(X2,X0) & ! [X3] : (EX(X0,X3) | ~EX(X2,X3)) & oP(X2,X1)) => (~oP(sK67(X0,X1),X0) &
     ! [X3] : (EX(X0,X3) | ~EX(sK67(X0,X1),X3)) & oP(sK67(X0,X1),X1))) [choice axiom]
708. ! [X4,X0] : (? [X5] : (~EX(X0,X5) & EX(X4,X5)) => (~EX(X0,sK68(X0,X4)) & EX(X4,sK68(X0,X4)))) [choice axiom]
709. ! [X0,X1] : ((tmP(X0,X1) | (~oP(sK67(X0,X1),X0) & ! [X3] : (EX(X0,X3) | ~EX(sK67(X0,X1),X3)) & oP(sK67(X0,X1)
     ,X1)) | ~oP(X0,X1)) & ((! [X4] : (oP(X4,X0) | (~EX(X0,sK68(X0,X4)) & EX(X4,sK68(X0,X4)))) | ~oP(X4,X1)) & oP(
     X0,X1)) | ~tmP(X0,X1))) [skolemisation 706,708,707]
710. ! [X0,X1] : ((::(X0,pbnd,X1) | (~AT(X1) | ~TLC(X0,X1) | ! [X2] : ~tmPP(X0,X2) | ~PD(X0))) & ((AT(X1) & TLC(X0
     ,X1) & ? [X2] : tmPP(X0,X2) & PD(X0)) | ~::(X0,pbnd,X1))) [nnf transformation 307]
711. ! [X0,X1] : ((::(X0,pbnd,X1) | ~AT(X1) | ~TLC(X0,X1) | ! [X2] : ~tmPP(X0,X2) | ~PD(X0)) & ((AT(X1) & TLC(X0,
     X1) & ? [X2] : tmPP(X0,X2) & PD(X0)) | ~::(X0,pbnd,X1))) [flattening 710]
712. ! [X0,X1] : ((::(X0,pbnd,X1) | ~AT(X1) | ~TLC(X0,X1) | ! [X2] : ~tmPP(X0,X2) | ~PD(X0)) & ((AT(X1) & TLC(X0,
     X1) & ? [X3] : tmPP(X0,X3) & PD(X0)) | ~::(X0,pbnd,X1))) [rectify 711]
713. ! [X0] : (? [X3] : tmPP(X0,X3) => tmPP(X0,sK69(X0))) [choice axiom]
714. ! [X0,X1] : ((::(X0,pbnd,X1) | ~AT(X1) | ~TLC(X0,X1) | ! [X2] : ~tmPP(X0,X2) | ~PD(X0)) & ((AT(X1) & TLC(X0,
     X1) & tmPP(X0,sK69(X0)) & PD(X0)) | ~::(X0,pbnd,X1))) [skolemisation 712,713]
715. ! [X0,X1] : ((::(X0,proc,X1) | (::(X0,pbnd,X1) | ~PRE(X0,X1) | ~PD(X0))) & ((~::(X0,pbnd,X1) & PRE(X0,X1) &
     PD(X0)) | ~::(X0,proc,X1))) [nnf transformation 308]
716. ! [X0,X1] : ((::(X0,proc,X1) | ::(X0,pbnd,X1) | ~PRE(X0,X1) | ~PD(X0)) & ((~::(X0,pbnd,X1) & PRE(X0,X1) & PD(
     X0)) | ~::(X0,proc,X1))) [flattening 715]
717. ! [X0,X1] : ((::(X0,treg,X1) | (~EX(X0,X1) | ~T(X0))) & ((EX(X0,X1) & T(X0)) | ~::(X0,treg,X1))) [nnf
     transformation 309]
718. ! [X0,X1] : ((::(X0,treg,X1) | ~EX(X0,X1) | ~T(X0)) & ((EX(X0,X1) & T(X0)) | ~::(X0,treg,X1))) [flattening
     717]
719. ! [X0,X1] : ((::(X0,occ,X1) | (~::(X0,treg,X1) & ~::(X0,proc,X1) & ~::(X0,pbnd,X1))) & ((::(X0,treg,X1) | ::(
     X0,proc,X1) | ::(X0,pbnd,X1)) | ~::(X0,occ,X1))) [nnf transformation 310]
720. ! [X0,X1] : ((::(X0,occ,X1) | (~::(X0,treg,X1) & ~::(X0,proc,X1) & ~::(X0,pbnd,X1))) & (::(X0,treg,X1) | ::(
     X0,proc,X1) | ::(X0,pbnd,X1) | ~::(X0,occ,X1))) [flattening 719]
766. ? [X0] : (~tmP(X0,X0) & ? [X1] : ::(X0,occ,X1)) => (~tmP(sK84,sK84) & ? [X1] : ::(sK84,occ,X1)) [choice axiom
     ]
767. ? [X1] : ::(sK84,occ,X1) => ::(sK84,occ,sK85) [choice axiom]
768. ~tmP(sK84,sK84) & ::(sK84,occ,sK85) [skolemisation 564,767,766]
787. ~P(X0,X1) | ~T(X1) | T(X0) [cnf transformation 588]
790. P(X0,X0) | ~AB(X0) [cnf transformation 374]
791. P(X0,X0) | ~PD(X0) [cnf transformation 374]
993. ~TR(X0) | R(X0) [cnf transformation 533]
994. ~R(X0) | AB(X0) [cnf transformation 534]
1000. ~T(X0) | TR(X0) [cnf transformation 538]
1039. ~P(X0,X1) | ~PD(X1) | ~PD(X0) | oP(X0,X1) [cnf transformation 703]
1040. oP(X0,X1) | ~T(X1) | ~T(X0) | ~P(X0,X1) [cnf transformation 703]
1044. oP(sK67(X0,X1),X1) | tmP(X0,X1) | ~oP(X0,X1) [cnf transformation 709]
1046. ~oP(sK67(X0,X1),X0) | tmP(X0,X1) | ~oP(X0,X1) [cnf transformation 709]
1047. ~::(X0,pbnd,X1) | PD(X0) [cnf transformation 714]
1052. ~::(X0,proc,X1) | PD(X0) [cnf transformation 716]
1056. ~::(X0,treg,X1) | T(X0) [cnf transformation 718]
1059. ~::(X0,occ,X1) | ::(X0,proc,X1) | ::(X0,pbnd,X1) | ::(X0,treg,X1) [cnf transformation 720]
1122. ::(sK84,occ,sK85) [cnf transformation 768]
1123. ~tmP(sK84,sK84) [cnf transformation 768]
1131. ~P(X0,X1) | ~T(X1) | oP(X0,X1) [subsumption resolution 1040,787]
1367. oP(X1,X1) | ~T(X1) | ~AB(X1) [resolution 1131,790]
1792. ~PD(X0) | ~PD(X0) | oP(X0,X0) | ~PD(X0) [resolution 1039,791]
1808. oP(X0,X0) | ~PD(X0) [duplicate literal removal 1792]
1973. tmP(X0,X0) | ~oP(X0,X0) | tmP(X0,X0) | ~oP(X0,X0) [resolution 1046,1044]
1974. ~oP(X0,X0) | tmP(X0,X0) [duplicate literal removal 1973]
1975. tmP(X0,X0) | ~PD(X0) [resolution 1974,1808]
1976. tmP(X1,X1) | ~T(X1) | ~AB(X1) [resolution 1974,1367]
1981. ~PD(sK84) [resolution 1975,1123]
2037. ~T(sK84) | ~AB(sK84) [resolution 1976,1123]
2041. 1 <=> AB(sK84) [avatar definition]
2043. ~AB(sK84) <- (~1) [avatar component clause 2041]
2045. 2 <=> T(sK84) [avatar definition]
2046. T(sK84) <- (2) [avatar component clause 2045]
2048. ~1 | ~2 [avatar split clause 2037,2045,2041]
2647. ::(sK84,proc,sK85) | ::(sK84,pbnd,sK85) | ::(sK84,treg,sK85) [resolution 1059,1122]
2649. 3 <=> ::(sK84,treg,sK85) [avatar definition]
2651. ::(sK84,treg,sK85) <- (3) [avatar component clause 2649]
2653. 4 <=> ::(sK84,pbnd,sK85) [avatar definition]
2655. ::(sK84,pbnd,sK85) <- (4) [avatar component clause 2653]
2657. 5 <=> ::(sK84,proc,sK85) [avatar definition]
2659. ::(sK84,proc,sK85) <- (5) [avatar component clause 2657]
2660. 3 | 4 | 5 [avatar split clause 2647,2657,2653,2649]
2665. T(sK84) <- (3) [resolution 2651,1056]
2680. 2 | ~3 [avatar split clause 2665,2649,2045]
2697. PD(sK84) <- (4) [resolution 2655,1047]
2698. $false <- (4) [subsumption resolution 2697,1981]
2699. ~4 [avatar contradiction clause 2698]
2742. PD(sK84) <- (5) [resolution 2659,1052]
```

```
2743. $false <- (5) [subsumption resolution 2742,1981]
2744. ~5 [avatar contradiction clause 2743]
2750. TR(sK84) <- (2) [resolution 2046,1000]
2752. R(sK84) <- (2) [resolution 2750,993]
2753. AB(sK84) <- (2) [resolution 2752,994]
2754. $false <- (~1, 2) [subsumption resolution 2753,2043]
2755. 1 | ~2 [avatar contradiction clause 2754]
2756. $false [avatar sat refutation 2048,2660,2680,2699,2744,2755]
```

Proof of Theorem (t_{db}15)

```
9. ! [X0,X1] : ((P(X1,X0) & P(X0,X1)) => X0 = X1) [input]
163. ! [X0,X1] : (oP(X0,X1) <=> (((T(X1) & T(X0)) | (PD(X1) & PD(X0))) & P(X0,X1))) [input]
164. ! [X0,X1] : (tmP(X0,X1) <=> (! [X2] : ((! [X4] : (EX(X2,X4) => EX(X0,X4)) & oP(X2,X1)) => oP(X2,X0)) & oP(X0,
     X1))) [input]
189. ! [X0,X1] : ((tmP(X1,X0) & tmP(X0,X1)) => X0 = X1) [input]
190. ~! [X0,X1] : ((tmP(X1,X0) & tmP(X0,X1)) => X0 = X1) [negated conjecture 189]
305. ! [X0,X1] : (tmP(X0,X1) <=> (! [X2] : ((! [X3] : (EX(X2,X3) => EX(X0,X3)) & oP(X2,X1)) => oP(X2,X0)) & oP(X0,
     X1))) [rectify 164]
374. ! [X0,X1] : (X0 = X1 | (~P(X1,X0) | ~P(X0,X1))) [ennf transformation 9]
375. ! [X0,X1] : (X0 = X1 | ~P(X1,X0) | ~P(X0,X1)) [flattening 374]
557. ! [X0,X1] : (tmP(X0,X1) <=> (! [X2] : (oP(X2,X0) | (? [X3] : (~EX(X0,X3) & EX(X2,X3)) | ~oP(X2,X1))) & oP(X0,
     X1))) [ennf transformation 305]
558. ! [X0,X1] : (tmP(X0,X1) <=> (! [X2] : (oP(X2,X0) | ? [X3] : (~EX(X0,X3) & EX(X2,X3)) | ~oP(X2,X1)) & oP(X0,X1
     ))) [flattening 557]
563. ? [X0,X1] : (X0 != X1 & (tmP(X1,X0) & tmP(X0,X1))) [ennf transformation 190]
564. ? [X0,X1] : (X0 != X1 & tmP(X1,X0) & tmP(X0,X1)) [flattening 563]
702. ! [X0,X1] : ((oP(X0,X1) | (((~T(X1) | ~T(X0)) & (~PD(X1) | ~PD(X0))) | ~P(X0,X1))) & ((((T(X1) & T(X0)) | (PD
     (X1) & PD(X0))) & P(X0,X1)) | ~oP(X0,X1))) [nnf transformation 163]
703. ! [X0,X1] : ((oP(X0,X1) | (((~T(X1) | ~T(X0)) & (~PD(X1) | ~PD(X0))) | ~P(X0,X1)) & ((((T(X1) & T(X0)) | (PD(
     X1) & PD(X0))) & P(X0,X1)) | ~oP(X0,X1))) [flattening 702]
704. ! [X0,X1] : ((tmP(X0,X1) | (? [X2] : (~oP(X2,X0) & ! [X3] : (EX(X0,X3) | ~EX(X2,X3)) & oP(X2,X1)) | ~oP(X0,X1
     ))) & ((! [X2] : (oP(X2,X0) | ? [X3] : (~EX(X0,X3) & EX(X2,X3)) | ~oP(X2,X1)) & oP(X0,X1)) | ~tmP(X0,X1))) [
     nnf transformation 558]
705. ! [X0,X1] : ((tmP(X0,X1) | ? [X2] : (~oP(X2,X0) & ! [X3] : (EX(X0,X3) | ~EX(X2,X3)) & oP(X2,X1)) | ~oP(X0,X1)
     ) & ((! [X2] : (oP(X2,X0) | ? [X3] : (~EX(X0,X3) & EX(X2,X3)) | ~oP(X2,X1)) & oP(X0,X1)) | ~tmP(X0,X1))) [
     flattening 704]
706. ! [X0,X1] : ((tmP(X0,X1) | ? [X2] : (~oP(X2,X0) & ! [X3] : (EX(X0,X3) | ~EX(X2,X3)) & oP(X2,X1)) | ~oP(X0,X1)
     ) & ((! [X4] : (oP(X4,X0) | ? [X5] : (~EX(X0,X5) & EX(X4,X5)) | ~oP(X4,X1)) & oP(X0,X1)) | ~tmP(X0,X1))) [
     rectify 705]
707. ! [X1,X0] : (? [X2] : (~oP(X2,X0) & ! [X3] : (EX(X0,X3) | ~EX(X2,X3)) & oP(X2,X1)) => (~oP(sK67(X0,X1),X0) &
     ! [X3] : (EX(X0,X3) | ~EX(sK67(X0,X1),X3)) & oP(sK67(X0,X1),X1))) [choice axiom]
708. ! [X4,X0] : (? [X5] : (~EX(X0,X5) & EX(X4,X5)) => (~EX(X0,sK68(X0,X4)) & EX(X4,sK68(X0,X4)))) [choice axiom]
709. ! [X0,X1] : ((tmP(X0,X1) | (~oP(sK67(X0,X1),X0) & ! [X3] : (EX(X0,X3) | ~EX(sK67(X0,X1),X3)) & oP(sK67(X0,X1)
     ,X1)) | ~oP(X0,X1)) & ((! [X4] : (oP(X4,X0) | (~EX(X0,sK68(X0,X4)) & EX(X4,sK68(X0,X4)))) | ~oP(X4,X1)) & oP(
     X0,X1)) | ~tmP(X0,X1))) [skolemisation 706,708,707]
766. ? [X0,X1] : (X0 != X1 & tmP(X1,X0) & tmP(X0,X1)) => (sK84 != sK85 & tmP(sK85,sK84) & tmP(sK84,sK85)) [choice
     axiom]
767. sK84 != sK85 & tmP(sK85,sK84) & tmP(sK84,sK85) [skolemisation 564,766]
791. ~P(X1,X0) | X0 = X1 | ~P(X0,X1) [cnf transformation 375]
1033. ~oP(X0,X1) | P(X0,X1) [cnf transformation 703]
1040. ~tmP(X0,X1) | oP(X0,X1) [cnf transformation 709]
1121. tmP(sK84,sK85) [cnf transformation 767]
1122. tmP(sK85,sK84) [cnf transformation 767]
1123. sK84 != sK85 [cnf transformation 767]
1179. oP(sK84,sK85) [resolution 1040,1121]
1180. oP(sK85,sK84) [resolution 1040,1122]
1181. P(sK84,sK85) [resolution 1179,1033]
1182. P(sK85,sK84) [resolution 1180,1033]
1475. sK84 = sK85 | ~P(sK85,sK84) [resolution 791,1181]
1477. ~P(sK85,sK84) [subsumption resolution 1475,1123]
1478. $false [subsumption resolution 1477,1182]
```

Proof of Theorem (t_{db}16) Prover9 struggles with this theorem. In order to simplify the proof process we substitute "tmP(X,Z)" with its definiens, and we add to DOLCE axioms the theorems (t_{db}4) and (t_{db}6).

```
1 P(A,B) -> AB(A) & AB(B) | PD(A) & PD(B). [assumption].
2 P(A,B) -> (T(A) <-> T(B)). [assumption].
3 AB(A) | PD(A) -> P(A,A). [assumption].
4 PRE(A,B) -> (ED(A) | PD(A) | Q(A)) & T(B). [assumption].
5 AR(A) | PR(A) | TR(A) <-> R(A). [assumption].
6 R(A) -> AB(A). [assumption].
7 T(A) -> TR(A). [assumption].
8 -(exists A (AB(A) & ED(A))). [assumption].
9 -(exists A (AB(A) & PD(A))). [assumption].
10 -(exists A (AB(A) & Q(A))). [assumption].
11 EX(A,B) <-> PRE(A,B) | T(A) & P(B,A) | AB(A) & -T(A) & T(B). [assumption].
12 oP(A,B) <-> P(A,B) & (PD(A) & PD(B) | T(A) & T(B)). [assumption].
13 tmP(A,B) <-> oP(A,B) & (all C (oP(C,B) & (all D (EX(C,D) -> EX(A,D))) -> oP(C,A))). [assumption].
14 oP(A,B) & PRE(A,C) -> PRE(B,C). [assumption].
15 oP(A,B) & oP(B,C) -> oP(A,C). [assumption].
16 oP(A,B) & EX(A,C) -> EX(B,C). [assumption].
17 tmP(A,B) & tmP(B,C) -> oP(A,C) & (all D (oP(D,C) & (all E (EX(D,E) -> EX(A,E))) -> oP(D,A))). [goal].
18 -AB(A) | P(A,A). [clausify(3)].
19 -P(A,B) | AB(A) | PD(B). [clausify(1)].
20 -P(A,B) | AB(B) | PD(A). [clausify(1)].
```

```
21 -R(A) | AB(A). [clausify(6)].
22 -AB(A) | -ED(A). [clausify(8)].
23 -AB(A) | -PD(A). [clausify(9)].
24 -AB(A) | -Q(A). [clausify(10)].
25 -PRE(A,B) | ED(A) | PD(A) | Q(A). [clausify(4)].
26 -ED(A) | -R(A). [resolve(22,a,21,b)].
27 -Q(A) | -R(A). [resolve(24,a,21,b)].
28 -R(A) | -PRE(A,B) | PD(A) | Q(A). [resolve(26,a,25,b)].
29 -T(A) | TR(A). [clausify(7)].
30 -TR(A) | R(A). [clausify(5)].
31 -tmP(A,B) | oP(A,B). [clausify(13)].
32 -tmP(A,B) | -oP(C,B) | EX(C,f6(B,A,C)) | oP(C,A). [clausify(13)].
33 -tmP(A,B) | -oP(C,B) | -EX(A,f6(B,A,C)) | oP(C,A). [clausify(13)].
34 tmP(c3,c1). [deny(17)].
35 tmP(c1,c2). [deny(17)].
36 -P(A,B) | T(A) | -T(B). [clausify(2)].
37 -EX(A,B) | PRE(A,B) | P(B,A) | -T(A). [clausify(11)].
38 EX(A,B) | -T(A) | -P(B,A). [clausify(11)].
39 -oP(A,B) | P(A,B). [clausify(12)].
40 -oP(A,B) | PD(B) | T(A). [clausify(12)].
41 -oP(A,B) | PD(B) | T(B). [clausify(12)].
42 oP(A,B) | -P(A,B) | -PD(A) | -PD(B). [clausify(12)].
43 oP(A,B) | -P(A,B) | -T(A) | -T(B). [clausify(12)].
44 -oP(A,B) | -PRE(A,C) | PRE(B,C). [clausify(14)].
45 -oP(A,B) | -oP(B,C) | oP(A,C). [clausify(15)].
46 -oP(A,B) | -EX(A,C) | EX(B,C). [clausify(16)].
47 -oP(c3,c2) | oP(c4,c2). [deny(17)].
48 -oP(c3,c2) | -EX(c4,A) | EX(c3,A). [deny(17)].
49 -oP(c3,c2) | -oP(c4,c3). [deny(17)].
50 P(A,A) | -P(A,B) | PD(B). [resolve(18,a,19,b)].
51 -PD(A) | -P(B,A) | PD(B). [resolve(23,a,20,b)].
52 -R(A) | -PRE(A,B) | PD(A) | -R(A). [resolve(28,d,27,a)].
53 -R(A) | -PRE(A,B) | PD(A). [copy(52),merge(d)].
54 -T(A) | R(A). [resolve(29,b,30,a)].
55 oP(c3,c1). [resolve(34,a,31,a)].
56 -oP(A,c1) | EX(A,f6(c1,c3,A)) | oP(A,c3). [resolve(34,a,32,a)].
57 -oP(A,c1) | -EX(c3,f6(c1,c3,A)) | oP(A,c3). [resolve(34,a,33,a)].
58 oP(c1,c2). [resolve(35,a,31,a)].
59 -oP(A,c2) | EX(A,f6(c2,c1,A)) | oP(A,c1). [resolve(35,a,32,a)].
60 -oP(A,c2) | -EX(c1,f6(c2,c1,A)) | oP(A,c1). [resolve(35,a,33,a)].
61 -EX(c3,A) | EX(c1,A). [resolve(55,a,46,a)].
62 -oP(c1,A) | oP(c3,A). [resolve(55,a,45,a)].
63 P(c3,c1). [resolve(55,a,39,a)].
64 PD(c2) | T(c2). [resolve(58,a,41,a)].
65 PD(c2) | T(c1). [resolve(58,a,40,a)].
66 P(c1,c2). [resolve(58,a,39,a)].
67 T(c3) | -T(c1). [resolve(63,a,36,a)].
68 -PD(c2) | PD(c1). [resolve(66,a,51,b)].
69 PD(c2) | R(c2). [resolve(64,b,54,a)].
70 T(c3) | PD(c2). [resolve(67,b,65,b)].
71 oP(c3,c2). [resolve(62,a,58,a)].
72 -oP(c4,c3). [resolve(71,a,49,a)].
73 -EX(c4,A) | EX(c3,A). [resolve(71,a,48,a)].
74 oP(c4,c2). [resolve(71,a,47,a)].
75 -PRE(c3,A) | PRE(c2,A). [resolve(71,a,44,a)].
76 -EX(c1,f6(c2,c1,c4)) | oP(c4,c1). [resolve(74,a,60,a)].
77 EX(c4,f6(c2,c1,c4)) | oP(c4,c1). [resolve(74,a,59,a)].
78 PD(c2) | T(c4). [resolve(74,a,40,a)].
79 P(c4,c2). [resolve(74,a,39,a)].
80 -PD(c2) | PD(c4). [resolve(79,a,51,b)].
81 P(c4,c4) | PD(c2). [resolve(79,a,50,b)].
82 PD(c2) | EX(c4,c4) | -T(c4). [resolve(81,a,38,c)].
83A PD(c2) | EX(c4,c4) | PD(c2). [resolve(82,c,78,b)].
83 PD(c2) | EX(c4,c4). [copy(83A),merge(c)].
84 PD(c2) | EX(c3,c4). [resolve(83,b,73,a)].
85 PD(c2) | PRE(c3,c4) | P(c4,c3) | -T(c3). [resolve(84,b,37,a)].
86 EX(c4,f6(c2,c1,c4)) | P(c4,c1). [resolve(77,b,39,a)].
87 P(c4,c1) | EX(c3,f6(c2,c1,c4)). [resolve(86,a,73,a)].
88 P(c4,c1) | EX(c1,f6(c2,c1,c4)). [resolve(87,b,61,a)].
89 P(c4,c1) | oP(c4,c1). [resolve(88,b,76,a)].
90A P(c4,c1) | P(c4,c1). [resolve(89,b,39,a)].
90 P(c4,c1). [copy(90A),merge(b)].
91 oP(c4,c1) | -PD(c4) | -PD(c1). [resolve(90,a,42,b)].
92A PD(c2) | PRE(c3,c4) | P(c4,c3) | PD(c2). [resolve(85,d,70,a)].
92 PD(c2) | PRE(c3,c4) | P(c4,c3). [copy(92A),merge(d)].
93 PD(c2) | P(c4,c3) | PRE(c2,c4). [resolve(92,b,75,a)].
94A PD(c2) | P(c4,c3) | -R(c2) | PD(c2). [resolve(93,c,53,b)].
94 PD(c2) | P(c4,c3) | -R(c2). [copy(94A),merge(d)].
95A PD(c2) | P(c4,c3) | PD(c2). [resolve(94,c,69,b)].
95 PD(c2) | P(c4,c3). [copy(95A),merge(c)].
96A PD(c2) | oP(c4,c3) | -T(c4) | -T(c3). [resolve(95,b,43,b)].
96 PD(c2) | -T(c4) | -T(c3). [resolve(72,a,96A,b)].
97A PD(c2) | -T(c3) | PD(c2). [resolve(96,b,78,b)].
97 PD(c2) | -T(c3). [copy(97A),merge(c)].
98A PD(c2) | PD(c2). [resolve(97,b,70,a)].
98 PD(c2). [copy(98A),merge(b)].
99 PD(c4). [resolve(98,a,80,a)].
100 PD(c1). [resolve(98,a,68,a)].
101A oP(c4,c1) | -PD(c1). [resolve(99,a,91,b)].
101 oP(c4,c1). [resolve(100,a,101A,b)].
102A -EX(c3,f6(c1,c3,c4)) | oP(c4,c3). [resolve(101,a,57,a)].
```

```
102 -EX(c3,f6(c1,c3,c4)).  [resolve(72,a,102A,b)].
103A EX(c4,f6(c1,c3,c4)) | oP(c4,c3).  [resolve(101,a,56,a)].
103 EX(c4,f6(c1,c3,c4)).  [resolve(72,a,103A,b)].
104A EX(c3,f6(c1,c3,c4)).  [resolve(103,a,73,a)].
104 $F.  [resolve(102,a,104A,a)].
```

Proof of Theorem (t_{db}17)

```
1 AB(A) | PD(A) -> P(A,A).  [assumption].
2 PRE(A,B) -> (ED(A) | PD(A) | Q(A)) & T(B).  [assumption].
3 AR(A) | PR(A) | TR(A) <-> R(A).  [assumption].
4 R(A) -> AB(A).  [assumption].
5 T(A) -> TR(A).  [assumption].
6 -(exists A (AB(A) & ED(A))).  [assumption].
7 -(exists A (AB(A) & PD(A))).  [assumption].
8 -(exists A (AB(A) & Q(A))).  [assumption].
9 EX(A,B) <-> PRE(A,B) | T(A) & P(B,A) | AB(A) & -T(A) & T(B).  [assumption].
10 oP(A,B) <-> P(A,B) & (PD(A) & PD(B) | T(A) & T(B)).  [assumption].
11 tmP(A,B) <-> oP(A,B) & (all C (oP(C,B) & (all D (EX(C,D) -> EX(A,D))) -> oP(C,A))).  [assumption].
12 ::(A,treg,B) <-> T(A) & EX(A,B).  [assumption].
13 (exists A ::(B,treg,A)) & (exists A ::(C,treg,A)) -> (tmP(B,C) <-> oP(B,C)).  [goal].
14 -AB(A) | P(A,A).  [clausify(1)].
15 -R(A) | AB(A).  [clausify(4)].
16 -AB(A) | -ED(A).  [clausify(6)].
17 -AB(A) | -PD(A).  [clausify(7)].
18 -AB(A) | -Q(A).  [clausify(8)].
19 -TR(A) | R(A).  [clausify(3)].
20 -T(A) | TR(A).  [clausify(5)].
21 -PRE(A,B) | ED(A) | PD(A) | Q(A).  [clausify(2)].
22 -EX(A,B) | PRE(A,B) | P(B,A) | -T(A).  [clausify(9)].
23 EX(A,B) | -T(A) | -P(B,A).  [clausify(9)].
24 -oP(A,B) | PD(B) | T(A).  [clausify(10)].
25 oP(A,B) | -P(A,B) | -T(A) | -T(B).  [clausify(10)].
26 -tmP(A,B) | oP(A,B).  [clausify(11)].
27 tmP(A,B) | -oP(A,B) | oP(f79(B,A),B).  [clausify(11)].
28 tmP(A,B) | -oP(A,B) | -EX(f79(B,A),C) | EX(A,C).  [clausify(11)].
29 tmP(A,B) | -oP(A,B) | -oP(f79(B,A),A).  [clausify(11)].
30 -::(A,treg,B) | T(A).  [clausify(12)].
31 ::(c1,treg,c3).  [deny(13)].
32 ::(c2,treg,c4).  [deny(13)].
33 tmP(c1,c2) | oP(c1,c2).  [deny(13)].
34 -tmP(c1,c2) | -oP(c1,c2).  [deny(13)].
35 -R(A) | P(A,A).  [resolve(15,b,14,a)].
36 -ED(A) | -R(A).  [resolve(16,a,15,b)].
37 -PD(A) | -R(A).  [resolve(17,a,15,b)].
38 -Q(A) | -R(A).  [resolve(18,a,15,b)].
39 -T(A) | R(A).  [resolve(20,b,19,a)].
40 T(c1).  [resolve(31,a,30,a)].
41 T(c2).  [resolve(32,a,30,a)].
42 oP(c1,c2).  [resolve(33,a,26,a),merge(b)].
43 -tmP(c1,c2).  [back_unit_del(34),unit_del(b,42)].
44 R(c1).  [resolve(40,a,39,a)].
45 R(c2).  [resolve(41,a,39,a)].
46 -Q(c1).  [resolve(44,a,38,b)].
47 -PD(c1).  [resolve(44,a,37,b)].
48 -ED(c1).  [resolve(44,a,36,b)].
49 -PD(c2).  [resolve(45,a,37,b)].
50 -EX(f79(c2,c1),A) | EX(c1,A).  [resolve(42,a,28,b),unit_del(a,43)].
51 oP(f79(c2,c1),c2).  [resolve(42,a,27,b),unit_del(a,43)].
52 T(f79(c2,c1)).  [resolve(51,a,24,a),unit_del(a,49)].
53 R(f79(c2,c1)).  [resolve(52,a,39,a)].
54 P(f79(c2,c1),f79(c2,c1)).  [resolve(53,a,35,a)].
55 EX(f79(c2,c1),f79(c2,c1)).  [resolve(54,a,23,c),unit_del(b,52)].
56 EX(c1,f79(c2,c1)).  [resolve(55,a,50,a)].
57 PRE(c1,f79(c2,c1)) | P(f79(c2,c1),c1).  [resolve(56,a,22,a),unit_del(c,40)].
58 P(f79(c2,c1),c1).  [resolve(57,a,21,a),unit_del(b,48),unit_del(c,47),unit_del(d,46)].
59 oP(f79(c2,c1),c1).  [resolve(58,a,25,b),unit_del(b,52),unit_del(c,40)].
60 $F.  [resolve(59,a,29,c),unit_del(a,43),unit_del(b,42)].
```

Proof of Theorem (t_{db}18)

```
2. ! [X0,X1] : (O(X0,X1) <=> ? [X2] : (P(X2,X1) & P(X2,X0)))  [input]
8. ! [X0] : ((PD(X0) | AB(X0)) => P(X0,X0))  [input]
124. ! [X0] : ((TR(X0) | PR(X0) | AR(X0)) <=> R(X0))  [input]
125. ! [X0] : (R(X0) => AB(X0))  [input]
130. ! [X0] : (T(X0) => TR(X0))  [input]
133. ~? [X0] : (PD(X0) & AB(X0))  [input]
161. ! [X0] : (tmPP(X0,X1) <=> (X0 != X1 & tmP(X0,X1)))  [input]
163. ! [X0,X4] : (EX(X0,X4) <=> ((T(X4) & ~T(X0) & AB(X0)) | (P(X4,X0) & T(X0)) | PRE(X0,X4)))  [input]
164. ! [X0,X1] : (oP(X0,X1) <=> (((T(X1) & T(X0)) | (PD(X1) & PD(X0))) & P(X0,X1)))  [input]
165. ! [X0,X1] : (tmP(X0,X1) <=> (! [X2] : ((! [X4] : (EX(X2,X4) => EX(X0,X4)) & oP(X2,X1)) => oP(X2,X0)) & oP(X0,
     X1)))  [input]
169. ! [X0,X4] : (::(X0,treg,X4) <=> (EX(X0,X4) & T(X0)))  [input]
191. ! [X0,X1] : (~O(X0,X1) => ~tmO(X0,X1))  [input]
192. ! [X0,X1] : ((? [X4] : ::(X1,treg,X4) & ? [X4] : ::(X0,treg,X4)) => (tmP(X0,X1) <=> oP(X0,X1)))  [input]
193. ! [X0,X1] : ((tmPP(X0,X1) & ::(X1,treg,X1) & ::(X0,treg,X0)) => ? [X2] : (~O(X2,X0) & oP(X2,X1)))  [input]
194. ! [X0,X1] : ((tmPP(X0,X1) & ::(X1,treg,X1) & ::(X0,treg,X0)) => ? [X2] : (~tmO(X2,X0) & tmPP(X2,X1)))  [input]
195. ~! [X0,X1] : ((tmPP(X0,X1) & ::(X1,treg,X1) & ::(X0,treg,X0)) => ? [X2] : (~tmO(X2,X0) & tmPP(X2,X1)))  [
     negated conjecture 194]
```

```
307. ! [X1] : ! [X0] : (tmPP(X0,X1) <=> (X0 != X1 & tmP(X0,X1))) [closure 161]
308. ! [X1,X0] : (tmPP(X0,X1) <=> (X0 != X1 & tmP(X0,X1))) [flattening 307]
310. ! [X0,X1] : (EX(X0,X1) <=> ((~T(X1) & ~T(X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1))) [rectify 163]
311. ! [X0,X1] : (tmP(X0,X1) <=> (! [X2] : ((! [X3] : (EX(X2,X3) => EX(X0,X3)) & oP(X2,X1)) => oP(X2,X0)) & oP(X0,
     X1))) [rectify 165]
315. ! [X0,X1] : (::(X0,treg,X1) <=> (EX(X0,X1) & T(X0))) [rectify 169]
341. ! [X0,X1] : ((? [X2] : ::(X1,treg,X2) & ? [X3] : ::(X0,treg,X3)) => (tmP(X0,X1) <=> oP(X0,X1))) [rectify 192]
342. ! [X0] : ((TR(X0) | PR(X0) | AR(X0)) => R(X0)) [unused predicate definition removal 124]
380. ! [X0] : (P(X0,X0) | (~PD(X0) & ~AB(X0))) [ennf transformation 8]
539. ! [X0] : (R(X0) | (~TR(X0) & ~PR(X0) & ~AR(X0))) [ennf transformation 342]
540. ! [X0] : (AB(X0) | ~R(X0)) [ennf transformation 125]
544. ! [X0] : (TR(X0) | ~T(X0)) [ennf transformation 130]
547. ! [X0] : (~PD(X0) | ~AB(X0)) [ennf transformation 133]
564. ! [X0,X1] : (tmP(X0,X1) <=> (! [X2] : (oP(X2,X0) | (? [X3] : (~EX(X0,X3) & EX(X2,X3)) | ~oP(X2,X1))) & oP(X0,
     X1))) [ennf transformation 311]
565. ! [X0,X1] : (tmP(X0,X1) <=> (! [X2] : (oP(X2,X0) | ? [X3] : (~EX(X0,X3) & EX(X2,X3)) | ~oP(X2,X1)) & oP(X0,X1
     ))) [flattening 564]
571. ! [X0,X1] : (~tmO(X0,X1) | O(X0,X1)) [ennf transformation 191]
572. ! [X0,X1] : ((tmP(X0,X1) <=> oP(X0,X1)) | (! [X2] : ~::(X1,treg,X2) | ! [X3] : ~::(X0,treg,X3))) [ennf
     transformation 341]
573. ! [X0,X1] : ((tmP(X0,X1) <=> oP(X0,X1)) | ! [X2] : ~::(X1,treg,X2) | ! [X3] : ~::(X0,treg,X3)) [flattening
     572]
574. ! [X0,X1] : (? [X2] : (~O(X2,X0) & oP(X2,X1)) | (~tmPP(X0,X1) | ~::(X1,treg,X1) | ~::(X0,treg,X0))) [ennf
     transformation 193]
575. ! [X0,X1] : (? [X2] : (~O(X2,X0) & oP(X2,X1)) | ~tmPP(X0,X1) | ~::(X1,treg,X1) | ~::(X0,treg,X0)) [flattening
      574]
576. ? [X0,X1] : (! [X2] : (tmO(X2,X0) | ~tmPP(X2,X1)) & (tmPP(X0,X1) & ::(X1,treg,X1) & ::(X0,treg,X0))) [ennf
     transformation 195]
577. ? [X0,X1] : (! [X2] : (tmO(X2,X0) | ~tmPP(X2,X1)) & tmPP(X0,X1) & ::(X1,treg,X1) & ::(X0,treg,X0)) [
     flattening 576]
580. ! [X1,X0] : (sP1(X1,X0) <=> ((T(X1) & ~T(X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1))) [predicate
     definition introduction]
581. ! [X0,X1] : (EX(X0,X1) <=> sP1(X1,X0)) [definition folding 310,580]
588. ! [X0,X1] : ((O(X0,X1) | ! [X2] : (~P(X2,X1) | ~P(X2,X0))) & (? [X2] : (P(X2,X1) & P(X2,X0)) | ~O(X0,X1))) [
     nnf transformation 2]
589. ! [X0,X1] : ((O(X0,X1) | ! [X2] : (~P(X2,X1) | ~P(X2,X0))) & (? [X3] : (P(X3,X1) & P(X3,X0)) | ~O(X0,X1))) [
     rectify 588]
590. ! [X1,X0] : (? [X3] : (P(X3,X1) & P(X3,X0)) => (P(sK4(X0,X1),X1) & P(sK4(X0,X1),X0))) [choice axiom]
591. ! [X0,X1] : ((O(X0,X1) | ! [X2] : (~P(X2,X1) | ~P(X2,X0))) & ((P(sK4(X0,X1),X1) & P(sK4(X0,X1),X0)) | ~O(X0,
     X1))) [skolemisation 589,590]
712. ! [X1,X0] : ((tmPP(X0,X1) | (X0 = X1 | ~tmP(X0,X1))) & ((X0 != X1 & tmP(X0,X1)) | ~tmPP(X0,X1))) [nnf
     transformation 308]
713. ! [X1,X0] : ((tmPP(X0,X1) | X0 = X1 | ~tmP(X0,X1)) & ((X0 != X1 & tmP(X0,X1)) | ~tmPP(X0,X1))) [flattening
     712]
714. ! [X0,X1] : ((tmPP(X1,X0) | X0 = X1 | ~tmP(X1,X0)) & ((X0 != X1 & tmP(X1,X0)) | ~tmPP(X1,X0))) [rectify 713]
715. ! [X1,X0] : ((sP1(X1,X0) | ((~T(X1) | T(X0) | ~AB(X0)) & (~P(X1,X0) | ~T(X0)) & ~PRE(X0,X1))) & (((T(X1) & ~T
     (X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1)) | ~sP1(X1,X0))) [nnf transformation 580]
716. ! [X1,X0] : ((sP1(X1,X0) | ((~T(X1) | T(X0) | ~AB(X0)) & (~P(X1,X0) | ~T(X0)) & ~PRE(X0,X1))) & ((T(X1) & ~T(
     X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1) | ~sP1(X1,X0))) [flattening 715]
717. ! [X0,X1] : ((sP1(X0,X1) | ((~T(X0) | T(X1) | ~AB(X1)) & (~P(X0,X1) | ~T(X1)) & ~PRE(X1,X0))) & ((T(X0) & ~T(
     X1) & AB(X1)) | (P(X0,X1) & T(X1)) | PRE(X1,X0) | ~sP1(X0,X1))) [rectify 716]
718. ! [X0,X1] : ((EX(X0,X1) | ~sP1(X1,X0)) & (sP1(X1,X0) | ~EX(X0,X1))) [nnf transformation 581]
719. ! [X0,X1] : ((oP(X0,X1) | (((~T(X1) | ~T(X0)) & (~PD(X1) | ~PD(X0))) | ~P(X0,X1))) & ((((T(X1) & T(X0)) | (PD
     (X1) & PD(X0))) & P(X0,X1)) | ~oP(X0,X1))) [nnf transformation 164]
720. ! [X0,X1] : ((oP(X0,X1) | ((~T(X1) | ~T(X0)) & (~PD(X1) | ~PD(X0))) | ~P(X0,X1)) & ((((T(X1) & T(X0)) | (PD(
     X1) & PD(X0))) & P(X0,X1)) | ~oP(X0,X1))) [flattening 719]
721. ! [X0,X1] : ((tmP(X0,X1) | (? [X2] : (~oP(X2,X0) & ! [X3] : (EX(X0,X3) | ~EX(X2,X3)) & oP(X2,X1)) | ~oP(X0,X1
     ))) & ((! [X2] : (oP(X2,X0) | ? [X3] : (~EX(X0,X3) & EX(X2,X3)) | ~oP(X2,X1)) & oP(X0,X1)) | ~tmP(X0,X1))) [
     nnf transformation 565]
722. ! [X0,X1] : ((tmP(X0,X1) | ? [X2] : (~oP(X2,X0) & ! [X3] : (EX(X0,X3) | ~EX(X2,X3)) & oP(X2,X1)) | ~oP(X0,X1)
     ) & ((! [X2] : (oP(X2,X0) | ? [X3] : (~EX(X0,X3) & EX(X2,X3)) | ~oP(X2,X1)) & oP(X0,X1)) | ~tmP(X0,X1))) [
     flattening 721]
723. ! [X0,X1] : ((tmP(X0,X1) | ? [X2] : (~oP(X2,X0) & ! [X3] : (EX(X0,X3) | ~EX(X2,X3)) & oP(X2,X1)) | ~oP(X0,X1)
     ) & ((! [X4] : (oP(X4,X0) | ? [X5] : (~EX(X0,X5) & EX(X4,X5)) | ~oP(X4,X1)) & oP(X0,X1)) | ~tmP(X0,X1))) [
     rectify 722]
724. ! [X1,X0] : (? [X2] : (~oP(X2,X0) & ! [X3] : (EX(X0,X3) | ~EX(X2,X3)) & oP(X2,X1)) => (~oP(sK68(X0,X1),X0) &
     ! [X3] : (EX(X0,X3) | ~EX(sK68(X0,X1),X3)) & oP(sK68(X0,X1),X1))) [choice axiom]
725. ! [X4,X0] : (? [X5] : (~EX(X0,X5) & EX(X4,X5)) => (~EX(X0,sK69(X0,X4)) & EX(X4,sK69(X0,X4)))) [choice axiom]
726. ! [X0,X1] : ((tmP(X0,X1) | (~oP(sK68(X0,X1),X0) & ! [X3] : (EX(X0,X3) | ~EX(sK68(X0,X1),X3)) & oP(sK68(X0,X1)
     ,X1)) | ~oP(X0,X1)) & ((! [X4] : (oP(X4,X0) | (~EX(X0,sK69(X0,X4)) & EX(X4,sK69(X0,X4)))) | ~oP(X4,X1)) & oP(
     X0,X1)) | ~tmP(X0,X1))) [skolemisation 723,725,724]
734. ! [X0,X1] : ((::(X0,treg,X1) | (~EX(X0,X1) | ~T(X0))) & ((EX(X0,X1) & T(X0)) | ~::(X0,treg,X1))) [nnf
     transformation 315]
735. ! [X0,X1] : ((::(X0,treg,X1) | ~EX(X0,X1) | ~T(X0)) & ((EX(X0,X1) & T(X0)) | ~::(X0,treg,X1))) [flattening
     734]
785. ! [X0,X1] : ((((tmP(X0,X1) | ~oP(X0,X1)) & (oP(X0,X1) | ~tmP(X0,X1))) | ! [X2] : ~::(X1,treg,X2) | ! [X3] :
     ~::(X0,treg,X3)) [nnf transformation 573]
786. ! [X1,X0] : (? [X2] : (~O(X2,X0) & oP(X2,X1)) => (~O(sK86(X0,X1),X0) & oP(sK86(X0,X1),X1))) [choice axiom]
787. ! [X0,X1] : ((~O(sK86(X0,X1),X0) & oP(sK86(X0,X1),X1)) | ~tmPP(X0,X1) | ~::(X1,treg,X1) | ~::(X0,treg,X0)) [
     skolemisation 575,786]
788. ? [X0,X1] : (! [X2] : (tmO(X2,X0) | ~tmPP(X2,X1)) & tmPP(X0,X1) & ::(X1,treg,X1) & ::(X0,treg,X0)) => (! [X2]
      : (tmO(X2,sK87) | ~tmPP(X2,sK88)) & tmPP(sK87,sK88) & ::(sK88,treg,sK88) & ::(sK87,treg,sK87)) [choice
     axiom]
789. ! [X2] : (tmO(X2,sK87) | ~tmPP(X2,sK88)) & tmPP(sK87,sK88) & ::(sK88,treg,sK88) & ::(sK87,treg,sK87) [
     skolemisation 577,788]
795. ~P(X2,X1) | O(X0,X1) | ~P(X2,X0) [cnf transformation 591]
811. P(X0,X0) | ~AB(X0) [cnf transformation 380]
1014. ~TR(X0) | R(X0) [cnf transformation 539]
1015. ~R(X0) | AB(X0) [cnf transformation 540]
1021. ~T(X0) | TR(X0) [cnf transformation 544]
1024. ~AB(X0) | ~PD(X0) [cnf transformation 547]
```

```
1044. ~tmPP(X1,X0) | tmP(X1,X0) [cnf transformation 714]
1046. ~tmP(X1,X0) | X0 = X1 | tmPP(X1,X0) [cnf transformation 714]
1054. ~P(X0,X1) | sP1(X0,X1) | ~T(X1) [cnf transformation 717]
1057. ~sP1(X1,X0) | EX(X0,X1) [cnf transformation 718]
1058. ~oP(X0,X1) | P(X0,X1) [cnf transformation 720]
1060. ~oP(X0,X1) | PD(X1) | T(X0) [cnf transformation 720]
1065. ~tmP(X0,X1) | oP(X0,X1) [cnf transformation 726]
1080. ~::(X0,treg,X1) | T(X0) [cnf transformation 735]
1082. ::(X0,treg,X1) | ~EX(X0,X1) | ~T(X0) [cnf transformation 735]
1148. ~tmO(X0,X1) | O(X0,X1) [cnf transformation 571]
1150. ~::(X1,treg,X2) | ~oP(X0,X1) | tmP(X0,X1) | ~::(X0,treg,X3) [cnf transformation 785]
1151. ~::(X1,treg,X1) | ~tmPP(X0,X1) | oP(sK86(X0,X1),X1) | ~::(X0,treg,X0) [cnf transformation 787]
1152. ~O(sK86(X0,X1),X0) | ~tmPP(X0,X1) | ~::(X1,treg,X1) | ~::(X0,treg,X0) [cnf transformation 787]
1153. ::(sK87,treg,sK87) [cnf transformation 789]
1154. ::(sK88,treg,sK88) [cnf transformation 789]
1155. tmPP(sK87,sK88) [cnf transformation 789]
1156. ~tmPP(X2,sK88) | tmO(X2,sK87) [cnf transformation 789]
1209. tmP(sK87,sK88) [resolution 1044,1155]
1213. oP(sK87,sK88) [resolution 1065,1209]
1214. P(sK87,sK88) [resolution 1213,1058]
1217. T(sK87) [resolution 1080,1153]
1218. T(sK88) [resolution 1080,1154]
1219. TR(sK87) [resolution 1217,1021]
1220. TR(sK88) [resolution 1218,1021]
1221. R(sK87) [resolution 1219,1014]
1222. R(sK88) [resolution 1220,1014]
1223. AB(sK87) [resolution 1221,1015]
1224. AB(sK88) [resolution 1222,1015]
1226. ~PD(sK88) [resolution 1224,1024]
1487. ~P(X3,X2) | O(X2,X3) | ~AB(X3) [resolution 795,811]
2605. O(sK88,sK87) | ~AB(sK87) [resolution 1487,1214]
2726. ~::(X2,treg,X3) | tmP(X2,sK88) | ~oP(X2,sK88) [resolution 1150,1154]
2929. ~::(X1,treg,X1) | oP(sK86(X1,sK88),sK88) | ~tmPP(X1,sK88) [resolution 1151,1154]
3598. ~EX(X0,X1) | ~oP(X0,sK88) | tmP(X0,sK88) | ~T(X0) [resolution 2726,1082]
5164. oP(sK86(sK87,sK88),sK88) | ~tmPP(sK87,sK88) [resolution 2929,1153]
5167. oP(sK86(sK87,sK88),sK88) [subsumption resolution 5164,1155]
5170. PD(sK88) | T(sK86(sK87,sK88)) [resolution 5167,1060]
5173. T(sK86(sK87,sK88)) [subsumption resolution 5170,1226]
5174. TR(sK86(sK87,sK88)) [resolution 5173,1021]
5175. R(sK86(sK87,sK88)) [resolution 5174,1014]
5176. AB(sK86(sK87,sK88)) [resolution 5175,1015]
5219. 29 <=> sK88 = sK86(sK87,sK88) [avatar definition]
5220. sK88 != sK86(sK87,sK88) <- (~29) [avatar component clause 5219]
5221. sK88 = sK86(sK87,sK88) <- (29) [avatar component clause 5219]
17160. 38 <=> tmPP(sK86(sK87,sK88),sK88) [avatar definition]
17161. ~tmPP(sK86(sK87,sK88),sK88) <- (~38) [avatar component clause 17160]
17162. tmPP(sK86(sK87,sK88),sK88) <- (38) [avatar component clause 17160]
17280. 39 <=> P(sK86(sK87,sK88),sK86(sK87,sK88)) [avatar definition]
17281. ~P(sK86(sK87,sK88),sK86(sK87,sK88)) <- (~39) [avatar component clause 17280]
17282. P(sK86(sK87,sK88),sK86(sK87,sK88)) <- (39) [avatar component clause 17280]
18583. tmO(sK86(sK87,sK88),sK87) <- (38) [resolution 17162,1156]
18664. O(sK86(sK87,sK88),sK87) <- (38) [resolution 18583,1148]
18818. ~tmPP(sK87,sK88) | ~::(sK88,treg,sK88) | ~::(sK87,treg,sK87) <- (38) [resolution 18664,1152]
18835. ~::(sK88,treg,sK88) | ~::(sK87,treg,sK87) <- (38) [subsumption resolution 18818,1155]
18836. ~::(sK87,treg,sK87) <- (38) [subsumption resolution 18835,1154]
18837. $false <- (38) [subsumption resolution 18836,1153]
18838. ~38 [avatar contradiction clause 18837]
19389. oP(sK86(sK87,sK88),sK88) | ~tmPP(sK87,sK88) [resolution 2929,1153]
19392. oP(sK86(sK87,sK88),sK88) [subsumption resolution 19389,1155]
20439. ~O(sK88,sK87) | ~tmPP(sK87,sK88) | ~::(sK88,treg,sK88) | ~::(sK87,treg,sK87) <- (29) [superposition
       1152,5221]
20445. ~O(sK88,sK87) | ~::(sK88,treg,sK88) | ~::(sK87,treg,sK87) <- (29) [subsumption resolution 20439,1155]
20446. ~O(sK88,sK87) | ~::(sK87,treg,sK87) <- (29) [subsumption resolution 20445,1154]
20447. ~O(sK88,sK87) <- (29) [subsumption resolution 20446,1153]
20497. ~AB(sK87) <- (29) [subsumption resolution 2605,20447]
20498. $false <- (29) [subsumption resolution 20497,1223]
20499. ~29 [avatar contradiction clause 20498]
23684. sP1(sK86(sK87,sK88),sK86(sK87,sK88)) | ~T(sK86(sK87,sK88)) <- (39) [resolution 17282,1054]
23691. sP1(sK86(sK87,sK88),sK86(sK87,sK88)) <- (39) [subsumption resolution 23684,5173]
23812. EX(sK86(sK87,sK88),sK86(sK87,sK88)) <- (39) [resolution 23691,1057]
23932. ~oP(sK86(sK87,sK88),sK88) | tmP(sK86(sK87,sK88),sK88) | ~T(sK86(sK87,sK88)) <- (39) [resolution 23812,3598]
23938. tmP(sK86(sK87,sK88),sK88) | ~T(sK86(sK87,sK88)) <- (39) [subsumption resolution 23932,19392]
23939. tmP(sK86(sK87,sK88),sK88) <- (39) [subsumption resolution 23938,5173]
24025. sK88 = sK86(sK87,sK88) | tmPP(sK86(sK87,sK88),sK88) <- (39) [resolution 23939,1046]
24028. tmPP(sK86(sK87,sK88),sK88) <- (~29, 39) [subsumption resolution 24025,5220]
24029. $false <- (~29, ~38, 39) [subsumption resolution 24028,17161]
24030. 29 | 38 | ~39 [avatar contradiction clause 24029]
24142. ~AB(sK86(sK87,sK88)) <- (~39) [resolution 17281,811]
24143. $false <- (~39) [subsumption resolution 24142,5176]
24144. 39 [avatar contradiction clause 24143]
24145. $false [avatar sat refutation 18838,20499,24030,24144]
```

Proof of Theorem (t_{db} 19)

```
164. ! [X0,X1] : (tmP(X0,X1) <=> (! [X2] : ((! [X4] : (EX(X2,X4) => EX(X0,X4)) & oP(X2,X1)) => oP(X2,X0)) & oP(X0,
       X1))) [input]
189. ! [X0,X1] : (tmP(X0,X1) => oP(X0,X1)) [input]
190. ~! [X0,X1] : (tmP(X0,X1) => oP(X0,X1)) [negated conjecture 189]
305. ! [X0,X1] : (tmP(X0,X1) <=> (! [X2] : ((! [X3] : (EX(X2,X3) => EX(X0,X3)) & oP(X2,X1)) => oP(X2,X0)) & oP(X0,
       X1))) [rectify 164]
```

557. ! [X0,X1] : (tmP(X0,X1) <=> (! [X2] : (oP(X2,X0) | (? [X3] : (~EX(X0,X3) & EX(X2,X3)) | ~oP(X2,X1))) & oP(X0,X1))) [ennf transformation 305]
558. ! [X0,X1] : (tmP(X0,X1) <=> (! [X2] : (oP(X2,X0) | ? [X3] : (~EX(X0,X3) & EX(X2,X3)) | ~oP(X2,X1)) & oP(X0,X1))) [flattening 557]
563. ? [X0,X1] : (~oP(X0,X1) & tmP(X0,X1)) [ennf transformation 190]
703. ! [X0,X1] : ((tmP(X0,X1) | (? [X2] : (~oP(X2,X0) & ! [X3] : (EX(X0,X3) | ~EX(X2,X3)) & oP(X2,X1)) | ~oP(X0,X1))) & ((! [X2] : (oP(X2,X0) | ? [X3] : (~EX(X0,X3) & EX(X2,X3)) | ~oP(X2,X1)) & oP(X0,X1)) | ~tmP(X0,X1))) [nnf transformation 558]
704. ! [X0,X1] : ((tmP(X0,X1) | ? [X2] : (~oP(X2,X0) & ! [X3] : (EX(X0,X3) | ~EX(X2,X3)) & oP(X2,X1)) | ~oP(X0,X1)) & ((! [X2] : (oP(X2,X0) | ? [X3] : (~EX(X0,X3) & EX(X2,X3)) | ~oP(X2,X1)) & oP(X0,X1)) | ~tmP(X0,X1))) [flattening 703]
705. ! [X0,X1] : ((tmP(X0,X1) | ? [X2] : (~oP(X2,X0) & ! [X3] : (EX(X0,X3) | ~EX(X2,X3)) & oP(X2,X1)) | ~oP(X0,X1)) & ((! [X4] : (oP(X4,X0) | ? [X5] : (~EX(X0,X5) & EX(X4,X5)) | ~oP(X4,X1)) & oP(X0,X1)) | ~tmP(X0,X1))) [rectify 704]
706. ! [X1,X0] : (? [X2] : (~oP(X2,X0) & ! [X3] : (EX(X0,X3) | ~EX(X2,X3)) & oP(X2,X1)) => (~oP(sK67(X0,X1),X0) & ! [X3] : (EX(X0,X3) | ~EX(sK67(X0,X1),X3)) & oP(sK67(X0,X1),X1))) [choice axiom]
707. ! [X4,X0] : (? [X5] : (~EX(X0,X5) & EX(X4,X5)) => (~EX(X0,sK68(X0,X4)) & EX(X4,sK68(X0,X4)))) [choice axiom]
708. ! [X0,X1] : ((tmP(X0,X1) | (~oP(sK67(X0,X1),X0) & ! [X3] : (EX(X0,X3) | ~EX(sK67(X0,X1),X3)) & oP(sK67(X0,X1),X1)) | ~oP(X0,X1)) & ((! [X4] : (oP(X4,X0) | (~EX(X0,sK68(X0,X4)) & EX(X4,sK68(X0,X4))) | ~oP(X4,X1)) & oP(X0,X1)) | ~tmP(X0,X1))) [skolemisation 705,707,706]
765. ? [X0,X1] : (~oP(X0,X1) & tmP(X0,X1)) => (~oP(sK84,sK85) & tmP(sK84,sK85)) [choice axiom]
766. ~oP(sK84,sK85) & tmP(sK84,sK85) [skolemisation 563,765]
1039. ~tmP(X0,X1) | oP(X0,X1) [cnf transformation 708]
1120. tmP(sK84,sK85) [cnf transformation 766]
1121. ~oP(sK84,sK85) [cnf transformation 766]
1177. oP(sK84,sK85) [resolution 1039,1120]
1178. $false [subsumption resolution 1177,1121]

Proof of Theorem (t_{ab}20)

10. ! [X0,X1,X2] : ((P(X1,X2) & P(X0,X1)) => P(X0,X2)) [input]
34. ! [X0,X4] : (PRE(X0,X4) => (T(X4) & (Q(X0) | PD(X0) | ED(X0)))) [input]
36. ! [X0,X4,X1] : ((P(X0,X1) & PRE(X0,X4)) => PRE(X1,X4)) [input]
124. ! [X0] : ((TR(X0) | PR(X0) | AR(X0)) <=> R(X0)) [input]
125. ! [X0] : (R(X0) => AB(X0)) [input]
130. ! [X0] : (T(X0) => TR(X0)) [input]
132. ~? [X0] : (ED(X0) & AB(X0)) [input]
133. ~? [X0] : (PD(X0) & AB(X0)) [input]
134. ~? [X0] : (Q(X0) & AB(X0)) [input]
162. ! [X0,X4] : (EX(X0,X4) <=> ((T(X4) & ~T(X0) & AB(X0)) | (P(X4,X0) & T(X0)) | PRE(X0,X4))) [input]
163. ! [X0,X1] : (oP(X0,X1) <=> (((T(X1) & T(X0)) | (PD(X1) & PD(X0))) & P(X0,X1))) [input]
164. ! [X0,X1] : (tmP(X0,X1) <=> (! [X2] : ((! [X4] : (EX(X2,X4) => EX(X0,X4)) & oP(X2,X1)) => oP(X2,X0)) & oP(X0,X1))) [input]
189. ! [X0,X1,X4] : ((EX(X0,X4) & tmP(X0,X1)) => EX(X1,X4)) [input]
190. ~! [X0,X1,X4] : ((EX(X0,X4) & tmP(X0,X1)) => EX(X1,X4)) [negated conjecture 189]
200. ! [X0,X1] : (PRE(X0,X1) => (T(X1) & (Q(X0) | PD(X0) | ED(X0)))) [rectify 34]
202. ! [X0,X1,X2] : ((P(X0,X2) & PRE(X0,X1)) => PRE(X2,X1)) [rectify 36]
304. ! [X0,X1] : (EX(X0,X1) <=> ((T(X1) & ~T(X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1))) [rectify 162]
305. ! [X0,X1] : (tmP(X0,X1) <=> (! [X2] : ((! [X3] : (EX(X2,X3) => EX(X0,X3)) & oP(X2,X1)) => oP(X2,X0)) & oP(X0,X1))) [rectify 164]
335. ~! [X0,X1,X2] : ((EX(X0,X2) & tmP(X0,X1)) => EX(X1,X2)) [rectify 190]
336. ! [X0] : ((TR(X0) | PR(X0) | AR(X0)) => R(X0)) [unused predicate definition removal 124]
377. ! [X0,X1,X2] : (P(X0,X2) | (~P(X1,X2) | ~P(X0,X1))) [ennf transformation 10]
378. ! [X0,X1,X2] : (P(X0,X2) | ~P(X1,X2) | ~P(X0,X1)) [flattening 377]
411. ! [X0,X1] : ((T(X1) & (Q(X0) | PD(X0) | ED(X0))) | ~PRE(X0,X1)) [ennf transformation 200]
414. ! [X0,X1,X2] : (PRE(X2,X1) | (~P(X0,X2) | ~PRE(X0,X1))) [ennf transformation 202]
415. ! [X0,X1,X2] : (PRE(X2,X1) | ~P(X0,X2) | ~PRE(X0,X1)) [flattening 414]
533. ! [X0] : (R(X0) | (~TR(X0) & ~PR(X0) & ~AR(X0))) [ennf transformation 336]
534. ! [X0] : (AB(X0) | ~R(X0)) [ennf transformation 125]
538. ! [X0] : (TR(X0) | ~T(X0)) [ennf transformation 130]
540. ! [X0] : (~ED(X0) | ~AB(X0)) [ennf transformation 132]
541. ! [X0] : (~PD(X0) | ~AB(X0)) [ennf transformation 133]
542. ! [X0] : (~Q(X0) | ~AB(X0)) [ennf transformation 134]
558. ! [X0,X1] : (tmP(X0,X1) <=> (! [X2] : (oP(X2,X0) | (? [X3] : (~EX(X0,X3) & EX(X2,X3)) | ~oP(X2,X1))) & oP(X0,X1))) [ennf transformation 305]
559. ! [X0,X1] : (tmP(X0,X1) <=> (! [X2] : (oP(X2,X0) | ? [X3] : (~EX(X0,X3) & EX(X2,X3)) | ~oP(X2,X1)) & oP(X0,X1))) [flattening 558]
564. ? [X0,X1,X2] : (~EX(X1,X2) & (EX(X0,X2) & tmP(X0,X1))) [ennf transformation 335]
565. ? [X0,X1,X2] : (~EX(X1,X2) & EX(X0,X2) & tmP(X0,X1)) [flattening 564]
568. ! [X1,X0] : (sP1(X1,X0) <=> ((T(X1) & ~T(X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1))) [predicate definition introduction]
569. ! [X0,X1] : (EX(X0,X1) <=> sP1(X1,X0)) [definition folding 304,568]
699. ! [X1,X0] : ((sP1(X1,X0) | ((~T(X1) | T(X0) | ~AB(X0)) & (~P(X1,X0) | ~T(X0)) & ~PRE(X0,X1))) & (((T(X1) & ~T(X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1)) | ~sP1(X1,X0))) [nnf transformation 568]
700. ! [X1,X0] : ((sP1(X1,X0) | ((~T(X1) | T(X0) | ~AB(X0)) & (~P(X1,X0) | ~T(X0)) & ~PRE(X0,X1))) & ((T(X1) & ~T(X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1) | ~sP1(X1,X0))) [flattening 699]
701. ! [X0,X1] : ((sP1(X0,X1) | ((~T(X0) | T(X1) | ~AB(X1)) & (~P(X0,X1) | ~T(X1)) & ~PRE(X1,X0))) & ((T(X0) & ~T(X1) & AB(X1)) | (P(X0,X1) & T(X1)) | PRE(X1,X0) | ~sP1(X0,X1))) [rectify 700]
702. ! [X0,X1] : ((EX(X0,X1) | ~sP1(X1,X0)) & (sP1(X1,X0) | ~EX(X0,X1))) [nnf transformation 569]
703. ! [X0,X1] : ((oP(X0,X1) | (((~T(X1) | ~T(X0)) & (~PD(X1) | ~PD(X0))) | ~P(X0,X1))) & ((((T(X1) & T(X0)) | (PD(X1) & PD(X0))) & P(X0,X1)) | ~oP(X0,X1))) [nnf transformation 163]
704. ! [X0,X1] : ((oP(X0,X1) | ((~T(X1) | ~T(X0)) & (~PD(X1) | ~PD(X0))) | ~P(X0,X1)) & ((((T(X1) & T(X0)) | (PD(X1) & PD(X0))) & P(X0,X1)) | ~oP(X0,X1))) [flattening 703]
705. ! [X0,X1] : ((tmP(X0,X1) | (? [X2] : (~oP(X2,X0) & ! [X3] : (EX(X0,X3) | ~EX(X2,X3)) & oP(X2,X1)) | ~oP(X0,X1))) & ((! [X2] : (oP(X2,X0) | ? [X3] : (~EX(X0,X3) & EX(X2,X3)) | ~oP(X2,X1)) & oP(X0,X1)) | ~tmP(X0,X1))) [nnf transformation 559]
706. ! [X0,X1] : ((tmP(X0,X1) | ? [X2] : (~oP(X2,X0) & ! [X3] : (EX(X0,X3) | ~EX(X2,X3)) & oP(X2,X1)) | ~oP(X0,X1)) & ((! [X2] : (oP(X2,X0) | ? [X3] : (~EX(X0,X3) & EX(X2,X3)) | ~oP(X2,X1)) & oP(X0,X1)) | ~tmP(X0,X1))) [flattening 705]

```
707. ! [X0,X1] : ((tmP(X0,X1) | ? [X2] : (~oP(X2,X0) & ! [X3] : (EX(X0,X3) | ~EX(X2,X3)) & oP(X2,X1)) | ~oP(X0,X1)
       ) & ((! [X4] : (oP(X4,X0) | ? [X5] : (~EX(X0,X5) & EX(X4,X5)) | ~oP(X4,X1)) & oP(X0,X1)) | ~tmP(X0,X1))) [
       rectify 706]
708. ! [X1,X0] : (? [X2] : (~oP(X2,X0) & ! [X3] : (EX(X0,X3) | ~EX(X2,X3)) & oP(X2,X1)) => (~oP(sK67(X0,X1),X0) &
       ! [X3] : (EX(X0,X3) | ~EX(sK67(X0,X1),X3)) & oP(sK67(X0,X1),X1))) [choice axiom]
709. ! [X4,X0] : (? [X5] : (~EX(X0,X5) & EX(X4,X5)) => (~EX(X0,sK68(X0,X4)) & EX(X4,sK68(X0,X4)))) [choice axiom]
710. ! [X0,X1] : ((tmP(X0,X1) | (~oP(sK67(X0,X1),X0) & ! [X3] : (EX(X0,X3) | ~EX(sK67(X0,X1),X3)) & oP(sK67(X0,X1)
       ,X1)) | ~oP(X0,X1)) & ((! [X4] : (oP(X4,X0) | (~EX(X0,sK68(X0,X4)) & EX(X4,sK68(X0,X4))) | ~oP(X4,X1)) & oP(
       X0,X1)) | ~tmP(X0,X1))) [skolemisation 707,709,708]
767. ? [X0,X1,X2] : (~EX(X1,X2) & EX(X0,X2) & tmP(X0,X1)) => (~EX(sK85,sK86) & EX(sK84,sK86) & tmP(sK84,sK85)) [
       choice axiom]
768. ~EX(sK85,sK86) & EX(sK84,sK86) & tmP(sK84,sK85) [skolemisation 565,767]
793. ~P(X1,X2) | P(X0,X2) | ~P(X0,X1) [cnf transformation 378]
841. ~PRE(X0,X1) | PD(X0) | ED(X0) | Q(X0) [cnf transformation 411]
844. ~PRE(X0,X1) | ~P(X0,X2) | PRE(X2,X1) [cnf transformation 415]
993. ~TR(X0) | R(X0) [cnf transformation 533]
994. ~R(X0) | AB(X0) [cnf transformation 534]
1000. ~T(X0) | TR(X0) [cnf transformation 538]
1002. ~ED(X0) | ~AB(X0) [cnf transformation 540]
1003. ~AB(X0) | ~PD(X0) [cnf transformation 541]
1004. ~Q(X0) | ~AB(X0) [cnf transformation 542]
1023. ~sP1(X0,X1) | T(X1) | PRE(X1,X0) | AB(X1) [cnf transformation 701]
1024. ~sP1(X0,X1) | P(X0,X1) | PRE(X1,X0) | AB(X1) [cnf transformation 701]
1026. ~sP1(X0,X1) | P(X0,X1) | PRE(X1,X0) | ~T(X1) [cnf transformation 701]
1029. ~PRE(X1,X0) | sP1(X0,X1) [cnf transformation 701]
1030. ~P(X0,X1) | sP1(X0,X1) | ~T(X1) [cnf transformation 701]
1032. ~EX(X0,X1) | sP1(X1,X0) [cnf transformation 702]
1033. ~sP1(X1,X0) | EX(X0,X1) [cnf transformation 702]
1034. ~oP(X0,X1) | P(X0,X1) [cnf transformation 704]
1035. ~oP(X0,X1) | PD(X0) | T(X0) [cnf transformation 704]
1037. ~oP(X0,X1) | PD(X0) | T(X1) [cnf transformation 704]
1041. ~tmP(X0,X1) | oP(X0,X1) [cnf transformation 710]
1122. tmP(sK84,sK85) [cnf transformation 768]
1123. EX(sK84,sK86) [cnf transformation 768]
1124. ~EX(sK85,sK86) [cnf transformation 768]
1177. sP1(sK86,sK84) [resolution 1032,1123]
1181. oP(sK84,sK85) [resolution 1041,1122]
1182. P(sK84,sK85) [resolution 1181,1034]
1194. 1 <=> PD(sK84) [avatar definition]
1195. ~PD(sK84) <- (~1) [avatar component clause 1194]
1198. 2 <=> AB(sK84) [avatar definition]
1199. ~AB(sK84) <- (~2) [avatar component clause 1198]
1200. AB(sK84) <- (2) [avatar component clause 1198]
1202. ~PD(sK84) <- (2) [resolution 1200,1003]
1203. ~1 | ~2 [avatar split clause 1202,1198,1194]
1219. 3 <=> T(sK85) [avatar definition]
1221. T(sK85) <- (3) [avatar component clause 1219]
1223. 4 <=> T(sK84) [avatar definition]
1224. T(sK84) <- (4) [avatar component clause 1223]
1225. ~T(sK84) <- (~4) [avatar component clause 1223]
1285. PD(sK84) | T(sK84) [resolution 1035,1181]
1303. PD(sK84) | T(sK85) [resolution 1037,1181]
1487. ~P(X7,sK84) | P(X7,sK85) [resolution 793,1182]
1926. T(sK84) | PRE(sK84,sK86) | AB(sK84) [resolution 1023,1177]
1928. PRE(sK84,sK86) | AB(sK84) <- (~4) [subsumption resolution 1926,1225]
1929. PRE(sK84,sK86) <- (~2, ~4) [subsumption resolution 1928,1199]
1933. ~P(sK84,X1) | PRE(X1,sK86) <- (~2, ~4) [resolution 1929,844]
1973. PRE(sK85,sK86) <- (~2, ~4) [resolution 1933,1182]
1982. sP1(sK86,sK85) <- (~2, ~4) [resolution 1973,1029]
1988. EX(sK85,sK86) <- (~2, ~4) [resolution 1982,1033]
1989. $false <- (~2, ~4) [subsumption resolution 1988,1124]
1990. 2 | 4 [avatar contradiction clause 1989]
2007. 3 | 1 [avatar split clause 1303,1194,1219]
2017. 16 <=> PRE(sK84,sK86) [avatar definition]
2019. PRE(sK84,sK86) <- (16) [avatar component clause 2017]
2027. 18 <=> P(sK86,sK84) [avatar definition]
2029. P(sK86,sK84) <- (18) [avatar component clause 2027]
2032. 4 | 1 [avatar split clause 1285,1194,1223]
2065. TR(sK84) <- (4) [resolution 1224,1000]
2072. R(sK84) <- (4) [resolution 2065,993]
2077. AB(sK84) <- (4) [resolution 2072,994]
2133. P(sK86,sK84) | PRE(sK84,sK86) | AB(sK84) [resolution 1024,1177]
2173. PD(sK84) | ED(sK84) | Q(sK84) <- (16) [resolution 2019,841]
2180. ED(sK84) | Q(sK84) <- (~1, 16) [subsumption resolution 2173,1195]
2182. 24 <=> Q(sK84) [avatar definition]
2184. Q(sK84) <- (24) [avatar component clause 2182]
2186. 25 <=> ED(sK84) [avatar definition]
2188. ED(sK84) <- (25) [avatar component clause 2186]
2189. 24 | 25 | 1 | ~16 [avatar split clause 2180,2017,1194,2186,2182]
2197. ~AB(sK84) <- (25) [resolution 2188,1002]
2212. $false <- (2, 25) [subsumption resolution 2197,1200]
2213. ~2 | ~25 [avatar contradiction clause 2212]
2283. P(sK86,sK84) | PRE(sK84,sK86) | ~T(sK84) [resolution 1177,1026]
2390. P(sK86,sK84) | PRE(sK84,sK86) <- (4) [subsumption resolution 2283,1224]
2391. 16 | 18 | ~4 [avatar split clause 2390,1223,2027,2017]
2394. ~AB(sK84) <- (24) [resolution 2184,1004]
2408. $false <- (2, 24) [subsumption resolution 2394,1200]
2409. ~2 | ~24 [avatar contradiction clause 2408]
2424. P(sK86,sK85) <- (18) [resolution 2029,1487]
2522. sP1(sK86,sK85) | ~T(sK85) <- (18) [resolution 2424,1030]
2525. sP1(sK86,sK85) <- (3, 18) [subsumption resolution 2522,1221]
```

```
2552. EX(sK85,sK86) <- (3, 18) [resolution 2525,1033]
2553. $false <- (3, 18) [subsumption resolution 2552,1124]
2554. ~3 | ~18 [avatar contradiction clause 2553]
2557. 2 | 16 | 18 [avatar split clause 2133,2027,2017,1198]
2558. 2 | ~4 [avatar split clause 2077,1223,1198]
2567. $false [avatar sat refutation 1203,1990,2007,2032,2189,2213,2391,2409,2554,2557,2558]
```

Proof of Theorem (t_{db}21)

```
1 P(A,B) -> AB(A) & AB(B) | PD(A) & PD(B). [assumption].
2 P(A,B) -> (T(A) <-> T(B)). [assumption].
3 AB(A) | PD(A) -> P(A,A). [assumption].
4 PRE(A,B) -> (ED(A) | PD(A) | Q(A)) & T(B). [assumption].
5 AR(A) | PR(A) | TR(A) <-> R(A). [assumption].
6 R(A) -> AB(A). [assumption].
7 T(A) -> TR(A). [assumption].
8 -(exists A (AB(A) & ED(A))). [assumption].
9 -(exists A (AB(A) & PD(A))). [assumption].
10 -(exists A (AB(A) & Q(A))). [assumption].
11 EX(A,B) <-> PRE(A,B) | T(A) & P(B,A) | AB(A) & -T(A) & T(B). [assumption].
12 oP(A,B) <-> P(A,B) & (PD(A) & PD(B) | T(A) & T(B)). [assumption].
13 tmP(A,B) <-> oP(A,B) & (all C (oP(C,B) & (all D (EX(C,D) -> EX(A,D))) -> oP(C,A))). [assumption].
14 ::(A,treg,B) <-> T(A) & EX(A,B). [assumption].
15 ::(A,treg,B) -> tmP(B,A). [goal].
16 -AB(A) | P(A,A). [clausify(3)].
17 -P(A,B) | AB(A) | PD(B). [clausify(1)].
18 -R(A) | AB(A). [clausify(6)].
19 -AB(A) | -ED(A). [clausify(8)].
20 -AB(A) | -PD(A). [clausify(9)].
21 -AB(A) | -Q(A). [clausify(10)].
22 -TR(A) | R(A). [clausify(5)].
23 -T(A) | TR(A). [clausify(7)].
24 -P(A,B) | T(A) | -T(B). [clausify(2)].
25 -PRE(A,B) | ED(A) | PD(A) | Q(A). [clausify(4)].
26 -EX(A,B) | PRE(A,B) | P(B,A) | -T(A). [clausify(11)].
27 EX(A,B) | -T(A) | -P(B,A). [clausify(11)].
28 -oP(A,B) | PD(B) | T(A). [clausify(12)].
29 oP(A,B) | -P(A,B) | -T(A) | -T(B). [clausify(12)].
30 tmP(A,B) | -oP(A,B) | oP(f79(B,A),B). [clausify(13)].
31 tmP(A,B) | -oP(A,B) | -EX(f79(B,A),C) | EX(A,C). [clausify(13)].
32 tmP(A,B) | -oP(A,B) | -oP(f79(B,A),A). [clausify(13)].
33 -::(A,treg,B) | T(A). [clausify(14)].
34 -::(A,treg,B) | EX(A,B). [clausify(14)].
35 ::(c1,treg,c2). [deny(15)].
36 -tmP(c2,c1). [deny(15)].
37 -R(A) | P(A,A). [resolve(18,b,16,a)].
38 -ED(A) | -P(A,B) | PD(B). [resolve(19,a,17,b)].
39 -ED(A) | -R(A). [resolve(19,a,18,b)].
40 -PD(A) | -P(A,B) | PD(B). [resolve(20,a,17,b)].
41 -PD(A) | -R(A). [resolve(20,a,18,b)].
42 -Q(A) | -P(A,B) | PD(B). [resolve(21,a,17,b)].
43 -Q(A) | -R(A). [resolve(21,a,18,b)].
44 -T(A) | R(A). [resolve(23,b,22,a)].
45 EX(c1,c2). [resolve(35,a,34,a)].
46 T(c1). [resolve(35,a,33,a)].
47 R(c1). [resolve(46,a,44,a)].
48 -Q(c1). [resolve(47,a,43,b)].
49 -PD(c1). [resolve(47,a,41,b)].
50 -ED(c1). [resolve(47,a,39,b)].
51 PRE(c1,c2) | P(c2,c1). [resolve(45,a,26,a),unit_del(c,46)].
52 P(c2,c1). [resolve(51,a,25,a),unit_del(b,50),unit_del(c,49),unit_del(d,48)].
53 -Q(c2). [resolve(52,a,42,b),unit_del(b,49)].
54 -PD(c2). [resolve(52,a,40,b),unit_del(b,49)].
55 -ED(c2). [resolve(52,a,38,b),unit_del(b,49)].
56 oP(c2,c1) | -T(c2). [resolve(52,a,29,b),unit_del(c,46)].
57 T(c2). [resolve(52,a,24,a),unit_del(b,46)].
58 oP(c2,c1). [back_unit_del(56),unit_del(b,57)].
59 -EX(f79(c1,c2),A) | EX(c2,A). [resolve(58,a,31,b),unit_del(a,36)].
60 oP(f79(c1,c2),c1). [resolve(58,a,30,b),unit_del(a,36)].
61 T(f79(c1,c2)). [resolve(60,a,28,a),unit_del(a,49)].
62 R(f79(c1,c2)). [resolve(61,a,44,a)].
63 P(f79(c1,c2),f79(c1,c2)). [resolve(62,a,37,a)].
64 EX(f79(c1,c2),f79(c1,c2)). [resolve(63,a,27,c),unit_del(b,61)].
65 EX(c2,f79(c1,c2)). [resolve(59,a,64,a)].
66 PRE(c2,f79(c1,c2)) | P(f79(c1,c2),c2). [resolve(65,a,26,a),unit_del(c,57)].
67 P(f79(c1,c2),c2). [resolve(66,a,25,a),unit_del(b,55),unit_del(c,54),unit_del(d,53)].
68 oP(f79(c1,c2),c2). [resolve(67,a,29,b),unit_del(b,61),unit_del(c,57)].
69 $F. [resolve(68,a,32,c),unit_del(a,36),unit_del(b,58)].
```

Proof of Theorem (t_{db}22)

```
9. ! [X0,X1] : ((P(X1,X0) & P(X0,X1)) => X0 = X1) [input]
161. ! [X0] : (tmPP(X0,X1) <=> (X0 != X1 & tmP(X0,X1))) [input]
163. ! [X0,X1] : (oP(X0,X1) <=> (((T(X1) & T(X0)) | (PD(X1) & PD(X0))) & P(X0,X1))) [input]
164. ! [X0,X1] : (tmP(X0,X1) <=> (! [X2] : ((! [X4] : (EX(X2,X4) => EX(X0,X4)) & oP(X2,X1)) => oP(X2,X0)) & oP(X0,
     X1))) [input]
189. ! [X0,X1] : (tmPP(X0,X1) => ~tmPP(X1,X0)) [input]
190. ~! [X0,X1] : (tmPP(X0,X1) => ~tmPP(X1,X0)) [negated conjecture 189]
302. ! [X1] : ! [X0] : (tmPP(X0,X1) <=> (X0 != X1 & tmP(X0,X1))) [closure 161]
```

```
303. ! [X1,X0] : (tmPP(X0,X1) <=> (X0 != X1 & tmP(X0,X1))) [flattening 302]
305. ! [X0,X1] : (tmP(X0,X1) <=> (! [X2] : ((! [X3] : (EX(X2,X3) => EX(X0,X3)) & oP(X2,X1)) => oP(X2,X0)) & oP(X0,
     X1))) [rectify 164]
374. ! [X0,X1] : (X0 = X1 | ~P(X1,X0) | ~P(X0,X1))) [ennf transformation 9]
375. ! [X0,X1] : (X0 = X1 | ~P(X1,X0) | ~P(X0,X1)) [flattening 374]
557. ! [X0,X1] : (tmP(X0,X1) <=> (! [X2] : (oP(X2,X0) | (? [X3] : (~EX(X0,X3) & EX(X2,X3)) | ~oP(X2,X1))) & oP(X0,
     X1))) [ennf transformation 305]
558. ! [X0,X1] : (tmP(X0,X1) <=> (! [X2] : (oP(X2,X0) | ? [X3] : (~EX(X0,X3) & EX(X2,X3)) | ~oP(X2,X1)) & oP(X0,X1
     ))) [flattening 557]
563. ? [X0,X1] : (tmPP(X1,X0) & tmPP(X0,X1)) [ennf transformation 190]
694. ! [X1,X0] : ((tmPP(X0,X1) | (X0 = X1 | ~tmP(X0,X1))) & ((X0 != X1 & tmP(X0,X1)) | ~tmPP(X0,X1))) [nnf
     transformation 303]
695. ! [X1,X0] : ((tmPP(X0,X1) | X0 = X1 | ~tmP(X0,X1)) & ((X0 != X1 & tmP(X0,X1)) | ~tmPP(X0,X1))) [flattening
     694]
696. ! [X0,X1] : ((tmPP(X1,X0) | X0 = X1 | ~tmP(X1,X0)) & ((X0 != X1 & tmP(X1,X0)) | ~tmPP(X1,X0))) [rectify 695]
701. ! [X0,X1] : ((oP(X0,X1) | (((~T(X1) | ~T(X0)) & (~PD(X1) | ~PD(X0))) | ~P(X0,X1))) & ((((T(X1) & T(X0)) | (PD
     (X1) & PD(X0))) & P(X0,X1)) | ~oP(X0,X1))) [nnf transformation 163]
702. ! [X0,X1] : ((oP(X0,X1) | ((~T(X1) | ~T(X0)) & (~PD(X1) | ~PD(X0))) | ~P(X0,X1)) & ((((T(X1) & T(X0)) | (PD(
     X1) & PD(X0))) & P(X0,X1)) | ~oP(X0,X1))) [flattening 701]
703. ! [X0,X1] : ((tmP(X0,X1) | (? [X2] : (~oP(X2,X0) & ! [X3] : (EX(X0,X3) | ~EX(X2,X3)) & oP(X2,X1)) | ~oP(X0,X1
     ))) & ((! [X2] : (oP(X2,X0) | ? [X3] : (~EX(X0,X3) & EX(X2,X3)) | ~oP(X2,X1)) & oP(X0,X1)) | ~tmP(X0,X1))) [
     nnf transformation 558]
704. ! [X0,X1] : ((tmP(X0,X1) | ? [X2] : (~oP(X2,X0) & ! [X3] : (EX(X0,X3) | ~EX(X2,X3)) & oP(X2,X1)) | ~oP(X0,X1)
     ) & ((! [X2] : (oP(X2,X0) | ? [X3] : (~EX(X0,X3) & EX(X2,X3)) | ~oP(X2,X1)) & oP(X0,X1)) | ~tmP(X0,X1))) [
     flattening 703]
705. ! [X0,X1] : ((tmP(X0,X1) | ? [X2] : (~oP(X2,X0) & ! [X3] : (EX(X0,X3) | ~EX(X2,X3)) & oP(X2,X1)) | ~oP(X0,X1)
     ) & ((! [X4] : (oP(X4,X0) | ? [X5] : (~EX(X0,X5) & EX(X4,X5)) | ~oP(X4,X1)) & oP(X0,X1)) | ~tmP(X0,X1))) [
     rectify 704]
706. ! [X1,X0] : (? [X2] : (~oP(X2,X0) & ! [X3] : (EX(X0,X3) | ~EX(X2,X3)) & oP(X2,X1)) => (~oP(sK67(X0,X1),X0) &
     ! [X3] : (EX(X0,X3) | ~EX(sK67(X0,X1),X3)) & oP(sK67(X0,X1),X1))) [choice axiom]
707. ! [X4,X0] : (? [X5] : (~EX(X0,X5) & EX(X4,X5)) => (~EX(X0,sK68(X0,X4)) & EX(X4,sK68(X0,X4)))) [choice axiom]
708. ! [X0,X1] : ((tmP(X0,X1) | (~oP(sK67(X0,X1),X0) & ! [X3] : (EX(X0,X3) | ~EX(sK67(X0,X1),X3)) & oP(sK67(X0,X1)
     ,X1)) | ~oP(X0,X1)) & ((! [X4] : (oP(X4,X0) | (~EX(X0,sK68(X0,X4)) & EX(X4,sK68(X0,X4)))) | ~oP(X4,X1)) & oP(
     X0,X1)) | ~tmP(X0,X1))) [skolemisation 705,707,706]
765. ? [X0,X1] : (tmPP(X1,X0) & tmPP(X0,X1)) => (tmPP(sK85,sK84) & tmPP(sK84,sK85)) [choice axiom]
766. tmPP(sK85,sK84) & tmPP(sK84,sK85) [skolemisation 563,765]
790. ~P(X1,X0) | X0 = X1 | ~P(X0,X1) [cnf transformation 375]
1018. ~tmPP(X1,X0) | tmP(X1,X0) [cnf transformation 696]
1019. X0 != X1 | ~tmPP(X1,X0) [cnf transformation 696]
1032. ~oP(X0,X1) | P(X0,X1) [cnf transformation 702]
1039. ~tmP(X0,X1) | oP(X0,X1) [cnf transformation 708]
1120. tmPP(sK84,sK85) [cnf transformation 766]
1121. tmPP(sK85,sK84) [cnf transformation 766]
1125. ~tmPP(X1,X1) [equality resolution 1019]
1173. tmP(sK84,sK85) [resolution 1018,1120]
1174. tmP(sK85,sK84) [resolution 1018,1121]
1178. oP(sK84,sK85) [resolution 1039,1173]
1179. oP(sK85,sK84) [resolution 1039,1174]
1180. P(sK84,sK85) [resolution 1178,1032]
1181. P(sK85,sK84) [resolution 1179,1032]
1475. sK84 = sK85 | ~P(sK85,sK84) [resolution 790,1180]
1477. sK84 = sK85 [subsumption resolution 1475,1181]
1478. tmPP(sK84,sK84) [backward demodulation 1120,1477]
1493. $false [subsumption resolution 1478,1125]
```

Proof of Theorem (t_{db}23)

```
1 P(A,B) & P(B,A) -> A = B.  [assumption].
2 tmPP(A,B) <-> tmP(A,B) & A != B.  [assumption].
3 oP(A,B) <-> P(A,B) & (PD(A) & PD(B) | T(A) & T(B)).  [assumption].
4 tmP(A,B) <-> oP(A,B) & (all C (oP(C,B) & (all D (EX(C,D) -> EX(A,D))) -> oP(C,A))).  [assumption].
5 tmP(A,B) & tmP(B,C) -> tmP(A,C).  [assumption].
6 tmPP(A,B) & tmPP(B,C) -> tmPP(A,C).  [goal].
7 -P(A,B) | -P(B,A) | B = A.  [clausify(1)].
8 -tmPP(A,B) | tmP(A,B).  [clausify(2)].
9 -tmPP(A,B) | B != A.  [clausify(2)].
10 tmPP(A,B) | -tmP(A,B) | B = A.  [clausify(2)].
11 -oP(A,B) | P(A,B).  [clausify(3)].
12 -tmP(A,B) | oP(A,B).  [clausify(4)].
13 -tmP(A,B) | -tmP(B,C) | tmP(A,C).  [clausify(5)].
14 tmPP(c1,c2).  [deny(6)].
15 tmPP(c2,c3).  [deny(6)].
16 -tmPP(c1,c3).  [deny(6)].
17 tmP(c1,c2).  [resolve(14,a,8,a)].
18 c3 != c2.  [resolve(15,a,9,a)].
19 tmP(c2,c3).  [resolve(15,a,8,a)].
20 -tmP(c2,A) | tmP(c1,A).  [resolve(17,a,13,a)].
21 oP(c1,c2).  [resolve(17,a,12,a)].
22 oP(c2,c3).  [resolve(19,a,12,a)].
23 P(c1,c2).  [resolve(21,a,11,a)].
24 P(c2,c3).  [resolve(22,a,11,a)].
25 -P(c3,c2).  [resolve(24,a,7,b),flip(b),unit_del(b,18)].
26 tmP(c1,c3).  [resolve(20,a,19,a)].
27 c3 = c1.  [resolve(26,a,10,b),unit_del(a,16)].
28 $F.  [back_rewrite(25),rewrite([27(1)]),unit_del(a,23)].
```

Proof of Theorem (t_{db}25)

```
8. ! [X0] : ((PD(X0) | AB(X0)) => P(X0,X0)) [input]
27. ! [X0,X4] : (TLC(X0,X4) => (T(X4) & (Q(X0) | PD(X0) | ED(X0)))) [input]
29. ! [X0,X4,X5] : ((TLC(X0,X5) & TLC(X0,X4)) => X4 = X5) [input]
124. ! [X0] : ((TR(X0) | PR(X0) | AR(X0)) <=> R(X0)) [input]
125. ! [X0] : (R(X0) => AB(X0)) [input]
130. ! [X0] : (T(X0) => TR(X0)) [input]
162. ! [X0,X4] : (EX(X0,X4) <=> ((T(X4) & ~T(X0) & AB(X0)) | (P(X4,X0) & T(X0)) | PRE(X0,X4))) [input]
165. ! [X0,X4] : (TREG(X0,X4) <=> (TLC(X0,X4) & PD(X0))) [input]
166. ! [X0,X4] : (::(X0,pbnd,X4) <=> (AT(X4) & TLC(X0,X4) & ? [X1] : tmPP(X0,X1) & PD(X0))) [input]
168. ! [X0,X4] : (::(X0,treg,X4) <=> (EX(X0,X4) & T(X0))) [input]
170. ! [X0,X4] : (::(X0,tinst,X4) <=> (AT(X0) & ::(X0,treg,X4))) [input]
189. ! [X0,X4] : ((TREG(X0,X4) & ? [X4] : ::(X0,pbnd,X4)) => ::(X4,tinst,X4)) [input]
190. ~! [X0,X4] : ((TREG(X0,X4) & ? [X4] : ::(X0,pbnd,X4)) => ::(X4,tinst,X4)) [negated conjecture 189]
193. ! [X0,X1] : (TLC(X0,X1) => (T(X1) & (Q(X0) | PD(X0) | ED(X0)))) [rectify 27]
195. ! [X0,X1,X2] : ((TLC(X0,X2) & TLC(X0,X1)) => X1 = X2) [rectify 29]
304. ! [X0,X1] : (EX(X0,X1) <=> ((T(X1) & ~T(X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1))) [rectify 162]
306. ! [X0,X1] : (TREG(X0,X1) <=> (TLC(X0,X1) & PD(X0))) [rectify 165]
307. ! [X0,X1] : (::(X0,pbnd,X1) <=> (AT(X1) & TLC(X0,X1) & ? [X2] : tmPP(X0,X2) & PD(X0))) [rectify 166]
309. ! [X0,X1] : (::(X0,treg,X1) <=> (EX(X0,X1) & T(X0))) [rectify 168]
311. ! [X0,X1] : (::(X0,tinst,X1) <=> (AT(X0) & ::(X0,treg,X1))) [rectify 170]
335. ~! [X0,X1] : ((TREG(X0,X1) & ? [X2] : ::(X0,pbnd,X2)) => ::(X1,tinst,X1)) [rectify 190]
336. ! [X0,X1] : (TREG(X0,X1) => (TLC(X0,X1) & PD(X0))) [unused predicate definition removal 306]
337. ! [X0] : ((TR(X0) | PR(X0) | AR(X0)) => R(X0)) [unused predicate definition removal 124]
375. ! [X0] : (P(X0,X0) | (~PD(X0) & ~AB(X0))) [ennf transformation 8]
401. ! [X0,X1] : ((T(X1) & (Q(X0) | PD(X0) | ED(X0))) | ~TLC(X0,X1)) [ennf transformation 193]
403. ! [X0,X1,X2] : (X1 = X2 | (~TLC(X0,X2) | ~TLC(X0,X1))) [ennf transformation 195]
404. ! [X0,X1,X2] : (X1 = X2 | ~TLC(X0,X2) | ~TLC(X0,X1)) [flattening 403]
534. ! [X0] : (R(X0) | (~TR(X0) & ~PR(X0) & ~AR(X0))) [ennf transformation 337]
535. ! [X0] : (AB(X0) | ~R(X0)) [ennf transformation 125]
539. ! [X0] : (TR(X0) | ~T(X0)) [ennf transformation 130]
561. ! [X0,X1] : ((TLC(X0,X1) & PD(X0)) | ~TREG(X0,X1)) [ennf transformation 336]
566. ? [X0,X1] : (~::(X1,tinst,X1) & (TREG(X0,X1) & ? [X2] : ::(X0,pbnd,X2))) [ennf transformation 335]
567. ? [X0,X1] : (~::(X1,tinst,X1) & TREG(X0,X1) & ? [X2] : ::(X0,pbnd,X2)) [flattening 566]
570. ! [X1,X0] : (sP1(X1,X0) <=> ((T(X1) & ~T(X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1))) [predicate
      definition introduction]
571. ! [X0,X1] : (EX(X0,X1) <=> sP1(X1,X0)) [definition folding 304,570]
701. ! [X1,X0] : ((sP1(X1,X0) | ((~T(X1) | T(X0) | ~AB(X0)) & (~P(X1,X0) | ~T(X0)) & ~PRE(X0,X1))) & (((T(X1) & ~T
      (X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1)) | ~sP1(X1,X0))) [nnf transformation 570]
702. ! [X1,X0] : ((sP1(X1,X0) | ((~T(X1) | T(X0) | ~AB(X0)) & (~P(X1,X0) | ~T(X0)) & ~PRE(X0,X1))) & ((T(X1) & ~T(
      X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1) | ~sP1(X1,X0))) [flattening 701]
703. ! [X0,X1] : ((sP1(X0,X1) | ((~T(X0) | T(X1) | ~AB(X1)) & (~P(X0,X1) | ~T(X1)) & ~PRE(X1,X0))) & ((T(X0) & ~T(
      X1) & AB(X1)) | (P(X0,X1) & T(X1)) | PRE(X1,X0) | ~sP1(X0,X1))) [rectify 702]
704. ! [X0,X1] : ((EX(X0,X1) | ~sP1(X1,X0)) & (sP1(X1,X0) | ~EX(X0,X1))) [nnf transformation 571]
713. ! [X0,X1] : ((::(X0,pbnd,X1) | (~AT(X1) | ~TLC(X0,X1) | ! [X2] : ~tmPP(X0,X2) | ~PD(X0))) & ((AT(X1) & TLC(X0
      ,X1) & ? [X2] : tmPP(X0,X2) & PD(X0)) | ~::(X0,pbnd,X1))) [nnf transformation 307]
714. ! [X0,X1] : ((::(X0,pbnd,X1) | ~AT(X1) | ~TLC(X0,X1) | ! [X2] : ~tmPP(X0,X2) | ~PD(X0)) & ((AT(X1) & TLC(X0,
      X1) & ? [X2] : tmPP(X0,X2) & PD(X0)) | ~::(X0,pbnd,X1))) [flattening 713]
715. ! [X0,X1] : ((::(X0,pbnd,X1) | ~AT(X1) | ~TLC(X0,X1) | ! [X2] : ~tmPP(X0,X2) | ~PD(X0)) & ((AT(X1) & TLC(X0,
      X1) & ? [X3] : tmPP(X0,X3) & PD(X0)) | ~::(X0,pbnd,X1))) [rectify 714]
716. ! [X0] : (? [X3] : tmPP(X0,X3) => tmPP(X0,sK69(X0))) [choice axiom]
717. ! [X0,X1] : ((::(X0,pbnd,X1) | ~AT(X1) | ~TLC(X0,X1) | ! [X2] : ~tmPP(X0,X2) | ~PD(X0)) & ((AT(X1) & TLC(X0,
      X1) & tmPP(X0,sK69(X0)) & PD(X0)) | ~::(X0,pbnd,X1))) [skolemisation 715,716]
720. ! [X0,X1] : ((::(X0,treg,X1) | (~EX(X0,X1) | ~T(X0))) & ((EX(X0,X1) & T(X0)) | ~::(X0,treg,X1))) [nnf
      transformation 309]
721. ! [X0,X1] : ((::(X0,treg,X1) | ~EX(X0,X1) | ~T(X0)) & ((EX(X0,X1) & T(X0)) | ~::(X0,treg,X1))) [flattening
      720]
724. ! [X0,X1] : ((::(X0,tinst,X1) | (~AT(X0) | ~::(X0,treg,X1))) & ((AT(X0) & ::(X0,treg,X1)) | ~::(X0,tinst,X1))
      ) [nnf transformation 311]
725. ! [X0,X1] : ((::(X0,tinst,X1) | ~AT(X0) | ~::(X0,treg,X1)) & ((AT(X0) & ::(X0,treg,X1)) | ~::(X0,tinst,X1)))
      [flattening 724]
769. ? [X0,X1] : (~::(X1,tinst,X1) & TREG(X0,X1) & ? [X2] : ::(X0,pbnd,X2)) => (~::(sK85,tinst,sK85) & TREG(sK84,
      sK85) & ? [X2] : ::(sK84,pbnd,X2)) [choice axiom]
770. ? [X2] : ::(sK84,pbnd,X2) => ::(sK84,pbnd,sK86) [choice axiom]
771. ~::(sK85,tinst,sK85) & TREG(sK84,sK85) & ::(sK84,pbnd,sK86) [skolemisation 567,770,769]
793. P(X0,X0) | ~AB(X0) [cnf transformation 375]
832. ~TLC(X0,X1) | T(X1) [cnf transformation 401]
836. ~TLC(X0,X2) | X1 = X2 | ~TLC(X0,X1) [cnf transformation 404]
996. ~TR(X0) | R(X0) [cnf transformation 534]
997. ~R(X0) | AB(X0) [cnf transformation 535]
1003. ~T(X0) | TR(X0) [cnf transformation 539]
1033. ~P(X0,X1) | sP1(X0,X1) | ~T(X1) [cnf transformation 703]
1036. ~sP1(X1,X0) | EX(X0,X1) [cnf transformation 704]
1051. ~TREG(X0,X1) | TLC(X0,X1) [cnf transformation 561]
1054. ~::(X0,pbnd,X1) | TLC(X0,X1) [cnf transformation 717]
1055. ~::(X0,pbnd,X1) | AT(X1) [cnf transformation 717]
1063. ::(X0,treg,X1) | ~EX(X0,X1) | ~T(X0) [cnf transformation 721]
1070. ~::(X0,treg,X1) | ~AT(X0) | ::(X0,tinst,X1) [cnf transformation 725]
1127. ::(sK84,pbnd,sK86) [cnf transformation 771]
1128. TREG(sK84,sK85) [cnf transformation 771]
1129. ~::(sK85,tinst,sK85) [cnf transformation 771]
1186. TLC(sK84,sK85) [resolution 1051,1128]
1187. T(sK85) [resolution 1186,832]
1189. TR(sK85) [resolution 1187,1003]
1190. AT(sK86) [resolution 1055,1127]
1201. R(sK85) [resolution 1189,996]
1202. AB(sK85) [resolution 1201,997]
1268. TLC(sK84,sK86) [resolution 1054,1127]
1379. sP1(X1,X1) | ~T(X1) | ~AB(X1) [resolution 1033,793]
1545. ~TLC(sK84,X9) | sK85 = X9 [resolution 836,1186]
```

```
1620. sK85 = sK86 [resolution 1545,1268]
1626. AT(sK85) [backward demodulation 1190,1620]
1946. EX(X2,X2) | ~AB(X2) | ~T(X2) [resolution 1379,1036]
1994. ::(X0,tinst,X1) | ~AT(X0) | ~EX(X0,X1) | ~T(X0) [resolution 1070,1063]
4589. ~AT(sK85) | ~EX(sK85,sK85) | ~T(sK85) [resolution 1994,1129]
4592. ~EX(sK85,sK85) | ~T(sK85) [subsumption resolution 4589,1626]
4593. ~EX(sK85,sK85) [subsumption resolution 4592,1187]
4594. ~AB(sK85) | ~T(sK85) [resolution 4593,1946]
4597. ~T(sK85) [subsumption resolution 4594,1202]
4598. $false [subsumption resolution 4597,1187]
```

Proof of Theorem (t_{db}27)

```
1. ! [X0,X1] : (pP(X0,X1) <=> (~P(X1,X0) & P(X0,X1))) [input]
3. ! [X0] : (AT(X0) <=> (~? [X1] : pP(X1,X0) & (AB(X0) | PD(X0)))) [input]
6. ! [X0,X1] : (P(X0,X1) => (T(X0) <=> T(X1))) [input]
8. ! [X0] : ((PD(X0) | AB(X0)) => P(X0,X0)) [input]
9. ! [X0,X1] : ((P(X1,X0) & P(X0,X1)) => X0 = X1) [input]
27. ! [X0,X4] : (TLC(X0,X4) => (T(X4) & (Q(X0) | PD(X0) | ED(X0)))) [input]
28. ! [X0] : ((Q(X0) | PD(X0) | ED(X0)) => ? [X4] : TLC(X0,X4)) [input]
29. ! [X0,X4,X5] : ((TLC(X0,X5) & TLC(X0,X4)) => X4 = X5) [input]
30. ! [X0,X1,X4,X5] : ((TLC(X1,X5) & TLC(X0,X4) & P(X0,X1)) => P(X4,X5)) [input]
159. ! [X0] : (oPP(X0,X1) <=> (X0 != X1 & oP(X0,X1))) [input]
163. ! [X0,X1] : (oP(X0,X1) <=> (((T(X1) & T(X0)) | (PD(X1) & PD(X0))) & P(X0,X1))) [input]
165. ! [X0,X4] : (TREG(X0,X4) <=> (TLC(X0,X4) & PD(X0))) [input]
166. ! [X0,X4] : (::(X0,pbnd,X4) <=> (AT(X4) & TLC(X0,X4) & ? [X1] : tmPP(X0,X1) & PD(X0))) [input]
167. ! [X0,X4] : (::(X0,proc,X4) <=> (~::(X0,pbnd,X4) & PRE(X0,X4) & PD(X0))) [input]
189. ! [X0,X4] : ((TM(X4) & ? [X4] : (::(X0,pbnd,X4) | ::(X0,proc,X4))) => (TREG(X0,X4) <=> (~? [X9] : (TREG(X0,X9
     ) & oPP(X9,X4)) & ! [X1] : ! [X9] : ((TREG(X1,X9) & oP(X1,X0)) => oP(X9,X4))))) [input]
190. ~! [X0,X4] : ((TM(X4) & ? [X4] : (::(X0,pbnd,X4) | ::(X0,proc,X4))) => (TREG(X0,X4) <=> (~? [X9] : (TREG(X0,
     X9) & oPP(X9,X4)) & ! [X1] : ! [X9] : ((TREG(X1,X9) & oP(X1,X0)) => oP(X9,X4))))) [negated conjecture 189]
193. ! [X0,X1] : (TLC(X0,X1) => (T(X1) & (Q(X0) | PD(X0) | ED(X0)))) [rectify 27]
194. ! [X0] : ((Q(X0) | PD(X0) | ED(X0)) => ? [X1] : TLC(X0,X1)) [rectify 28]
195. ! [X0,X1,X2] : ((TLC(X0,X2) & TLC(X0,X1)) => X1 = X2) [rectify 29]
196. ! [X0,X1,X2,X3] : ((TLC(X1,X3) & TLC(X0,X2) & P(X0,X1)) => P(X2,X3)) [rectify 30]
300. ! [X1] : ! [X0] : (oPP(X0,X1) <=> (X0 != X1 & oP(X0,X1))) [closure 159]
301. ! [X1,X0] : (oPP(X0,X1) <=> (X0 != X1 & oP(X0,X1))) [flattening 300]
306. ! [X0,X1] : (TREG(X0,X1) <=> (TLC(X0,X1) & PD(X0))) [rectify 165]
307. ! [X0,X1] : (::(X0,pbnd,X1) <=> (AT(X1) & TLC(X0,X1) & ? [X2] : tmPP(X0,X2) & PD(X0))) [rectify 166]
308. ! [X0,X1] : (::(X0,proc,X1) <=> (~::(X0,pbnd,X1) & PRE(X0,X1) & PD(X0))) [rectify 167]
335. ~! [X0,X1] : ((TM(X1) & ? [X2] : (::(X0,pbnd,X2) | ::(X0,proc,X2))) => (TREG(X0,X1) <=> (~? [X3] : (TREG(X0,
     X3) & oPP(X3,X1)) & ! [X4] : ! [X5] : ((TREG(X4,X5) & oP(X4,X0)) => oP(X5,X1))))) [rectify 190]
336. ~! [X0,X1] : ((TM(X1) & ? [X2] : (::(X0,pbnd,X2) | ::(X0,proc,X2))) => (TREG(X0,X1) <=> (~? [X3] : (TREG(X0,
     X3) & oPP(X3,X1)) & ! [X4,X5] : ((TREG(X4,X5) & oP(X4,X0)) => oP(X5,X1))))) [flattening 335]
354. ~! [X0,X1] : (? [X2] : (::(X0,pbnd,X2) | ::(X0,proc,X2)) => (TREG(X0,X1) <=> (~? [X3] : (TREG(X0,X3) & oPP(X3
     ,X1)) & ! [X4,X5] : ((TREG(X4,X5) & oP(X4,X0)) => oP(X5,X1))))) [pure predicate removal 336]
370. ! [X0] : (AT(X0) <=> (! [X1] : ~pP(X1,X0) & (AB(X0) | PD(X0)))) [ennf transformation 3]
374. ! [X0,X1] : ((T(X0) <=> T(X1)) | ~P(X0,X1)) [ennf transformation 6]
376. ! [X0] : (P(X0,X0) | (~PD(X0) & ~AB(X0))) [ennf transformation 8]
377. ! [X0,X1] : (X0 = X1 | (~P(X1,X0) | ~P(X0,X1))) [ennf transformation 9]
378. ! [X0,X1] : (X0 = X1 | ~P(X1,X0) | ~P(X0,X1)) [flattening 377]
402. ! [X0,X1] : ((T(X1) & (Q(X0) | PD(X0) | ED(X0))) | ~TLC(X0,X1)) [ennf transformation 193]
403. ! [X0] : (? [X1] : TLC(X0,X1) | (~Q(X0) & ~PD(X0) & ~ED(X0))) [ennf transformation 194]
404. ! [X0,X1,X2] : (X1 = X2 | (~TLC(X0,X2) | ~TLC(X0,X1))) [ennf transformation 195]
405. ! [X0,X1,X2] : (X1 = X2 | ~TLC(X0,X2) | ~TLC(X0,X1)) [flattening 404]
406. ! [X0,X1,X2,X3] : (P(X2,X3) | (~TLC(X1,X3) | ~TLC(X0,X2) | ~P(X0,X1))) [ennf transformation 196]
407. ! [X0,X1,X2,X3] : (P(X2,X3) | ~TLC(X1,X3) | ~TLC(X0,X2) | ~P(X0,X1)) [flattening 406]
566. ? [X0,X1] : ((TREG(X0,X1) <~> (! [X3] : (~TREG(X0,X3) | ~oPP(X3,X1)) & ! [X4,X5] : (oP(X5,X1) | (~TREG(X4,X5)
     | ~oP(X4,X0))))) & ? [X2] : (::(X0,pbnd,X2) | ::(X0,proc,X2))) [ennf transformation 354]
567. ? [X0,X1] : ((TREG(X0,X1) <~> (! [X3] : (~TREG(X0,X3) | ~oPP(X3,X1)) & ! [X4,X5] : (oP(X5,X1) | ~TREG(X4,X5)
     | ~oP(X4,X0)))) & ? [X2] : (::(X0,pbnd,X2) | ::(X0,proc,X2))) [flattening 566]
576. ! [X0,X1] : ((pP(X0,X1) | (P(X1,X0) | ~P(X0,X1))) & ((~P(X1,X0) & P(X0,X1)) | ~pP(X0,X1))) [nnf
     transformation 1]
577. ! [X0,X1] : ((pP(X0,X1) | P(X1,X0) | ~P(X0,X1)) & ((~P(X1,X0) & P(X0,X1)) | ~pP(X0,X1))) [flattening 576]
582. ! [X0] : ((AT(X0) | (? [X1] : pP(X1,X0) | (~AB(X0) & ~PD(X0)))) & ((! [X1] : ~pP(X1,X0) & (AB(X0) | PD(X0)))
     | ~AT(X0))) [nnf transformation 370]
583. ! [X0] : ((AT(X0) | ? [X1] : pP(X1,X0) | (~AB(X0) & ~PD(X0))) & ((! [X1] : ~pP(X1,X0) & (AB(X0) | PD(X0))) |
     ~AT(X0))) [flattening 582]
584. ! [X0] : ((AT(X0) | ? [X1] : pP(X1,X0) | (~AB(X0) & ~PD(X0))) & ((! [X2] : ~pP(X2,X0) & (AB(X0) | PD(X0))) |
     ~AT(X0))) [rectify 583]
585. ! [X0] : (? [X1] : pP(X1,X0) => pP(sK5(X0),X0)) [choice axiom]
586. ! [X0] : ((AT(X0) | pP(sK5(X0),X0) | (~AB(X0) & ~PD(X0))) & ((! [X2] : ~pP(X2,X0) & (AB(X0) | PD(X0))) | ~AT(
     X0))) [skolemisation 584,585]
591. ! [X0,X1] : (((T(X0) | ~T(X1)) & (T(X1) | ~T(X0))) | ~P(X0,X1)) [nnf transformation 374]
613. ! [X0] : (? [X1] : TLC(X0,X1) => TLC(X0,sK14(X0))) [choice axiom]
614. ! [X0] : (TLC(X0,sK14(X0)) | (~Q(X0) & ~PD(X0) & ~ED(X0))) [skolemisation 403,613]
698. ! [X1,X0] : ((oPP(X0,X1) | (X0 = X1 | ~oP(X0,X1))) & ((X0 != X1 & oP(X0,X1)) | ~oPP(X0,X1))) [nnf
     transformation 301]
699. ! [X1,X0] : ((oPP(X0,X1) | X0 = X1 | ~oP(X0,X1)) & ((X0 != X1 & oP(X0,X1)) | ~oPP(X0,X1))) [flattening 698]
700. ! [X0,X1] : ((oPP(X1,X0) | X0 = X1 | ~oP(X1,X0)) & ((X0 != X1 & oP(X1,X0)) | ~oPP(X1,X0))) [rectify 699]
708. ! [X0,X1] : ((oP(X0,X1) | (((~T(X1) | ~T(X0)) & (~PD(X1) | ~PD(X0))) | ~P(X0,X1))) & ((((T(X1) & T(X0)) | (PD
     (X1) & PD(X0))) & P(X0,X1)) | ~oP(X0,X1))) [nnf transformation 163]
709. ! [X0,X1] : ((oP(X0,X1) | ((~T(X1) | ~T(X0)) & (~PD(X1) | ~PD(X0))) | ~P(X0,X1)) & ((((T(X1) & T(X0)) | (PD(
     X1) & PD(X0))) & P(X0,X1)) | ~oP(X0,X1))) [flattening 708]
716. ! [X0,X1] : ((TREG(X0,X1) | (~TLC(X0,X1) | ~PD(X0))) & ((TLC(X0,X1) & PD(X0)) | ~TREG(X0,X1))) [nnf
     transformation 306]
717. ! [X0,X1] : ((TREG(X0,X1) | ~TLC(X0,X1) | ~PD(X0)) & ((TLC(X0,X1) & PD(X0)) | ~TREG(X0,X1))) [flattening 716]
718. ! [X0,X1] : ((::(X0,pbnd,X1) | (~AT(X1) | ~TLC(X0,X1) | ! [X2] : ~tmPP(X0,X2) | ~PD(X0))) & ((AT(X1) & TLC(X0
     ,X1) & ? [X2] : tmPP(X0,X2) & PD(X0)) | ~::(X0,pbnd,X1))) [nnf transformation 307]
```

719. ! [X0,X1] : ((::(X0,pbnd,X1) | ~AT(X1) | ~TLC(X0,X1) | ! [X2] : ~tmPP(X0,X2) | ~PD(X0)) & ((AT(X1) & TLC(X0, X1) & ? [X2] : tmPP(X0,X2) & PD(X0)) | ~::(X0,pbnd,X1))) [flattening 718]
720. ! [X0,X1] : ((::(X0,pbnd,X1) | ~AT(X1) | ~TLC(X0,X1) | ! [X2] : ~tmPP(X0,X2) | ~PD(X0)) & ((AT(X1) & TLC(X0, X1) & ? [X3] : tmPP(X0,X3) & PD(X0)) | ~::(X0,pbnd,X1))) [rectify 719]
721. ! [X0] : (? [X3] : tmPP(X0,X3) => tmPP(X0,sK69(X0))) [choice axiom]
722. ! [X0,X1] : ((::(X0,pbnd,X1) | ~AT(X1) | ~TLC(X0,X1) | ! [X2] : ~tmPP(X0,X2) | ~PD(X0)) & ((AT(X1) & TLC(X0, X1) & tmPP(X0,sK69(X0)) & PD(X0)) | ~::(X0,pbnd,X1))) [skolemisation 720,721]
723. ! [X0,X1] : ((::(X0,proc,X1) | (::(X0,pbnd,X1) | ~PRE(X0,X1) | ~PD(X0))) & ((~::(X0,pbnd,X1) & PRE(X0,X1) & PD(X0)) | ~::(X0,proc,X1))) [nnf transformation 308]
724. ! [X0,X1] : ((::(X0,proc,X1) | ::(X0,pbnd,X1) | ~PRE(X0,X1) | ~PD(X0)) & ((~::(X0,pbnd,X1) & PRE(X0,X1) & PD(X0)) | ~::(X0,proc,X1))) [flattening 723]
774. ? [X0,X1] : ((((? [X3] : (TREG(X0,X3) & oPP(X3,X1)) | ? [X4,X5] : (~oP(X5,X1) & TREG(X4,X5) & oP(X4,X0))) | ~ TREG(X0,X1)) & ((! [X3] : (~TREG(X0,X3) | ~oPP(X3,X1)) & ! [X4,X5] : (oP(X5,X1) | ~TREG(X4,X5) | ~oP(X4,X0))) | TREG(X0,X1))) & ? [X2] : (::(X0,pbnd,X2) | ::(X0,proc,X2))) [nnf transformation 567]
775. ? [X0,X1] : ((? [X3] : (TREG(X0,X3) & oPP(X3,X1)) | ? [X4,X5] : (~oP(X5,X1) & TREG(X4,X5) & oP(X4,X0)) | ~ TREG(X0,X1)) & ((! [X3] : (~TREG(X0,X3) | ~oPP(X3,X1)) & ! [X4,X5] : (oP(X5,X1) | ~TREG(X4,X5) | ~oP(X4,X0))) | TREG(X0,X1)) & ? [X2] : (::(X0,pbnd,X2) | ::(X0,proc,X2))) [flattening 774]
776. ? [X0,X1] : ((? [X2] : (TREG(X0,X2) & oPP(X2,X1)) | ? [X3,X4] : (~oP(X4,X1) & TREG(X3,X4) & oP(X3,X0)) | ~ TREG(X0,X1)) & ((! [X5] : (~TREG(X0,X5) | ~oPP(X5,X1)) & ! [X6,X7] : (oP(X7,X1) | ~TREG(X6,X7) | ~oP(X6,X0))) | TREG(X0,X1)) & ? [X8] : (::(X0,pbnd,X8) | ::(X0,proc,X8))) [rectify 775]
777. ? [X0,X1] : ((? [X2] : (TREG(X0,X2) & oPP(X2,X1)) | ? [X3,X4] : (~oP(X4,X1) & TREG(X3,X4) & oP(X3,X0)) | ~ TREG(X0,X1)) & ((! [X5] : (~TREG(X0,X5) | ~oPP(X5,X1)) & ! [X6,X7] : (oP(X7,X1) | ~TREG(X6,X7) | ~oP(X6,X0))) | TREG(X0,X1)) & ? [X8] : (::(X0,pbnd,X8) | ::(X0,proc,X8))) => ((? [X2] : (TREG(sK84,X2) & oPP(X2,sK85)) | ? [X4,X3] : (~oP(X4,sK85) & TREG(X3,X4) & oP(X3,sK84)) | ~TREG(sK84,sK85)) & ((! [X5] : (~TREG(sK84,X5) | ~oPP(X5,sK85)) & ! [X7,X6] : (oP(X7,sK85) | ~TREG(X6,X7) | ~oP(X6,sK84))) | TREG(sK84,sK85)) & ? [X8] : (::(sK84,pbnd,X8) | ::(sK84,proc,X8))) [choice axiom]
778. ? [X2] : (TREG(sK84,X2) & oPP(X2,sK85)) => (TREG(sK84,sK86) & oPP(sK86,sK85)) [choice axiom]
779. ? [X4,X3] : (~oP(X4,sK85) & TREG(X3,X4) & oP(X3,sK84)) => (~oP(sK88,sK85) & TREG(sK87,sK88) & oP(sK87,sK84)) [choice axiom]
780. ? [X8] : (::(sK84,pbnd,X8) | ::(sK84,proc,X8)) => (::(sK84,pbnd,sK89) | ::(sK84,proc,sK89)) [choice axiom]
781. ((TREG(sK84,sK86) & oPP(sK86,sK85)) | (~oP(sK88,sK85) & TREG(sK87,sK88) & oP(sK87,sK84)) | ~TREG(sK84,sK85)) & ((! [X5] : (~TREG(sK84,X5) | ~oPP(X5,sK85)) & ! [X6,X7] : (oP(X7,sK85) | ~TREG(X6,X7) | ~oP(X6,sK84))) | TREG(sK84,sK85)) & (::(sK84,pbnd,sK89) | ::(sK84,proc,sK89)) [skolemisation 776,780,779,778,777]
784. ~P(X0,X1) | P(X1,X0) | pP(X0,X1) [cnf transformation 577]
789. ~pP(X2,X0) | ~AT(X0) [cnf transformation 586]
800. ~P(X0,X1) | ~T(X1) | T(X0) [cnf transformation 591]
804. P(X0,X0) | ~PD(X0) [cnf transformation 376]
805. ~P(X1,X0) | X0 = X1 | ~P(X0,X1) [cnf transformation 378]
842. ~TLC(X0,X1) | T(X1) [cnf transformation 402]
844. TLC(X0,sK14(X0)) | ~PD(X0) [cnf transformation 614]
846. ~TLC(X0,X2) | X1 = X2 | ~TLC(X0,X1) [cnf transformation 405]
847. ~TLC(X1,X3) | P(X2,X3) | ~TLC(X0,X2) | ~P(X0,X1) [cnf transformation 407]
1033. ~oPP(X1,X0) | oP(X1,X0) [cnf transformation 700]
1034. X0 != X1 | ~oPP(X1,X0) [cnf transformation 700]
1035. ~oP(X1,X0) | X0 = X1 | oPP(X1,X0) [cnf transformation 700]
1050. ~oP(X0,X1) | P(X0,X1) [cnf transformation 709]
1055. ~P(X0,X1) | ~PD(X1) | ~PD(X0) | oP(X0,X1) [cnf transformation 709]
1056. oP(X0,X1) | ~T(X1) | ~T(X0) | ~P(X0,X1) [cnf transformation 709]
1064. ~TREG(X0,X1) | TLC(X0,X1) [cnf transformation 717]
1065. ~TLC(X0,X1) | TREG(X0,X1) | ~PD(X0) [cnf transformation 717]
1066. ~::(X0,pbnd,X1) | PD(X0) [cnf transformation 722]
1068. ~::(X0,pbnd,X1) | TLC(X0,X1) [cnf transformation 722]
1069. ~::(X0,pbnd,X1) | AT(X1) [cnf transformation 722]
1071. ~::(X0,proc,X1) | PD(X0) [cnf transformation 724]
1141. ::(sK84,pbnd,sK89) | ::(sK84,proc,sK89) [cnf transformation 781]
1142. oP(X7,sK85) | ~TREG(X6,X7) | ~oP(X6,sK84) | TREG(sK84,sK85) [cnf transformation 781]
1143. ~TREG(sK84,X5) | ~oPP(X5,sK85) | TREG(sK84,sK85) [cnf transformation 781]
1144. oPP(sK86,sK85) | oP(sK87,sK84) | ~TREG(sK84,sK85) [cnf transformation 781]
1145. oPP(sK86,sK85) | TREG(sK87,sK88) | ~TREG(sK84,sK85) [cnf transformation 781]
1146. oPP(sK86,sK85) | ~oP(sK88,sK85) | ~TREG(sK84,sK85) [cnf transformation 781]
1147. TREG(sK84,sK86) | oP(sK87,sK84) | ~TREG(sK84,sK85) [cnf transformation 781]
1148. TREG(sK84,sK86) | TREG(sK87,sK88) | ~TREG(sK84,sK85) [cnf transformation 781]
1149. TREG(sK84,sK86) | ~oP(sK88,sK85) | ~TREG(sK84,sK85) [cnf transformation 781]
1153. ~oPP(X1,X1) [equality resolution 1034]
1158. ~P(X0,X1) | ~T(X1) | oP(X0,X1) [subsumption resolution 1056,800]
1166. 1 <=> TREG(sK84,sK85) [avatar definition]
1167. TREG(sK84,sK85) <- (1) [avatar component clause 1166]
1170. 2 <=> oP(sK88,sK85) [avatar definition]
1172. ~oP(sK88,sK85) <- (~2) [avatar component clause 1170]
1174. 3 <=> TREG(sK84,sK86) [avatar definition]
1176. TREG(sK84,sK86) <- (3) [avatar component clause 1174]
1177. ~1 | ~2 | 3 [avatar split clause 1149,1174,1170,1166]
1179. 4 <=> TREG(sK87,sK88) [avatar definition]
1181. TREG(sK87,sK88) <- (4) [avatar component clause 1179]
1182. ~1 | 4 | 3 [avatar split clause 1148,1174,1179,1166]
1184. 5 <=> oP(sK87,sK84) [avatar definition]
1186. oP(sK87,sK84) <- (5) [avatar component clause 1184]
1187. ~1 | 5 | 3 [avatar split clause 1147,1174,1184,1166]
1189. 6 <=> oPP(sK86,sK85) [avatar definition]
1191. oPP(sK86,sK85) <- (6) [avatar component clause 1189]
1192. ~1 | ~2 | 6 [avatar split clause 1146,1189,1170,1166]
1193. ~1 | 4 | 6 [avatar split clause 1145,1189,1179,1166]
1194. ~1 | 5 | 6 [avatar split clause 1144,1189,1184,1166]
1196. 7 <=> ! [X5] : (~TREG(sK84,X5) | ~oPP(X5,sK85)) [avatar definition]
1197. ~TREG(sK84,X5) | ~oPP(X5,sK85) <- (7) [avatar component clause 1196]
1198. 1 | 7 [avatar split clause 1143,1196,1166]
1200. 8 <=> ! [X7,X6] : (oP(X7,sK85) | ~oP(X6,sK84) | ~TREG(X6,X7)) [avatar definition]
1201. ~TREG(X6,X7) | ~oP(X6,sK84) | oP(X7,sK85) <- (8) [avatar component clause 1200]
1202. 1 | 8 [avatar split clause 1142,1200,1166]
1204. 9 <=> ::(sK84,proc,sK89) [avatar definition]
1206. ::(sK84,proc,sK89) <- (9) [avatar component clause 1204]

```
1208. 10 <=> ::(sK84,pbnd,sK89) [avatar definition]
1210. ::(sK84,pbnd,sK89) <- (10) [avatar component clause 1208]
1211. 9 | 10 [avatar split clause 1141,1208,1204]
1436. TREG(X1,sK14(X1)) | ~PD(X1) | ~PD(X1) [resolution 1065,844]
1439. TREG(X1,sK14(X1)) | ~PD(X1) [duplicate literal removal 1436]
1480. ~PD(sK84) | ~oPP(sK14(sK84),sK85) <- (7) [resolution 1439,1197]
1560. ~TLC(X3,X2) | sK14(X3) = X2 | ~PD(X3) [resolution 846,844]
1909. ~PD(X0) | ~PD(X0) | oP(X0,X0) | ~PD(X0) [resolution 1055,804]
1923. oP(X0,X0) | ~PD(X0) [duplicate literal removal 1909]
2002. 14 <=> AT(sK89) [avatar definition]
2004. AT(sK89) <- (14) [avatar component clause 2002]
2007. 15 <=> oPP(sK14(sK84),sK85) [avatar definition]
2009. ~oPP(sK14(sK84),sK85) <- (~15) [avatar component clause 2007]
2011. 16 <=> PD(sK84) [avatar definition]
2012. PD(sK84) <- (16) [avatar component clause 2011]
2014. ~15 | ~16 | ~7 [avatar split clause 1480,1196,2011,2007]
2018. AT(sK89) <- (10) [resolution 1210,1069]
2019. PD(sK84) <- (10) [resolution 1210,1066]
2020. 14 | ~10 [avatar split clause 2018,1208,2002]
2021. 16 | ~10 [avatar split clause 2019,1208,2011]
2025. PD(sK84) <- (9) [resolution 1206,1071]
2026. 16 | ~9 [avatar split clause 2025,1204,2011]
2039. TLC(sK84,sK89) <- (10) [resolution 1210,1068]
2048. ~TLC(sK84,X0) | sK89 = X0 <- (10) [resolution 2039,846]
2078. 20 <=> oP(sK84,sK84) [avatar definition]
2079. oP(sK84,sK84) <- (20) [avatar component clause 2078]
2080. ~oP(sK84,sK84) <- (~20) [avatar component clause 2078]
2095. ~PD(sK84) <- (~20) [resolution 2080,1923]
2098. $false <- (16, ~20) [subsumption resolution 2095,2012]
2099. ~16 | 20 [avatar contradiction clause 2098]
2119. oP(sK86,sK85) <- (6) [resolution 1191,1033]
2226. P(sK86,sK85) <- (6) [resolution 2119,1050]
2228. P(sK85,sK86) | pP(sK86,sK85) <- (6) [resolution 2226,784]
2238. sK85 = sK86 | ~P(sK85,sK86) <- (6) [resolution 2226,805]
2244. 23 <=> pP(sK86,sK85) [avatar definition]
2246. pP(sK86,sK85) <- (23) [avatar component clause 2244]
2248. 24 <=> P(sK85,sK86) [avatar definition]
2251. 23 | 24 | ~6 [avatar split clause 2228,1189,2248,2244]
2263. 27 <=> sK85 = sK86 [avatar definition]
2264. sK85 != sK86 <- (~27) [avatar component clause 2263]
2265. sK85 = sK86 <- (27) [avatar component clause 2263]
2266. ~24 | 27 | ~6 [avatar split clause 2238,1189,2263,2248]
2268. TLC(sK87,sK88) <- (4) [resolution 1181,1064]
2659. sK89 = sK14(sK84) | ~PD(sK84) <- (10) [resolution 2048,844]
2706. sK89 = sK14(sK84) <- (10, 16) [subsumption resolution 2659,2012]
2948. oPP(sK85,sK85) <- (6, 27) [forward demodulation 1191,2265]
2949. $false <- (6, 27) [subsumption resolution 2948,1153]
2950. ~6 | ~27 [avatar contradiction clause 2949]
2955. TLC(sK84,sK85) <- (1) [resolution 1167,1064]
2964. P(sK87,sK84) <- (5) [resolution 1186,1050]
3106. ~AT(sK85) <- (23) [resolution 2246,789]
3110. 46 <=> T(sK85) [avatar definition]
3162. 55 <=> sK85 = sK89 [avatar definition]
3164. sK85 = sK89 <- (55) [avatar component clause 3162]
3280. T(sK85) <- (1) [resolution 2955,842]
3282. ~TLC(X3,X2) | P(X2,sK85) | ~P(X3,sK84) <- (1) [resolution 2955,847]
3742. sK85 = sK14(sK84) | ~PD(sK84) <- (1) [resolution 1560,2955]
3746. sK85 = sK14(sK84) <- (1, 16) [subsumption resolution 3742,2012]
3747. sK85 = sK89 <- (1, 10, 16) [backward demodulation 2706,3746]
3748. 55 | ~1 | ~10 | ~16 [avatar split clause 3747,2011,1208,1166,3162]
3752. AT(sK85) <- (14, 55) [backward demodulation 2004,3164]
3791. $false <- (14, 23, 55) [subsumption resolution 3752,3106]
3792. ~14 | ~23 | ~55 [avatar contradiction clause 3791]
13247. P(sK88,sK85) | ~P(sK87,sK84) <- (1, 4) [resolution 3282,2268]
13249. P(sK88,sK85) <- (1, 4, 5) [subsumption resolution 13247,2964]
13269. ~T(sK85) | oP(sK88,sK85) <- (1, 4, 5) [resolution 13249,1158]
13303. ~T(sK85) <- (1, ~2, 4, 5) [subsumption resolution 13269,1172]
13304. ~46 | ~1 | 2 | ~4 | ~5 [avatar split clause 13303,1184,1179,1170,1166,3110]
13344. 46 | ~1 [avatar split clause 3280,1166,3110]
13803. oP(sK14(X0),sK85) | ~oP(X0,sK84) | ~PD(X0) <- (8) [resolution 1201,1439]
19681. 326 <=> sK85 = sK14(sK84) [avatar definition]
19682. sK85 != sK14(sK84) <- (~326) [avatar component clause 19681]
19683. sK85 = sK14(sK84) <- (326) [avatar component clause 19681]
36550. 572 <=> oP(sK14(sK84),sK85) [avatar definition]
36551. ~oP(sK14(sK84),sK85) <- (~572) [avatar component clause 36550]
36552. oP(sK14(sK84),sK85) <- (572) [avatar component clause 36550]
39539. sK85 = sK14(sK84) | oPP(sK14(sK84),sK85) <- (572) [resolution 36552,1035]
39546. oPP(sK14(sK84),sK85) <- (~326, 572) [subsumption resolution 39539,19682]
39547. $false <- (~15, ~326, 572) [subsumption resolution 39546,2009]
39548. 15 | 326 | ~572 [avatar contradiction clause 39547]
39590. ~oP(sK84,sK84) | ~PD(sK84) <- (8, ~572) [resolution 36551,13803]
39591. ~PD(sK84) <- (8, 20, ~572) [subsumption resolution 39590,2079]
39592. $false <- (8, 16, 20, ~572) [subsumption resolution 39591,2012]
39593. ~8 | ~16 | ~20 | 572 [avatar contradiction clause 39592]
43007. TREG(sK84,sK85) | ~PD(sK84) <- (326) [superposition 1439,19683]
43124. TREG(sK84,sK85) <- (16, 326) [subsumption resolution 43007,2012]
43125. 1 | ~16 | ~326 [avatar split clause 43124,19681,2011,1166]
43127. TLC(sK84,sK85) <- (1) [resolution 1167,1064]
43130. TLC(sK84,sK86) <- (3) [resolution 1176,1064]
43143. sK86 = sK14(sK84) | ~PD(sK84) <- (3) [resolution 43130,1560]
43166. sK86 = sK14(sK84) <- (3, 16) [subsumption resolution 43143,2012]
```

```
43167. sK85 = sK86 <- (3, 16, 326) [forward demodulation 43166,19683]
43168. $false <- (3, 16, ~27, 326) [subsumption resolution 43167,2264]
43169. ~3 | ~16 | 27 | ~326 [avatar contradiction clause 43168]
43337. sK85 = sK14(sK84) | ~PD(sK84) <- (1) [resolution 43127,1560]
43360. ~PD(sK84) <- (1, ~326) [subsumption resolution 43337,19682]
43361. $false <- (1, 16, ~326) [subsumption resolution 43360,2012]
43362. ~1 | ~16 | 326 [avatar contradiction clause 43361]
43363. $false [avatar sat refutation 1177,1182,1187,1192,1193,1194,1198,1202,1211,2014,2020,2021,2026,2099,
2251,2266,2950,3748,3792,13304,13344,39548,39593,43125,43169,43362]
```

Proof of Theorem (t_{db}28)

```
6. ! [X0,X1] : (P(X0,X1) => (T(X0) <=> T(X1))) [input]
27. ! [X0,X4] : (TLC(X0,X4) => (T(X4) & (Q(X0) | PD(X0) | ED(X0)))) [input]
30. ! [X0,X1,X4,X5] : ((TLC(X1,X5) & TLC(X0,X4) & P(X0,X1)) => P(X4,X5)) [input]
163. ! [X0,X1] : (oP(X0,X1) <=> (((T(X1) & T(X0)) | (PD(X1) & PD(X0))) & P(X0,X1))) [input]
165. ! [X0,X4] : (TREG(X0,X4) <=> (TLC(X0,X4) & PD(X0))) [input]
189. ! [X0,X1,X4,X9] : ((TREG(X1,X9) & TREG(X0,X4) & oP(X0,X1)) => oP(X4,X9)) [input]
190. ~! [X0,X1,X4,X9] : ((TREG(X1,X9) & TREG(X0,X4) & oP(X0,X1)) => oP(X4,X9)) [negated conjecture 189]
193. ! [X0,X1] : (TLC(X0,X1) => (T(X1) & (Q(X0) | PD(X0) | ED(X0)))) [rectify 27]
196. ! [X0,X1,X2,X3] : ((TLC(X1,X3) & TLC(X0,X2) & P(X0,X1)) => P(X2,X3)) [rectify 30]
306. ! [X0,X1] : (TREG(X0,X1) <=> (TLC(X0,X1) & PD(X0))) [rectify 165]
335. ~! [X0,X1,X2,X3] : ((TREG(X1,X3) & TREG(X0,X2) & oP(X0,X1)) => oP(X2,X3)) [rectify 190]
336. ! [X0,X1] : (TREG(X0,X1) => (TLC(X0,X1) & PD(X0))) [unused predicate definition removal 306]
373. ! [X0,X1] : ((T(X0) <=> T(X1)) | ~P(X0,X1)) [ennf transformation 6]
401. ! [X0,X1] : ((T(X1) & (Q(X0) | PD(X0) | ED(X0))) | ~TLC(X0,X1)) [ennf transformation 193]
405. ! [X0,X1,X2,X3] : (P(X2,X3) | (~TLC(X1,X3) | ~TLC(X0,X2) | ~P(X0,X1))) [ennf transformation 196]
406. ! [X0,X1,X2,X3] : (P(X2,X3) | ~TLC(X1,X3) | ~TLC(X0,X2) | ~P(X0,X1)) [flattening 405]
561. ! [X0,X1] : ((TLC(X0,X1) & PD(X0)) | ~TREG(X0,X1)) [ennf transformation 336]
566. ? [X0,X1,X2,X3] : (~oP(X2,X3) & (TREG(X1,X3) & TREG(X0,X2) & oP(X0,X1))) [ennf transformation 335]
567. ? [X0,X1,X2,X3] : (~oP(X2,X3) & TREG(X1,X3) & TREG(X0,X2) & oP(X0,X1)) [flattening 566]
591. ! [X0,X1] : (((T(X0) | ~T(X1)) & (T(X1) | ~T(X0))) | ~P(X0,X1)) [nnf transformation 373]
705. ! [X0,X1] : ((oP(X0,X1) | (((~T(X1) | ~T(X0)) & (~PD(X1) | ~PD(X0))) | ~P(X0,X1))) & ((((T(X1) & T(X0)) | (PD
(X1) & PD(X0))) & P(X0,X1)) | ~oP(X0,X1))) [nnf transformation 163]
706. ! [X0,X1] : ((oP(X0,X1) | ((~T(X1) | ~T(X0)) & (~PD(X1) | ~PD(X0))) | ~P(X0,X1)) & ((((T(X1) & T(X0)) | (PD(
X1) & PD(X0))) & P(X0,X1)) | ~oP(X0,X1))) [flattening 705]
769. ? [X0,X1,X2,X3] : (~oP(X2,X3) & TREG(X1,X3) & TREG(X0,X2) & oP(X0,X1)) => (~oP(sK86,sK87) & TREG(sK85,sK87) &
TREG(sK84,sK86) & oP(sK84,sK85)) [choice axiom]
770. ~oP(sK86,sK87) & TREG(sK85,sK87) & TREG(sK84,sK86) & oP(sK84,sK85) [skolemisation 567,769]
789. ~P(X0,X1) | ~T(X1) | T(X0) [cnf transformation 591]
831. ~TLC(X0,X1) | T(X1) [cnf transformation 401]
836. ~TLC(X1,X3) | P(X2,X3) | ~TLC(X0,X2) | ~P(X0,X1) [cnf transformation 406]
1036. ~oP(X0,X1) | P(X0,X1) [cnf transformation 706]
1042. oP(X0,X1) | ~T(X1) | ~T(X0) | ~P(X0,X1) [cnf transformation 706]
1050. ~TREG(X0,X1) | TLC(X0,X1) [cnf transformation 561]
1126. oP(sK84,sK85) [cnf transformation 770]
1127. TREG(sK84,sK86) [cnf transformation 770]
1128. TREG(sK85,sK87) [cnf transformation 770]
1129. ~oP(sK86,sK87) [cnf transformation 770]
1137. ~P(X0,X1) | ~T(X1) | oP(X0,X1) [subsumption resolution 1042,789]
1186. P(sK84,sK85) [resolution 1036,1126]
1188. TLC(sK84,sK86) [resolution 1050,1127]
1189. TLC(sK85,sK87) [resolution 1050,1128]
1195. T(sK87) [resolution 1189,831]
2181. ~TLC(X16,X15) | P(X15,sK87) | ~P(X16,sK85) [resolution 836,1189]
12263. P(sK86,sK87) | ~P(sK84,sK85) [resolution 2181,1188]
12265. P(sK86,sK87) [subsumption resolution 12263,1186]
12285. ~T(sK87) | oP(sK86,sK87) [resolution 12265,1137]
12303. oP(sK86,sK87) [subsumption resolution 12285,1195]
12304. $false [subsumption resolution 12303,1129]
```

Proof of Theorem (t_{db}30)

```
101. ! [X0,X3,X4,X8] : ((SLC(X0,X8,X4) & SLC(X0,X3,X4)) => X3 = X8) [input]
172. ! [X0,X3,X4] : (SREG_O(X0,X3,X4) <=> (SLC(X0,X3,X4) & (::(X0,pbnd,X4) | ::(X0,proc,X4)))) [input]
189. ! [X0,X8,X4,X3] : ((SREG_O(X0,X3,X4) & SREG_O(X0,X8,X4)) => X3 = X8) [input]
190. ~! [X0,X8,X4,X3] : ((SREG_O(X0,X3,X4) & SREG_O(X0,X8,X4)) => X3 = X8) [negated conjecture 189]
285. ! [X0,X1,X2,X3] : ((SLC(X0,X3,X2) & SLC(X0,X1,X2)) => X1 = X3) [rectify 101]
313. ! [X0,X1,X2] : (SREG_O(X0,X1,X2) <=> (SLC(X0,X1,X2) & (::(X0,pbnd,X2) | ::(X0,proc,X2)))) [rectify 172]
335. ~! [X0,X1,X2,X3] : ((SREG_O(X0,X3,X2) & SREG_O(X0,X1,X2)) => X1 = X3) [rectify 190]
336. ! [X0,X1,X2] : (SREG_O(X0,X1,X2) => (SLC(X0,X1,X2) & (::(X0,pbnd,X2) | ::(X0,proc,X2)))) [unused predicate
definition removal 313]
508. ! [X0,X1,X2,X3] : (X1 = X3 | (~SLC(X0,X3,X2) | ~SLC(X0,X1,X2))) [ennf transformation 285]
509. ! [X0,X1,X2,X3] : (X1 = X3 | ~SLC(X0,X3,X2) | ~SLC(X0,X1,X2)) [flattening 508]
561. ! [X0,X1,X2] : ((SLC(X0,X1,X2) & (::(X0,pbnd,X2) | ::(X0,proc,X2))) | ~SREG_O(X0,X1,X2)) [ennf transformation
336]
566. ? [X0,X1,X2,X3] : (X1 != X3 & (SREG_O(X0,X3,X2) & SREG_O(X0,X1,X2))) [ennf transformation 335]
567. ? [X0,X1,X2,X3] : (X1 != X3 & SREG_O(X0,X3,X2) & SREG_O(X0,X1,X2)) [flattening 566]
769. ? [X0,X1,X2,X3] : (X1 != X3 & SREG_O(X0,X3,X2) & SREG_O(X0,X1,X2)) => (sK85 != sK87 & SREG_O(sK84,sK87,sK86)
& SREG_O(sK84,sK85,sK86)) [choice axiom]
770. sK85 != sK87 & SREG_O(sK84,sK87,sK86) & SREG_O(sK84,sK85,sK86) [skolemisation 567,769]
961. ~SLC(X0,X3,X2) | X1 = X3 | ~SLC(X0,X1,X2) [cnf transformation 509]
1072. ~SREG_O(X0,X1,X2) | SLC(X0,X1,X2) [cnf transformation 561]
1126. SREG_O(sK84,sK85,sK86) [cnf transformation 770]
1127. SREG_O(sK84,sK87,sK86) [cnf transformation 770]
1128. sK85 != sK87 [cnf transformation 770]
1358. SLC(sK84,sK85,sK86) [resolution 1072,1126]
1359. SLC(sK84,sK87,sK86) [resolution 1072,1127]
1914. ~SLC(sK84,X0,sK86) | sK85 = X0 [resolution 961,1358]
```

```
1918. sK85 = sK87 [resolution 1914,1359]
1920. $false [subsumption resolution 1918,1128]
```

Proof of Theorem (t_{db}31)

```
1 PRE(A,B) -> (ED(A) | PD(A) | Q(A)) & T(B). [assumption].
2 SLC(A,B,C) -> (ED(A) & -AS(A) | PQ(A) | PD(A)) & S(B). [assumption].
3 SLC(A,B,C) -> PRE(A,C). [assumption].
4 P(A,B) & SLC(A,C,D) & SLC(B,E,D) -> P(C,E). [assumption].
5 AR(A) | PR(A) | TR(A) <-> R(A). [assumption].
6 R(A) -> AB(A). [assumption].
7 S(A) -> PR(A). [assumption].
8 T(A) -> TR(A). [assumption].
9 -(exists A (TR(A) & PR(A))). [assumption].
10 EX(A,B) <-> PRE(A,B) | T(A) & P(B,A) | AB(A) & -T(A) & T(B). [assumption].
11 oP(A,B) <-> P(A,B) & (PD(A) & PD(B) | T(A) & T(B)). [assumption].
12 SREG_O(A,B,C) <-> (::(A,proc,C) | ::(A,pbnd,C)) & SLC(A,B,C). [assumption].
13 ::(A,sreg,B) <-> S(A) & EX(A,B). [assumption].
14 ::(A,imen,B) <-> ::(A,site,B) | ::(A,cfbnd,B) | ::(A,sreg,B). [assumption].
15 ::(A,idcnt,B) <-> ::(A,mten,B) | ::(A,imen,B). [assumption].
16 ::(A,cnt,B) <-> ::(A,idcnt,B) | ::(A,sdcnt,B) | ::(A,gdcnt,B). [assumption].
17 cP(A,B,C) <-> ::(A,cnt,C) & ::(B,cnt,C) & (tP(A,B,C) | P(A,B) | (exists D exists E (DQT(A,D) & DQT(B,E) & tP(D,
     E,C)))). [assumption].
18 oP(A,B) & SREG_O(A,C,D) & SREG_O(B,E,D) -> cP(C,E,D). [goal].
19 -R(A) | AB(A). [clausify(6)].
20 EX(A,B) | -AB(A) | T(A) | -T(B). [clausify(10)].
21 -T(A) | TR(A). [clausify(8)].
22 -TR(A) | -PR(A). [clausify(9)].
23 -PR(A) | R(A). [clausify(5)].
24 -S(A) | PR(A). [clausify(7)].
25 -PR(A) | -T(A). [resolve(22,a,21,b)].
26 -SREG_O(A,B,C) | SLC(A,B,C). [clausify(12)].
27 SREG_O(c1,c3,c4). [deny(18)].
28 SREG_O(c2,c5,c4). [deny(18)].
29 -PRE(A,B) | T(B). [clausify(1)].
30 -SLC(A,B,C) | S(B). [clausify(2)].
31 -SLC(A,B,C) | PRE(A,C). [clausify(3)].
32 -P(A,B) | -SLC(A,C,D) | -SLC(B,E,D) | P(C,E). [clausify(4)].
33 -oP(A,B) | P(A,B). [clausify(11)].
34 ::(A,sreg,B) | -S(A) | -EX(A,B). [clausify(13)].
35 ::(A,imen,B) | -::(A,sreg,B). [clausify(14)].
36 ::(A,idcnt,B) | -::(A,imen,B). [clausify(15)].
37 ::(A,cnt,B) | -::(A,idcnt,B). [clausify(16)].
38 cP(A,B,C) | -::(A,cnt,C) | -::(B,cnt,C) | -P(A,B). [clausify(17)].
39 oP(c1,c2). [deny(18)].
40 -cP(c3,c5,c4). [deny(18)].
41 EX(A,B) | T(A) | -T(B) | -R(A). [resolve(20,b,19,b)].
42 -S(A) | R(A). [resolve(24,b,23,a)].
43 -T(A) | -S(A). [resolve(25,a,24,b)].
44 SLC(c1,c3,c4). [resolve(27,a,26,a)].
45 SLC(c2,c5,c4). [resolve(28,a,26,a)].
46 P(c1,c2). [resolve(39,a,33,a)].
47 -P(c1,A) | -SLC(A,B,c4) | P(c3,B). [resolve(44,a,32,b)].
48 PRE(c1,c4). [resolve(44,a,31,a)].
49 S(c3). [resolve(44,a,30,a)].
50 S(c5). [resolve(45,a,30,a)].
51 -T(c3). [resolve(49,a,43,b)].
52 R(c3). [resolve(49,a,42,a)].
53 -T(c5). [resolve(50,a,43,b)].
54 R(c5). [resolve(50,a,42,a)].
55 EX(c3,A) | -T(A). [resolve(52,a,41,d),unit_del(b,51)].
56 EX(c5,A) | -T(A). [resolve(54,a,41,d),unit_del(b,53)].
57 T(c4). [resolve(48,a,29,a)].
58 EX(c3,c4). [resolve(55,b,57,a)].
59 ::(c3,sreg,c4). [resolve(58,a,34,c),unit_del(b,49)].
60 ::(c3,imen,c4). [resolve(59,a,35,b)].
61 ::(c3,idcnt,c4). [resolve(60,a,36,b)].
62 ::(c3,cnt,c4). [resolve(61,a,37,b)].
63 cP(c3,A,c4) | -::(A,cnt,c4) | -P(c3,A). [resolve(62,a,38,b)].
64 EX(c5,c4). [resolve(56,b,57,a)].
65 ::(c5,sreg,c4). [resolve(64,a,34,c),unit_del(b,50)].
66 ::(c5,imen,c4). [resolve(65,a,35,b)].
67 ::(c5,idcnt,c4). [resolve(66,a,36,b)].
68 ::(c5,cnt,c4). [resolve(67,a,37,b)].
69 P(c3,c5). [resolve(47,b,45,a),unit_del(a,46)].
70 $F. [resolve(63,b,68,a),unit_del(a,40),unit_del(b,69)].
```

Proof of Theorem (t_{db}33)

```
1 ED(A) | PD(A) | Q(A) -> (exists B PRE(A,B)). [assumption].
2 PP(A,B) <-> P(A,B) & -P(B,A). [assumption].
3 AT(A) <-> (PD(A) | AB(A)) & -(exists B PP(B,A)). [assumption].
4 AB(A) | PD(A) -> P(A,A). [assumption].
5 P(A,B) & P(B,A) -> A = B. [assumption].
6 PRE(A,B) <-> (exists C (TLC(A,C) & P(B,C))). [assumption].
7 TLC(A,B) -> (ED(A) | PD(A) | Q(A)) & T(B). [assumption].
8 ED(A) | PD(A) | Q(A) -> (exists B TLC(A,B)). [assumption].
9 TLC(A,B) & TLC(A,C) -> B = C. [assumption].
10 PRE(A,B) -> (ED(A) | PD(A) | Q(A)) & T(B). [assumption].
```

```
11 AR(A) | PR(A) | TR(A) <-> R(A). [assumption].
12 R(A) -> AB(A). [assumption].
13 T(A) -> TR(A). [assumption].
14 ::(A,B) <-> (exists C ::(A,B,C)). [assumption].
15 EX(A,B) <-> PRE(A,B) | T(A) & P(B,A) | AB(A) & -T(A) & T(B). [assumption].
16 ::(A,treg,B) <-> T(A) & EX(A,B). [assumption].
17 ::(AtintB) <-> ::(A,treg,B) & -AT(A). [assumption].
18 ::(A,mten,B) <-> ED(A) & PRE(A,B) & (exists C (PRE(A,C) & -AT(C))) & (all C (PRE(A,C) -> (exists D exists E
       exists F (M(D) & SLC(A,E,C) & SLC(D,F,C) & P(F,E))))). [assumption].
19 ::(A,mten) -> (exists B (::(BtintB) & EX(A,B))). [goal].
20 PP(A,B) | -P(A,B) | P(B,A). [clausify(2)].
21 -AT(A) | -PP(B,A). [clausify(3)].
22 -AB(A) | P(A,A). [clausify(4)].
23 -R(A) | AB(A). [clausify(12)].
24 -TR(A) | R(A). [clausify(11)].
25 -T(A) | TR(A). [clausify(13)].
26 -::(A,B) | ::(A,B,f79(A,B)). [clausify(14)].
27 ::(c1,mten). [deny(19)].
28 -ED(A) | PRE(A,f2(A)). [clausify(1)].
29 -P(A,B) | -P(B,A) | B = A. [clausify(5)].
30 -PRE(A,B) | TLC(A,f12(A,B)). [clausify(6)].
31 -PRE(A,B) | P(B,f12(A,B)). [clausify(6)].
32 PRE(A,B) | -TLC(A,C) | -P(B,C). [clausify(6)].
33 -TLC(A,B) | T(B). [clausify(7)].
34 -ED(A) | TLC(A,f13(A)). [clausify(8)].
35 -TLC(A,B) | -TLC(A,C) | C = B. [clausify(9)].
36 -PRE(A,B) | T(B). [clausify(10)].
37 EX(A,B) | -PRE(A,B). [clausify(15)].
38 EX(A,B) | -T(A) | -P(B,A). [clausify(15)].
39 ::(A,treg,B) | -T(A) | -EX(A,B). [clausify(16)].
40 ::(AtintB) | -::(A,treg,B) | AT(A). [clausify(17)].
41 -::(A,mten,B) | ED(A). [clausify(18)].
42 -::(A,mten,B) | PRE(A,B). [clausify(18)].
43 -::(A,mten,B) | PRE(A,f88(B,A)). [clausify(18)].
44 -::(A,mten,B) | -AT(f88(B,A)). [clausify(18)].
45 -::(AtintA) | -EX(c1,A). [deny(19)].
46 -AT(A) | -P(B,A) | P(A,B). [resolve(21,b,20,a)].
47 -R(A) | P(A,A). [resolve(23,b,22,a)].
48 -T(A) | R(A). [resolve(25,b,24,a)].
49 ::(c1,mten,f79(c1,mten)). [resolve(27,a,26,a)].
50 -AT(f88(f79(c1,mten),c1)). [resolve(49,a,44,a)].
51 PRE(c1,f88(f79(c1,mten),c1)). [resolve(49,a,43,a)].
52 PRE(c1,f79(c1,mten)). [resolve(49,a,42,a)].
53 ED(c1). [resolve(49,a,41,a)].
54 TLC(c1,f13(c1)). [resolve(53,a,34,a)].
55 PRE(c1,f2(c1)). [resolve(53,a,28,a)].
56 -TLC(c1,A) | f13(c1) = A. [resolve(54,a,35,b)].
57 T(f13(c1)). [resolve(54,a,33,a)].
58 R(f13(c1)). [resolve(57,a,48,a)].
59 P(f13(c1),f13(c1)). [resolve(58,a,47,a)].
60 TLC(c1,f12(c1,f2(c1))). [resolve(55,a,30,a)].
61 EX(c1,f79(c1,mten)). [resolve(52,a,37,b)].
62 T(f79(c1,mten)). [resolve(52,a,36,a)].
63 P(f79(c1,mten),f12(c1,f79(c1,mten))). [resolve(52,a,31,a)].
64 TLC(c1,f12(c1,f79(c1,mten))). [resolve(52,a,30,a)].
65 R(f79(c1,mten)). [resolve(62,a,48,a)].
66 P(f79(c1,mten),f79(c1,mten)). [resolve(65,a,47,a)].
67 EX(f13(c1),f13(c1)). [resolve(59,a,38,c),unit_del(b,57)].
68 ::(f13(c1),treg,f13(c1)). [resolve(67,a,39,c),unit_del(b,57)].
69 -TLC(c1,A) | f12(c1,f2(c1)) = A. [resolve(60,a,35,b)].
70 ::(f13(c1)tintf13(c1)) | AT(f13(c1)). [resolve(68,a,40,b)].
71 P(f88(f79(c1,mten),c1),f12(c1,f88(f79(c1,mten),c1))). [resolve(51,a,31,a)].
72 TLC(c1,f12(c1,f88(f79(c1,mten),c1))). [resolve(51,a,30,a)].
73 f13(c1) = f12(c1,f2(c1)). [resolve(56,a,60,a)].
74 ::(f12(c1,f2(c1))tintf12(c1,f2(c1))) | AT(f12(c1,f2(c1))). [back_rewrite(70),rewrite([73(2),73(7),73(12)])].
75 -TLC(c1,A) | f12(c1,f79(c1,mten)) = A. [resolve(64,a,35,b)].
76 T(f12(c1,f79(c1,mten))). [resolve(64,a,33,a)].
77 PRE(c1,A) | -P(A,f12(c1,f79(c1,mten))). [resolve(64,a,32,b)].
78 R(f12(c1,f79(c1,mten))). [resolve(76,a,48,a)].
79 P(f12(c1,f79(c1,mten)),f12(c1,f79(c1,mten))). [resolve(78,a,47,a)].
80 EX(f79(c1,mten),f79(c1,mten)). [resolve(66,a,38,c),unit_del(b,62)].
81 ::(f79(c1,mten),treg,f79(c1,mten)). [resolve(80,a,39,c),unit_del(b,62)].
82 ::(f79(c1,mten)tintf79(c1,mten)) | AT(f79(c1,mten)). [resolve(81,a,40,b)].
83 -AT(f12(c1,f79(c1,mten))) | P(f12(c1,f79(c1,mten)),f79(c1,mten)). [resolve(63,a,46,b)].
84 -P(f12(c1,f79(c1,mten)),f79(c1,mten)) | f12(c1,f79(c1,mten)) = f79(c1,mten). [resolve(63,a,29,b),flip(b)].
85 f12(c1,f2(c1)) = f12(c1,f79(c1,mten)). [resolve(69,a,64,a)].
86 ::(f12(c1,f79(c1,mten))tintf12(c1,f79(c1,mten))) | AT(f12(c1,f79(c1,mten))). [back_rewrite(74),rewrite([85(4)
       ,85(10),85(16)])].
87 f12(c1,f88(f79(c1,mten),c1)) = f12(c1,f79(c1,mten)). [resolve(75,a,72,a),flip(a)].
88 P(f88(f79(c1,mten),c1),f12(c1,f79(c1,mten))). [back_rewrite(71),rewrite([87(12)])].
89 PRE(c1,f12(c1,f79(c1,mten))). [resolve(79,a,77,b)].
90 EX(c1,f12(c1,f79(c1,mten))). [resolve(89,a,37,b)].
91 AT(f79(c1,mten)). [resolve(82,a,45,a),unit_del(b,61)].
92 AT(f12(c1,f79(c1,mten))). [resolve(86,a,45,a),unit_del(b,90)].
93 P(f12(c1,f79(c1,mten)),f79(c1,mten)). [back_unit_del(83),unit_del(a,92)].
94 f12(c1,f79(c1,mten)) = f79(c1,mten). [back_unit_del(84),unit_del(a,93)].
95 P(f88(f79(c1,mten),c1),f79(c1,mten)). [back_rewrite(88),rewrite([94(10)])].
96 P(f79(c1,mten),f88(f79(c1,mten),c1)). [resolve(95,a,46,b),unit_del(a,91)].
97 f88(f79(c1,mten),c1) = f79(c1,mten). [resolve(95,a,29,b),unit_del(a,96)].
98 $F. [back_rewrite(50),rewrite([97(5)]),unit_del(a,91)].
```

Proof of Theorem (t_{ab}34)

```
8. ! [X0,X3] : (P(X0,X3) => (T(X0) <=> T(X3))) [input]
10. ! [X0] : ((PD(X0) | AB(X0)) => P(X0,X0)) [input]
36. ! [X0,X1] : (PRE(X0,X1) => (T(X1) & (Q(X0) | PD(X0) | ED(X0)))) [input]
126. ! [X0] : ((TR(X0) | PR(X0) | AR(X0)) <=> R(X0)) [input]
127. ! [X0] : (R(X0) => AB(X0)) [input]
132. ! [X0] : (T(X0) => TR(X0)) [input]
163. ! [X0] : (TM(X0) <=> ::(X0,treg,X0)) [input]
170. ! [X0,X1] : (EX(X0,X1) <=> ((T(X1) & ~T(X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1))) [input]
178. ! [X0,X1] : (::(X0,treg,X1) <=> (EX(X0,X1) & T(X0))) [input]
183. ! [X0,X1] : (::(X0,mten,X1) <=> (! [X5] : (PRE(X0,X5) => ? [X3] : ? [X2] : ? [X8] : (P(X8,X2) & SLC(X3,X8,X5)
     & SLC(X0,X2,X5) & M(X3))) & ? [X5] : (~AT(X5) & PRE(X0,X5)) & PRE(X0,X1) & ED(X0))) [input]
184. ! [X0,X1] : (((::(X0,cfbnd,X1) | ::(X0,site,X1)) <=> (! [X5] : (PRE(X0,X5) => ~? [X3] : ? [X2] : ? [X8] : (P(
     X8,X2) & SLC(X3,X8,X5) & SLC(X0,X2,X5) & M(X3))) & ? [X2] : SLC(X0,X2,X1) & PRE(X0,X1) & F(X0))) [input]
185. ! [X0,X1] : (::(X0,sreg,X1) <=> (EX(X0,X1) & S(X0))) [input]
186. ! [X0,X1] : (::(X0,imen,X1) <=> (::(X0,sreg,X1) | ::(X0,cfbnd,X1) | ::(X0,site,X1))) [input]
187. ! [X0,X1] : (::(X0,idcnt,X1) <=> (::(X0,imen,X1) | ::(X0,mten,X1))) [input]
188. ! [X0,X1] : (::(X0,sdcnt,X1) <=> (? [X3] : (DQT(X0,X3) & ~S(X3) & ::(X3,idcnt,X1)) & PRE(X0,X1) & Q(X0))) [
     input]
190. ! [X0,X1] : (::(X0,gdcnt,X1) <=> (! [X5] : (PRE(X0,X5) => ? [X3] : CONCR(X3,X0,X5)) & PRE(X0,X1))) [input]
191. ! [X0,X1] : (::(X0,cnt,X1) <=> (::(X0,gdcnt,X1) | ::(X0,sdcnt,X1) | ::(X0,idcnt,X1))) [input]
192. ! [X0,X3,X1] : (cP(X0,X3,X1) <=> ((? [X4] : ? [X5] : (tP(X4,X5,X1) & DQT(X3,X5) & DQT(X0,X4)) | P(X0,X3) | tP
     (X0,X3,X1)) & ::(X3,cnt,X1) & ::(X0,cnt,X1))) [input]
199. ! [X0,X3,X1] : (cP(X0,X3,X1) => (TM(X1) & ::(X3,cnt,X1) & ::(X0,cnt,X1))) [input]
200. ~! [X0,X3,X1] : (cP(X0,X3,X1) => (TM(X1) & ::(X3,cnt,X1) & ::(X0,cnt,X1))) [negated conjecture 199]
206. ! [X0,X1] : (P(X0,X1) => (T(X0) <=> T(X1))) [rectify 8]
344. ! [X0,X1] : (::(X0,mten,X1) <=> (! [X2] : (PRE(X0,X2) => ? [X3] : ? [X4] : ? [X5] : (P(X5,X4) & SLC(X3,X5,X2)
     & SLC(X0,X4,X2) & M(X3))) & ? [X6] : (~AT(X6) & PRE(X0,X6)) & PRE(X0,X1) & ED(X0))) [rectify 183]
345. ! [X0,X1] : (::(X0,mten,X1) <=> (! [X2] : (PRE(X0,X2) => ? [X3,X4,X5] : (P(X5,X4) & SLC(X3,X5,X2) & SLC(X0,X4
     ,X2) & M(X3))) & ? [X6] : (~AT(X6) & PRE(X0,X6)) & PRE(X0,X1) & ED(X0))) [flattening 344]
346. ! [X0,X1] : (((::(X0,cfbnd,X1) | ::(X0,site,X1)) <=> (! [X2] : (PRE(X0,X2) => ~? [X3] : ? [X4] : ? [X5] : (P(
     X5,X4) & SLC(X3,X5,X2) & SLC(X0,X4,X2) & M(X3))) & ? [X6] : SLC(X0,X6,X1) & PRE(X0,X1) & F(X0))) [rectify
     184]
347. ! [X0,X1] : (((::(X0,cfbnd,X1) | ::(X0,site,X1)) <=> (! [X2] : (PRE(X0,X2) => ~? [X3,X4,X5] : (P(X5,X4) & SLC(
     X3,X5,X2) & SLC(X0,X4,X2) & M(X3))) & ? [X6] : SLC(X0,X6,X1) & PRE(X0,X1) & F(X0))) [flattening 346]
348. ! [X0,X1] : (::(X0,sdcnt,X1) <=> (? [X2] : (DQT(X0,X2) & ~S(X2) & ::(X2,idcnt,X1)) & PRE(X0,X1) & Q(X0))) [
     rectify 188]
350. ! [X0,X1] : (::(X0,gdcnt,X1) <=> (! [X2] : (PRE(X0,X2) => ? [X3] : CONCR(X3,X0,X2)) & PRE(X0,X1))) [rectify
     190]
351. ! [X0,X1,X2] : (cP(X0,X1,X2) <=> ((? [X3] : ? [X4] : (tP(X3,X4,X2) & DQT(X1,X4) & DQT(X0,X3)) | P(X0,X1) | tP
     (X0,X1,X2)) & ::(X1,cnt,X2) & ::(X0,cnt,X2))) [rectify 192]
352. ! [X0,X1,X2] : (cP(X0,X1,X2) <=> ((? [X3,X4] : (tP(X3,X4,X2) & DQT(X1,X4) & DQT(X0,X3)) | P(X0,X1) | tP(X0,X1
     ,X2)) & ::(X1,cnt,X2) & ::(X0,cnt,X2))) [flattening 351]
362. ~! [X0,X1,X2] : (cP(X0,X1,X2) => (TM(X2) & ::(X1,cnt,X2) & ::(X0,cnt,X2))) [rectify 200]
363. ! [X0,X1,X2] : (cP(X0,X1,X2) => ((? [X3,X4] : (tP(X3,X4,X2) & DQT(X1,X4) & DQT(X0,X3)) | P(X0,X1) | tP(X0,X1,
     X2)) & ::(X1,cnt,X2) & ::(X0,cnt,X2))) [unused predicate definition removal 352]
364. ! [X0] : (::(X0,treg,X0) => TM(X0)) [unused predicate definition removal 163]
365. ! [X0] : ((TR(X0) | PR(X0) | AR(X0)) => R(X0)) [unused predicate definition removal 126]
404. ! [X0,X1] : ((T(X0) <=> T(X1)) | ~P(X0,X1)) [ennf transformation 206]
406. ! [X0] : (P(X0,X0) | (~PD(X0) & ~AB(X0))) [ennf transformation 10]
443. ! [X0,X1] : ((T(X1) & (Q(X0) | PD(X0) | ED(X0))) | ~PRE(X0,X1)) [ennf transformation 36]
565. ! [X0] : (R(X0) | (~TR(X0) & ~PR(X0) & ~AR(X0))) [ennf transformation 365]
566. ! [X0] : (AB(X0) | ~R(X0)) [ennf transformation 127]
570. ! [X0] : (TR(X0) | ~T(X0)) [ennf transformation 132]
590. ! [X0] : (TM(X0) | ~::(X0,treg,X0)) [ennf transformation 364]
593. ! [X0,X1] : (::(X0,mten,X1) <=> (! [X2] : (? [X3,X4,X5] : (P(X5,X4) & SLC(X3,X5,X2) & SLC(X0,X4,X2) & M(X3))
     | ~PRE(X0,X2)) & ? [X6] : (~AT(X6) & PRE(X0,X6)) & PRE(X0,X1) & ED(X0))) [ennf transformation 345]
594. ! [X0,X1] : (((::(X0,cfbnd,X1) | ::(X0,site,X1)) <=> (! [X2] : (! [X3,X4,X5] : (~P(X5,X4) | ~SLC(X3,X5,X2) | ~
     SLC(X0,X4,X2) | ~M(X3)) | ~PRE(X0,X2)) & ? [X6] : SLC(X0,X6,X1) & PRE(X0,X1) & F(X0))) [ennf transformation
     347]
596. ! [X0,X1] : (::(X0,gdcnt,X1) <=> (! [X2] : (? [X3] : CONCR(X3,X0,X2) | ~PRE(X0,X2)) & PRE(X0,X1))) [ennf
     transformation 350]
597. ! [X0,X1,X2] : (((? [X3,X4] : (tP(X3,X4,X2) & DQT(X1,X4) & DQT(X0,X3)) | P(X0,X1) | tP(X0,X1,X2)) & ::(X1,cnt
     ,X2) & ::(X0,cnt,X2)) | ~cP(X0,X1,X2)) [ennf transformation 363]
598. ? [X0,X1,X2] : ((~TM(X2) | ~::(X1,cnt,X2) | ~::(X0,cnt,X2)) & cP(X0,X1,X2)) [ennf transformation 362]
601. ! [X1,X0] : (sP1(X1,X0) <=> ((T(X1) & ~T(X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1))) [predicate
     definition introduction]
602. ! [X0,X1] : (EX(X0,X1) <=> sP1(X1,X0)) [definition folding 170,601]
603. ! [X0,X1] : (sP2(X0,X1) <=> (! [X2] : (? [X3,X4,X5] : (P(X5,X4) & SLC(X3,X5,X2) & SLC(X0,X4,X2) & M(X3)) | ~
     PRE(X0,X2)) & ? [X6] : (~AT(X6) & PRE(X0,X6)) & PRE(X0,X1) & ED(X0))) [predicate definition introduction]
604. ! [X0,X1] : (::(X0,mten,X1) <=> sP2(X0,X1)) [definition folding 593,603]
605. ! [X0,X1] : (sP3(X0,X1) <=> (! [X2] : (! [X3,X4,X5] : (~P(X5,X4) | ~SLC(X3,X5,X2) | ~SLC(X0,X4,X2) | ~M(X3))
     | ~PRE(X0,X2)) & ? [X6] : SLC(X0,X6,X1) & PRE(X0,X1) & F(X0))) [predicate definition introduction]
606. ! [X0,X1] : (((::(X0,cfbnd,X1) | ::(X0,site,X1)) <=> sP3(X0,X1)) [definition folding 594,605]
626. ! [X0,X1] : (((T(X0) | ~T(X1)) & (T(X1) | ~T(X0))) | ~P(X0,X1)) [nnf transformation 404]
735. ! [X1,X0] : ((sP1(X1,X0) | ((~T(X1) | T(X0) | ~AB(X0)) & (~P(X1,X0) | ~T(X0)) & ~PRE(X0,X1))) & (((T(X1) & ~T
     (X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1)) | ~sP1(X1,X0))) [nnf transformation 601]
736. ! [X1,X0] : ((sP1(X1,X0) | ((~T(X1) | T(X0) | ~AB(X0)) & (~P(X1,X0) | ~T(X0)) & ~PRE(X0,X1))) & ((T(X1) & ~T(
     X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1) | ~sP1(X1,X0))) [flattening 735]
737. ! [X0,X1] : ((sP1(X0,X1) | ((~T(X0) | T(X1) | ~AB(X1)) & (~P(X0,X1) | ~T(X1)) & ~PRE(X1,X0))) & ((T(X0) & ~T(
     X1) & AB(X1)) | (P(X0,X1) & T(X1)) | PRE(X1,X0) | ~sP1(X0,X1))) [rectify 736]
738. ! [X0,X1] : ((EX(X0,X1) | ~sP1(X1,X0)) & (sP1(X1,X0) | ~EX(X0,X1))) [nnf transformation 602]
754. ! [X0,X1] : ((::(X0,treg,X1) | (~EX(X0,X1) | ~T(X0))) & ((EX(X0,X1) & T(X0)) | ~::(X0,treg,X1))) [nnf
     transformation 178]
755. ! [X0,X1] : ((::(X0,treg,X1) | ~EX(X0,X1) | ~T(X0)) & ((EX(X0,X1) & T(X0)) | ~::(X0,treg,X1))) [flattening
     754]
762. ! [X0,X1] : ((sP2(X0,X1) | (? [X2] : (! [X3,X4,X5] : (~P(X5,X4) | ~SLC(X3,X5,X2) | ~SLC(X0,X4,X2) | ~M(X3)) &
     PRE(X0,X2)) | ! [X6] : (AT(X6) | ~PRE(X0,X6)) | ~PRE(X0,X1) | ~ED(X0))) & ((! [X2] : (? [X3,X4,X5] : (P(X5,
     X4) & SLC(X3,X5,X2) & SLC(X0,X4,X2) & M(X3)) | ~PRE(X0,X2)) & ? [X6] : (~AT(X6) & PRE(X0,X6)) & PRE(X0,X1) &
     ED(X0)) | ~sP2(X0,X1))) [nnf transformation 603]
```

763. ! [X0,X1] : ((sP2(X0,X1) | ? [X2] : (! [X3,X4,X5] : (~P(X5,X4) | ~SLC(X3,X5,X2) | ~SLC(X0,X4,X2) | ~M(X3)) & PRE(X0,X2)) | ! [X6] : (AT(X6) | ~PRE(X0,X6)) | ~PRE(X0,X1) | ~ED(X0)) & ((! [X2] : (? [X3,X4,X5] : (P(X5,X4) & SLC(X3,X5,X2) & SLC(X0,X4,X2) & M(X3)) | ~PRE(X0,X2)) & ? [X6] : (~AT(X6) & PRE(X0,X6)) & PRE(X0,X1) & ED(X0)) | ~sP2(X0,X1))) [flattening 762]
764. ! [X0,X1] : ((sP2(X0,X1) | ? [X2] : (! [X3,X4,X5] : (~P(X5,X4) | ~SLC(X3,X5,X2) | ~SLC(X0,X4,X2) | ~M(X3)) & PRE(X0,X2)) | ! [X6] : (AT(X6) | ~PRE(X0,X6)) | ~PRE(X0,X1) | ~ED(X0)) & ((! [X7] : (? [X8,X9,X10] : (P(X10,X9) & SLC(X8,X10,X7) & SLC(X0,X9,X7) & M(X8)) | ~PRE(X0,X7)) & ? [X11] : (~AT(X11) & PRE(X0,X11)) & PRE(X0,X1) & ED(X0)) | ~sP2(X0,X1))) [rectify 763]
765. ! [X0] : (? [X2] : (! [X3,X4,X5] : (~P(X5,X4) | ~SLC(X3,X5,X2) | ~SLC(X0,X4,X2) | ~M(X3)) & PRE(X0,X2)) => (! [X5,X4,X3] : (~P(X5,X4) | ~SLC(X3,X5,sK72(X0)) | ~SLC(X0,X4,sK72(X0)) | ~M(X3)) & PRE(X0,sK72(X0)))) [choice axiom]
766. ! [X7,X0] : (? [X8,X9,X10] : (P(X10,X9) & SLC(X8,X10,X7) & SLC(X0,X9,X7) & M(X8)) => (P(sK75(X0,X7),sK74(X0,X7)) & SLC(sK73(X0,X7),sK75(X0,X7),X7) & SLC(X0,sK74(X0,X7),X7) & M(sK73(X0,X7)))) [choice axiom]
767. ! [X0] : (? [X11] : (~AT(X11) & PRE(X0,X11)) => (~AT(sK76(X0)) & PRE(X0,sK76(X0)))) [choice axiom]
768. ! [X0,X1] : ((sP2(X0,X1) | (! [X3,X4,X5] : (~P(X5,X4) | ~SLC(X3,X5,sK72(X0)) | ~SLC(X0,X4,sK72(X0)) | ~M(X3)) & PRE(X0,sK72(X0))) | ! [X6] : (AT(X6) | ~PRE(X0,X6)) | ~PRE(X0,X1) | ~ED(X0)) & ((! [X7] : ((P(sK75(X0,X7),sK74(X0,X7)) & SLC(sK73(X0,X7),sK75(X0,X7),X7) & SLC(X0,sK74(X0,X7),X7) & M(sK73(X0,X7))) | ~PRE(X0,X7)) & (~AT(sK76(X0)) & PRE(X0,sK76(X0)))) & PRE(X0,X1) & ED(X0)) | ~sP2(X0,X1))) [skolemisation 764,767,766,765]
769. ! [X0,X1] : ((::(X0,mten,X1) | ~sP2(X0,X1)) & (sP2(X0,X1) | ~::(X0,mten,X1))) [nnf transformation 604]
770. ! [X0,X1] : ((sP3(X0,X1) | (? [X2] : (? [X3,X4,X5] : (P(X5,X4) & SLC(X3,X5,X2) & SLC(X0,X4,X2) & M(X3)) & PRE(X0,X2)) | ! [X6] : ~SLC(X0,X6,X1) | ~PRE(X0,X1) | ~F(X0))) & ((! [X2] : (! [X3,X4,X5] : (~P(X5,X4) | ~SLC(X3,X5,X2) | ~SLC(X0,X4,X2) | ~M(X3)) | ~PRE(X0,X2)) & ? [X6] : SLC(X0,X6,X1) & PRE(X0,X1) & F(X0)) | ~sP3(X0,X1))) [nnf transformation 605]
771. ! [X0,X1] : ((sP3(X0,X1) | ? [X2] : (? [X3,X4,X5] : (P(X5,X4) & SLC(X3,X5,X2) & SLC(X0,X4,X2) & M(X3)) & PRE(X0,X2)) | ! [X6] : ~SLC(X0,X6,X1) | ~PRE(X0,X1) | ~F(X0)) & ((! [X2] : (! [X3,X4,X5] : (~P(X5,X4) | ~SLC(X3,X5,X2) | ~SLC(X0,X4,X2) | ~M(X3)) | ~PRE(X0,X2)) & ? [X6] : SLC(X0,X6,X1) & PRE(X0,X1) & F(X0)) | ~sP3(X0,X1))) [flattening 770]
772. ! [X0,X1] : ((sP3(X0,X1) | ? [X2] : (? [X3,X4,X5] : (P(X5,X4) & SLC(X3,X5,X2) & SLC(X0,X4,X2) & M(X3)) & PRE(X0,X2)) | ! [X6] : ~SLC(X0,X6,X1) | ~PRE(X0,X1) | ~F(X0)) & ((! [X7] : (! [X8,X9,X10] : (~P(X10,X9) | ~SLC(X8,X10,X7) | ~SLC(X0,X9,X7) | ~M(X8)) | ~PRE(X0,X7)) & ? [X11] : SLC(X0,X11,X1) & PRE(X0,X1) & F(X0)) | ~sP3(X0,X1))) [rectify 771]
773. ! [X0] : (? [X2] : (? [X3,X4,X5] : (P(X5,X4) & SLC(X3,X5,X2) & SLC(X0,X4,X2) & M(X3)) & PRE(X0,X2)) => (? [X5,X4,X3] : (P(X5,X4) & SLC(X3,X5,sK77(X0)) & SLC(X0,X4,sK77(X0)) & M(X3)) & PRE(X0,sK77(X0)))) [choice axiom]
774. ! [X0] : (? [X5,X4,X3] : (P(X5,X4) & SLC(X3,X5,sK77(X0)) & SLC(X0,X4,sK77(X0)) & M(X3)) => (P(sK80(X0),sK79(X0)) & SLC(sK78(X0),sK80(X0),sK77(X0)) & SLC(X0,sK79(X0),sK77(X0)) & M(sK78(X0)))) [choice axiom]
775. ! [X1,X0] : (? [X11] : SLC(X0,X11,X1) => SLC(X0,sK81(X0,X1),X1)) [choice axiom]
776. ! [X0,X1] : ((sP3(X0,X1) | ((P(sK80(X0),sK79(X0)) & SLC(sK78(X0),sK80(X0),sK77(X0)) & SLC(X0,sK79(X0),sK77(X0)) & M(sK78(X0))) & PRE(X0,sK77(X0))) | ! [X6] : ~SLC(X0,X6,X1) | ~PRE(X0,X1) | ~F(X0)) & ((! [X7] : (! [X8,X9,X10] : (~P(X10,X9) | ~SLC(X8,X10,X7) | ~SLC(X0,X9,X7) | ~M(X8)) | ~PRE(X0,X7)) & SLC(X0,sK81(X0,X1),X1) & PRE(X0,X1) & F(X0)) | ~sP3(X0,X1))) [skolemisation 772,775,774,773]
777. ! [X0,X1] : (((::(X0,cfbnd,X1) | ::(X0,site,X1)) | ~sP3(X0,X1)) & (sP3(X0,X1) | (~::(X0,cfbnd,X1) & ~::(X0,site,X1)))) [nnf transformation 606]
778. ! [X0,X1] : ((::(X0,cfbnd,X1) | ::(X0,site,X1) | ~sP3(X0,X1)) & (sP3(X0,X1) | (~::(X0,cfbnd,X1) & ~::(X0,site,X1)))) [flattening 777]
779. ! [X0,X1] : ((::(X0,sreg,X1) | (~EX(X0,X1) | ~S(X0))) & ((EX(X0,X1) & S(X0)) | ~::(X0,sreg,X1))) [nnf transformation 185]
780. ! [X0,X1] : ((::(X0,sreg,X1) | ~EX(X0,X1) | ~S(X0)) & ((EX(X0,X1) & S(X0)) | ~::(X0,sreg,X1))) [flattening 779]
781. ! [X0,X1] : ((::(X0,imen,X1) | (~::(X0,sreg,X1) & ~::(X0,cfbnd,X1) & ~::(X0,site,X1))) & ((::(X0,sreg,X1) | ::(X0,cfbnd,X1) | ::(X0,site,X1)) | ~::(X0,imen,X1))) [nnf transformation 186]
782. ! [X0,X1] : ((::(X0,imen,X1) | (~::(X0,sreg,X1) & ~::(X0,cfbnd,X1) & ~::(X0,site,X1))) & (::(X0,sreg,X1) | ::(X0,cfbnd,X1) | ::(X0,site,X1) | ~::(X0,imen,X1))) [flattening 781]
783. ! [X0,X1] : ((::(X0,idcnt,X1) | (~::(X0,imen,X1) & ~::(X0,mten,X1))) & ((::(X0,imen,X1) | ::(X0,mten,X1)) | ~::(X0,idcnt,X1))) [nnf transformation 187]
784. ! [X0,X1] : ((::(X0,idcnt,X1) | (~::(X0,imen,X1) & ~::(X0,mten,X1))) & (::(X0,imen,X1) | ::(X0,mten,X1) | ~::(X0,idcnt,X1))) [flattening 783]
785. ! [X0,X1] : ((::(X0,sdcnt,X1) | (! [X2] : (~DQT(X0,X2) | S(X2) | ~::(X2,idcnt,X1)) | ~PRE(X0,X1) | ~Q(X0))) & ((? [X2] : (DQT(X0,X2) & ~S(X2) & ::(X2,idcnt,X1)) & PRE(X0,X1) & Q(X0)) | ~::(X0,sdcnt,X1))) [nnf transformation 348]
786. ! [X0,X1] : ((::(X0,sdcnt,X1) | ! [X2] : (~DQT(X0,X2) | S(X2) | ~::(X2,idcnt,X1)) | ~PRE(X0,X1) | ~Q(X0)) & ((? [X2] : (DQT(X0,X2) & ~S(X2) & ::(X2,idcnt,X1)) & PRE(X0,X1) & Q(X0)) | ~::(X0,sdcnt,X1))) [flattening 785]
787. ! [X0,X1] : ((::(X0,sdcnt,X1) | ! [X2] : (~DQT(X0,X2) | S(X2) | ~::(X2,idcnt,X1)) | ~PRE(X0,X1) | ~Q(X0)) & ((? [X3] : (DQT(X0,X3) & ~S(X3) & ::(X3,idcnt,X1)) & PRE(X0,X1) & Q(X0)) | ~::(X0,sdcnt,X1))) [rectify 786]
788. ! [X1,X0] : (? [X3] : (DQT(X0,X3) & ~S(X3) & ::(X3,idcnt,X1)) => (DQT(X0,sK82(X0,X1)) & ~S(sK82(X0,X1)) & ::(sK82(X0,X1),idcnt,X1))) [choice axiom]
789. ! [X0,X1] : ((::(X0,sdcnt,X1) | ! [X2] : (~DQT(X0,X2) | S(X2) | ~::(X2,idcnt,X1)) | ~PRE(X0,X1) | ~Q(X0)) & (((DQT(X0,sK82(X0,X1)) & ~S(sK82(X0,X1)) & ::(sK82(X0,X1),idcnt,X1)) & PRE(X0,X1) & Q(X0)) | ~::(X0,sdcnt,X1))) [skolemisation 787,788]
795. ! [X0,X1] : ((::(X0,gdcnt,X1) | (? [X2] : (! [X3] : ~CONCR(X3,X0,X2) & PRE(X0,X2)) | ~PRE(X0,X1))) & ((! [X2] : (? [X3] : CONCR(X3,X0,X2) | ~PRE(X0,X2)) & PRE(X0,X1)) | ~::(X0,gdcnt,X1))) [nnf transformation 596]
796. ! [X0,X1] : ((::(X0,gdcnt,X1) | ? [X2] : (! [X3] : ~CONCR(X3,X0,X2) & PRE(X0,X2)) | ~PRE(X0,X1)) & ((! [X2] : (? [X3] : CONCR(X3,X0,X2) | ~PRE(X0,X2)) & PRE(X0,X1)) | ~::(X0,gdcnt,X1))) [flattening 795]
797. ! [X0,X1] : ((::(X0,gdcnt,X1) | ? [X2] : (! [X3] : ~CONCR(X3,X0,X2) & PRE(X0,X2)) | ~PRE(X0,X1)) & ((! [X4] : (? [X5] : CONCR(X5,X0,X4) | ~PRE(X0,X4)) & PRE(X0,X1)) | ~::(X0,gdcnt,X1))) [rectify 796]
798. ! [X0] : (? [X2] : (! [X3] : ~CONCR(X3,X0,X2) & PRE(X0,X2)) => (! [X3] : ~CONCR(X3,X0,sK84(X0)) & PRE(X0,sK84(X0)))) [choice axiom]
799. ! [X4,X0] : (? [X5] : CONCR(X5,X0,X4) => CONCR(sK85(X0,X4),X0,X4)) [choice axiom]
800. ! [X0,X1] : ((::(X0,gdcnt,X1) | (! [X3] : ~CONCR(X3,X0,sK84(X0)) & PRE(X0,sK84(X0))) | ~PRE(X0,X1)) & ((! [X4] : (CONCR(sK85(X0,X4),X0,X4) | ~PRE(X0,X4)) & PRE(X0,X1)) | ~::(X0,gdcnt,X1))) [skolemisation 797,799,798]
801. ! [X0,X1] : ((::(X0,cnt,X1) | (~::(X0,gdcnt,X1) & ~::(X0,sdcnt,X1) & ~::(X0,idcnt,X1))) & ((::(X0,gdcnt,X1) | ::(X0,sdcnt,X1) | ::(X0,idcnt,X1)) | ~::(X0,cnt,X1))) [nnf transformation 191]
802. ! [X0,X1] : ((::(X0,cnt,X1) | (~::(X0,gdcnt,X1) & ~::(X0,sdcnt,X1) & ~::(X0,idcnt,X1))) & (::(X0,gdcnt,X1) | ::(X0,sdcnt,X1) | ::(X0,idcnt,X1) | ~::(X0,cnt,X1))) [flattening 801]
803. ! [X2,X1,X0] : (? [X3,X4] : (tP(X3,X4,X2) & DQT(X1,X4) & DQT(X0,X3)) => (tP(sK86(X0,X1,X2),sK87(X0,X1,X2),X2) & DQT(X1,sK87(X0,X1,X2)) & DQT(X0,sK86(X0,X1,X2)))) [choice axiom]
804. ! [X0,X1,X2] : ((((tP(sK86(X0,X1,X2),sK87(X0,X1,X2),X2) & DQT(X1,sK87(X0,X1,X2)) & DQT(X0,sK86(X0,X1,X2))) | P(X0,X1) | tP(X0,X1,X2)) & ::(X1,cnt,X2) & ::(X0,cnt,X2)) | ~cP(X0,X1,X2)) [skolemisation 597,803]
805. ? [X0,X1,X2] : ((~TM(X2) | ~::(X1,cnt,X2) | ~::(X0,cnt,X2)) & cP(X0,X1,X2)) => ((~TM(sK90) | ~::(sK89,cnt,sK90) | ~::(sK88,cnt,sK90)) & cP(sK88,sK89,sK90)) [choice axiom]
806. (~TM(sK90) | ~::(sK89,cnt,sK90) | ~::(sK88,cnt,sK90)) & cP(sK88,sK89,sK90) [skolemisation 598,805]

```
829. ~P(X0,X1) | ~T(X1) | T(X0) [cnf transformation 626]
832. P(X0,X0) | ~AB(X0) [cnf transformation 406]
884. ~PRE(X0,X1) | T(X1) [cnf transformation 443]
1036. ~TR(X0) | R(X0) [cnf transformation 565]
1037. ~R(X0) | AB(X0) [cnf transformation 566]
1043. ~T(X0) | TR(X0) [cnf transformation 570]
1066. ~::(X0,treg,X0) | TM(X0) [cnf transformation 590]
1071. T(X0) | T(X1) | PRE(X1,X0) | ~sP1(X0,X1) [cnf transformation 737]
1072. T(X0) | P(X0,X1) | PRE(X1,X0) | ~sP1(X0,X1) [cnf transformation 737]
1074. ~P(X0,X1) | sP1(X0,X1) | ~T(X1) [cnf transformation 737]
1076. ~EX(X0,X1) | sP1(X1,X0) [cnf transformation 738]
1077. ~sP1(X1,X0) | EX(X0,X1) [cnf transformation 738]
1102. ::(X0,treg,X1) | ~EX(X0,X1) | ~T(X0) [cnf transformation 755]
1114. ~sP2(X0,X1) | PRE(X0,X1) [cnf transformation 768]
1123. ~::(X0,mten,X1) | sP2(X0,X1) [cnf transformation 769]
1126. ~sP3(X0,X1) | PRE(X0,X1) [cnf transformation 776]
1134. ~::(X0,site,X1) | sP3(X0,X1) [cnf transformation 778]
1135. ~::(X0,cfbnd,X1) | sP3(X0,X1) [cnf transformation 778]
1138. ~::(X0,sreg,X1) | EX(X0,X1) [cnf transformation 780]
1140. ~::(X0,imen,X1) | ::(X0,cfbnd,X1) | ::(X0,site,X1) | ::(X0,sreg,X1) [cnf transformation 782]
1144. ~::(X0,idcnt,X1) | ::(X0,mten,X1) | ::(X0,imen,X1) [cnf transformation 784]
1148. ~::(X0,sdcnt,X1) | PRE(X0,X1) [cnf transformation 789]
1158. ~::(X0,gdcnt,X1) | PRE(X0,X1) [cnf transformation 800]
1162. ~::(X0,cnt,X1) | ::(X0,sdcnt,X1) | ::(X0,idcnt,X1) | ::(X0,gdcnt,X1) [cnf transformation 802]
1166. ~cP(X0,X1,X2) | ::(X0,cnt,X2) [cnf transformation 804]
1167. ~cP(X0,X1,X2) | ::(X1,cnt,X2) [cnf transformation 804]
1171. cP(sK88,sK89,sK90) [cnf transformation 806]
1172. ~TM(sK90) | ~::(sK89,cnt,sK90) | ~::(sK88,cnt,sK90) [cnf transformation 806]
1178. ~sP1(X0,X1) | P(X0,X1) | T(X0) [subsumption resolution 1072,884]
1179. ~sP1(X0,X1) | T(X1) | T(X0) [subsumption resolution 1071,884]
1188. 1 <=> ::(sK88,cnt,sK90) [avatar definition]
1189. ::(sK88,cnt,sK90) <- (1) [avatar component clause 1188]
1192. 2 <=> ::(sK89,cnt,sK90) [avatar definition]
1196. 3 <=> TM(sK90) [avatar definition]
1198. ~TM(sK90) <- (~3) [avatar component clause 1196]
1199. ~1 | ~2 | ~3 [avatar split clause 1172,1196,1192,1188]
1435. sP1(X1,X1) | ~T(X1) | ~AB(X1) [resolution 1074,832]
1457. ::(sK88,cnt,sK90) [resolution 1166,1171]
1460. 1 [avatar split clause 1457,1188]
1463. ::(sK89,cnt,sK90) [resolution 1167,1171]
1466. 2 [avatar split clause 1463,1192]
1618. EX(X2,X2) | ~AB(X2) | ~T(X2) [resolution 1435,1077]
1666. ~EX(X0,X0) | ~T(X0) | TM(X0) [resolution 1102,1066]
1823. ~AB(X1) | ~T(X1) | ~T(X1) | TM(X1) [resolution 1618,1666]
1824. ~T(X1) | ~AB(X1) | TM(X1) [duplicate literal removal 1823]
2734. ::(sK88,sdcnt,sK90) | ::(sK88,idcnt,sK90) | ::(sK88,gdcnt,sK90) <- (1) [resolution 1162,1189]
2737. 4 <=> ::(sK88,gdcnt,sK90) [avatar definition]
2739. ::(sK88,gdcnt,sK90) <- (4) [avatar component clause 2737]
2741. 5 <=> ::(sK88,idcnt,sK90) [avatar definition]
2743. ::(sK88,idcnt,sK90) <- (5) [avatar component clause 2741]
2745. 6 <=> ::(sK88,sdcnt,sK90) [avatar definition]
2747. ::(sK88,sdcnt,sK90) <- (6) [avatar component clause 2745]
2748. 4 | 5 | 6 | ~1 [avatar split clause 2734,1188,2745,2741,2737]
2765. 10 <=> ::(sK88,imen,sK90) [avatar definition]
2767. ::(sK88,imen,sK90) <- (10) [avatar component clause 2765]
2769. 11 <=> ::(sK88,mten,sK90) [avatar definition]
2771. ::(sK88,mten,sK90) <- (11) [avatar component clause 2769]
2808. 16 <=> AB(sK90) [avatar definition]
2809. ~AB(sK90) <- (~16) [avatar component clause 2808]
2810. AB(sK90) <- (16) [avatar component clause 2808]
3084. 38 <=> PRE(sK88,sK90) [avatar definition]
3085. PRE(sK88,sK90) <- (38) [avatar component clause 3084]
3086. ~PRE(sK88,sK90) <- (~38) [avatar component clause 3084]
3094. ::(sK88,mten,sK90) | ::(sK88,imen,sK90) <- (5) [resolution 2743,1144]
3236. 47 <=> ::(sK88,sreg,sK90) [avatar definition]
3238. ::(sK88,sreg,sK90) <- (47) [avatar component clause 3236]
3240. 48 <=> ::(sK88,site,sK90) [avatar definition]
3242. ::(sK88,site,sK90) <- (48) [avatar component clause 3240]
3244. 49 <=> ::(sK88,cfbnd,sK90) [avatar definition]
3245. ~::(sK88,cfbnd,sK90) <- (~49) [avatar component clause 3244]
3246. ::(sK88,cfbnd,sK90) <- (49) [avatar component clause 3244]
3401. sP3(sK88,sK90) <- (49) [resolution 3246,1135]
3418. PRE(sK88,sK90) <- (49) [resolution 3401,1126]
3420. $false <- (~38, 49) [subsumption resolution 3418,3086]
3421. 38 | ~49 [avatar contradiction clause 3420]
3428. EX(sK88,sK90) <- (47) [resolution 3238,1138]
3435. sP1(sK90,sK88) <- (47) [resolution 3428,1076]
3444. P(sK90,sK88) | T(sK90) <- (47) [resolution 3435,1178]
3445. T(sK88) | T(sK90) <- (47) [resolution 3435,1179]
3449. 58 <=> T(sK88) [avatar definition]
3450. T(sK88) <- (58) [avatar component clause 3449]
3453. 59 <=> P(sK90,sK88) [avatar definition]
3455. P(sK90,sK88) <- (59) [avatar component clause 3453]
3458. 60 <=> T(sK90) [avatar definition]
3459. ~T(sK90) <- (~60) [avatar component clause 3458]
3460. T(sK90) <- (60) [avatar component clause 3458]
3461. 60 | 59 | ~47 [avatar split clause 3444,3236,3453,3458]
3462. 60 | 58 | ~47 [avatar split clause 3445,3236,3449,3458]
3484. TR(sK90) <- (60) [resolution 3460,1043]
3499. R(sK90) <- (60) [resolution 3484,1036]
3502. AB(sK90) <- (60) [resolution 3499,1037]
```

3503. $false <- (~16, 60) [subsumption resolution 3502,2809]
3504. 16 | ~60 [avatar contradiction clause 3503]
3505. 10 | 11 | ~5 [avatar split clause 3094,2741,2769,2765]
3518. sP2(sK88,sK90) <- (11) [resolution 2771,1123]
3521. PRE(sK88,sK90) <- (11) [resolution 3518,1114]
3530. $false <- (11, ~38) [subsumption resolution 3521,3086]
3531. ~11 | 38 [avatar contradiction clause 3530]
3540. PRE(sK88,sK90) <- (6) [resolution 2747,1148]
3542. $false <- (6, ~38) [subsumption resolution 3540,3086]
3543. ~6 | 38 [avatar contradiction clause 3542]
3548. PRE(sK88,sK90) <- (4) [resolution 2739,1158]
3551. 38 | ~4 [avatar split clause 3548,2737,3084]
3570. T(sK90) <- (38) [resolution 3085,884]
3574. $false <- (38, ~60) [subsumption resolution 3570,3459]
3575. ~38 | 60 [avatar contradiction clause 3574]
3580. ::(sK88,cfbnd,sK90) | ::(sK88,site,sK90) | ::(sK88,sreg,sK90) <- (10) [resolution 2767,1140]
3593. sP3(sK88,sK90) <- (48) [resolution 3242,1134]
3597. PRE(sK88,sK90) <- (48) [resolution 3593,1126]
3599. $false <- (~38, 48) [subsumption resolution 3597,3086]
3600. 38 | ~48 [avatar contradiction clause 3599]
3601. ::(sK88,site,sK90) | ::(sK88,sreg,sK90) <- (10, ~49) [subsumption resolution 3580,3245]
3602. 47 | 48 | ~10 | 49 [avatar split clause 3601,3244,2765,3240,3236]
3648. ~T(sK88) | T(sK90) <- (59) [resolution 3455,829]
3679. T(sK90) <- (58, 59) [subsumption resolution 3648,3450]
3680. $false <- (58, 59, ~60) [subsumption resolution 3679,3459]
3681. ~58 | ~59 | 60 [avatar contradiction clause 3680]
3730. ~AB(sK90) | TM(sK90) <- (60) [resolution 3460,1824]
3733. TM(sK90) <- (16, 60) [subsumption resolution 3730,2810]
3734. $false <- (~3, 16, 60) [subsumption resolution 3733,1198]
3735. 3 | ~16 | ~60 [avatar contradiction clause 3734]
3736. $false [avatar sat refutation
1199,1460,1466,2748,3421,3461,3462,3504,3505,3531,3543,3551,3575,3600,3602,3681,3735]

Proof of Theorem ($t_{db}35$)

8. ! [X0] : ((PD(X0) | AB(X0)) => P(X0,X0)) [input]
51. ! [X0,X4] : ((PRE(X0,X4) & ED(X0)) => tP(X0,X0,X4)) [input]
116. ! [X0] : ((PED(X0) | NPED(X0) | AS(X0)) <=> ED(X0)) [input]
124. ! [X0] : ((TR(X0) | PR(X0) | AR(X0)) <=> R(X0)) [input]
125. ! [X0] : (R(X0) => AB(X0)) [input]
126. ! [X0] : ((POB(X0) | M(X0) | F(X0)) => PED(X0)) [input]
128. ! [X0] : (S(X0) => PR(X0)) [input]
173. ! [X0,X4] : (::(X0,mten,X4) <=> (! [X5] : (PRE(X0,X5) => ? [X1] : ? [X3] : ? [X8] : (P(X8,X3) & SLC(X1,X8,X5) & SLC(X0,X3,X5) & M(X1))) & ? [X5] : (~AT(X5) & PRE(X0,X5)) & PRE(X0,X4) & ED(X0))) [input]
174. ! [X0,X4] : ((::(X0,cfbnd,X4) | ::(X0,site,X4)) <=> (! [X5] : (PRE(X0,X5) => ~? [X1] : ? [X3] : ? [X8] : (P(X8,X3) & SLC(X1,X8,X5) & SLC(X0,X3,X5) & M(X1))) & ? [X3] : SLC(X0,X3,X4) & PRE(X0,X4) & F(X0))) [input]
175. ! [X0,X4] : (::(X0,sreg,X4) <=> (EX(X0,X4) & S(X0))) [input]
176. ! [X0,X4] : (::(X0,imen,X4) <=> (::(X0,sreg,X4) | ::(X0,cfbnd,X4) | ::(X0,site,X4))) [input]
177. ! [X0,X4] : (::(X0,idcnt,X4) <=> (::(X0,imen,X4) | ::(X0,mten,X4))) [input]
181. ! [X0,X4] : (::(X0,cnt,X4) <=> (::(X0,gdcnt,X4) | ::(X0,sdcnt,X4) | ::(X0,idcnt,X4))) [input]
182. ! [X0,X1,X4] : (cP(X0,X1,X4) <=> ((? [X2] : ? [X5] : (tP(X2,X5,X4) & DQT(X1,X5) & DQT(X0,X2)) | P(X0,X1) | tP(X0,X1,X4)) & ::(X1,cnt,X4) & ::(X0,cnt,X4))) [input]
189. ! [X0,X4] : (::(X0,idcnt,X4) => cP(X0,X0,X4)) [input]
190. ~! [X0,X4] : (::(X0,idcnt,X4) => cP(X0,X0,X4)) [negated conjecture 189]
217. ! [X0,X1] : ((PRE(X0,X1) & ED(X0)) => tP(X0,X0,X1)) [rectify 51]
314. ! [X0,X1] : (::(X0,mten,X1) <=> (! [X2] : (PRE(X0,X2) => ? [X3] : ? [X4] : ? [X5] : (P(X5,X4) & SLC(X3,X5,X2) & SLC(X0,X4,X2) & M(X3))) & ? [X6] : (~AT(X6) & PRE(X0,X6)) & PRE(X0,X1) & ED(X0))) [rectify 173]
315. ! [X0,X1] : (::(X0,mten,X1) <=> (! [X2] : (PRE(X0,X2) => ? [X3,X4,X5] : (P(X5,X4) & SLC(X3,X5,X2) & SLC(X0,X4,X2) & M(X3))) & ? [X6] : (~AT(X6) & PRE(X0,X6)) & PRE(X0,X1) & ED(X0))) [flattening 314]
316. ! [X0,X1] : (((::(X0,cfbnd,X1) | ::(X0,site,X1)) <=> (! [X2] : (PRE(X0,X2) => ~? [X3] : ? [X4] : ? [X5] : (P(X5,X4) & SLC(X3,X5,X2) & SLC(X0,X4,X2) & M(X3))) & ? [X6] : SLC(X0,X6,X1) & PRE(X0,X1) & F(X0))) [rectify 174]
317. ! [X0,X1] : (((::(X0,cfbnd,X1) | ::(X0,site,X1)) <=> (! [X2] : (PRE(X0,X2) => ~? [X3,X4,X5] : (P(X5,X4) & SLC(X3,X5,X2) & SLC(X0,X4,X2) & M(X3))) & ? [X6] : SLC(X0,X6,X1) & PRE(X0,X1) & F(X0))) [flattening 316]
318. ! [X0,X1] : (::(X0,sreg,X1) <=> (EX(X0,X1) & S(X0))) [rectify 175]
319. ! [X0,X1] : (::(X0,imen,X1) <=> (::(X0,sreg,X1) | ::(X0,cfbnd,X1) | ::(X0,site,X1))) [rectify 176]
320. ! [X0,X1] : (::(X0,idcnt,X1) <=> (::(X0,imen,X1) | ::(X0,mten,X1))) [rectify 177]
324. ! [X0,X1] : (::(X0,cnt,X1) <=> (::(X0,gdcnt,X1) | ::(X0,sdcnt,X1) | ::(X0,idcnt,X1))) [rectify 181]
325. ! [X0,X1,X2] : (cP(X0,X1,X2) <=> ((? [X3] : ? [X4] : (tP(X3,X4,X2) & DQT(X1,X4) & DQT(X0,X3)) | P(X0,X1) | tP(X0,X1,X2)) & ::(X1,cnt,X2) & ::(X0,cnt,X2))) [rectify 182]
326. ! [X0,X1,X2] : (cP(X0,X1,X2) <=> ((? [X3,X4] : (tP(X3,X4,X2) & DQT(X1,X4) & DQT(X0,X3)) | P(X0,X1) | tP(X0,X1,X2)) & ::(X1,cnt,X2) & ::(X0,cnt,X2))) [flattening 325]
335. ~! [X0,X1] : (::(X0,idcnt,X1) => cP(X0,X0,X1)) [rectify 190]
336. ! [X0,X1,X2] : (((? [X3,X4] : (tP(X3,X4,X2) & DQT(X1,X4) & DQT(X0,X3)) | P(X0,X1) | tP(X0,X1,X2)) & ::(X1,cnt,X2) & ::(X0,cnt,X2)) => cP(X0,X1,X2)) [unused predicate definition removal 326]
337. ! [X0] : ((TR(X0) | PR(X0) | AR(X0)) => R(X0)) [unused predicate definition removal 124]
375. ! [X0] : (P(X0,X0) | (~PD(X0) & ~AB(X0))) [ennf transformation 8]
438. ! [X0,X1] : (tP(X0,X0,X1) | (~PRE(X0,X1) | ~ED(X0))) [ennf transformation 217]
439. ! [X0,X1] : (tP(X0,X0,X1) | ~PRE(X0,X1) | ~ED(X0)) [flattening 438]
534. ! [X0] : (R(X0) | (~TR(X0) & ~PR(X0) & ~AR(X0))) [ennf transformation 337]
535. ! [X0] : (AB(X0) | ~R(X0)) [ennf transformation 125]
536. ! [X0] : (PED(X0) | (~POB(X0) & ~M(X0) & ~F(X0))) [ennf transformation 126]
537. ! [X0] : (PR(X0) | ~S(X0)) [ennf transformation 128]
561. ! [X0,X1] : (::(X0,mten,X1) <=> (! [X2] : (? [X3,X4,X5] : (P(X5,X4) & SLC(X3,X5,X2) & SLC(X0,X4,X2) & M(X3)) | ~PRE(X0,X2)) & ? [X6] : (~AT(X6) & PRE(X0,X6)) & PRE(X0,X1) & ED(X0))) [ennf transformation 315]
562. ! [X0,X1] : (((::(X0,cfbnd,X1) | ::(X0,site,X1)) <=> (! [X2] : (! [X3,X4,X5] : (~P(X5,X4) | ~SLC(X3,X5,X2) | ~SLC(X0,X4,X2) | ~M(X3)) | ~PRE(X0,X2)) & ? [X6] : SLC(X0,X6,X1) & PRE(X0,X1) & F(X0))) [ennf transformation 317]
565. ! [X0,X1,X2] : (cP(X0,X1,X2) | ((! [X3,X4] : (~tP(X3,X4,X2) | ~DQT(X1,X4) | ~DQT(X0,X3)) & ~P(X0,X1) & ~tP(X0,X1,X2)) | ~::(X1,cnt,X2) | ~::(X0,cnt,X2))) [ennf transformation 336]

566. ! [X0,X1,X2] : (cP(X0,X1,X2) | (! [X3,X4] : (~tP(X3,X4,X2) | ~DQT(X1,X4) | ~DQT(X0,X3)) & ~P(X0,X1) & ~tP(X0,X1,X2)) | ~::(X1,cnt,X2) | ~::(X0,cnt,X2)) [flattening 565]
567. ? [X0,X1] : (~cP(X0,X0,X1) & ::(X0,idcnt,X1)) [ennf transformation 335]
572. ! [X0,X1] : (sP2(X0,X1) <=> (! [X2] : (? [X3,X4,X5] : (P(X5,X4) & SLC(X3,X5,X2) & SLC(X0,X4,X2) & M(X3)) | ~PRE(X0,X2)) & ? [X6] : (~AT(X6) & PRE(X0,X6)) & PRE(X0,X1) & ED(X0))) [predicate definition introduction]
573. ! [X0,X1] : (::(X0,mten,X1) <=> sP2(X0,X1)) [definition folding 561,572]
574. ! [X0,X1] : (sP3(X0,X1) <=> (! [X2] : (! [X3,X4,X5] : (~P(X5,X4) | ~SLC(X3,X5,X2) | ~SLC(X0,X4,X2) | ~M(X3)) | ~PRE(X0,X2)) & ? [X6] : SLC(X0,X6,X1) & PRE(X0,X1) & F(X0))) [predicate definition introduction]
575. ! [X0,X1] : ((::(X0,cfbnd,X1) | ::(X0,site,X1)) <=> sP3(X0,X1)) [definition folding 562,574]
694. ! [X0] : (((PED(X0) | NPED(X0) | AS(X0)) | ~ED(X0)) & (ED(X0) | (~PED(X0) & ~NPED(X0) & ~AS(X0)))) [nnf transformation 116]
695. ! [X0] : ((PED(X0) | NPED(X0) | AS(X0) | ~ED(X0)) & (ED(X0) | (~PED(X0) & ~NPED(X0) & ~AS(X0)))) [flattening 694]
728. ! [X0,X1] : ((sP2(X0,X1) | (? [X2] : (! [X3,X4,X5] : (~P(X5,X4) | ~SLC(X3,X5,X2) | ~SLC(X0,X4,X2) | ~M(X3)) & PRE(X0,X2)) | ! [X6] : (AT(X6) | ~PRE(X0,X6)) | ~PRE(X0,X1) | ~ED(X0))) & ((! [X2] : (? [X3,X4,X5] : (P(X5,X4) & SLC(X3,X5,X2) & SLC(X0,X4,X2) & M(X3)) | ~PRE(X0,X2)) & ? [X6] : (~AT(X6) & PRE(X0,X6)) & PRE(X0,X1) & ED(X0)) | ~sP2(X0,X1))) [nnf transformation 572]
729. ! [X0,X1] : ((sP2(X0,X1) | ? [X2] : (! [X3,X4,X5] : (~P(X5,X4) | ~SLC(X3,X5,X2) | ~SLC(X0,X4,X2) | ~M(X3)) & PRE(X0,X2)) | ! [X6] : (AT(X6) | ~PRE(X0,X6)) | ~PRE(X0,X1) | ~ED(X0)) & ((! [X2] : (? [X3,X4,X5] : (P(X5,X4) & SLC(X3,X5,X2) & SLC(X0,X4,X2) & M(X3)) | ~PRE(X0,X2)) & ? [X6] : (~AT(X6) & PRE(X0,X6)) & PRE(X0,X1) & ED(X0)) | ~sP2(X0,X1))) [flattening 728]
730. ! [X0,X1] : ((sP2(X0,X1) | ? [X2] : (! [X3,X4,X5] : (~P(X5,X4) | ~SLC(X3,X5,X2) | ~SLC(X0,X4,X2) | ~M(X3)) & PRE(X0,X2)) | ! [X6] : (AT(X6) | ~PRE(X0,X6)) | ~PRE(X0,X1) | ~ED(X0)) & ((! [X7] : (? [X8,X9,X10] : (P(X10,X9) & SLC(X8,X10,X7) & SLC(X0,X9,X7) & M(X8)) | ~PRE(X0,X7)) & ? [X11] : (~AT(X11) & PRE(X0,X11)) & PRE(X0,X1) & ED(X0)) | ~sP2(X0,X1))) [rectify 729]
731. ! [X0] : (? [X2] : (! [X3,X4,X5] : (~P(X5,X4) | ~SLC(X3,X5,X2) | ~SLC(X0,X4,X2) | ~M(X3)) & PRE(X0,X2)) => (! [X5,X4,X3] : (~P(X5,X4) | ~SLC(X3,X5,sK70(X0)) | ~SLC(X0,X4,sK70(X0)) | ~M(X3)) & PRE(X0,sK70(X0)))) [choice axiom]
732. ! [X7,X0] : (? [X8,X9,X10] : (P(X10,X9) & SLC(X8,X10,X7) & SLC(X0,X9,X7) & M(X8)) => (P(sK73(X0,X7),sK72(X0,X7)) & SLC(sK71(X0,X7),sK73(X0,X7),X7) & SLC(X0,sK72(X0,X7),X7) & M(sK71(X0,X7)))) [choice axiom]
733. ! [X0] : (? [X11] : (~AT(X11) & PRE(X0,X11)) => (~AT(sK74(X0)) & PRE(X0,sK74(X0)))) [choice axiom]
734. ! [X0,X1] : ((sP2(X0,X1) | (! [X3,X4,X5] : (~P(X5,X4) | ~SLC(X3,X5,sK70(X0)) | ~SLC(X0,X4,sK70(X0)) | ~M(X3)) & PRE(X0,sK70(X0))) | ! [X6] : (AT(X6) | ~PRE(X0,X6)) | ~PRE(X0,X1) | ~ED(X0)) & ((! [X7] : ((P(sK73(X0,X7),sK72(X0,X7)) & SLC(sK71(X0,X7),sK73(X0,X7),X7) & SLC(X0,sK72(X0,X7),X7) & M(sK71(X0,X7))) | ~PRE(X0,X7)) & (~AT(sK74(X0)) & PRE(X0,sK74(X0))) & PRE(X0,X1) & ED(X0)) | ~sP2(X0,X1))) [skolemisation 730,733,732,731]
735. ! [X0,X1] : ((::(X0,mten,X1) | ~sP2(X0,X1)) & (sP2(X0,X1) | ~::(X0,mten,X1))) [nnf transformation 573]
736. ! [X0,X1] : ((sP3(X0,X1) | (? [X2] : (? [X3,X4,X5] : (P(X5,X4) & SLC(X3,X5,X2) & SLC(X0,X4,X2) & M(X3)) & PRE(X0,X2)) | ! [X6] : ~SLC(X0,X6,X1) | ~PRE(X0,X1) | ~F(X0))) & ((! [X2] : (! [X3,X4,X5] : (~P(X5,X4) | ~SLC(X3,X5,X2) | ~SLC(X0,X4,X2) | ~M(X3)) | ~PRE(X0,X2)) & ? [X6] : SLC(X0,X6,X1) & PRE(X0,X1) & F(X0)) | ~sP3(X0,X1))) [nnf transformation 574]
737. ! [X0,X1] : ((sP3(X0,X1) | ? [X2] : (? [X3,X4,X5] : (P(X5,X4) & SLC(X3,X5,X2) & SLC(X0,X4,X2) & M(X3)) & PRE(X0,X2)) | ! [X6] : ~SLC(X0,X6,X1) | ~PRE(X0,X1) | ~F(X0)) & ((! [X2] : (! [X3,X4,X5] : (~P(X5,X4) | ~SLC(X3,X5,X2) | ~SLC(X0,X4,X2) | ~M(X3)) | ~PRE(X0,X2)) & ? [X6] : SLC(X0,X6,X1) & PRE(X0,X1) & F(X0)) | ~sP3(X0,X1))) [flattening 736]
738. ! [X0,X1] : ((sP3(X0,X1) | ? [X2] : (? [X3,X4,X5] : (P(X5,X4) & SLC(X3,X5,X2) & SLC(X0,X4,X2) & M(X3)) & PRE(X0,X2)) | ! [X6] : ~SLC(X0,X6,X1) | ~PRE(X0,X1) | ~F(X0)) & ((! [X7] : (! [X8,X9,X10] : (~P(X10,X9) | ~SLC(X8,X10,X7) | ~SLC(X0,X9,X7) | ~M(X8)) | ~PRE(X0,X7)) & ? [X11] : SLC(X0,X11,X1) & PRE(X0,X1) & F(X0)) | ~sP3(X0,X1))) [rectify 737]
739. ! [X0] : (? [X2] : (? [X3,X4,X5] : (P(X5,X4) & SLC(X3,X5,X2) & SLC(X0,X4,X2) & M(X3)) & PRE(X0,X2)) => (? [X5,X4,X3] : (P(X5,X4) & SLC(X3,X5,sK75(X0)) & SLC(X0,X4,sK75(X0)) & M(X3)) & PRE(X0,sK75(X0)))) [choice axiom]
740. ! [X0] : (? [X5,X4,X3] : (P(X5,X4) & SLC(X3,X5,sK75(X0)) & SLC(X0,X4,sK75(X0)) & M(X3)) => (P(sK78(X0),sK77(X0)) & SLC(sK76(X0),sK78(X0),sK75(X0)) & SLC(X0,sK77(X0),sK75(X0)) & M(sK76(X0)))) [choice axiom]
741. ! [X1,X0] : (? [X11] : SLC(X0,X11,X1) => SLC(X0,sK79(X0,X1),X1)) [choice axiom]
742. ! [X0,X1] : ((sP3(X0,X1) | ((P(sK78(X0),sK77(X0)) & SLC(sK76(X0),sK78(X0),sK75(X0)) & SLC(X0,sK77(X0),sK75(X0)) & M(sK76(X0))) & PRE(X0,sK75(X0))) | ! [X6] : ~SLC(X0,X6,X1) | ~PRE(X0,X1) | ~F(X0)) & ((! [X7] : (! [X8,X9,X10] : (~P(X10,X9) | ~SLC(X8,X10,X7) | ~SLC(X0,X9,X7) | ~M(X8)) | ~PRE(X0,X7)) & SLC(X0,sK79(X0,X1),X1) & PRE(X0,X1) & F(X0)) | ~sP3(X0,X1))) [skolemisation 738,741,740,739]
743. ! [X0,X1] : (((::(X0,cfbnd,X1) | ::(X0,site,X1)) | ~sP3(X0,X1)) & (sP3(X0,X1) | (~::(X0,cfbnd,X1) & ~::(X0,site,X1)))) [nnf transformation 575]
744. ! [X0,X1] : ((::(X0,cfbnd,X1) | ::(X0,site,X1) | ~sP3(X0,X1)) & (sP3(X0,X1) | (~::(X0,cfbnd,X1) & ~::(X0,site,X1)))) [flattening 743]
745. ! [X0,X1] : ((::(X0,sreg,X1) | (~EX(X0,X1) | ~S(X0))) & ((EX(X0,X1) & S(X0)) | ~::(X0,sreg,X1))) [nnf transformation 318]
746. ! [X0,X1] : ((::(X0,sreg,X1) | ~EX(X0,X1) | ~S(X0)) & ((EX(X0,X1) & S(X0)) | ~::(X0,sreg,X1))) [flattening 745]
747. ! [X0,X1] : ((::(X0,imen,X1) | (~::(X0,sreg,X1) & ~::(X0,cfbnd,X1) & ~::(X0,site,X1))) & ((::(X0,sreg,X1) | ::(X0,cfbnd,X1) | ::(X0,site,X1)) | ~::(X0,imen,X1))) [nnf transformation 319]
748. ! [X0,X1] : ((::(X0,imen,X1) | (~::(X0,sreg,X1) & ~::(X0,cfbnd,X1) & ~::(X0,site,X1))) & (::(X0,sreg,X1) | ::(X0,cfbnd,X1) | ::(X0,site,X1) | ~::(X0,imen,X1))) [flattening 747]
749. ! [X0,X1] : ((::(X0,idcnt,X1) | (~::(X0,imen,X1) & ~::(X0,mten,X1))) & ((::(X0,imen,X1) | ::(X0,mten,X1)) | ~::(X0,idcnt,X1))) [nnf transformation 320]
750. ! [X0,X1] : ((::(X0,idcnt,X1) | (~::(X0,imen,X1) & ~::(X0,mten,X1))) & (::(X0,imen,X1) | ::(X0,mten,X1) | ~::(X0,idcnt,X1))) [flattening 749]
767. ! [X0,X1] : ((::(X0,cnt,X1) | (~::(X0,gdcnt,X1) & ~::(X0,sdcnt,X1) & ~::(X0,idcnt,X1))) & ((::(X0,gdcnt,X1) | ::(X0,sdcnt,X1) | ::(X0,idcnt,X1)) | ~::(X0,cnt,X1))) [nnf transformation 324]
768. ! [X0,X1] : ((::(X0,cnt,X1) | (~::(X0,gdcnt,X1) & ~::(X0,sdcnt,X1) & ~::(X0,idcnt,X1))) & (::(X0,gdcnt,X1) | ::(X0,sdcnt,X1) | ::(X0,idcnt,X1) | ~::(X0,cnt,X1))) [flattening 767]
769. ? [X0,X1] : (~cP(X0,X0,X1) & ::(X0,idcnt,X1)) => (~cP(sK84,sK84,sK85) & ::(sK84,idcnt,sK85)) [choice axiom]
770. ~cP(sK84,sK84,sK85) & ::(sK84,idcnt,sK85) [skolemisation 567,769]
792. P(X0,X0) | ~AB(X0) [cnf transformation 375]
873. tP(X0,X0,X1) | ~PRE(X0,X1) | ~ED(X0) [cnf transformation 439]
986. ~PED(X0) | ED(X0) [cnf transformation 695]
994. ~PR(X0) | R(X0) [cnf transformation 534]
996. ~R(X0) | AB(X0) [cnf transformation 535]
997. ~F(X0) | PED(X0) [cnf transformation 536]
1000. ~S(X0) | PR(X0) [cnf transformation 537]
1071. ~sP2(X0,X1) | ED(X0) [cnf transformation 734]
1072. ~sP2(X0,X1) | PRE(X0,X1) [cnf transformation 734]
1081. ~::(X0,mten,X1) | sP2(X0,X1) [cnf transformation 735]
1083. ~sP3(X0,X1) | F(X0) [cnf transformation 742]
1084. ~sP3(X0,X1) | PRE(X0,X1) [cnf transformation 742]

```
1092. ~::(X0,site,X1) | sP3(X0,X1) [cnf transformation 744]
1093. ~::(X0,cfbnd,X1) | sP3(X0,X1) [cnf transformation 744]
1095. ~::(X0,sreg,X1) | S(X0) [cnf transformation 746]
1098. ~::(X0,imen,X1) | ::(X0,cfbnd,X1) | ::(X0,site,X1) | ::(X0,sreg,X1) [cnf transformation 748]
1102. ~::(X0,idcnt,X1) | ::(X0,mten,X1) | ::(X0,imen,X1) [cnf transformation 750]
1121. ~::(X0,idcnt,X1) | ::(X0,cnt,X1) [cnf transformation 768]
1124. ~::(X1,cnt,X2) | ~tP(X0,X1,X2) | cP(X0,X1,X2) | ~::(X0,cnt,X2) [cnf transformation 566]
1125. ~::(X1,cnt,X2) | ~P(X0,X1) | cP(X0,X1,X2) | ~::(X0,cnt,X2) [cnf transformation 566]
1127. ::(sK84,idcnt,sK85) [cnf transformation 770]
1128. ~cP(sK84,sK84,sK85) [cnf transformation 770]
1366. ::(sK84,cnt,sK85) [resolution 1121,1127]
2112. ::(sK84,mten,sK85) | ::(sK84,imen,sK85) [resolution 1102,1127]
2117. 1 <=> ::(sK84,imen,sK85) [avatar definition]
2119. ::(sK84,imen,sK85) <- (1) [avatar component clause 2117]
2121. 2 <=> ::(sK84,mten,sK85) [avatar definition]
2123. ::(sK84,mten,sK85) <- (2) [avatar component clause 2121]
2124. 1 | 2 [avatar split clause 2112,2121,2117]
2139. 3 <=> PED(sK84) [avatar definition]
2140. ~PED(sK84) <- (~3) [avatar component clause 2139]
2141. PED(sK84) <- (3) [avatar component clause 2139]
2533. ~::(X0,cnt,sK85) | cP(X0,sK84,sK85) | ~P(X0,sK84) [resolution 1125,1366]
2633. 10 <=> ::(sK84,sreg,sK85) [avatar definition]
2635. ::(sK84,sreg,sK85) <- (10) [avatar component clause 2633]
2637. 11 <=> ::(sK84,site,sK85) [avatar definition]
2639. ::(sK84,site,sK85) <- (11) [avatar component clause 2637]
2641. 12 <=> ::(sK84,cfbnd,sK85) [avatar definition]
2642. ~::(sK84,cfbnd,sK85) <- (~12) [avatar component clause 2641]
2643. ::(sK84,cfbnd,sK85) <- (12) [avatar component clause 2641]
2649. ~::(X0,cnt,sK85) | cP(X0,sK84,sK85) | ~tP(X0,sK84,sK85) [resolution 1124,1366]
2731. 16 <=> S(sK84) [avatar definition]
2732. ~S(sK84) <- (~16) [avatar component clause 2731]
2733. S(sK84) <- (16) [avatar component clause 2731]
2846. 19 <=> PRE(sK84,sK85) [avatar definition]
2847. ~PRE(sK84,sK85) <- (~19) [avatar component clause 2846]
2848. PRE(sK84,sK85) <- (19) [avatar component clause 2846]
2868. sP3(sK84,sK85) <- (11) [resolution 2639,1092]
2869. PRE(sK84,sK85) <- (11) [resolution 2868,1084]
2870. F(sK84) <- (11) [resolution 2868,1083]
2871. 19 | ~11 [avatar split clause 2869,2637,2846]
2876. PED(sK84) <- (11) [resolution 2870,997]
2879. 3 | ~11 [avatar split clause 2876,2637,2139]
2940. sP2(sK84,sK85) <- (2) [resolution 2123,1081]
2942. PRE(sK84,sK85) <- (2) [resolution 2940,1072]
2949. $false <- (2, ~19) [subsumption resolution 2942,2847]
2950. ~2 | 19 [avatar contradiction clause 2949]
2951. ::(sK84,cfbnd,sK85) | ::(sK84,site,sK85) | ::(sK84,sreg,sK85) <- (1) [resolution 2119,1098]
2963. S(sK84) <- (10) [resolution 2635,1095]
2965. $false <- (10, ~16) [subsumption resolution 2963,2732]
2966. ~10 | 16 [avatar contradiction clause 2965]
2990. sP3(sK84,sK85) <- (12) [resolution 2643,1093]
2997. PRE(sK84,sK85) <- (12) [resolution 2990,1084]
2999. $false <- (12, ~19) [subsumption resolution 2997,2847]
3000. ~12 | 19 [avatar contradiction clause 2999]
3001. PR(sK84) <- (16) [resolution 2733,1000]
3003. R(sK84) <- (16) [resolution 3001,994]
3036. AB(sK84) <- (16) [resolution 3003,996]
3278. 23 <=> ED(sK84) [avatar definition]
3280. ~ED(sK84) <- (~23) [avatar component clause 3278]
14150. cP(sK84,sK84,sK85) | ~P(sK84,sK84) [resolution 2533,1366]
14157. ~P(sK84,sK84) [subsumption resolution 14150,1128]
14159. ~AB(sK84) [resolution 14157,792]
14160. $false <- (16) [subsumption resolution 14159,3036]
14161. ~16 [avatar contradiction clause 14160]
14240. 68 <=> sP3(sK84,sK85) [avatar definition]
14241. sP3(sK84,sK85) <- (68) [avatar component clause 14240]
14244. ::(sK84,site,sK85) | ::(sK84,sreg,sK85) <- (1, ~12) [subsumption resolution 2951,2642]
14245. 10 | 11 | ~1 | 12 [avatar split clause 14244,2641,2117,2637,2633]
14392. ED(sK84) <- (3) [resolution 2141,986]
14395. 23 | ~3 [avatar split clause 14392,2139,3278]
16347. cP(sK84,sK84,sK85) | ~tP(sK84,sK84,sK85) [resolution 2649,1366]
16355. ~tP(sK84,sK84,sK85) [subsumption resolution 16347,1128]
16359. ~PRE(sK84,sK85) | ~ED(sK84) [resolution 16355,873]
16360. ~ED(sK84) <- (19) [subsumption resolution 16359,2848]
16399. ~23 | ~19 [avatar split clause 16360,2846,3278]
16478. sP2(sK84,sK85) <- (2) [resolution 2123,1081]
16486. ED(sK84) <- (2) [resolution 16478,1071]
16494. $false <- (2, ~23) [subsumption resolution 16486,3280]
16495. ~2 | 23 [avatar contradiction clause 16494]
16517. sP3(sK84,sK85) <- (12) [resolution 2643,1093]
16520. 68 | ~12 [avatar split clause 16517,2641,14240]
16537. F(sK84) <- (68) [resolution 14241,1083]
16567. PED(sK84) <- (68) [resolution 16537,997]
16568. $false <- (~3, 68) [subsumption resolution 16567,2140]
16569. 3 | ~68 [avatar contradiction clause 16568]
16570. $false [avatar sat refutation 2124,2871,2879,2950,2966,3000,14161,14245,14395,16399,16495,16520,16569]
```

Proof of Theorem (t_{ab}36)

This went KO with both provers

Proof of Theorem (t_{db}39)

This went KO with both provers

Proof of Theorem (t_{db}40)

```
162. ! [X0,X4] : (EX(X0,X4) <=> ((T(X4) & ~T(X0) & AB(X0)) | (P(X4,X0) & T(X0)) | PRE(X0,X4))) [input]
173. ! [X0,X4] : (::(X0,mten,X4) <=> (! [X5] : (PRE(X0,X5) => ? [X1] : ? [X3] : ? [X8] : (P(X8,X3) & SLC(X1,X8,X5)
     & SLC(X0,X3,X5) & M(X1))) & ? [X5] : (~AT(X5) & PRE(X0,X5)) & PRE(X0,X4) & ED(X0))) [input]
174. ! [X0,X4] : (((::(X0,cfbnd,X4) | ::(X0,site,X4)) <=> (! [X5] : (PRE(X0,X5) => ~? [X1] : ? [X3] : ? [X8] : (P(
     X8,X3) & SLC(X1,X8,X5) & SLC(X0,X3,X5) & M(X1))) & ? [X3] : SLC(X0,X3,X4) & PRE(X0,X4) & F(X0))) [input]
175. ! [X0,X4] : (::(X0,sreg,X4) <=> (EX(X0,X4) & S(X0))) [input]
176. ! [X0,X4] : (::(X0,imen,X4) <=> (::(X0,sreg,X4) | ::(X0,cfbnd,X4) | ::(X0,site,X4))) [input]
177. ! [X0,X4] : (::(X0,idcnt,X4) <=> (::(X0,imen,X4) | ::(X0,mten,X4))) [input]
178. ! [X0,X4] : (::(X0,sdcnt,X4) <=> (? [X1] : (DQT(X0,X1) & ~S(X1) & ::(X1,idcnt,X4)) & PRE(X0,X4) & Q(X0))) [
     input]
180. ! [X0,X4] : (::(X0,gdcnt,X4) <=> (! [X5] : (PRE(X0,X5) => ? [X1] : CONCR(X1,X0,X5)) & PRE(X0,X4))) [input]
181. ! [X0,X4] : (::(X0,cnt,X4) <=> (::(X0,gdcnt,X4) | ::(X0,sdcnt,X4) | ::(X0,idcnt,X4))) [input]
182. ! [X0,X1,X4] : (cP(X0,X1,X4) <=> ((? [X2] : ? [X5] : (tP(X2,X5,X4) & DQT(X1,X5) & DQT(X0,X2)) | P(X0,X1) | tP
     (X0,X1,X4)) & ::(X1,cnt,X4) & ::(X0,cnt,X4))) [input]
189. ! [X0,X1,X4] : ((EX(X1,X4) & cP(X0,X1,X4)) => EX(X0,X4)) [input]
190. ~! [X0,X1,X4] : ((EX(X1,X4) & cP(X0,X1,X4)) => EX(X0,X4)) [negated conjecture 189]
304. ! [X0,X1] : (EX(X0,X1) <=> ((T(X1) & ~T(X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1))) [rectify 162]
314. ! [X0,X1] : (::(X0,mten,X1) <=> (! [X2] : (PRE(X0,X2) => ? [X3] : ? [X4] : ? [X5] : (P(X5,X4) & SLC(X3,X5,X2)
     & SLC(X0,X4,X2) & M(X3))) & ? [X6] : (~AT(X6) & PRE(X0,X6)) & PRE(X0,X1) & ED(X0))) [rectify 173]
315. ! [X0,X1] : (::(X0,mten,X1) <=> (! [X2] : (PRE(X0,X2) => ? [X3,X4,X5] : (P(X5,X4) & SLC(X3,X5,X2) & SLC(X0,X4
     ,X2) & M(X3))) & ? [X6] : (~AT(X6) & PRE(X0,X6)) & PRE(X0,X1) & ED(X0))) [flattening 314]
316. ! [X0,X1] : (((::(X0,cfbnd,X1) | ::(X0,site,X1)) <=> (! [X2] : (PRE(X0,X2) => ~? [X3] : ? [X4] : ? [X5] : (P(
     X5,X4) & SLC(X3,X5,X2) & SLC(X0,X4,X2) & M(X3))) & ? [X6] : SLC(X0,X6,X1) & PRE(X0,X1) & F(X0))) [rectify
     174]
317. ! [X0,X1] : (((::(X0,cfbnd,X1) | ::(X0,site,X1)) <=> (! [X2] : (PRE(X0,X2) => ~? [X3,X4,X5] : (P(X5,X4) & SLC(
     X3,X5,X2) & SLC(X0,X4,X2) & M(X3))) & ? [X6] : SLC(X0,X6,X1) & PRE(X0,X1) & F(X0))) [flattening 316]
318. ! [X0,X1] : (::(X0,sreg,X1) <=> (EX(X0,X1) & S(X0))) [rectify 175]
319. ! [X0,X1] : (::(X0,imen,X1) <=> (::(X0,sreg,X1) | ::(X0,cfbnd,X1) | ::(X0,site,X1))) [rectify 176]
320. ! [X0,X1] : (::(X0,idcnt,X1) <=> (::(X0,imen,X1) | ::(X0,mten,X1))) [rectify 177]
321. ! [X0,X1] : (::(X0,sdcnt,X1) <=> (? [X2] : (DQT(X0,X2) & ~S(X2) & ::(X2,idcnt,X1)) & PRE(X0,X1) & Q(X0))) [
     rectify 178]
323. ! [X0,X1] : (::(X0,gdcnt,X1) <=> (! [X2] : (PRE(X0,X2) => ? [X3] : CONCR(X3,X0,X2)) & PRE(X0,X1))) [rectify
     180]
324. ! [X0,X1] : (::(X0,cnt,X1) <=> (::(X0,gdcnt,X1) | ::(X0,sdcnt,X1) | ::(X0,idcnt,X1))) [rectify 181]
325. ! [X0,X1,X2] : (cP(X0,X1,X2) <=> ((? [X3] : ? [X4] : (tP(X3,X4,X2) & DQT(X1,X4) & DQT(X0,X3)) | P(X0,X1) | tP
     (X0,X1,X2)) & ::(X1,cnt,X2) & ::(X0,cnt,X2))) [rectify 182]
326. ! [X0,X1,X2] : (cP(X0,X1,X2) <=> ((? [X3,X4] : (tP(X3,X4,X2) & DQT(X1,X4) & DQT(X0,X3)) | P(X0,X1) | tP(X0,X1
     ,X2)) & ::(X1,cnt,X2) & ::(X0,cnt,X2))) [flattening 325]
335. ~! [X0,X1,X2] : ((EX(X1,X2) & cP(X0,X1,X2)) => EX(X0,X2)) [rectify 190]
336. ! [X0,X1,X2] : (cP(X0,X1,X2) <=> ((? [X3,X4] : (tP(X3,X4,X2) & DQT(X1,X4) & DQT(X0,X3)) | P(X0,X1) | tP(X0,X1,
     X2)) & ::(X1,cnt,X2) & ::(X0,cnt,X2))) [unused predicate definition removal 326]
561. ! [X0,X1] : (::(X0,mten,X1) <=> (! [X2] : (? [X3,X4,X5] : (P(X5,X4) & SLC(X3,X5,X2) & SLC(X0,X4,X2) & M(X3))
     | ~PRE(X0,X2)) & ? [X6] : (~AT(X6) & PRE(X0,X6)) & PRE(X0,X1) & ED(X0))) [ennf transformation 315]
562. ! [X0,X1] : (((::(X0,cfbnd,X1) | ::(X0,site,X1)) <=> (! [X2] : (! [X3,X4,X5] : (~P(X5,X4) | ~SLC(X3,X5,X2) | ~
     SLC(X0,X4,X2) | ~M(X3)) | ~PRE(X0,X2)) & ? [X6] : SLC(X0,X6,X1) & PRE(X0,X1) & F(X0))) [ennf transformation
     317]
564. ! [X0,X1] : (::(X0,gdcnt,X1) <=> (! [X2] : (? [X3] : CONCR(X3,X0,X2) | ~PRE(X0,X2)) & PRE(X0,X1))) [ennf
     transformation 323]
565. ! [X0,X1,X2] : (((? [X3,X4] : (tP(X3,X4,X2) & DQT(X1,X4) & DQT(X0,X3)) | P(X0,X1) | tP(X0,X1,X2)) & ::(X1,cnt
     ,X2) & ::(X0,cnt,X2)) | ~cP(X0,X1,X2)) [ennf transformation 336]
566. ? [X0,X1,X2] : (~EX(X0,X2) & (EX(X1,X2) & cP(X0,X1,X2))) [ennf transformation 335]
567. ? [X0,X1,X2] : (~EX(X0,X2) & EX(X1,X2) & cP(X0,X1,X2)) [flattening 566]
570. ! [X1,X0] : (sP1(X1,X0) <=> ((T(X1) & ~T(X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1))) [predicate
     definition introduction]
571. ! [X0,X1] : (EX(X0,X1) <=> sP1(X1,X0)) [definition folding 304,570]
572. ! [X0,X1] : (sP2(X0,X1) <=> (! [X2] : (? [X3,X4,X5] : (P(X5,X4) & SLC(X3,X5,X2) & SLC(X0,X4,X2) & M(X3)) | ~
     PRE(X0,X2)) & ? [X6] : (~AT(X6) & PRE(X0,X6)) & PRE(X0,X1) & ED(X0))) [predicate definition introduction]
573. ! [X0,X1] : (::(X0,mten,X1) <=> sP2(X0,X1)) [definition folding 561,572]
574. ! [X0,X1] : (sP3(X0,X1) <=> (! [X2] : (! [X3,X4,X5] : (~P(X5,X4) | ~SLC(X3,X5,X2) | ~SLC(X0,X4,X2) | ~M(X3))
     | ~PRE(X0,X2)) & ? [X6] : SLC(X0,X6,X1) & PRE(X0,X1) & F(X0))) [predicate definition introduction]
575. ! [X0,X1] : (((::(X0,cfbnd,X1) | ::(X0,site,X1)) <=> sP3(X0,X1)) [definition folding 562,574]
701. ! [X1,X0] : ((sP1(X1,X0) | ((~T(X1) | T(X0) | ~AB(X0)) & (~P(X1,X0) | ~T(X0)) & ~PRE(X0,X1))) & (((T(X1) & ~T
     (X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1)) | ~sP1(X1,X0))) [nnf transformation 570]
702. ! [X1,X0] : ((sP1(X1,X0) | ((~T(X1) | T(X0) | ~AB(X0)) & (~P(X1,X0) | ~T(X0)) & ~PRE(X0,X1))) & ((T(X1) & ~T(
     X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1) | ~sP1(X1,X0))) [flattening 701]
703. ! [X0,X1] : ((sP1(X0,X1) | ((~T(X0) | T(X1) | ~AB(X1)) & (~P(X0,X1) | ~T(X1)) & ~PRE(X1,X0))) & ((T(X0) & ~T(
     X1) & AB(X1)) | (P(X0,X1) & T(X1)) | PRE(X1,X0) | ~sP1(X0,X1))) [rectify 702]
704. ! [X0,X1] : ((EX(X0,X1) | ~sP1(X1,X0)) & (sP1(X1,X0) | ~EX(X0,X1))) [nnf transformation 571]
728. ! [X0,X1] : ((sP2(X0,X1) | (? [X2] : (! [X3,X4,X5] : (~P(X5,X4) | ~SLC(X3,X5,X2) | ~SLC(X0,X4,X2) | ~M(X3)) &
     PRE(X0,X2)) | ! [X6] : (AT(X6) | ~PRE(X0,X6)) | ~PRE(X0,X1) | ~ED(X0))) & ((! [X2] : (? [X3,X4,X5] : (P(X5,
     X4) & SLC(X3,X5,X2) & SLC(X0,X4,X2) & M(X3)) | ~PRE(X0,X2)) & ? [X6] : (~AT(X6) & PRE(X0,X6)) & PRE(X0,X1) &
     ED(X0)) | ~sP2(X0,X1))) [nnf transformation 572]
729. ! [X0,X1] : ((sP2(X0,X1) | ? [X2] : (! [X3,X4,X5] : (~P(X5,X4) | ~SLC(X3,X5,X2) | ~SLC(X0,X4,X2) | ~M(X3)) &
     PRE(X0,X2)) | ! [X6] : (AT(X6) | ~PRE(X0,X6)) | ~PRE(X0,X1) | ~ED(X0)) & ((! [X2] : (? [X3,X4,X5] : (P(X5,X4
     ) & SLC(X3,X5,X2) & SLC(X0,X4,X2) & M(X3)) | ~PRE(X0,X2)) & ? [X6] : (~AT(X6) & PRE(X0,X6)) & PRE(X0,X1) &
     ED(X0)) | ~sP2(X0,X1))) [flattening 728]
730. ! [X0,X1] : ((sP2(X0,X1) | ? [X2] : (! [X3,X4,X5] : (~P(X5,X4) | ~SLC(X3,X5,X2) | ~SLC(X0,X4,X2) | ~M(X3)) &
     PRE(X0,X2)) | ! [X6] : (AT(X6) | ~PRE(X0,X6)) | ~PRE(X0,X1) | ~ED(X0)) & ((! [X7] : (? [X8,X9,X10] : (P(X10,
     X9) & SLC(X8,X10,X7) & SLC(X0,X9,X7) & M(X8)) | ~PRE(X0,X7)) & ? [X11] : (~AT(X11) & PRE(X0,X11)) & PRE(X0,
     X1) & ED(X0)) | ~sP2(X0,X1))) [rectify 729]
731. ! [X0] : (? [X2] : (! [X3,X4,X5] : (~P(X5,X4) | ~SLC(X3,X5,X2) | ~SLC(X0,X4,X2) | ~M(X3)) & PRE(X0,X2)) => (!
     [X5,X4,X3] : (~P(X5,X4) | ~SLC(X3,X5,sK70(X0)) | ~SLC(X0,X4,sK70(X0)) | ~M(X3)) & PRE(X0,sK70(X0)))) [
     choice axiom]
```

```
732. ! [X7,X0] : (? [X8,X9,X10] : (P(X10,X9) & SLC(X8,X10,X7) & SLC(X0,X9,X7) & M(X8)) => (P(sK73(X0,X7),sK72(X0,
       X7)) & SLC(sK71(X0,X7),sK73(X0,X7),X7) & SLC(X0,sK72(X0,X7),X7) & M(sK71(X0,X7)))) [choice axiom]
733. ! [X0] : (? [X11] : (~AT(X11) & PRE(X0,X11)) => (~AT(sK74(X0)) & PRE(X0,sK74(X0)))) [choice axiom]
734. ! [X0,X1] : ((sP2(X0,X1) | (! [X3,X4,X5] : (~P(X5,X4) | ~SLC(X3,X5,sK70(X0)) | ~SLC(X0,X4,sK70(X0)) | ~M(X3))
       & PRE(X0,sK70(X0)))) | ! [X6] : (AT(X6) | ~PRE(X0,X6)) | ~PRE(X0,X1) | ~ED(X0)) & ((! [X7] : ((P(sK73(X0,X7)
       ,sK72(X0,X7)) & SLC(sK71(X0,X7),sK73(X0,X7),X7) & SLC(X0,sK72(X0,X7),X7) & M(sK71(X0,X7))) | ~PRE(X0,X7)) &
       (~AT(sK74(X0)) & PRE(X0,sK74(X0)))) & PRE(X0,X1) & ED(X0)) | ~sP2(X0,X1))) [skolemisation 730,733,732,731]
735. ! [X0,X1] : ((::(X0,mten,X1) | ~sP2(X0,X1)) & (sP2(X0,X1) | ~::(X0,mten,X1))) [nnf transformation 573]
736. ! [X0,X1] : ((sP3(X0,X1) | (? [X2] : (? [X3,X4,X5] : (P(X5,X4) & SLC(X3,X5,X2) & SLC(X0,X4,X2) & M(X3)) & PRE
       (X0,X2)) | ! [X6] : ~SLC(X0,X6,X1) | ~PRE(X0,X1) | ~F(X0))) & ((! [X2] : (! [X3,X4,X5] : (~P(X5,X4) | ~SLC(
       X3,X5,X2) | ~SLC(X0,X4,X2) | ~M(X3)) | ~PRE(X0,X2)) & ? [X6] : SLC(X0,X6,X1) & PRE(X0,X1) & F(X0)) | ~sP3(X0
       ,X1))) [nnf transformation 574]
737. ! [X0,X1] : ((sP3(X0,X1) | ? [X2] : (? [X3,X4,X5] : (P(X5,X4) & SLC(X3,X5,X2) & SLC(X0,X4,X2) & M(X3)) & PRE(
       X0,X2)) | ! [X6] : ~SLC(X0,X6,X1) | ~PRE(X0,X1) | ~F(X0)) & ((! [X2] : (! [X3,X4,X5] : (~P(X5,X4) | ~SLC(X3,
       X5,X2) | ~SLC(X0,X4,X2) | ~M(X3)) | ~PRE(X0,X2)) & ? [X6] : SLC(X0,X6,X1) & PRE(X0,X1) & F(X0)) | ~sP3(X0
       ,X1))) [flattening 736]
738. ! [X0,X1] : ((sP3(X0,X1) | ? [X2] : (? [X3,X4,X5] : (P(X5,X4) & SLC(X3,X5,X2) & SLC(X0,X4,X2) & M(X3)) & PRE(
       X0,X2)) | ! [X6] : ~SLC(X0,X6,X1) | ~PRE(X0,X1) | ~F(X0)) & ((! [X7] : (! [X8,X9,X10] : (~P(X10,X9) | ~SLC(
       X8,X10,X7) | ~SLC(X0,X9,X7) | ~M(X8)) | ~PRE(X0,X7)) & ? [X11] : SLC(X0,X11,X1) & PRE(X0,X1) & F(X0)) | ~sP3
       (X0,X1))) [rectify 737]
739. ! [X0] : (? [X2] : (? [X3,X4,X5] : (P(X5,X4) & SLC(X3,X5,X2) & SLC(X0,X4,X2) & M(X3)) & PRE(X0,X2)) => (? [X5
       ,X4,X3] : (P(X5,X4) & SLC(X3,X5,sK75(X0)) & SLC(X0,X4,sK75(X0)) & M(X3)) & PRE(X0,sK75(X0)))) [choice axiom]
740. ! [X0] : (? [X5,X4,X3] : (P(X5,X4) & SLC(X3,X5,sK75(X0)) & SLC(X0,X4,sK75(X0)) & M(X3)) => (P(sK78(X0),sK77(
       X0)) & SLC(sK76(X0),sK78(X0),sK75(X0)) & SLC(X0,sK77(X0),sK75(X0)) & M(sK76(X0)))) [choice axiom]
741. ! [X1,X0] : (? [X11] : SLC(X0,X11,X1) => SLC(X0,sK79(X0,X1),X1)) [choice axiom]
742. ! [X0,X1] : ((sP3(X0,X1) | ((P(sK78(X0),sK77(X0)) & SLC(sK76(X0),sK78(X0),sK75(X0)) & SLC(X0,sK77(X0),sK75(X0
       )) & M(sK76(X0))) & PRE(X0,sK75(X0))) | ! [X6] : ~SLC(X0,X6,X1) | ~PRE(X0,X1) | ~F(X0)) & ((! [X7] : (! [X8,
       X9,X10] : (~P(X10,X9) | ~SLC(X8,X10,X7) | ~SLC(X0,X9,X7) | ~M(X8)) | ~PRE(X0,X7)) & SLC(X0,sK79(X0,X1),X1) &
       PRE(X0,X1) & F(X0)) | ~sP3(X0,X1))) [skolemisation 738,741,740,739]
743. ! [X0,X1] : (((::(X0,cfbnd,X1) | ::(X0,site,X1)) | ~sP3(X0,X1)) & (sP3(X0,X1) | (~::(X0,cfbnd,X1) & ~::(X0,
       site,X1)))) [nnf transformation 575]
744. ! [X0,X1] : ((::(X0,cfbnd,X1) | ::(X0,site,X1) | ~sP3(X0,X1)) & (sP3(X0,X1) | (~::(X0,cfbnd,X1) & ~::(X0,site
       ,X1)))) [flattening 743]
745. ! [X0,X1] : (((::(X0,sreg,X1) | (~EX(X0,X1) | ~S(X0))) & ((EX(X0,X1) & S(X0)) | ~::(X0,sreg,X1))) [nnf
       transformation 318]
746. ! [X0,X1] : ((::(X0,sreg,X1) | ~EX(X0,X1) | ~S(X0)) & ((EX(X0,X1) & S(X0)) | ~::(X0,sreg,X1))) [flattening
       745]
747. ! [X0,X1] : ((::(X0,imen,X1) | (~::(X0,sreg,X1) & ~::(X0,cfbnd,X1) & ~::(X0,site,X1))) & ((::(X0,sreg,X1) |
       ::(X0,cfbnd,X1) | ::(X0,site,X1)) | ~::(X0,imen,X1))) [nnf transformation 319]
748. ! [X0,X1] : ((::(X0,imen,X1) | (~::(X0,sreg,X1) & ~::(X0,cfbnd,X1) & ~::(X0,site,X1))) & (::(X0,sreg,X1) |
       ::(X0,cfbnd,X1) | ::(X0,site,X1) | ~::(X0,imen,X1))) [flattening 747]
749. ! [X0,X1] : ((::(X0,idcnt,X1) | (~::(X0,imen,X1) & ~::(X0,mten,X1))) & ((::(X0,imen,X1) | ::(X0,mten,X1)) |
       ~::(X0,idcnt,X1))) [nnf transformation 320]
750. ! [X0,X1] : ((::(X0,idcnt,X1) | (~::(X0,imen,X1) & ~::(X0,mten,X1))) & (::(X0,imen,X1) | ::(X0,mten,X1) |
       ~::(X0,idcnt,X1))) [flattening 749]
751. ! [X0,X1] : ((::(X0,sdcnt,X1) | (! [X2] : (~DQT(X0,X2) | S(X2) | ~::(X2,idcnt,X1)) | ~PRE(X0,X1) | ~Q(X0))) &
       ((? [X2] : (DQT(X0,X2) & ~S(X2) & ::(X2,idcnt,X1)) & PRE(X0,X1) & Q(X0)) | ~::(X0,sdcnt,X1))) [nnf
       transformation 321]
752. ! [X0,X1] : ((::(X0,sdcnt,X1) | ! [X2] : (~DQT(X0,X2) | S(X2) | ~::(X2,idcnt,X1)) | ~PRE(X0,X1) | ~Q(X0)) &
       ((? [X2] : (DQT(X0,X2) & ~S(X2) & ::(X2,idcnt,X1)) & PRE(X0,X1) & Q(X0)) | ~::(X0,sdcnt,X1))) [flattening
       751]
753. ! [X0,X1] : ((::(X0,sdcnt,X1) | ! [X2] : (~DQT(X0,X2) | S(X2) | ~::(X2,idcnt,X1)) | ~PRE(X0,X1) | ~Q(X0)) &
       ((? [X3] : (DQT(X0,X3) & ~S(X3) & ::(X3,idcnt,X1)) & PRE(X0,X1) & Q(X0)) | ~::(X0,sdcnt,X1))) [rectify 752]
754. ! [X1,X0] : (? [X3] : (DQT(X0,X3) & ~S(X3) & ::(X3,idcnt,X1)) => (DQT(X0,sK80(X0,X1)) & ~S(sK80(X0,X1)) & ::(
       sK80(X0,X1),idcnt,X1))) [choice axiom]
755. ! [X0,X1] : ((::(X0,sdcnt,X1) | ! [X2] : (~DQT(X0,X2) | S(X2) | ~::(X2,idcnt,X1)) | ~PRE(X0,X1) | ~Q(X0)) &
       (((DQT(X0,sK80(X0,X1)) & ~S(sK80(X0,X1)) & ::(sK80(X0,X1),idcnt,X1)) & PRE(X0,X1) & Q(X0)) | ~::(X0,sdcnt,X1
       ))) [skolemisation 753,754]
761. ! [X0,X1] : ((::(X0,gdcnt,X1) | (? [X2] : (! [X3] : ~CONCR(X3,X0,X2) & PRE(X0,X2)) | ~PRE(X0,X1))) & ((! [X2]
       : (? [X3] : CONCR(X3,X0,X2) | ~PRE(X0,X2)) & PRE(X0,X1)) | ~::(X0,gdcnt,X1))) [nnf transformation 564]
762. ! [X0,X1] : ((::(X0,gdcnt,X1) | ? [X2] : (! [X3] : ~CONCR(X3,X0,X2) & PRE(X0,X2)) | ~PRE(X0,X1)) & ((! [X2] :
       (? [X3] : CONCR(X3,X0,X2) | ~PRE(X0,X2)) & PRE(X0,X1)) | ~::(X0,gdcnt,X1))) [flattening 761]
763. ! [X0,X1] : ((::(X0,gdcnt,X1) | ? [X2] : (! [X3] : ~CONCR(X3,X0,X2) & PRE(X0,X2)) | ~PRE(X0,X1)) & ((! [X4] :
       (? [X5] : CONCR(X5,X0,X4) | ~PRE(X0,X4)) & PRE(X0,X1)) | ~::(X0,gdcnt,X1))) [rectify 762]
764. ! [X0] : (? [X2] : (! [X3] : ~CONCR(X3,X0,X2) & PRE(X0,X2)) => (! [X3] : ~CONCR(X3,X0,sK82(X0)) & PRE(X0,sK82
       (X0)))) [choice axiom]
765. ! [X4,X0] : (? [X5] : CONCR(X5,X0,X4) => CONCR(sK83(X0,X4),X0,X4)) [choice axiom]
766. ! [X0,X1] : ((::(X0,gdcnt,X1) | (! [X3] : ~CONCR(X3,X0,sK82(X0)) & PRE(X0,sK82(X0))) | ~PRE(X0,X1)) & ((! [X4
       ] : (CONCR(sK83(X0,X4),X0,X4) | ~PRE(X0,X4)) & PRE(X0,X1)) | ~::(X0,gdcnt,X1))) [skolemisation 763,765,764]
767. ! [X0,X1] : ((::(X0,cnt,X1) | (~::(X0,gdcnt,X1) & ~::(X0,sdcnt,X1) & ~::(X0,idcnt,X1))) & ((::(X0,gdcnt,X1) |
       ::(X0,sdcnt,X1) | ::(X0,idcnt,X1)) | ~::(X0,cnt,X1))) [nnf transformation 324]
768. ! [X0,X1] : ((::(X0,cnt,X1) | (~::(X0,gdcnt,X1) & ~::(X0,sdcnt,X1) & ~::(X0,idcnt,X1))) & (::(X0,gdcnt,X1) |
       ::(X0,sdcnt,X1) | ::(X0,idcnt,X1) | ~::(X0,cnt,X1))) [flattening 767]
769. ! [X2,X1,X0] : (? [X3,X4] : (tP(X3,X4,X2) & DQT(X1,X4) & DQT(X0,X3)) => (tP(sK84(X0,X1,X2),sK85(X0,X1,X2),X2)
       & DQT(X1,sK85(X0,X1,X2)) & DQT(X0,sK84(X0,X1,X2)))) [choice axiom]
770. ! [X0,X1,X2] : ((((tP(sK84(X0,X1,X2),sK85(X0,X1,X2),X2) & DQT(X1,sK85(X0,X1,X2)) & DQT(X0,sK84(X0,X1,X2))) |
       P(X0,X1) | tP(X0,X1,X2)) & ::(X1,cnt,X2) & ::(X0,cnt,X2)) | ~cP(X0,X1,X2)) [skolemisation 565,769]
771. ? [X0,X1,X2] : (~EX(X0,X2) & EX(X1,X2) & cP(X0,X1,X2)) => (~EX(sK86,sK88) & EX(sK87,sK88) & cP(sK86,sK87,sK88
       )) [choice axiom]
772. ~EX(sK86,sK88) & EX(sK87,sK88) & cP(sK86,sK87,sK88) [skolemisation 567,771]
1033. ~PRE(X1,X0) | sP1(X0,X1) [cnf transformation 703]
1037. ~sP1(X1,X0) | EX(X0,X1) [cnf transformation 704]
1074. ~sP2(X0,X1) | PRE(X0,X1) [cnf transformation 734]
1083. ~::(X0,mten,X1) | sP2(X0,X1) [cnf transformation 735]
1086. ~sP3(X0,X1) | PRE(X0,X1) [cnf transformation 742]
1094. ~::(X0,site,X1) | sP3(X0,X1) [cnf transformation 744]
1095. ~::(X0,cfbnd,X1) | sP3(X0,X1) [cnf transformation 744]
1098. ~::(X0,sreg,X1) | EX(X0,X1) [cnf transformation 746]
1100. ~::(X0,imen,X1) | ::(X0,cfbnd,X1) | ::(X0,site,X1) | ::(X0,sreg,X1) [cnf transformation 748]
1104. ~::(X0,idcnt,X1) | ::(X0,mten,X1) | ::(X0,imen,X1) [cnf transformation 750]
1108. ~::(X0,sdcnt,X1) | PRE(X0,X1) [cnf transformation 755]
```

```
1118. ~::(X0,gdcnt,X1) | PRE(X0,X1) [cnf transformation 766]
1122. ~::(X0,cnt,X1) | ::(X0,sdcnt,X1) | ::(X0,idcnt,X1) | ::(X0,gdcnt,X1) [cnf transformation 768]
1126. ~cP(X0,X1,X2) | ::(X0,cnt,X2) [cnf transformation 770]
1131. cP(sK86,sK87,sK88) [cnf transformation 772]
1133. ~EX(sK86,sK88) [cnf transformation 772]
1389. ::(sK86,cnt,sK88) [resolution 1126,1131]
2761. ::(sK86,sdcnt,sK88) | ::(sK86,idcnt,sK88) | ::(sK86,gdcnt,sK88) [resolution 1122,1389]
2764. 27 <=> ::(sK86,gdcnt,sK88) [avatar definition]
2766. ::(sK86,gdcnt,sK88) <- (27) [avatar component clause 2764]
2768. 28 <=> ::(sK86,idcnt,sK88) [avatar definition]
2770. ::(sK86,idcnt,sK88) <- (28) [avatar component clause 2768]
2772. 29 <=> ::(sK86,sdcnt,sK88) [avatar definition]
2774. ::(sK86,sdcnt,sK88) <- (29) [avatar component clause 2772]
2775. 27 | 28 | 29 [avatar split clause 2761,2772,2768,2764]
2840. 40 <=> PRE(sK86,sK88) [avatar definition]
2841. PRE(sK86,sK88) <- (40) [avatar component clause 2840]
2842. ~PRE(sK86,sK88) <- (~40) [avatar component clause 2840]
2889. ::(sK86,mten,sK88) | ::(sK86,imen,sK88) <- (28) [resolution 2770,1104]
2892. 42 <=> ::(sK86,imen,sK88) [avatar definition]
2894. ::(sK86,imen,sK88) <- (42) [avatar component clause 2892]
2896. 43 <=> ::(sK86,mten,sK88) [avatar definition]
2898. ::(sK86,mten,sK88) <- (43) [avatar component clause 2896]
2899. 42 | 43 | ~28 [avatar split clause 2889,2768,2896,2892]
2927. ::(sK86,cfbnd,sK88) | ::(sK86,site,sK88) | ::(sK86,sreg,sK88) <- (42) [resolution 2894,1100]
2930. 46 <=> ::(sK86,sreg,sK88) [avatar definition]
2932. ::(sK86,sreg,sK88) <- (46) [avatar component clause 2930]
2934. 47 <=> ::(sK86,site,sK88) [avatar definition]
2936. ::(sK86,site,sK88) <- (47) [avatar component clause 2934]
2938. 48 <=> ::(sK86,cfbnd,sK88) [avatar definition]
2940. ::(sK86,cfbnd,sK88) <- (48) [avatar component clause 2938]
2941. 46 | 47 | 48 | ~42 [avatar split clause 2927,2892,2938,2934,2930]
2974. sP3(sK86,sK88) <- (47) [resolution 2936,1094]
2975. PRE(sK86,sK88) <- (47) [resolution 2974,1086]
2977. $false <- (~40, 47) [subsumption resolution 2975,2842]
2978. 40 | ~47 [avatar contradiction clause 2977]
3003. sP3(sK86,sK88) <- (48) [resolution 2940,1095]
3024. PRE(sK86,sK88) <- (48) [resolution 3003,1086]
3026. $false <- (~40, 48) [subsumption resolution 3024,2842]
3027. 40 | ~48 [avatar contradiction clause 3026]
3062. sP1(sK88,sK86) <- (40) [resolution 2841,1033]
3164. EX(sK86,sK88) <- (40) [resolution 3062,1037]
3165. $false <- (40) [subsumption resolution 3164,1133]
3166. ~40 [avatar contradiction clause 3165]
3208. EX(sK86,sK88) <- (46) [resolution 2932,1098]
3210. $false <- (46) [subsumption resolution 3208,1133]
3211. ~46 [avatar contradiction clause 3210]
3217. PRE(sK86,sK88) <- (27) [resolution 2766,1118]
3218. $false <- (27, ~40) [subsumption resolution 3217,2842]
3219. ~27 | 40 [avatar contradiction clause 3218]
3237. PRE(sK86,sK88) <- (29) [resolution 2774,1108]
3239. $false <- (29, ~40) [subsumption resolution 3237,2842]
3240. ~29 | 40 [avatar contradiction clause 3239]
3276. sP2(sK86,sK88) <- (43) [resolution 2898,1083]
3286. PRE(sK86,sK88) <- (43) [resolution 3276,1074]
3295. $false <- (~40, 43) [subsumption resolution 3286,2842]
3296. 40 | ~43 [avatar contradiction clause 3295]
3297. $false [avatar sat refutation 2775,2899,2941,2978,3027,3166,3211,3219,3240,3296]
```

Proof of Theorem (t_{ab}44)

```
7. ! [X0,X1] : (P(X0,X1) => (S(X0) <=> S(X1))) [input]
12. ! [X0,X1] : (DQT(X0,X1) => ((NPED(X1) & AQ(X0)) | (PED(X1) & PQ(X0)) | (PD(X1) & tQ(X0)))) [input]
34. ! [X0,X4] : (PRE(X0,X4) => (T(X4) & (Q(X0) | PD(X0) | ED(X0)))) [input]
47. ! [X0,X1,X4] : (tP(X0,X1,X4) => (T(X4) & ED(X1) & ED(X0))) [input]
116. ! [X0] : ((PED(X0) | NPED(X0) | AS(X0)) <=> ED(X0)) [input]
118. ! [X0] : ((tQ(X0) | PQ(X0) | AQ(X0)) <=> Q(X0)) [input]
124. ! [X0] : ((TR(X0) | PR(X0) | AR(X0)) <=> R(X0)) [input]
125. ! [X0] : (R(X0) => AB(X0)) [input]
126. ! [X0] : ((POB(X0) | M(X0) | F(X0)) => PED(X0)) [input]
128. ! [X0] : (S(X0) => PR(X0)) [input]
130. ! [X0] : (T(X0) => TR(X0)) [input]
132. ~? [X0] : (ED(X0) & AB(X0)) [input]
134. ~? [X0] : (Q(X0) & AB(X0)) [input]
153. ~? [X0] : (PR(X0) & TR(X0)) [input]
162. ! [X0,X4] : (EX(X0,X4) <=> ((T(X4) & ~T(X0) & AB(X0)) | (P(X4,X0) & T(X0)) | PRE(X0,X4))) [input]
173. ! [X0,X4] : (::(X0,mten,X4) <=> (! [X5] : (PRE(X0,X5) => ? [X1] : ? [X3] : ? [X8] : (P(X8,X3) & SLC(X1,X8,X5)
       & SLC(X0,X3,X5) & M(X1))) & ? [X5] : (~AT(X5) & PRE(X0,X5)) & PRE(X0,X4) & ED(X0))) [input]
174. ! [X0,X4] : ((::(X0,cfbnd,X4) | ::(X0,site,X4)) <=> (! [X5] : (PRE(X0,X5) => ~? [X1] : ? [X3] : ? [X8] : (P(
       X8,X3) & SLC(X1,X8,X5) & SLC(X0,X3,X5) & M(X1))) & ? [X3] : SLC(X0,X3,X4) & PRE(X0,X4) & F(X0))) [input]
175. ! [X0,X4] : (::(X0,sreg,X4) <=> (EX(X0,X4) & S(X0))) [input]
176. ! [X0,X4] : (::(X0,imen,X4) <=> (::(X0,sreg,X4) | ::(X0,cfbnd,X4) | ::(X0,site,X4))) [input]
177. ! [X0,X4] : (::(X0,idcnt,X4) <=> (::(X0,imen,X4) | ::(X0,mten,X4))) [input]
178. ! [X0,X4] : (::(X0,sdcnt,X4) <=> (? [X1] : (DQT(X0,X1) & ~S(X1) & ::(X1,idcnt,X4)) & PRE(X0,X4) & Q(X0))) [
       input]
180. ! [X0,X4] : (::(X0,gdcnt,X4) <=> (! [X5] : (PRE(X0,X5) => ? [X1] : CONCR(X1,X0,X5)) & PRE(X0,X4))) [input]
181. ! [X0,X4] : (::(X0,cnt,X4) <=> (::(X0,gdcnt,X4) | ::(X0,sdcnt,X4) | ::(X0,idcnt,X4))) [input]
182. ! [X0,X1,X4] : (cP(X0,X1,X4) <=> ((? [X2] : ? [X5] : (tP(X2,X5,X4) & DQT(X1,X5) & DQT(X0,X2)) | P(X0,X1) | tP
       (X0,X1,X4)) & ::(X1,cnt,X4) & ::(X0,cnt,X4))) [input]
189. ! [X0,X1,X4] : (cP(X0,X1,X4) => (::(X0,sreg,X4) <=> ::(X1,sreg,X4))) [input]
190. ~! [X0,X1,X4] : (cP(X0,X1,X4) => (::(X0,sreg,X4) <=> ::(X1,sreg,X4))) [negated conjecture 189]
```

```
200. ! [X0,X1] : (PRE(X0,X1) => (T(X1) & (Q(X0) | PD(X0) | ED(X0)))) [rectify 34]
213. ! [X0,X1,X2] : (tP(X0,X1,X2) => (T(X2) & ED(X1) & ED(X0))) [rectify 47]
304. ! [X0,X1] : (EX(X0,X1) <=> ((T(X1) & ~T(X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1))) [rectify 162]
314. ! [X0,X1] : (::(X0,mten,X1) <=> (! [X2] : (PRE(X0,X2) => ? [X3] : ? [X4] : ? [X5] : (P(X5,X4) & SLC(X3,X5,X2)
     & SLC(X0,X4,X2) & M(X3))) & ? [X6] : (~AT(X6) & PRE(X0,X6)) & PRE(X0,X1) & ED(X0))) [rectify 173]
315. ! [X0,X1] : (::(X0,mten,X1) <=> (! [X2] : (PRE(X0,X2) => ? [X3,X4,X5] : (P(X5,X4) & SLC(X3,X5,X2) & SLC(X0,X4
     ,X2) & M(X3))) & ? [X6] : (~AT(X6) & PRE(X0,X6)) & PRE(X0,X1) & ED(X0))) [flattening 314]
316. ! [X0,X1] : ((::(X0,cfbnd,X1) | ::(X0,site,X1)) <=> (! [X2] : (PRE(X0,X2) => ~? [X3] : ? [X4] : ? [X5] : (P(
     X5,X4) & SLC(X3,X5,X2) & SLC(X0,X4,X2) & M(X3))) & ? [X6] : SLC(X0,X6,X1) & PRE(X0,X1) & F(X0))) [rectify
     174]
317. ! [X0,X1] : ((::(X0,cfbnd,X1) | ::(X0,site,X1)) <=> (! [X2] : (PRE(X0,X2) => ~? [X3,X4,X5] : (P(X5,X4) & SLC(
     X3,X5,X2) & SLC(X0,X4,X2) & M(X3))) & ? [X6] : SLC(X0,X6,X1) & PRE(X0,X1) & F(X0))) [flattening 316]
318. ! [X0,X1] : (::(X0,sreg,X1) <=> (EX(X0,X1) & S(X0))) [rectify 175]
319. ! [X0,X1] : (::(X0,imen,X1) <=> (::(X0,sreg,X1) | ::(X0,cfbnd,X1) | ::(X0,site,X1))) [rectify 176]
320. ! [X0,X1] : (::(X0,idcnt,X1) <=> (::(X0,imen,X1) | ::(X0,mten,X1))) [rectify 177]
321. ! [X0,X1] : (::(X0,sdcnt,X1) <=> (? [X2] : (DQT(X0,X2) & ~S(X2) & ::(X2,idcnt,X1)) & PRE(X0,X1) & Q(X0))) [
     rectify 178]
323. ! [X0,X1] : (::(X0,gdcnt,X1) <=> (! [X2] : (PRE(X0,X2) => ? [X3] : CONCR(X3,X0,X2)) & PRE(X0,X1))) [rectify
     180]
324. ! [X0,X1] : (::(X0,cnt,X1) <=> (::(X0,gdcnt,X1) | ::(X0,sdcnt,X1) | ::(X0,idcnt,X1))) [rectify 181]
325. ! [X0,X1,X2] : (cP(X0,X1,X2) <=> ((? [X3] : ? [X4] : (tP(X3,X4,X2) & DQT(X1,X4) & DQT(X0,X3)) | P(X0,X1) | tP
     (X0,X1,X2)) & ::(X1,cnt,X2) & ::(X0,cnt,X2))) [rectify 182]
326. ! [X0,X1,X2] : (cP(X0,X1,X2) <=> ((? [X3,X4] : (tP(X3,X4,X2) & DQT(X1,X4) & DQT(X0,X3)) | P(X0,X1) | tP(X0,X1
     ,X2)) & ::(X1,cnt,X2) & ::(X0,cnt,X2))) [flattening 325]
335. ~! [X0,X1,X2] : (cP(X0,X1,X2) => (::(X0,sreg,X2) <=> ::(X1,sreg,X2))) [rectify 190]
336. ! [X0,X1,X2] : (cP(X0,X1,X2) => ((? [X3,X4] : (tP(X3,X4,X2) & DQT(X1,X4) & DQT(X0,X3)) | P(X0,X1) | tP(X0,X1,
     X2)) & ::(X1,cnt,X2) & ::(X0,cnt,X2))) [unused predicate definition removal 326]
337. ! [X0] : ((TR(X0) | PR(X0) | AR(X0)) => R(X0)) [unused predicate definition removal 124]
374. ! [X0,X1] : ((S(X0) <=> S(X1)) | ~P(X0,X1)) [ennf transformation 7]
382. ! [X0,X1] : (((NPED(X1) & AQ(X0)) | (PED(X1) & PQ(X0)) | (PD(X1) & tQ(X0))) | ~DQT(X0,X1)) [ennf
     transformation 12]
383. ! [X0,X1] : ((NPED(X1) & AQ(X0)) | (PED(X1) & PQ(X0)) | (PD(X1) & tQ(X0)) | ~DQT(X0,X1)) [flattening 382]
412. ! [X0,X1] : ((T(X1) & (Q(X0) | PD(X0) | ED(X0))) | ~PRE(X0,X1)) [ennf transformation 200]
432. ! [X0,X1,X2] : ((T(X2) & ED(X1) & ED(X0)) | ~tP(X0,X1,X2)) [ennf transformation 213]
534. ! [X0] : (R(X0) | (~TR(X0) & ~PR(X0) & ~AR(X0))) [ennf transformation 337]
535. ! [X0] : (AB(X0) | ~R(X0)) [ennf transformation 125]
536. ! [X0] : (PED(X0) | (~POB(X0) & ~M(X0) & ~F(X0))) [ennf transformation 126]
537. ! [X0] : (PR(X0) | ~S(X0)) [ennf transformation 128]
539. ! [X0] : (TR(X0) | ~T(X0)) [ennf transformation 130]
541. ! [X0] : (~ED(X0) | ~AB(X0)) [ennf transformation 132]
543. ! [X0] : (~Q(X0) | ~AB(X0)) [ennf transformation 134]
556. ! [X0] : (~PR(X0) | ~TR(X0)) [ennf transformation 153]
561. ! [X0,X1] : (::(X0,mten,X1) <=> (! [X2] : (? [X3,X4,X5] : (P(X5,X4) & SLC(X3,X5,X2) & SLC(X0,X4,X2) & M(X3))
     | ~PRE(X0,X2)) & ? [X6] : (~AT(X6) & PRE(X0,X6)) & PRE(X0,X1) & ED(X0))) [ennf transformation 315]
562. ! [X0,X1] : ((::(X0,cfbnd,X1) | ::(X0,site,X1)) <=> (! [X2] : (! [X3,X4,X5] : (~P(X5,X4) | ~SLC(X3,X5,X2) | ~
     SLC(X0,X4,X2) | ~M(X3)) | ~PRE(X0,X2)) & ? [X6] : SLC(X0,X6,X1) & PRE(X0,X1) & F(X0))) [ennf transformation
     317]
564. ! [X0,X1] : (::(X0,gdcnt,X1) <=> (! [X2] : (? [X3] : CONCR(X3,X0,X2) | ~PRE(X0,X2)) & PRE(X0,X1))) [ennf
     transformation 323]
565. ! [X0,X1,X2] : (((? [X3,X4] : (tP(X3,X4,X2) & DQT(X1,X4) & DQT(X0,X3)) | P(X0,X1) | tP(X0,X1,X2)) & ::(X1,cnt
     ,X2) & ::(X0,cnt,X2)) | ~cP(X0,X1,X2)) [ennf transformation 336]
566. ? [X0,X1,X2] : ((::(X0,sreg,X2) <~> ::(X1,sreg,X2)) & cP(X0,X1,X2)) [ennf transformation 335]
569. ! [X1,X0] : (sP1(X1,X0) <=> ((T(X1) & ~T(X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1))) [predicate
     definition introduction]
570. ! [X0,X1] : (EX(X0,X1) <=> sP1(X1,X0)) [definition folding 304,569]
571. ! [X0,X1] : (sP2(X0,X1) <=> (! [X2] : (? [X3,X4,X5] : (P(X5,X4) & SLC(X3,X5,X2) & SLC(X0,X4,X2) & M(X3)) | ~
     PRE(X0,X2)) & ? [X6] : (~AT(X6) & PRE(X0,X6)) & PRE(X0,X1) & ED(X0))) [predicate definition introduction]
572. ! [X0,X1] : (::(X0,mten,X1) <=> sP2(X0,X1)) [definition folding 561,571]
573. ! [X0,X1] : (sP3(X0,X1) <=> (! [X2] : (! [X3,X4,X5] : (~P(X5,X4) | ~SLC(X3,X5,X2) | ~SLC(X0,X4,X2) | ~M(X3))
     | ~PRE(X0,X2)) & ? [X6] : SLC(X0,X6,X1) & PRE(X0,X1) & F(X0))) [predicate definition introduction]
574. ! [X0,X1] : ((::(X0,cfbnd,X1) | ::(X0,site,X1)) <=> sP3(X0,X1)) [definition folding 562,573]
591. ! [X0,X1] : (((S(X0) | ~S(X1)) & (S(X1) | ~S(X0))) | ~P(X0,X1)) [nnf transformation 374]
693. ! [X0] : (((PED(X0) | NPED(X0) | AS(X0)) | ~ED(X0)) & (ED(X0) | (~PED(X0) & ~NPED(X0) & ~AS(X0)))) [nnf
     transformation 116]
694. ! [X0] : ((PED(X0) | NPED(X0) | AS(X0) | ~ED(X0)) & (ED(X0) | (~PED(X0) & ~NPED(X0) & ~AS(X0)))) [flattening
     693]
695. ! [X0] : (((tQ(X0) | PQ(X0) | AQ(X0)) | ~Q(X0)) & (Q(X0) | (~tQ(X0) & ~PQ(X0) & ~AQ(X0)))) [nnf
     transformation 118]
696. ! [X0] : ((tQ(X0) | PQ(X0) | AQ(X0) | ~Q(X0)) & (Q(X0) | (~tQ(X0) & ~PQ(X0) & ~AQ(X0)))) [flattening 695]
700. ! [X1,X0] : ((sP1(X1,X0) | ((~T(X1) | T(X0) | ~AB(X0)) & (~P(X1,X0) | ~T(X0)) & ~PRE(X0,X1))) & (((T(X1) & ~T
     (X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1)) | ~sP1(X1,X0))) [nnf transformation 569]
701. ! [X1,X0] : ((sP1(X1,X0) | ((~T(X1) | T(X0) | ~AB(X0)) & (~P(X1,X0) | ~T(X0)) & ~PRE(X0,X1))) & ((T(X1) & ~T(
     X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1) | ~sP1(X1,X0))) [flattening 700]
702. ! [X0,X1] : ((sP1(X0,X1) | ((~T(X0) | T(X1) | ~AB(X1)) & (~P(X0,X1) | ~T(X1)) & ~PRE(X1,X0))) & ((T(X0) & ~T(
     X1) & AB(X1)) | (P(X0,X1) & T(X1)) | PRE(X1,X0) | ~sP1(X0,X1))) [rectify 701]
703. ! [X0,X1] : ((EX(X0,X1) | ~sP1(X1,X0)) & (sP1(X1,X0) | ~EX(X0,X1))) [nnf transformation 570]
727. ! [X0,X1] : ((sP2(X0,X1) | (? [X2] : (! [X3,X4,X5] : (~P(X5,X4) | ~SLC(X3,X5,X2) | ~SLC(X0,X4,X2) | ~M(X3)) &
     PRE(X0,X2)) | ! [X6] : (AT(X6) | ~PRE(X0,X6)) | ~PRE(X0,X1) | ~ED(X0))) & ((! [X2] : (? [X3,X4,X5] : (P(X5,
     X4) & SLC(X3,X5,X2) & SLC(X0,X4,X2) & M(X3)) | ~PRE(X0,X2)) & ? [X6] : (~AT(X6) & PRE(X0,X6)) & PRE(X0,X1) &
     ED(X0)) | ~sP2(X0,X1))) [nnf transformation 571]
728. ! [X0,X1] : ((sP2(X0,X1) | ? [X2] : (! [X3,X4,X5] : (~P(X5,X4) | ~SLC(X3,X5,X2) | ~SLC(X0,X4,X2) | ~M(X3)) &
     PRE(X0,X2)) | ! [X6] : (AT(X6) | ~PRE(X0,X6)) | ~PRE(X0,X1) | ~ED(X0)) & ((! [X2] : (? [X3,X4,X5] : (P(X5,X4
     ) & SLC(X3,X5,X2) & SLC(X0,X4,X2) & M(X3)) | ~PRE(X0,X2)) & ? [X6] : (~AT(X6) & PRE(X0,X6)) & PRE(X0,X1) &
     ED(X0)) | ~sP2(X0,X1))) [flattening 727]
729. ! [X0,X1] : ((sP2(X0,X1) | ? [X2] : (! [X3,X4,X5] : (~P(X5,X4) | ~SLC(X3,X5,X2) | ~SLC(X0,X4,X2) | ~M(X3)) &
     PRE(X0,X2)) | ! [X6] : (AT(X6) | ~PRE(X0,X6)) | ~PRE(X0,X1) | ~ED(X0)) & ((! [X7] : (? [X8,X9,X10] : (P(X10,
     X9) & SLC(X8,X10,X7) & SLC(X0,X9,X7) & M(X8)) | ~PRE(X0,X7)) & ? [X11] : (~AT(X11) & PRE(X0,X11)) & PRE(X0,
     X1) & ED(X0)) | ~sP2(X0,X1))) [rectify 728]
730. ! [X0] : (? [X2] : (! [X3,X4,X5] : (~P(X5,X4) | ~SLC(X3,X5,X2) | ~SLC(X0,X4,X2) | ~M(X3)) & PRE(X0,X2)) => (!
     [X5,X4,X3] : (~P(X5,X4) | ~SLC(X3,X5,sK70(X0)) | ~SLC(X0,X4,sK70(X0)) | ~M(X3)) & PRE(X0,sK70(X0)))) [
     choice axiom]
```

731. ! [X7,X0] : (? [X8,X9,X10] : (P(X10,X9) & SLC(X8,X10,X7) & SLC(X0,X9,X7) & M(X8)) => (P(sK73(X0,X7),sK72(X0,X7)) & SLC(sK71(X0,X7),sK73(X0,X7),X7) & SLC(X0,sK72(X0,X7),X7) & M(sK71(X0,X7)))) [choice axiom]
732. ! [X0] : (? [X11] : (~AT(X11) & PRE(X0,X11)) => (~AT(sK74(X0)) & PRE(X0,sK74(X0)))) [choice axiom]
733. ! [X0,X1] : ((sP2(X0,X1) | (! [X3,X4,X5] : (~P(X5,X4) | ~SLC(X3,X5,sK70(X0)) | ~SLC(X0,X4,sK70(X0)) | ~M(X3)) & PRE(X0,sK70(X0)))) | ! [X6] : (AT(X6) | ~PRE(X0,X6)) | ~PRE(X0,X1) | ~ED(X0)) & ((! [X7] : ((P(sK73(X0,X7),sK72(X0,X7)) & SLC(sK71(X0,X7),sK73(X0,X7),X7) & SLC(X0,sK72(X0,X7),X7) & M(sK71(X0,X7))) | ~PRE(X0,X7)) & (~AT(sK74(X0)) & PRE(X0,sK74(X0)))) & PRE(X0,X1) & ED(X0)) | ~sP2(X0,X1))) [skolemisation 729,732,731,730]
734. ! [X0,X1] : ((::(X0,mten,X1) | ~sP2(X0,X1)) & (sP2(X0,X1) | ~::(X0,mten,X1))) [nnf transformation 572]
735. ! [X0,X1] : ((sP3(X0,X1) | (? [X2] : (? [X3,X4,X5] : (P(X5,X4) & SLC(X3,X5,X2) & SLC(X0,X4,X2) & M(X3)) & PRE(X0,X2)) | ! [X6] : ~SLC(X0,X6,X1) | ~PRE(X0,X1) | ~F(X0))) & ((! [X2] : (! [X3,X4,X5] : (~P(X5,X4) | ~SLC(X3,X5,X2) | ~SLC(X0,X4,X2) | ~M(X3)) | ~PRE(X0,X2)) & ? [X6] : SLC(X0,X6,X1) & PRE(X0,X1) & F(X0)) | ~sP3(X0,X1))) [nnf transformation 573]
736. ! [X0,X1] : ((sP3(X0,X1) | ? [X2] : (? [X3,X4,X5] : (P(X5,X4) & SLC(X3,X5,X2) & SLC(X0,X4,X2) & M(X3)) & PRE(X0,X2)) | ! [X6] : ~SLC(X0,X6,X1) | ~PRE(X0,X1) | ~F(X0)) & ((! [X2] : (! [X3,X4,X5] : (~P(X5,X4) | ~SLC(X3,X5,X2) | ~SLC(X0,X4,X2) | ~M(X3)) | ~PRE(X0,X2)) & ? [X6] : SLC(X0,X6,X1) & PRE(X0,X1) & F(X0)) | ~sP3(X0,X1))) [flattening 735]
737. ! [X0,X1] : ((sP3(X0,X1) | ? [X2] : (? [X3,X4,X5] : (P(X5,X4) & SLC(X3,X5,X2) & SLC(X0,X4,X2) & M(X3)) & PRE(X0,X2)) | ! [X6] : ~SLC(X0,X6,X1) | ~PRE(X0,X1) | ~F(X0)) & ((! [X7] : (! [X8,X9,X10] : (~P(X10,X9) | ~SLC(X8,X10,X7) | ~SLC(X0,X9,X7) | ~M(X8)) | ~PRE(X0,X7)) & ? [X11] : SLC(X0,X11,X1) & PRE(X0,X1) & F(X0)) | ~sP3(X0,X1))) [rectify 736]
738. ! [X0] : (? [X2] : (? [X3,X4,X5] : (P(X5,X4) & SLC(X3,X5,X2) & SLC(X0,X4,X2) & M(X3)) & PRE(X0,X2)) => (? [X5,X4,X3] : (P(X5,X4) & SLC(X3,X5,sK75(X0)) & SLC(X0,X4,sK75(X0)) & M(X3)) & PRE(X0,sK75(X0)))) [choice axiom]
739. ! [X0] : (? [X5,X4,X3] : (P(X5,X4) & SLC(X3,X5,sK75(X0)) & SLC(X0,X4,sK75(X0)) & M(X3)) => (P(sK78(X0),sK77(X0)) & SLC(sK76(X0),sK78(X0),sK75(X0)) & SLC(X0,sK77(X0),sK75(X0)) & M(sK76(X0)))) [choice axiom]
740. ! [X1,X0] : (? [X11] : SLC(X0,X11,X1) => SLC(X0,sK79(X0,X1),X1)) [choice axiom]
741. ! [X0,X1] : ((sP3(X0,X1) | ((P(sK78(X0),sK77(X0)) & SLC(sK76(X0),sK78(X0),sK75(X0)) & SLC(X0,sK77(X0),sK75(X0)) & M(sK76(X0))) & PRE(X0,sK75(X0))) | ! [X6] : ~SLC(X0,X6,X1) | ~PRE(X0,X1) | ~F(X0)) & ((! [X7] : (! [X8,X9,X10] : (~P(X10,X9) | ~SLC(X8,X10,X7) | ~SLC(X0,X9,X7) | ~M(X8)) | ~PRE(X0,X7)) & SLC(X0,sK79(X0,X1),X1) & PRE(X0,X1) & F(X0)) | ~sP3(X0,X1))) [skolemisation 737,740,739,738]
742. ! [X0,X1] : (((::(X0,cfbnd,X1) | ::(X0,site,X1)) | ~sP3(X0,X1)) & (sP3(X0,X1) | (~::(X0,cfbnd,X1) & ~::(X0,site,X1)))) [nnf transformation 574]
743. ! [X0,X1] : ((::(X0,cfbnd,X1) | ::(X0,site,X1) | ~sP3(X0,X1)) & (sP3(X0,X1) | (~::(X0,cfbnd,X1) & ~::(X0,site,X1)))) [flattening 742]
744. ! [X0,X1] : ((::(X0,sreg,X1) | (~EX(X0,X1) | ~S(X0))) & ((EX(X0,X1) & S(X0)) | ~::(X0,sreg,X1))) [nnf transformation 318]
745. ! [X0,X1] : ((::(X0,sreg,X1) | ~EX(X0,X1) | ~S(X0)) & ((EX(X0,X1) & S(X0)) | ~::(X0,sreg,X1))) [flattening 744]
746. ! [X0,X1] : ((::(X0,imen,X1) | (~::(X0,sreg,X1) & ~::(X0,cfbnd,X1) & ~::(X0,site,X1))) & ((::(X0,sreg,X1) | ::(X0,cfbnd,X1) | ::(X0,site,X1)) | ~::(X0,imen,X1))) [nnf transformation 319]
747. ! [X0,X1] : ((::(X0,imen,X1) | (~::(X0,sreg,X1) & ~::(X0,cfbnd,X1) & ~::(X0,site,X1))) & (::(X0,sreg,X1) | ::(X0,cfbnd,X1) | ::(X0,site,X1) | ~::(X0,imen,X1))) [flattening 746]
748. ! [X0,X1] : ((::(X0,idcnt,X1) | (~::(X0,imen,X1) & ~::(X0,mten,X1))) & ((::(X0,imen,X1) | ::(X0,mten,X1)) | ~::(X0,idcnt,X1))) [nnf transformation 320]
749. ! [X0,X1] : ((::(X0,idcnt,X1) | (~::(X0,imen,X1) & ~::(X0,mten,X1))) & (::(X0,imen,X1) | ::(X0,mten,X1) | ~::(X0,idcnt,X1))) [flattening 748]
750. ! [X0,X1] : ((::(X0,sdcnt,X1) | (! [X2] : (~DQT(X0,X2) | S(X2) | ~::(X2,idcnt,X1)) | ~PRE(X0,X1) | ~Q(X0))) & ((? [X2] : (DQT(X0,X2) & ~S(X2) & ::(X2,idcnt,X1)) & PRE(X0,X1) & Q(X0)) | ~::(X0,sdcnt,X1))) [nnf transformation 321]
751. ! [X0,X1] : ((::(X0,sdcnt,X1) | ! [X2] : (~DQT(X0,X2) | S(X2) | ~::(X2,idcnt,X1)) | ~PRE(X0,X1) | ~Q(X0)) & ((? [X2] : (DQT(X0,X2) & ~S(X2) & ::(X2,idcnt,X1)) & PRE(X0,X1) & Q(X0)) | ~::(X0,sdcnt,X1))) [flattening 750]
752. ! [X0,X1] : ((::(X0,sdcnt,X1) | ! [X2] : (~DQT(X0,X2) | S(X2) | ~::(X2,idcnt,X1)) | ~PRE(X0,X1) | ~Q(X0)) & ((? [X3] : (DQT(X0,X3) & ~S(X3) & ::(X3,idcnt,X1)) & PRE(X0,X1) & Q(X0)) | ~::(X0,sdcnt,X1))) [rectify 751]
753. ! [X1,X0] : (? [X3] : (DQT(X0,X3) & ~S(X3) & ::(X3,idcnt,X1)) => (DQT(X0,sK80(X0,X1)) & ~S(sK80(X0,X1)) & ::(sK80(X0,X1),idcnt,X1))) [choice axiom]
754. ! [X0,X1] : ((::(X0,sdcnt,X1) | ! [X2] : (~DQT(X0,X2) | S(X2) | ~::(X2,idcnt,X1)) | ~PRE(X0,X1) | ~Q(X0)) & (((DQT(X0,sK80(X0,X1)) & ~S(sK80(X0,X1)) & ::(sK80(X0,X1),idcnt,X1)) & PRE(X0,X1) & Q(X0)) | ~::(X0,sdcnt,X1))) [skolemisation 752,753]
760. ! [X0,X1] : ((::(X0,gdcnt,X1) | (? [X2] : (! [X3] : ~CONCR(X3,X0,X2) & PRE(X0,X2)) | ~PRE(X0,X1))) & ((! [X2] : (? [X3] : CONCR(X3,X0,X2) | ~PRE(X0,X2)) & PRE(X0,X1)) | ~::(X0,gdcnt,X1))) [nnf transformation 564]
761. ! [X0,X1] : ((::(X0,gdcnt,X1) | ? [X2] : (! [X3] : ~CONCR(X3,X0,X2) & PRE(X0,X2)) | ~PRE(X0,X1)) & ((! [X2] : (? [X3] : CONCR(X3,X0,X2) | ~PRE(X0,X2)) & PRE(X0,X1)) | ~::(X0,gdcnt,X1))) [flattening 760]
762. ! [X0,X1] : ((::(X0,gdcnt,X1) | ? [X2] : (! [X3] : ~CONCR(X3,X0,X2) & PRE(X0,X2)) | ~PRE(X0,X1)) & ((! [X4] : (? [X5] : CONCR(X5,X0,X4) | ~PRE(X0,X4)) & PRE(X0,X1)) | ~::(X0,gdcnt,X1))) [rectify 761]
763. ! [X0] : (? [X2] : (! [X3] : ~CONCR(X3,X0,X2) & PRE(X0,X2)) => (! [X3] : ~CONCR(X3,X0,sK82(X0)) & PRE(X0,sK82(X0)))) [choice axiom]
764. ! [X4,X0] : (? [X5] : CONCR(X5,X0,X4) => CONCR(sK83(X0,X4),X0,X4)) [choice axiom]
765. ! [X0,X1] : ((::(X0,gdcnt,X1) | (! [X3] : ~CONCR(X3,X0,sK82(X0)) & PRE(X0,sK82(X0))) | ~PRE(X0,X1)) & ((! [X4] : (CONCR(sK83(X0,X4),X0,X4) | ~PRE(X0,X4)) & PRE(X0,X1)) | ~::(X0,gdcnt,X1))) [skolemisation 762,764,763]
766. ! [X0,X1] : ((::(X0,cnt,X1) | (~::(X0,gdcnt,X1) & ~::(X0,sdcnt,X1) & ~::(X0,idcnt,X1))) & ((::(X0,gdcnt,X1) | ::(X0,sdcnt,X1) | ::(X0,idcnt,X1)) | ~::(X0,cnt,X1))) [nnf transformation 324]
767. ! [X0,X1] : ((::(X0,cnt,X1) | (~::(X0,gdcnt,X1) & ~::(X0,sdcnt,X1) & ~::(X0,idcnt,X1))) & (::(X0,gdcnt,X1) | ::(X0,sdcnt,X1) | ::(X0,idcnt,X1) | ~::(X0,cnt,X1))) [flattening 766]
768. ! [X2,X1,X0] : (? [X3,X4] : (tP(X3,X4,X2) & DQT(X1,X4) & DQT(X0,X3)) => (tP(sK84(X0,X1,X2),sK85(X0,X1,X2),X2) & DQT(X1,sK85(X0,X1,X2)) & DQT(X0,sK84(X0,X1,X2)))) [choice axiom]
769. ! [X0,X1,X2] : ((((tP(sK84(X0,X1,X2),sK85(X0,X1,X2),X2) & DQT(X1,sK85(X0,X1,X2)) & DQT(X0,sK84(X0,X1,X2))) | P(X0,X1) | tP(X0,X1,X2)) & ::(X1,cnt,X2) & ::(X0,cnt,X2)) | ~cP(X0,X1,X2)) [skolemisation 565,768]
770. ? [X0,X1,X2] : (((~::(X1,sreg,X2) | ~::(X0,sreg,X2)) & (::(X1,sreg,X2) | ::(X0,sreg,X2))) & cP(X0,X1,X2)) [nnf transformation 566]
771. ? [X0,X1,X2] : ((~::(X1,sreg,X2) | ~::(X0,sreg,X2)) & (::(X1,sreg,X2) | ::(X0,sreg,X2)) & cP(X0,X1,X2)) [flattening 770]
772. ? [X0,X1,X2] : ((~::(X1,sreg,X2) | ~::(X0,sreg,X2)) & (::(X1,sreg,X2) | ::(X0,sreg,X2)) & cP(X0,X1,X2)) => ((~::(sK87,sreg,sK88) | ~::(sK86,sreg,sK88)) & (::(sK87,sreg,sK88) | ::(sK86,sreg,sK88)) & cP(sK86,sK87,sK88)) [choice axiom]
773. (~::(sK87,sreg,sK88) | ~::(sK86,sreg,sK88)) & (::(sK87,sreg,sK88) | ::(sK86,sreg,sK88)) & cP(sK86,sK87,sK88) [skolemisation 771,772]
793. ~P(X0,X1) | ~S(X0) | S(X1) [cnf transformation 591]
794. ~P(X0,X1) | ~S(X1) | S(X0) [cnf transformation 591]
803. ~DQT(X0,X1) | PQ(X0) | tQ(X0) | AQ(X0) [cnf transformation 383]
847. ~PRE(X0,X1) | T(X1) [cnf transformation 412]
869. ~tP(X0,X1,X2) | ED(X0) [cnf transformation 432]

```
870. ~tP(X0,X1,X2) | ED(X1) [cnf transformation 432]
989. ~PED(X0) | ED(X0) [cnf transformation 694]
991. ~AQ(X0) | Q(X0) [cnf transformation 696]
992. ~PQ(X0) | Q(X0) [cnf transformation 696]
993. ~tQ(X0) | Q(X0) [cnf transformation 696]
997. ~PR(X0) | R(X0) [cnf transformation 534]
999. ~R(X0) | AB(X0) [cnf transformation 535]
1000. ~F(X0) | PED(X0) [cnf transformation 536]
1003. ~S(X0) | PR(X0) [cnf transformation 537]
1005. ~T(X0) | TR(X0) [cnf transformation 539]
1007. ~ED(X0) | ~AB(X0) [cnf transformation 541]
1009. ~Q(X0) | ~AB(X0) [cnf transformation 543]
1022. ~PR(X0) | ~TR(X0) [cnf transformation 556]
1032. T(X0) | T(X1) | PRE(X1,X0) | ~sP1(X0,X1) [cnf transformation 702]
1034. ~PRE(X1,X0) | sP1(X0,X1) [cnf transformation 702]
1036. sP1(X0,X1) | ~T(X0) | T(X1) | ~AB(X1) [cnf transformation 702]
1037. ~EX(X0,X1) | sP1(X1,X0) [cnf transformation 703]
1038. ~sP1(X1,X0) | EX(X0,X1) [cnf transformation 703]
1074. ~sP2(X0,X1) | ED(X0) [cnf transformation 733]
1084. ~::(X0,mten,X1) | sP2(X0,X1) [cnf transformation 734]
1086. ~sP3(X0,X1) | F(X0) [cnf transformation 741]
1087. ~sP3(X0,X1) | PRE(X0,X1) [cnf transformation 741]
1095. ~::(X0,site,X1) | sP3(X0,X1) [cnf transformation 743]
1096. ~::(X0,cfbnd,X1) | sP3(X0,X1) [cnf transformation 743]
1098. ~::(X0,sreg,X1) | S(X0) [cnf transformation 745]
1099. ~::(X0,sreg,X1) | EX(X0,X1) [cnf transformation 745]
1100. ::(X0,sreg,X1) | ~EX(X0,X1) | ~S(X0) [cnf transformation 745]
1101. ~::(X0,imen,X1) | ::(X0,cfbnd,X1) | ::(X0,site,X1) | ::(X0,sreg,X1) [cnf transformation 747]
1105. ~::(X0,idcnt,X1) | ::(X0,mten,X1) | ::(X0,imen,X1) [cnf transformation 749]
1108. ~::(X0,sdcnt,X1) | Q(X0) [cnf transformation 754]
1119. ~::(X0,gdcnt,X1) | PRE(X0,X1) [cnf transformation 765]
1123. ~::(X0,cnt,X1) | ::(X0,sdcnt,X1) | ::(X0,idcnt,X1) | ::(X0,gdcnt,X1) [cnf transformation 767]
1128. ~cP(X0,X1,X2) | ::(X1,cnt,X2) [cnf transformation 769]
1129. DQT(X0,sK84(X0,X1,X2)) | P(X0,X1) | tP(X0,X1,X2) | ~cP(X0,X1,X2) [cnf transformation 769]
1130. DQT(X1,sK85(X0,X1,X2)) | P(X0,X1) | tP(X0,X1,X2) | ~cP(X0,X1,X2) [cnf transformation 769]
1132. cP(sK86,sK87,sK88) [cnf transformation 773]
1133. ::(sK87,sreg,sK88) | ::(sK86,sreg,sK88) [cnf transformation 773]
1134. ~::(sK87,sreg,sK88) | ~::(sK86,sreg,sK88) [cnf transformation 773]
1141. ~sP1(X0,X1) | T(X1) | T(X0) [subsumption resolution 1032,847]
1150. 1 <=> ::(sK86,sreg,sK88) [avatar definition]
1151. ::(sK86,sreg,sK88) <- (1) [avatar component clause 1150]
1152. ~::(sK86,sreg,sK88) <- (~1) [avatar component clause 1150]
1154. 2 <=> ::(sK87,sreg,sK88) [avatar definition]
1155. ::(sK87,sreg,sK88) <- (2) [avatar component clause 1154]
1156. ~::(sK87,sreg,sK88) <- (~2) [avatar component clause 1154]
1157. ~1 | ~2 [avatar split clause 1134,1154,1150]
1158. 1 | 2 [avatar split clause 1133,1154,1150]
1205. S(sK87) <- (2) [resolution 1098,1155]
1206. PR(sK87) <- (2) [resolution 1205,1003]
1207. ~TR(sK87) <- (2) [resolution 1206,1022]
1275. EX(sK87,sK88) <- (2) [resolution 1099,1155]
1276. sP1(sK88,sK87) <- (2) [resolution 1275,1037]
1279. T(sK87) | T(sK88) <- (2) [resolution 1141,1276]
1281. 3 <=> T(sK88) [avatar definition]
1283. T(sK88) <- (3) [avatar component clause 1281]
1285. 4 <=> T(sK87) [avatar definition]
1287. T(sK87) <- (4) [avatar component clause 1285]
1288. 3 | 4 | ~2 [avatar split clause 1279,1154,1285,1281]
1409. ::(sK87,cnt,sK88) [resolution 1128,1132]
1612. EX(X5,X4) | T(X5) | ~AB(X5) | ~T(X4) [resolution 1036,1038]
1626. ~EX(sK86,sK88) | ~S(sK86) <- (~1) [resolution 1100,1152]
1631. 5 <=> S(sK86) [avatar definition]
1632. S(sK86) <- (5) [avatar component clause 1631]
1633. ~S(sK86) <- (~5) [avatar component clause 1631]
1635. 6 <=> EX(sK86,sK88) [avatar definition]
1637. ~EX(sK86,sK88) <- (~6) [avatar component clause 1635]
1638. ~5 | ~6 | 1 [avatar split clause 1626,1150,1635,1631]
1976. 7 <=> PRE(sK87,sK88) [avatar definition]
1977. ~PRE(sK87,sK88) <- (~7) [avatar component clause 1976]
1978. PRE(sK87,sK88) <- (7) [avatar component clause 1976]
2739. 12 <=> Q(sK86) [avatar definition]
2740. ~Q(sK86) <- (~12) [avatar component clause 2739]
2741. Q(sK86) <- (12) [avatar component clause 2739]
2743. 13 <=> ED(sK86) [avatar definition]
2744. ~ED(sK86) <- (~13) [avatar component clause 2743]
2745. ED(sK86) <- (13) [avatar component clause 2743]
2815. ~cP(X0,X1,X2) | tP(X0,X1,X2) | P(X0,X1) | PQ(X0) | tQ(X0) | AQ(X0) [resolution 1129,803]
2841. ~cP(X0,X1,X2) | tP(X0,X1,X2) | P(X0,X1) | PQ(X1) | tQ(X1) | AQ(X1) [resolution 1130,803]
2876. 23 <=> tQ(sK86) [avatar definition]
2877. ~tQ(sK86) <- (~23) [avatar component clause 2876]
2878. tQ(sK86) <- (23) [avatar component clause 2876]
2880. 24 <=> AQ(sK86) [avatar definition]
2882. AQ(sK86) <- (24) [avatar component clause 2880]
2884. 25 <=> PQ(sK86) [avatar definition]
2886. PQ(sK86) <- (25) [avatar component clause 2884]
3018. 28 <=> T(sK86) [avatar definition]
3019. T(sK86) <- (28) [avatar component clause 3018]
3028. 30 <=> AB(sK86) [avatar definition]
3030. AB(sK86) <- (30) [avatar component clause 3028]
3085. TR(sK87) <- (4) [resolution 1287,1005]
3086. $false <- (2, 4) [subsumption resolution 3085,1207]
```

```
3087. ~2 | ~4 [avatar contradiction clause 3086]
3090. S(sK86) <- (1) [resolution 1151,1098]
3095. 5 | ~1 [avatar split clause 3090,1150,1631]
3097. PR(sK86) <- (5) [resolution 1632,1003]
3098. ~EX(sK87,sK88) | ~S(sK87) <- (~2) [resolution 1156,1100]
3100. 33 <=> S(sK87) [avatar definition]
3101. S(sK87) <- (33) [avatar component clause 3100]
3102. ~S(sK87) <- (~33) [avatar component clause 3100]
3104. 34 <=> EX(sK87,sK88) [avatar definition]
3107. ~33 | ~34 | 2 [avatar split clause 3098,1154,3104,3100]
3108. ::(sK87,sdcnt,sK88) | ::(sK87,idcnt,sK88) | ::(sK87,gdcnt,sK88) [resolution 1409,1123]
3110. 35 <=> ::(sK87,gdcnt,sK88) [avatar definition]
3112. ::(sK87,gdcnt,sK88) <- (35) [avatar component clause 3110]
3114. 36 <=> ::(sK87,idcnt,sK88) [avatar definition]
3116. ::(sK87,idcnt,sK88) <- (36) [avatar component clause 3114]
3118. 37 <=> ::(sK87,sdcnt,sK88) [avatar definition]
3120. ::(sK87,sdcnt,sK88) <- (37) [avatar component clause 3118]
3121. 35 | 36 | 37 [avatar split clause 3108,3118,3114,3110]
3122. ~TR(sK86) <- (5) [resolution 3097,1022]
3123. R(sK86) <- (5) [resolution 3097,997]
3130. AB(sK86) <- (5) [resolution 3123,999]
3131. 30 | ~5 [avatar split clause 3130,1631,3028]
3141. ~AB(sK86) <- (12) [resolution 2741,1009]
3142. $false <- (12, 30) [subsumption resolution 3141,3030]
3143. ~12 | ~30 [avatar contradiction clause 3142]
3153. ~AB(sK86) <- (13) [resolution 2745,1007]
3167. $false <- (13, 30) [subsumption resolution 3153,3030]
3168. ~13 | ~30 [avatar contradiction clause 3167]
3196. Q(sK86) <- (23) [resolution 2878,993]
3197. $false <- (~12, 23) [subsumption resolution 3196,2740]
3198. 12 | ~23 [avatar contradiction clause 3197]
3277. TR(sK86) <- (28) [resolution 3019,1005]
3278. $false <- (5, 28) [subsumption resolution 3277,3122]
3279. ~5 | ~28 [avatar contradiction clause 3278]
3325. 41 <=> Q(sK87) [avatar definition]
3326. ~Q(sK87) <- (~41) [avatar component clause 3325]
3327. Q(sK87) <- (41) [avatar component clause 3325]
3329. 42 <=> ED(sK87) [avatar definition]
3330. ~ED(sK87) <- (~42) [avatar component clause 3329]
3331. ED(sK87) <- (42) [avatar component clause 3329]
3461. ::(sK87,mten,sK88) | ::(sK87,imen,sK88) <- (36) [resolution 3116,1105]
3465. 53 <=> ::(sK87,imen,sK88) [avatar definition]
3467. ::(sK87,imen,sK88) <- (53) [avatar component clause 3465]
3469. 54 <=> ::(sK87,mten,sK88) [avatar definition]
3471. ::(sK87,mten,sK88) <- (54) [avatar component clause 3469]
3472. 53 | 54 | ~36 [avatar split clause 3461,3114,3469,3465]
3537. 57 <=> ::(sK87,site,sK88) [avatar definition]
3539. ::(sK87,site,sK88) <- (57) [avatar component clause 3537]
3541. 58 <=> ::(sK87,cfbnd,sK88) [avatar definition]
3542. ~::(sK87,cfbnd,sK88) <- (~58) [avatar component clause 3541]
3543. ::(sK87,cfbnd,sK88) <- (58) [avatar component clause 3541]
3562. 59 <=> PQ(sK87) [avatar definition]
3563. ~PQ(sK87) <- (~59) [avatar component clause 3562]
3564. PQ(sK87) <- (59) [avatar component clause 3562]
3587. sP3(sK87,sK88) <- (58) [resolution 3543,1096]
3596. PRE(sK87,sK88) <- (58) [resolution 3587,1087]
3602. $false <- (~7, 58) [subsumption resolution 3596,1977]
3603. 7 | ~58 [avatar contradiction clause 3602]
3645. Q(sK87) <- (37) [resolution 3120,1108]
3648. 41 | ~37 [avatar split clause 3645,3118,3325]
3681. 61 <=> tQ(sK87) [avatar definition]
3682. ~tQ(sK87) <- (~61) [avatar component clause 3681]
3683. tQ(sK87) <- (61) [avatar component clause 3681]
3685. 62 <=> AQ(sK87) [avatar definition]
3687. AQ(sK87) <- (62) [avatar component clause 3685]
3698. sP2(sK87,sK88) <- (54) [resolution 3471,1084]
3701. ED(sK87) <- (54) [resolution 3698,1074]
3709. $false <- (~42, 54) [subsumption resolution 3701,3330]
3710. 42 | ~54 [avatar contradiction clause 3709]
3715. PRE(sK87,sK88) <- (35) [resolution 3112,1119]
3716. $false <- (~7, 35) [subsumption resolution 3715,1977]
3717. 7 | ~35 [avatar contradiction clause 3716]
3730. 63 <=> PED(sK87) [avatar definition]
3732. PED(sK87) <- (63) [avatar component clause 3730]
3779. sP1(sK88,sK87) <- (7) [resolution 1978,1034]
3791. EX(sK87,sK88) <- (7) [resolution 3779,1038]
3792. 34 | ~7 [avatar split clause 3791,1976,3104]
3809. ::(sK87,cfbnd,sK88) | ::(sK87,site,sK88) | ::(sK87,sreg,sK88) <- (53) [resolution 3467,1101]
3940. sP3(sK87,sK88) <- (57) [resolution 3539,1095]
3959. F(sK87) <- (57) [resolution 3940,1086]
3969. PED(sK87) <- (57) [resolution 3959,1000]
3972. 63 | ~57 [avatar split clause 3969,3537,3730]
4003. ED(sK87) <- (63) [resolution 3732,989]
9086. tP(sK86,sK87,sK88) | P(sK86,sK87) | PQ(sK86) | tQ(sK86) | AQ(sK86) [resolution 2815,1132]
9087. tP(sK86,sK87,sK88) | P(sK86,sK87) | PQ(sK86) | AQ(sK86) <- (~23) [subsumption resolution 9086,2877]
9089. 205 <=> P(sK86,sK87) [avatar definition]
9090. ~P(sK86,sK87) <- (~205) [avatar component clause 9089]
9091. P(sK86,sK87) <- (205) [avatar component clause 9089]
9093. 206 <=> tP(sK86,sK87,sK88) [avatar definition]
9095. tP(sK86,sK87,sK88) <- (206) [avatar component clause 9093]
9096. 24 | 25 | 205 | 206 | 23 [avatar split clause 9087,2876,9093,9089,2884,2880]
```

```
9098. Q(sK86) <- (25) [resolution 2886,992]
9099. $false <- (~12, 25) [subsumption resolution 9098,2740]
9100. 12 | ~25 [avatar contradiction clause 9099]
9106. tP(sK86,sK87,sK88) | P(sK86,sK87) | PQ(sK87) | tQ(sK87) | AQ(sK87) [resolution 2841,1132]
9116. ~S(sK86) | S(sK87) <- (205) [resolution 9091,793]
9140. S(sK87) <- (5, 205) [subsumption resolution 9116,1632]
9141. $false <- (5, ~33, 205) [subsumption resolution 9140,3102]
9142. ~5 | 33 | ~205 [avatar contradiction clause 9141]
9158. Q(sK86) <- (24) [resolution 2882,991]
9159. $false <- (~12, 24) [subsumption resolution 9158,2740]
9160. 12 | ~24 [avatar contradiction clause 9159]
9235. Q(sK87) <- (62) [resolution 3687,991]
9237. $false <- (~41, 62) [subsumption resolution 9235,3326]
9238. 41 | ~62 [avatar contradiction clause 9237]
9296. ED(sK86) <- (206) [resolution 9095,869]
9325. $false <- (~13, 206) [subsumption resolution 9296,2744]
9326. 13 | ~206 [avatar contradiction clause 9325]
9355. 216 <=> AB(sK87) [avatar definition]
9356. AB(sK87) <- (216) [avatar component clause 9355]
9559. 42 | ~63 [avatar split clause 4003,3730,3329]
9577. ::(sK87,site,sK88) | ::(sK87,sreg,sK88) <- (53, ~58) [subsumption resolution 3809,3542]
9578. 2 | 57 | ~53 | 58 [avatar split clause 9577,3541,3465,3537,1154]
9726. PR(sK87) <- (33) [resolution 3101,1003]
9729. S(sK87) <- (2) [resolution 1155,1098]
9732. R(sK87) <- (33) [resolution 9726,997]
9733. T(sK86) | ~AB(sK86) | ~T(sK88) <- (~6) [resolution 1637,1612]
9742. T(sK86) | ~T(sK88) <- (~6, 30) [subsumption resolution 9733,3030]
9743. T(sK86) <- (3, ~6, 30) [subsumption resolution 9742,1283]
9744. 28 | ~3 | 6 | ~30 [avatar split clause 9743,3028,1635,1281,3018]
9787. AB(sK87) <- (33) [resolution 9732,999]
9788. 216 | ~33 [avatar split clause 9787,3100,9355]
9906. ~S(sK87) | S(sK86) <- (205) [resolution 9091,794]
9923. ~S(sK87) <- (~5, 205) [subsumption resolution 9906,1633]
9924. ~33 | 5 | ~205 [avatar split clause 9923,9089,1631,3100]
9925. 33 | ~2 [avatar split clause 9729,1154,3100]
9969. Q(sK87) <- (59) [resolution 3564,992]
9971. $false <- (~41, 59) [subsumption resolution 9969,3326]
9972. 41 | ~59 [avatar contradiction clause 9971]
9975. Q(sK87) <- (61) [resolution 3683,993]
9976. $false <- (~41, 61) [subsumption resolution 9975,3326]
9977. 41 | ~61 [avatar contradiction clause 9976]
9985. ~AB(sK87) <- (41) [resolution 3327,1009]
9986. $false <- (41, 216) [subsumption resolution 9985,9356]
9987. ~41 | ~216 [avatar contradiction clause 9986]
9988. tP(sK86,sK87,sK88) | PQ(sK87) | tQ(sK87) | AQ(sK87) <- (~205) [subsumption resolution 9106,9090]
9989. tP(sK86,sK87,sK88) | tQ(sK87) | AQ(sK87) <- (~59, ~205) [subsumption resolution 9988,3563]
9990. tP(sK86,sK87,sK88) | AQ(sK87) <- (~59, ~61, ~205) [subsumption resolution 9989,3682]
9991. 62 | 206 | 59 | 61 | 205 [avatar split clause 9990,9089,3681,3562,9093,3685]
10392. ED(sK87) <- (206) [resolution 9095,870]
10442. 42 | ~206 [avatar split clause 10392,9093,3329]
10466. ~AB(sK87) <- (42) [resolution 3331,1007]
10467. $false <- (42, 216) [subsumption resolution 10466,9356]
10468. ~42 | ~216 [avatar contradiction clause 10467]
10469. $false [avatar sat refutation 1157,1158,1288,1638,3087,3095,3107,3121,3131,3143,3168,3198,3279,3472,
3603,3648,3710,3717,3792,3972,9096,9100,9142,9160,9238,9326,9559,9578,
9744,9788,9924,9925,9972,9977,9987,9991,10442,10468]
```

Proof of Theorem (t_{db}46)

```
This went KO with both provers
```

Proof of Theorem (t_{db}47)

```
8. ! [X0] : ((PD(X0) | AB(X0)) => P(X0,X0)) [input]
34. ! [X0,X4] : (PRE(X0,X4) => (T(X4) & (Q(X0) | PD(X0) | ED(X0)))) [input]
97. ! [X0,X3,X4] : (SLC(X0,X3,X4) => (S(X3) & (PD(X0) | PQ(X0) | (~AS(X0) & ED(X0))))) [input]
98. ! [X0,X3,X4] : (SLC(X0,X3,X4) => PRE(X0,X4)) [input]
122. ! [X0] : ((TR(X0) | PR(X0) | AR(X0)) <=> R(X0)) [input]
123. ! [X0] : (R(X0) => AB(X0)) [input]
126. ! [X0] : (S(X0) => PR(X0)) [input]
128. ! [X0] : (T(X0) => TR(X0)) [input]
151. ~? [X0] : (PR(X0) & TR(X0)) [input]
159. ! [X0] : (TM(X0) <=> ::(X0,treg,X0)) [input]
169. ! [X0,X4] : (EX(X0,X4) <=> ((T(X4) & ~T(X0) & AB(X0)) | (P(X4,X0) & T(X0)) | PRE(X0,X4))) [input]
175. ! [X0,X4] : (::(X0,treg,X4) <=> (EX(X0,X4) & T(X0))) [input]
182. ! [X0,X4] : (::(X0,sreg,X4) <=> (EX(X0,X4) & S(X0))) [input]
190. ! [X0,X3,X4] : (SREG(X0,X3,X4) <=> (SLC(X0,X3,X4) & ~S(X0) & ::(X0,idcnt,X4))) [input]
196. ! [X0,X1,X4] : (SREG(X0,X1,X4) => (TM(X4) & ::(X1,sreg,X4) & ~::(X0,sreg,X4) & ::(X0,idcnt,X4))) [input]
197. ~! [X0,X1,X4] : (SREG(X0,X1,X4) => (TM(X4) & ::(X1,sreg,X4) & ~::(X0,sreg,X4) & ::(X0,idcnt,X4))) [negated
     conjecture 196]
207. ! [X0,X1] : (PRE(X0,X1) => (T(X1) & (Q(X0) | PD(X0) | ED(X0)))) [rectify 34]
288. ! [X0,X1,X2] : (SLC(X0,X1,X2) => (S(X1) & (PD(X0) | PQ(X0) | (~AS(X0) & ED(X0))))) [rectify 97]
289. ! [X0,X1,X2] : (SLC(X0,X1,X2) => PRE(X0,X2)) [rectify 98]
313. ! [X0,X1] : (EX(X0,X1) <=> ((T(X1) & ~T(X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1))) [rectify 169]
318. ! [X0,X1] : (::(X0,treg,X1) <=> (EX(X0,X1) & T(X0))) [rectify 175]
327. ! [X0,X1] : (::(X0,sreg,X1) <=> (EX(X0,X1) & S(X0))) [rectify 182]
336. ! [X0,X1,X2] : (SREG(X0,X1,X2) <=> (SLC(X0,X1,X2) & ~S(X0) & ::(X0,idcnt,X2))) [rectify 190]
344. ~! [X0,X1,X2] : (SREG(X0,X1,X2) => (TM(X2) & ::(X1,sreg,X2) & ~::(X0,sreg,X2) & ::(X0,idcnt,X2))) [rectify
     197]
```

```
345. ! [X0,X1,X2] : (SREG(X0,X1,X2) => (SLC(X0,X1,X2) & ~S(X0) & ::(X0,idcnt,X2))) [unused predicate definition
     removal 336]
346. ! [X0] : (::(X0,treg,X0) => TM(X0)) [unused predicate definition removal 159]
347. ! [X0] : ((TR(X0) | PR(X0) | AR(X0)) => R(X0)) [unused predicate definition removal 122]
385. ! [X0] : (P(X0,X0) | (~PD(X0) & ~AB(X0))) [ennf transformation 8]
422. ! [X0,X1] : ((T(X1) & (Q(X0) | PD(X0) | ED(X0))) | ~PRE(X0,X1)) [ennf transformation 207]
515. ! [X0,X1,X2] : ((S(X1) & (PD(X0) | PQ(X0) | (~AS(X0) & ED(X0)))) | ~SLC(X0,X1,X2)) [ennf transformation 288]
516. ! [X0,X1,X2] : (PRE(X0,X2) | ~SLC(X0,X1,X2)) [ennf transformation 289]
543. ! [X0] : (R(X0) | (~TR(X0) & ~PR(X0) & ~AR(X0))) [ennf transformation 347]
544. ! [X0] : (AB(X0) | ~R(X0)) [ennf transformation 123]
546. ! [X0] : (PR(X0) | ~S(X0)) [ennf transformation 126]
548. ! [X0] : (TR(X0) | ~T(X0)) [ennf transformation 128]
565. ! [X0] : (~PR(X0) | ~TR(X0)) [ennf transformation 151]
571. ! [X0] : (TM(X0) | ~::(X0,treg,X0)) [ennf transformation 346]
578. ! [X0,X1,X2] : ((SLC(X0,X1,X2) & ~S(X0) & ::(X0,idcnt,X2)) | ~SREG(X0,X1,X2)) [ennf transformation 345]
579. ? [X0,X1,X2] : ((~TM(X2) | ~::(X1,sreg,X2) | ::(X0,sreg,X2) | ~::(X0,idcnt,X2)) & SREG(X0,X1,X2)) [ennf
     transformation 344]
582. ! [X1,X0] : (sP1(X1,X0) <=> ((T(X1) & ~T(X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1))) [predicate
     definition introduction]
583. ! [X0,X1] : (EX(X0,X1) <=> sP1(X1,X0)) [definition folding 313,582]
714. ! [X1,X0] : ((sP1(X1,X0) | ((~T(X1) | T(X0) | ~AB(X0)) & (~P(X1,X0) | ~T(X0)) & ~PRE(X0,X1))) & (((T(X1) & ~T
     (X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1)) | ~sP1(X1,X0))) [nnf transformation 582]
715. ! [X1,X0] : ((sP1(X1,X0) | ((~T(X1) | T(X0) | ~AB(X0)) & (~P(X1,X0) | ~T(X0)) & ~PRE(X0,X1))) & ((T(X1) & ~T(
     X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1) | ~sP1(X1,X0))) [flattening 714]
716. ! [X0,X1] : ((sP1(X0,X1) | ((~T(X0) | T(X1) | ~AB(X1)) & (~P(X0,X1) | ~T(X1)) & ~PRE(X1,X0))) & ((T(X0) & ~T(
     X1) & AB(X1)) | (P(X0,X1) & T(X1)) | PRE(X1,X0) | ~sP1(X0,X1))) [rectify 715]
717. ! [X0,X1] : ((EX(X0,X1) | ~sP1(X1,X0)) & (sP1(X1,X0) | ~EX(X0,X1))) [nnf transformation 583]
733. ! [X0,X1] : ((::(X0,treg,X1) | (~EX(X0,X1) | ~T(X0))) & ((EX(X0,X1) & T(X0)) | ~::(X0,treg,X1))) [nnf
     transformation 318]
734. ! [X0,X1] : ((::(X0,treg,X1) | ~EX(X0,X1) | ~T(X0)) & ((EX(X0,X1) & T(X0)) | ~::(X0,treg,X1))) [flattening
     733]
758. ! [X0,X1] : ((::(X0,sreg,X1) | (~EX(X0,X1) | ~S(X0))) & ((EX(X0,X1) & S(X0)) | ~::(X0,sreg,X1))) [nnf
     transformation 327]
759. ! [X0,X1] : ((::(X0,sreg,X1) | ~EX(X0,X1) | ~S(X0)) & ((EX(X0,X1) & S(X0)) | ~::(X0,sreg,X1))) [flattening
     758]
782. ? [X0,X1,X2] : ((~TM(X2) | ~::(X1,sreg,X2) | ::(X0,sreg,X2) | ~::(X0,idcnt,X2)) & SREG(X0,X1,X2)) => ((~TM(
     sK87) | ~::(sK86,sreg,sK87) | ::(sK85,sreg,sK87) | ~::(sK85,idcnt,sK87)) & SREG(sK85,sK86,sK87)) [choice
     axiom]
783. (~TM(sK87) | ~::(sK86,sreg,sK87) | ::(sK85,sreg,sK87) | ~::(sK85,idcnt,sK87)) & SREG(sK85,sK86,sK87) [
     skolemisation 579,782]
805. P(X0,X0) | ~AB(X0) [cnf transformation 385]
857. ~PRE(X0,X1) | T(X1) [cnf transformation 422]
968. ~SLC(X0,X1,X2) | S(X1) [cnf transformation 515]
969. ~SLC(X0,X1,X2) | PRE(X0,X2) [cnf transformation 516]
1004. ~PR(X0) | R(X0) [cnf transformation 543]
1005. ~TR(X0) | R(X0) [cnf transformation 543]
1006. ~R(X0) | AB(X0) [cnf transformation 544]
1010. ~S(X0) | PR(X0) [cnf transformation 546]
1012. ~T(X0) | TR(X0) [cnf transformation 548]
1029. ~PR(X0) | ~TR(X0) [cnf transformation 565]
1037. ~::(X0,treg,X0) | TM(X0) [cnf transformation 571]
1048. ~P(X0,X1) | sP1(X0,X1) | ~T(X1) [cnf transformation 716]
1049. sP1(X0,X1) | ~T(X0) | T(X1) | ~AB(X1) [cnf transformation 716]
1051. ~sP1(X1,X0) | EX(X0,X1) [cnf transformation 717]
1076. ::(X0,treg,X1) | ~EX(X0,X1) | ~T(X0) [cnf transformation 734]
1111. ~::(X0,sreg,X1) | S(X0) [cnf transformation 759]
1113. ::(X0,sreg,X1) | ~EX(X0,X1) | ~S(X0) [cnf transformation 759]
1140. ~SREG(X0,X1,X2) | ::(X0,idcnt,X2) [cnf transformation 578]
1141. ~SREG(X0,X1,X2) | ~S(X0) [cnf transformation 578]
1142. ~SREG(X0,X1,X2) | SLC(X0,X1,X2) [cnf transformation 578]
1143. SREG(sK85,sK86,sK87) [cnf transformation 783]
1144. ~TM(sK87) | ~::(sK86,sreg,sK87) | ::(sK85,sreg,sK87) | ~::(sK85,idcnt,sK87) [cnf transformation 783]
1160. 1 <=> ::(sK85,idcnt,sK87) [avatar definition]
1164. 2 <=> ::(sK85,sreg,sK87) [avatar definition]
1166. ::(sK85,sreg,sK87) <= (2) [avatar component clause 1164]
1168. 3 <=> ::(sK86,sreg,sK87) [avatar definition]
1170. ~::(sK86,sreg,sK87) <= (~3) [avatar component clause 1168]
1172. 4 <=> TM(sK87) [avatar definition]
1175. ~1 | 2 | ~3 | ~4 [avatar split clause 1144,1172,1168,1164,1160]
1225. ~S(sK85) [resolution 1141,1143]
1388. sP1(X1,X1) | ~T(X1) | ~AB(X1) [resolution 1048,805]
1412. ::(sK85,idcnt,sK87) [resolution 1140,1143]
1415. 1 [avatar split clause 1412,1160]
1417. SLC(sK85,sK86,sK87) [resolution 1142,1143]
1418. PRE(sK85,sK87) [resolution 1417,969]
1419. S(sK86) [resolution 1417,968]
1421. PR(sK86) [resolution 1419,1010]
1428. ~TR(sK86) [resolution 1421,1029]
1429. R(sK86) [resolution 1421,1004]
1440. AB(sK86) [resolution 1429,1006]
1456. T(sK87) [resolution 1418,857]
1460. TR(sK87) [resolution 1456,1012]
1464. R(sK87) [resolution 1460,1005]
1468. AB(sK87) [resolution 1464,1006]
1626. EX(X5,X4) | T(X5) | ~AB(X5) | ~T(X4) [resolution 1049,1051]
1634. ~EX(X0,X0) | ~T(X0) | TM(X0) [resolution 1076,1037]
1657. ~EX(sK86,sK87) | ~S(sK86) <= (~3) [resolution 1113,1170]
1658. ~EX(sK86,sK87) <= (~3) [subsumption resolution 1657,1419]
1730. EX(X2,X2) | ~AB(X2) | ~T(X2) [resolution 1388,1051]
1938. ~AB(X1) | ~T(X1) | ~T(X1) | TM(X1) [resolution 1730,1634]
1939. ~T(X1) | ~AB(X1) | TM(X1) [duplicate literal removal 1938]
```

```
1951. ~AB(sK87) | TM(sK87) [resolution 1939,1456]
1961. TM(sK87) [subsumption resolution 1951,1468]
1962. 4 [avatar split clause 1961,1172]
2932. S(sK85) <- (2) [resolution 1166,1111]
2933. $false <- (2) [subsumption resolution 2932,1225]
2934. ~2 [avatar contradiction clause 2933]
3351. T(sK86) | ~AB(sK86) | ~T(sK87) <- (~3) [resolution 1626,1658]
3354. T(sK86) | ~T(sK87) <- (~3) [subsumption resolution 3351,1440]
3355. T(sK86) <- (~3) [subsumption resolution 3354,1456]
3358. TR(sK86) <- (~3) [resolution 3355,1012]
3360. $false <- (~3) [subsumption resolution 3358,1428]
3361. 3 [avatar contradiction clause 3360]
3362. $false [avatar sat refutation 1175,1415,1962,2934,3361]
```

Proof of Theorem (t_{db}48)

```
5. ! [X0,X1] : (P(X0,X1) => ((PD(X1) & PD(X0)) | (AB(X1) & AB(X0)))) [input]
8. ! [X0] : ((PD(X0) | AB(X0)) => P(X0,X0)) [input]
26. ! [X0,X4] : (PRE(X0,X4) <=> ? [X5] : (P(X4,X5) & TLC(X0,X5))) [input]
34. ! [X0,X4] : (PRE(X0,X4) => (T(X4) & (Q(X0) | PD(X0) | ED(X0)))) [input]
98. ! [X0,X3,X4] : (SLC(X0,X3,X4) => PRE(X0,X4)) [input]
159. ! [X0] : (TM(X0) <=> ::(X0,treg,X0)) [input]
169. ! [X0,X4] : (EX(X0,X4) <=> ((T(X4) & ~T(X0) & AB(X0)) | (P(X4,X0) & T(X0)) | PRE(X0,X4))) [input]
175. ! [X0,X4] : (::(X0,treg,X4) <=> (EX(X0,X4) & T(X0))) [input]
182. ! [X0,X4] : (::(X0,sreg,X4) <=> (EX(X0,X4) & S(X0))) [input]
193. ! [X0,X1,X4] : (LOC(X0,X1,X4) <=> (? [X3] : ? [X8] : (P(X3,X8) & SLC(X1,X8,X4) & SLC(X0,X3,X4)) & ~S(X1) &
     ::(X1,idcnt,X4) & ~S(X0) & ::(X0,idcnt,X4))) [input]
196. ! [X0,X1,X4] : (LOC(X0,X1,X4) => (TM(X4) & ~::(X1,sreg,X4) & ::(X1,idcnt,X4) & ~::(X0,sreg,X4) & ::(X0,idcnt,
     X4))) [input]
197. ~! [X0,X1,X4] : (LOC(X0,X1,X4) => (TM(X4) & ~::(X1,sreg,X4) & ::(X1,idcnt,X4) & ~::(X0,sreg,X4) & ::(X0,idcnt
     ,X4))) [negated conjecture 196]
199. ! [X0,X1] : (PRE(X0,X1) <=> ? [X2] : (P(X1,X2) & TLC(X0,X2))) [rectify 26]
207. ! [X0,X1] : (PRE(X0,X1) => (T(X1) & (Q(X0) | PD(X0) | ED(X0)))) [rectify 34]
289. ! [X0,X1,X2] : (SLC(X0,X1,X2) => PRE(X0,X2)) [rectify 98]
313. ! [X0,X1] : (EX(X0,X1) <=> ((T(X1) & ~T(X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1))) [rectify 169]
318. ! [X0,X1] : (::(X0,treg,X1) <=> (EX(X0,X1) & T(X0))) [rectify 175]
327. ! [X0,X1] : (::(X0,sreg,X1) <=> (EX(X0,X1) & S(X0))) [rectify 182]
340. ! [X0,X1,X2] : (LOC(X0,X1,X2) <=> (? [X3] : ? [X4] : (P(X3,X4) & SLC(X1,X4,X2) & SLC(X0,X3,X2)) & ~S(X1) &
     ::(X1,idcnt,X2) & ~S(X0) & ::(X0,idcnt,X2))) [rectify 193]
341. ! [X0,X1,X2] : (LOC(X0,X1,X2) <=> (? [X3,X4] : (P(X3,X4) & SLC(X1,X4,X2) & SLC(X0,X3,X2)) & ~S(X1) & ::(X1,
     idcnt,X2) & ~S(X0) & ::(X0,idcnt,X2))) [flattening 340]
344. ~! [X0,X1,X2] : (LOC(X0,X1,X2) => (TM(X2) & ~::(X1,sreg,X2) & ::(X1,idcnt,X2) & ~::(X0,sreg,X2) & ::(X0,idcnt
     ,X2))) [rectify 197]
345. ! [X0,X1,X2] : (LOC(X0,X1,X2) => (? [X3,X4] : (P(X3,X4) & SLC(X1,X4,X2) & SLC(X0,X3,X2)) & ~S(X1) & ::(X1,
     idcnt,X2) & ~S(X0) & ::(X0,idcnt,X2))) [unused predicate definition removal 341]
346. ! [X0] : (::(X0,treg,X0) => TM(X0)) [unused predicate definition removal 159]
381. ! [X0,X1] : (((PD(X1) & PD(X0)) | (AB(X1) & AB(X0))) | ~P(X0,X1)) [ennf transformation 5]
382. ! [X0,X1] : ((PD(X1) & PD(X0)) | (AB(X1) & AB(X0)) | ~P(X0,X1)) [flattening 381]
385. ! [X0] : (P(X0,X0) | (~PD(X0) & ~AB(X0))) [ennf transformation 8]
422. ! [X0,X1] : ((T(X1) & (Q(X0) | PD(X0) | ED(X0))) | ~PRE(X0,X1)) [ennf transformation 207]
516. ! [X0,X1,X2] : (PRE(X0,X2) | ~SLC(X0,X1,X2)) [ennf transformation 289]
571. ! [X0] : (TM(X0) | ~::(X0,treg,X0)) [ennf transformation 346]
578. ! [X0,X1,X2] : ((? [X3,X4] : (P(X3,X4) & SLC(X1,X4,X2) & SLC(X0,X3,X2)) & ~S(X1) & ::(X1,idcnt,X2) & ~S(X0) &
     ::(X0,idcnt,X2)) | ~LOC(X0,X1,X2)) [ennf transformation 345]
579. ? [X0,X1,X2] : ((~TM(X2) | ::(X1,sreg,X2) | ~::(X1,idcnt,X2) | ::(X0,sreg,X2) | ~::(X0,idcnt,X2)) & LOC(X0,X1
     ,X2)) [ennf transformation 344]
582. ! [X1,X0] : (sP1(X1,X0) <=> ((T(X1) & ~T(X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1))) [predicate
     definition introduction]
583. ! [X0,X1] : (EX(X0,X1) <=> sP1(X1,X0)) [definition folding 313,582]
621. ! [X0,X1] : ((PRE(X0,X1) | ! [X2] : (~P(X1,X2) | ~TLC(X0,X2))) & (? [X2] : (P(X1,X2) & TLC(X0,X2)) | ~PRE(X0,
     X1))) [nnf transformation 199]
622. ! [X0,X1] : ((PRE(X0,X1) | ! [X2] : (~P(X1,X2) | ~TLC(X0,X2))) & (? [X3] : (P(X1,X3) & TLC(X0,X3)) | ~PRE(X0,
     X1))) [rectify 621]
623. ! [X1,X0] : (? [X3] : (P(X1,X3) & TLC(X0,X3)) => (P(X1,sK13(X0,X1)) & TLC(X0,sK13(X0,X1)))) [choice axiom]
624. ! [X0,X1] : ((PRE(X0,X1) | ! [X2] : (~P(X1,X2) | ~TLC(X0,X2))) & ((P(X1,sK13(X0,X1)) & TLC(X0,sK13(X0,X1))) |
     ~PRE(X0,X1))) [skolemisation 622,623]
714. ! [X1,X0] : ((sP1(X1,X0) | ((~T(X1) | T(X0) | ~AB(X0)) & (~P(X1,X0) | ~T(X0)) & ~PRE(X0,X1))) & (((T(X1) & ~T
     (X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1)) | ~sP1(X1,X0))) [nnf transformation 582]
715. ! [X1,X0] : ((sP1(X1,X0) | ((~T(X1) | T(X0) | ~AB(X0)) & (~P(X1,X0) | ~T(X0)) & ~PRE(X0,X1))) & ((T(X1) & ~T(
     X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1) | ~sP1(X1,X0))) [flattening 714]
716. ! [X0,X1] : ((sP1(X0,X1) | ((~T(X0) | T(X1) | ~AB(X1)) & (~P(X0,X1) | ~T(X1)) & ~PRE(X1,X0))) & ((T(X0) & ~T(
     X1) & AB(X1)) | (P(X0,X1) & T(X1)) | PRE(X1,X0) | ~sP1(X0,X1))) [rectify 715]
717. ! [X0,X1] : ((EX(X0,X1) | ~sP1(X1,X0)) & (sP1(X1,X0) | ~EX(X0,X1))) [nnf transformation 583]
733. ! [X0,X1] : ((::(X0,treg,X1) | (~EX(X0,X1) | ~T(X0))) & ((EX(X0,X1) & T(X0)) | ~::(X0,treg,X1))) [nnf
     transformation 318]
734. ! [X0,X1] : ((::(X0,treg,X1) | ~EX(X0,X1) | ~T(X0)) & ((EX(X0,X1) & T(X0)) | ~::(X0,treg,X1))) [flattening
     733]
758. ! [X0,X1] : ((::(X0,sreg,X1) | (~EX(X0,X1) | ~S(X0))) & ((EX(X0,X1) & S(X0)) | ~::(X0,sreg,X1))) [nnf
     transformation 327]
759. ! [X0,X1] : ((::(X0,sreg,X1) | ~EX(X0,X1) | ~S(X0)) & ((EX(X0,X1) & S(X0)) | ~::(X0,sreg,X1))) [flattening
     758]
782. ! [X2,X1,X0] : (? [X3,X4] : (P(X3,X4) & SLC(X1,X4,X2) & SLC(X0,X3,X2)) => (P(sK85(X0,X1,X2),sK86(X0,X1,X2)) &
     SLC(X1,sK86(X0,X1,X2),X2) & SLC(X0,sK85(X0,X1,X2),X2))) [choice axiom]
783. ! [X0,X1,X2] : (((P(sK85(X0,X1,X2),sK86(X0,X1,X2)) & SLC(X1,sK86(X0,X1,X2),X2) & SLC(X0,sK85(X0,X1,X2),X2)) &
     ~S(X1) & ::(X1,idcnt,X2) & ~S(X0) & ::(X0,idcnt,X2)) | ~LOC(X0,X1,X2)) [skolemisation 578,782]
784. ? [X0,X1,X2] : ((~TM(X2) | ::(X1,sreg,X2) | ~::(X1,idcnt,X2) | ::(X0,sreg,X2) | ~::(X0,idcnt,X2)) & LOC(X0,X1
     ,X2)) => ((~TM(sK89) | ::(sK88,sreg,sK89) | ~::(sK88,idcnt,sK89) | ::(sK87,sreg,sK89) | ~::(sK87,idcnt,sK89)
     ) & LOC(sK87,sK88,sK89)) [choice axiom]
785. (~TM(sK89) | ::(sK88,sreg,sK89) | ~::(sK88,idcnt,sK89) | ::(sK87,sreg,sK89) | ~::(sK87,idcnt,sK89)) & LOC(
     sK87,sK88,sK89) [skolemisation 579,784]
```

```
799. ~P(X0,X1) | AB(X0) | PD(X0) [cnf transformation 382]
807. P(X0,X0) | ~AB(X0) [cnf transformation 385]
808. P(X0,X0) | ~PD(X0) [cnf transformation 385]
843. P(X1,sK13(X0,X1)) | ~PRE(X0,X1) [cnf transformation 624]
859. ~PRE(X0,X1) | T(X1) [cnf transformation 422]
971. ~SLC(X0,X1,X2) | PRE(X0,X2) [cnf transformation 516]
1039. ~::(X0,treg,X0) | TM(X0) [cnf transformation 571]
1050. ~P(X0,X1) | sP1(X0,X1) | ~T(X1) [cnf transformation 716]
1053. ~sP1(X1,X0) | EX(X0,X1) [cnf transformation 717]
1078. ::(X0,treg,X1) | ~EX(X0,X1) | ~T(X0) [cnf transformation 734]
1113. ~::(X0,sreg,X1) | S(X0) [cnf transformation 759]
1142. ~LOC(X0,X1,X2) | ::(X0,idcnt,X2) [cnf transformation 783]
1143. ~LOC(X0,X1,X2) | ~S(X0) [cnf transformation 783]
1144. ~LOC(X0,X1,X2) | ::(X1,idcnt,X2) [cnf transformation 783]
1145. ~LOC(X0,X1,X2) | ~S(X1) [cnf transformation 783]
1146. SLC(X0,sK85(X0,X1,X2),X2) | ~LOC(X0,X1,X2) [cnf transformation 783]
1149. LOC(sK87,sK88,sK89) [cnf transformation 785]
1150. ~TM(sK89) | ::(sK88,sreg,sK89) | ~::(sK88,idcnt,sK89) | ::(sK87,sreg,sK89) | ~::(sK87,idcnt,sK89) [cnf
      transformation 785]
1166. 1 <=> ::(sK87,idcnt,sK89) [avatar definition]
1170. 2 <=> ::(sK87,sreg,sK89) [avatar definition]
1172. ::(sK87,sreg,sK89) <- (2) [avatar component clause 1170]
1174. 3 <=> ::(sK88,idcnt,sK89) [avatar definition]
1178. 4 <=> ::(sK88,sreg,sK89) [avatar definition]
1180. ::(sK88,sreg,sK89) <- (4) [avatar component clause 1178]
1182. 5 <=> TM(sK89) [avatar definition]
1185. ~1 | 2 | ~3 | 4 | ~5 [avatar split clause 1150,1182,1178,1174,1170,1166]
1235. ~S(sK87) [resolution 1143,1149]
1236. ~S(sK88) [resolution 1145,1149]
1329. ~PRE(X2,X3) | AB(X3) | PD(X3) [resolution 843,799]
1398. sP1(X0,X0) | ~T(X0) | ~PD(X0) [resolution 1050,808]
1399. sP1(X1,X1) | ~T(X1) | ~AB(X1) [resolution 1050,807]
1423. ::(sK87,idcnt,sK89) [resolution 1142,1149]
1426. 1 [avatar split clause 1423,1166]
1428. ::(sK88,idcnt,sK89) [resolution 1144,1149]
1431. 3 [avatar split clause 1428,1174]
1535. EX(X2,X2) | ~PD(X2) | ~T(X2) [resolution 1398,1053]
1548. EX(X2,X2) | ~AB(X2) | ~T(X2) [resolution 1399,1053]
1643. ~EX(X0,X0) | ~T(X0) | TM(X0) [resolution 1078,1039]
1649. ~T(X0) | TM(X0) | ~AB(X0) | ~T(X0) [resolution 1643,1548]
1650. ~T(X1) | TM(X1) | ~PD(X1) | ~T(X1) [resolution 1643,1535]
1651. ~T(X1) | TM(X1) | ~PD(X1) [duplicate literal removal 1650]
1652. ~T(X0) | TM(X0) | ~AB(X0) [duplicate literal removal 1649]
2044. ~LOC(X10,X11,X12) | PRE(X10,X12) [resolution 1146,971]
2046. PRE(sK87,sK89) [resolution 2044,1149]
2053. T(sK89) [resolution 2046,859]
2058. AB(sK89) | PD(sK89) [resolution 2046,1329]
2073. 9 <=> PD(sK89) [avatar definition]
2077. 10 <=> AB(sK89) [avatar definition]
2080. 9 | 10 [avatar split clause 2058,2077,2073]
2085. TM(sK89) | ~AB(sK89) [resolution 2053,1652]
2086. TM(sK89) | ~PD(sK89) [resolution 2053,1651]
2093. ~9 | 5 [avatar split clause 2086,1182,2073]
2094. ~10 | 5 [avatar split clause 2085,1182,2077]
2114. S(sK88) <- (4) [resolution 1180,1113]
2115. $false <- (4) [subsumption resolution 2114,1236]
2116. ~4 [avatar contradiction clause 2115]
2119. S(sK87) <- (2) [resolution 1172,1113]
2120. $false <- (2) [subsumption resolution 2119,1235]
2121. ~2 [avatar contradiction clause 2120]
2122. $false [avatar sat refutation 1185,1426,1431,2080,2093,2094,2116,2121]
```

Proof of Theorem (t_{db}49)

```
101. ! [X0,X3,X4,X8] : ((SLC(X0,X8,X4) & SLC(X0,X3,X4)) => X3 = X8) [input]
183. ! [X0,X3,X4] : (SREG(X0,X3,X4) <=> (SLC(X0,X3,X4) & ~S(X0) & ::(X0,idcnt,X4))) [input]
189. ! [X0,X1,X4,X2] : ((SREG(X0,X2,X4) & SREG(X0,X1,X4)) => X1 = X2) [input]
190. ~! [X0,X1,X4,X2] : ((SREG(X0,X2,X4) & SREG(X0,X1,X4)) => X1 = X2) [negated conjecture 189]
285. ! [X0,X1,X2,X3] : ((SLC(X0,X3,X2) & SLC(X0,X1,X2)) => X1 = X3) [rectify 101]
327. ! [X0,X1,X2] : (SREG(X0,X1,X2) <=> (SLC(X0,X1,X2) & ~S(X0) & ::(X0,idcnt,X2))) [rectify 183]
335. ~! [X0,X1,X2,X3] : ((SREG(X0,X3,X2) & SREG(X0,X1,X2)) => X1 = X3) [rectify 190]
336. ! [X0,X1,X2] : (SREG(X0,X1,X2) => (SLC(X0,X1,X2) & ~S(X0) & ::(X0,idcnt,X2))) [unused predicate definition
      removal 327]
508. ! [X0,X1,X2,X3] : (X1 = X3 | (~SLC(X0,X3,X2) | ~SLC(X0,X1,X2))) [ennf transformation 285]
509. ! [X0,X1,X2,X3] : (X1 = X3 | ~SLC(X0,X3,X2) | ~SLC(X0,X1,X2)) [flattening 508]
565. ! [X0,X1,X2] : ((SLC(X0,X1,X2) & ~S(X0) & ::(X0,idcnt,X2)) | ~SREG(X0,X1,X2)) [ennf transformation 336]
566. ? [X0,X1,X2,X3] : (X1 != X3 & (SREG(X0,X3,X2) & SREG(X0,X1,X2))) [ennf transformation 335]
567. ? [X0,X1,X2,X3] : (X1 != X3 & SREG(X0,X3,X2) & SREG(X0,X1,X2)) [flattening 566]
769. ? [X0,X1,X2,X3] : (X1 != X3 & SREG(X0,X3,X2) & SREG(X0,X1,X2)) => (sK85 != sK87 & SREG(sK84,sK87,sK86) & SREG
      (sK84,sK85,sK86)) [choice axiom]
770. sK85 != sK87 & SREG(sK84,sK87,sK86) & SREG(sK84,sK85,sK86) [skolemisation 567,769]
961. ~SLC(X0,X3,X2) | X1 = X3 | ~SLC(X0,X1,X2) [cnf transformation 509]
1126. ~SREG(X0,X1,X2) | SLC(X0,X1,X2) [cnf transformation 565]
1127. SREG(sK84,sK85,sK86) [cnf transformation 770]
1128. SREG(sK84,sK87,sK86) [cnf transformation 770]
1129. sK85 != sK87 [cnf transformation 770]
1374. SLC(sK84,sK85,sK86) [resolution 1126,1127]
1375. SLC(sK84,sK87,sK86) [resolution 1126,1128]
1920. ~SLC(sK84,X0,sK86) | sK85 = X0 [resolution 961,1374]
1924. sK85 = sK87 [resolution 1920,1375]
```

```
1926. $false [subsumption resolution 1924,1129]
```

Proof of Theorem (t_{db}52)

```
1 P(A,B) -> AB(A) & AB(B) | PD(A) & PD(B). [assumption].
2 P(A,B) -> (T(A) <-> T(B)). [assumption].
3 AB(A) | PD(A) -> P(A,A). [assumption].
4 PRE(A,B) -> (ED(A) | PD(A) | Q(A)) & T(B). [assumption].
5 SLC(A,B,C) -> (ED(A) & -AS(A) | PQ(A) | PD(A)) & S(B). [assumption].
6 SLC(A,B,C) -> PRE(A,C). [assumption].
7 SLC(A,B,C) & SLC(A,D,C) -> B = D. [assumption].
8 SLC(A,B,C) & SLC(A,D,E) & P(C,E) -> P(B,D). [assumption].
9 AR(A) | PR(A) | TR(A) <-> R(A). [assumption].
10 R(A) -> AB(A). [assumption].
11 S(A) -> PR(A). [assumption].
12 T(A) -> TR(A). [assumption].
13 -(exists A (AB(A) & PD(A))). [assumption].
14 -(exists A (TR(A) & PR(A))). [assumption].
15 EX(A,B) <-> PRE(A,B) | T(A) & P(B,A) | AB(A) & -T(A) & T(B). [assumption].
16 oP(A,B) <-> P(A,B) & (PD(A) & PD(B) | T(A) & T(B)). [assumption].
17 tmP(A,B) <-> oP(A,B) & (all C (oP(C,B) & (all D (EX(C,D) -> EX(A,D))) -> oP(C,A))). [assumption].
18 ::(A,sreg,B) <-> S(A) & EX(A,B). [assumption].
19 ::(A,imen,B) <-> ::(A,site,B) | ::(A,cfbnd,B) | ::(A,sreg,B). [assumption].
20 ::(A,idcnt,B) <-> ::(A,mten,B) | ::(A,imen,B). [assumption].
21 ::(A,cnt,B) <-> ::(A,idcnt,B) | ::(A,sdcnt,B) | ::(A,gdcnt,B). [assumption].
22 cP(A,B,C) <-> ::(A,cnt,C) & ::(B,cnt,C) & (tP(A,B,C) | P(A,B) | (exists D exists E (DQT(A,D) & DQT(B,E) & tP(D,
       E,C)))). [assumption].
23 SREG(A,B,C) <-> ::(A,idcnt,C) & -S(A) & SLC(A,B,C). [assumption].
24 LOC(A,B,C) <-> ::(A,idcnt,C) & -S(A) & ::(B,idcnt,C) & -S(B) & (exists D exists E (SLC(A,D,C) & SLC(B,E,C) & P(
       D,E))). [assumption].
25 LOC(A,B,C) -> (exists D exists E exists F (tmP(D,C) & SREG(A,E,D) & SREG(B,F,D) & cP(E,F,D))). [goal].
26 -AB(A) | P(A,A). [clausify(3)].
27 -P(A,B) | AB(B) | PD(A). [clausify(1)].
28 -R(A) | AB(A). [clausify(10)].
29 -AB(A) | -PD(A). [clausify(13)].
30 EX(A,B) | -AB(A) | T(A) | -T(B). [clausify(15)].
31 -TR(A) | R(A). [clausify(9)].
32 -T(A) | TR(A). [clausify(12)].
33 -TR(A) | -PR(A). [clausify(14)].
34 -PR(A) | R(A). [clausify(9)].
35 -S(A) | PR(A). [clausify(11)].
36 -PR(A) | -T(A). [resolve(33,a,32,b)].
37 cP(A,B,C) | -::(A,cnt,C) | -::(B,cnt,C) | -P(A,B). [clausify(22)].
38 -tmP(A,c1) | -SREG(c2,B,A) | -SREG(c3,C,A) | -cP(B,C,A). [deny(25)].
39 LOC(A,B,C) | -::(A,idcnt,C) | S(A) | -::(B,idcnt,C) | S(B) | -SLC(A,D,C) | -SLC(B,E,C) | -P(D,E). [clausify
       (24)].
40 -LOC(A,B,C) | ::(A,idcnt,C). [clausify(24)].
41 -LOC(A,B,C) | -S(A). [clausify(24)].
42 -LOC(A,B,C) | ::(B,idcnt,C). [clausify(24)].
43 -LOC(A,B,C) | -S(B). [clausify(24)].
44 -LOC(A,B,C) | SLC(A,f101(A,C,B),C). [clausify(24)].
45 -LOC(A,B,C) | SLC(B,f102(A,C,B),C). [clausify(24)].
46 -LOC(A,B,C) | P(f101(A,C,B),f102(A,C,B)). [clausify(24)].
47 LOC(c2,c3,c1). [deny(25)].
48 -P(A,B) | T(A) | -T(B). [clausify(2)].
49 -PRE(A,B) | T(B). [clausify(4)].
50 -SLC(A,B,C) | S(B). [clausify(5)].
51 -SLC(A,B,C) | PRE(A,C). [clausify(6)].
52 -SLC(A,B,C) | -SLC(A,D,C) | D = B. [clausify(7)].
53 -SLC(A,B,C) | -SLC(A,D,E) | -P(C,E) | P(B,D). [clausify(8)].
54 oP(A,B) | -P(A,B) | -T(A) | -T(B). [clausify(16)].
55 tmP(A,B) | -oP(A,B) | oP(f79(B,A),B). [clausify(17)].
56 tmP(A,B) | -oP(A,B) | -oP(f79(B,A),A). [clausify(17)].
57 ::(A,sreg,B) | -S(A) | -EX(A,B). [clausify(18)].
58 ::(A,imen,B) | -::(A,sreg,B). [clausify(19)].
59 ::(A,idcnt,B) | -::(A,imen,B). [clausify(20)].
60 ::(A,cnt,B) | -::(A,idcnt,B). [clausify(21)].
61 SREG(A,B,C) | -::(A,idcnt,C) | S(A) | -SLC(A,B,C). [clausify(23)].
62 -R(A) | P(A,A). [resolve(28,b,26,a)].
63 -PD(A) | -R(A). [resolve(29,a,28,b)].
64 EX(A,B) | T(A) | -T(B) | -P(C,A) | PD(C). [resolve(30,b,27,b)].
65 EX(A,B) | T(A) | -T(B) | -R(A). [resolve(30,b,28,b)].
66 -T(A) | R(A). [resolve(32,b,31,a)].
67 -S(A) | R(A). [resolve(35,b,34,a)].
68 -T(A) | -S(A). [resolve(36,a,35,b)].
69 -tmP(A,c1) | -SREG(c2,B,A) | -SREG(c3,C,A) | -::(B,cnt,A) | -::(C,cnt,A) | -P(B,C). [resolve(38,d,37,a)].
70 -::(A,idcnt,B) | S(A) | -::(C,idcnt,B) | S(C) | -SLC(A,D,B) | -SLC(C,E,B) | -P(D,E) | SLC(A,f101(A,B,C),B). [
       resolve(39,a,44,a)].
71 ::(c2,idcnt,c1). [resolve(47,a,40,a)].
72 -S(c2). [resolve(47,a,41,a)].
73 ::(c3,idcnt,c1). [resolve(47,a,42,a)].
74 -S(c3). [resolve(47,a,43,a)].
75 SLC(c2,f101(c2,c1,c3),c1). [resolve(47,a,44,a)].
76 SLC(c3,f102(c2,c1,c3),c1). [resolve(47,a,45,a)].
77 P(f101(c2,c1,c3),f102(c2,c1,c3)). [resolve(47,a,46,a)].
78 -SLC(A,B,C) | -P(C,C) | P(B,B). [factor(53,a,b)].
79 oP(A,A) | -P(A,A) | -T(A). [factor(54,c,d)].
80 -::(A,idcnt,c1) | S(A) | -SLC(A,B,c1) | -SLC(c2,C,c1) | -P(B,C) | SLC(A,f101(A,c1,c2),c1). [resolve(71,a,70,c)
       ,unit_del(c,72)].
81 SREG(c2,A,c1) | -SLC(c2,A,c1). [resolve(71,a,61,b),unit_del(b,72)].
```

```
82 -SLC(c2,A,c1) | -P(A,A) | SLC(c2,f101(c2,c1,c2),c1). [factor(80,c,d),unit_del(a,71),unit_del(b,72)].
83 SREG(c3,A,c1) | -SLC(c3,A,c1). [resolve(73,a,61,b),unit_del(b,74)].
84 -SLC(c2,A,c1) | f101(c2,c1,c3) = A. [resolve(75,a,52,b)].
85 PRE(c2,c1). [resolve(75,a,51,a)].
86 S(f101(c2,c1,c3)). [resolve(75,a,50,a)].
87 S(f102(c2,c1,c3)). [resolve(76,a,50,a)].
88 EX(f102(c2,c1,c3),A) | T(f102(c2,c1,c3)) | -T(A) | PD(f101(c2,c1,c3)). [resolve(77,a,64,d)].
89 T(f101(c2,c1,c3)) | -T(f102(c2,c1,c3)). [resolve(77,a,48,a)].
90 -P(c1,c1) | P(f101(c2,c1,c3),f101(c2,c1,c3)). [resolve(78,a,75,a)].
91 T(c1). [resolve(85,a,49,a)].
92 R(c1). [resolve(91,a,66,a)].
93 P(c1,c1). [resolve(92,a,62,a)].
94 P(f101(c2,c1,c3),f101(c2,c1,c3)). [back_unit_del(90),unit_del(a,93)].
95 oP(c1,c1). [resolve(93,a,79,b),unit_del(b,91)].
96 tmP(c1,c1) | oP(f79(c1,c1),c1). [resolve(95,a,55,b)].
97 -T(f101(c2,c1,c3)). [resolve(86,a,68,b)].
98 R(f101(c2,c1,c3)). [resolve(86,a,67,a)].
99 -T(f102(c2,c1,c3)). [back_unit_del(89),unit_del(a,97)].
100 EX(f102(c2,c1,c3),A) | -T(A) | PD(f101(c2,c1,c3)). [back_unit_del(88),unit_del(b,99)].
101 SREG(c2,f101(c2,c1,c3),c1). [resolve(81,b,75,a)].
102 -PD(f101(c2,c1,c3)). [resolve(98,a,63,b)].
103 EX(f102(c2,c1,c3),A) | -T(A). [back_unit_del(100),unit_del(c,102)].
104 SLC(c2,f101(c2,c1,c2),c1). [resolve(82,a,75,a),unit_del(a,94)].
105 SREG(c3,f102(c2,c1,c3),c1). [resolve(83,b,76,a)].
106 -tmP(c1,c1) | -SREG(c3,A,c1) | -::(f101(c2,c1,c3),cnt,c1) | -::(A,cnt,c1) | -P(f101(c2,c1,c3),A). [resolve
      (101,a,69,b)].
107 S(f101(c2,c1,c2)). [resolve(104,a,50,a)].
108 -T(f101(c2,c1,c2)). [resolve(107,a,68,b)].
109 R(f101(c2,c1,c2)). [resolve(107,a,67,a)].
110 EX(f101(c2,c1,c2),A) | -T(A). [resolve(109,a,65,d),unit_del(b,108)].
111 -tmP(c1,c1) | -::(f101(c2,c1,c3),cnt,c1) | -::(f102(c2,c1,c3),cnt,c1). [resolve(105,a,106,b),unit_del(d,77)].
112 f101(c2,c1,c3) = f101(c2,c1,c2). [resolve(84,a,104,a)].
113 -tmP(c1,c1) | -::(f101(c2,c1,c2),cnt,c1) | -::(f102(c2,c1,c3),cnt,c1). [back_rewrite(111),rewrite([112(7)])].
114 EX(f102(c2,c1,c3),c1). [resolve(103,b,91,a)].
115 ::(f102(c2,c1,c3),sreg,c1). [resolve(114,a,57,c),unit_del(b,87)].
116 ::(f102(c2,c1,c3),imen,c1). [resolve(115,a,58,b)].
117 ::(f102(c2,c1,c3),idcnt,c1). [resolve(116,a,59,b)].
118 ::(f102(c2,c1,c3),cnt,c1). [resolve(117,a,60,b)].
119 -tmP(c1,c1) | -::(f101(c2,c1,c2),cnt,c1). [back_unit_del(113),unit_del(c,118)].
120 EX(f101(c2,c1,c2),c1). [resolve(110,b,91,a)].
121 ::(f101(c2,c1,c2),sreg,c1). [resolve(120,a,57,c),unit_del(b,107)].
122 ::(f101(c2,c1,c2),imen,c1). [resolve(121,a,58,b)].
123 ::(f101(c2,c1,c2),idcnt,c1). [resolve(122,a,59,b)].
124 ::(f101(c2,c1,c2),cnt,c1). [resolve(123,a,60,b)].
125 -tmP(c1,c1). [back_unit_del(119),unit_del(b,124)].
126 oP(f79(c1,c1),c1). [back_unit_del(96),unit_del(a,125)].
127 $F. [resolve(126,a,56,c),unit_del(a,125),unit_del(b,95)].
```

Proof of Theorem (t_{db}55)

```
This went KO with both provers
```

Proof of Theorem (t_{db}57)

```
This went KO with both provers
```

Proof of Theorem (t_{db}58)

```
This went KO with both provers
```

Proof of Theorem (t_{db}59)

```
100. ! [X0,X3,X4] : (SLC(X0,X3,X4) => PRE(X0,X4)) [input]
124. ! [X0] : ((TR(X0) | PR(X0) | AR(X0)) <=> R(X0)) [input]
125. ! [X0] : (R(X0) => AB(X0)) [input]
130. ! [X0] : (T(X0) => TR(X0)) [input]
133. ~? [X0] : (PD(X0) & AB(X0)) [input]
162. ! [X0,X4] : (EX(X0,X4) <=> ((T(X4) & ~T(X0) & AB(X0)) | (P(X4,X0) & T(X0)) | PRE(X0,X4))) [input]
185. ! [X0,X1] : (OCCIN(X0,X1) <=> (! [X4] : (PRE(X0,X4) => ? [X3] : ? [X8] : (P(X3,X8) & SLC(X1,X8,X4) & SLC(X0,
       X3,X4))) & ? [X4] : (::(X1,cfbnd,X4) | ::(X1,site,X4) | ::(X1,mten,X4)) & PD(X0))) [input]
189. ! [X0,X1,X4] : ((EX(X0,X4) & OCCIN(X0,X1)) => EX(X1,X4)) [input]
190. ~! [X0,X1,X4] : ((EX(X0,X4) & OCCIN(X0,X1)) => EX(X1,X4)) [negated conjecture 189]
284. ! [X0,X1,X2] : (SLC(X0,X1,X2) => PRE(X0,X2)) [rectify 100]
304. ! [X0,X1] : (EX(X0,X1) <=> ((T(X1) & ~T(X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1))) [rectify 162]
329. ! [X0,X1] : (OCCIN(X0,X1) <=> (! [X2] : (PRE(X0,X2) => ? [X3] : ? [X4] : (P(X3,X4) & SLC(X1,X4,X2) & SLC(X0,
       X3,X2))) & ? [X5] : (::(X1,cfbnd,X5) | ::(X1,site,X5) | ::(X1,mten,X5)) & PD(X0))) [rectify 185]
330. ! [X0,X1] : (OCCIN(X0,X1) <=> (! [X2] : (PRE(X0,X2) => ? [X3,X4] : (P(X3,X4) & SLC(X1,X4,X2) & SLC(X0,X3,X2))
       ) & ? [X5] : (::(X1,cfbnd,X5) | ::(X1,site,X5) | ::(X1,mten,X5)) & PD(X0))) [flattening 329]
335. ~! [X0,X1,X2] : ((EX(X0,X2) & OCCIN(X0,X1)) => EX(X1,X2)) [rectify 190]
336. ! [X0,X1] : (OCCIN(X0,X1) => (! [X2] : (PRE(X0,X2) => ? [X3,X4] : (P(X3,X4) & SLC(X1,X4,X2) & SLC(X0,X3,X2)))
       & ? [X5] : (::(X1,cfbnd,X5) | ::(X1,site,X5) | ::(X1,mten,X5)) & PD(X0))) [unused predicate definition
      removal 330]
337. ! [X0] : ((TR(X0) | PR(X0) | AR(X0)) => R(X0)) [unused predicate definition removal 124]
507. ! [X0,X1,X2] : (PRE(X0,X2) | ~SLC(X0,X1,X2)) [ennf transformation 284]
534. ! [X0] : (R(X0) | (~TR(X0) & ~PR(X0) & ~AR(X0))) [ennf transformation 337]
535. ! [X0] : (AB(X0) | ~R(X0)) [ennf transformation 125]
539. ! [X0] : (TR(X0) | ~T(X0)) [ennf transformation 130]
```

```
542. ! [X0] : (~PD(X0) | ~AB(X0)) [ennf transformation 133]
565. ! [X0,X1] : ((! [X2] : (? [X3,X4] : (P(X3,X4) & SLC(X1,X4,X2) & SLC(X0,X3,X2)) | ~PRE(X0,X2)) & ? [X5] : (::(
       X1,cfbnd,X5) | ::(X1,site,X5) | ::(X1,mten,X5)) & PD(X0)) | ~OCCIN(X0,X1)) [ennf transformation 336]
566. ? [X0,X1,X2] : (~EX(X1,X2) & (EX(X0,X2) & OCCIN(X0,X1))) [ennf transformation 335]
567. ? [X0,X1,X2] : (~EX(X1,X2) & EX(X0,X2) & OCCIN(X0,X1)) [flattening 566]
570. ! [X1,X0] : (sP1(X1,X0) <=> ((T(X1) & ~T(X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1))) [predicate
       definition introduction]
571. ! [X0,X1] : (EX(X0,X1) <=> sP1(X1,X0)) [definition folding 304,570]
701. ! [X1,X0] : ((sP1(X1,X0) | ((~T(X1) | T(X0) | ~AB(X0)) & (~P(X1,X0) | ~T(X0)) & ~PRE(X0,X1))) & (((T(X1) & ~T
       (X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1)) | ~sP1(X1,X0))) [nnf transformation 570]
702. ! [X1,X0] : ((sP1(X1,X0) | ((~T(X1) | T(X0) | ~AB(X0)) & (~P(X1,X0) | ~T(X0)) & ~PRE(X0,X1))) & ((T(X1) & ~T(
       X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1) | ~sP1(X1,X0))) [flattening 701]
703. ! [X0,X1] : ((sP1(X0,X1) | ((~T(X0) | T(X1) | ~AB(X1)) & (~P(X0,X1) | ~T(X1)) & ~PRE(X1,X0))) & ((T(X0) & ~T(
       X1) & AB(X1)) | (P(X0,X1) & T(X1)) | PRE(X1,X0) | ~sP1(X0,X1))) [rectify 702]
704. ! [X0,X1] : ((EX(X0,X1) | ~sP1(X1,X0)) & (sP1(X1,X0) | ~EX(X0,X1))) [nnf transformation 571]
769. ! [X2,X1,X0] : (? [X3,X4] : (P(X3,X4) & SLC(X1,X4,X2) & SLC(X0,X3,X2)) => (P(sK84(X0,X1,X2),sK85(X0,X1,X2)) &
       SLC(X1,sK85(X0,X1,X2),X2) & SLC(X0,sK84(X0,X1,X2),X2))) [choice axiom]
770. ! [X1] : (? [X5] : (::(X1,cfbnd,X5) | ::(X1,site,X5) | ::(X1,mten,X5)) => (::(X1,cfbnd,sK86(X1)) | ::(X1,site
       ,sK86(X1)) | ::(X1,mten,sK86(X1)))) [choice axiom]
771. ! [X0,X1] : ((! [X2] : ((P(sK84(X0,X1,X2),sK85(X0,X1,X2)) & SLC(X1,sK85(X0,X1,X2),X2) & SLC(X0,sK84(X0,X1,X2)
       ,X2)) | ~PRE(X0,X2)) & (::(X1,cfbnd,sK86(X1)) | ::(X1,site,sK86(X1)) | ::(X1,mten,sK86(X1))) & PD(X0)) | ~
       OCCIN(X0,X1)) [skolemisation 565,770,769]
772. ? [X0,X1,X2] : (~EX(X1,X2) & EX(X0,X2) & OCCIN(X0,X1)) => (~EX(sK88,sK89) & EX(sK87,sK89) & OCCIN(sK87,sK88))
       [choice axiom]
773. ~EX(sK88,sK89) & EX(sK87,sK89) & OCCIN(sK87,sK88) [skolemisation 567,772]
963. ~SLC(X0,X1,X2) | PRE(X0,X2) [cnf transformation 507]
998. ~TR(X0) | R(X0) [cnf transformation 534]
999. ~R(X0) | AB(X0) [cnf transformation 535]
1005. ~T(X0) | TR(X0) [cnf transformation 539]
1008. ~AB(X0) | ~PD(X0) [cnf transformation 542]
1028. ~sP1(X0,X1) | T(X1) | PRE(X1,X0) | AB(X1) [cnf transformation 703]
1034. ~PRE(X1,X0) | sP1(X0,X1) [cnf transformation 703]
1037. ~EX(X0,X1) | sP1(X1,X0) [cnf transformation 704]
1038. ~sP1(X1,X0) | EX(X0,X1) [cnf transformation 704]
1127. ~OCCIN(X0,X1) | PD(X0) [cnf transformation 771]
1130. SLC(X1,sK85(X0,X1,X2),X2) | ~PRE(X0,X2) | ~OCCIN(X0,X1) [cnf transformation 771]
1132. OCCIN(sK87,sK88) [cnf transformation 773]
1133. EX(sK87,sK89) [cnf transformation 773]
1134. ~EX(sK88,sK89) [cnf transformation 773]
1155. PD(sK87) [resolution 1127,1132]
1188. sP1(sK89,sK87) [resolution 1037,1133]
1269. 2 <=> T(sK87) [avatar definition]
1270. ~T(sK87) <- (~2) [avatar component clause 1269]
1271. T(sK87) <- (2) [avatar component clause 1269]
1273. TR(sK87) <- (2) [resolution 1271,1005]
1283. R(sK87) <- (2) [resolution 1273,998]
1293. AB(sK87) <- (2) [resolution 1283,999]
1296. ~PD(sK87) <- (2) [resolution 1293,1008]
1297. $false <- (2) [subsumption resolution 1296,1155]
1298. ~2 [avatar contradiction clause 1297]
1805. T(sK87) | PRE(sK87,sK89) | AB(sK87) [resolution 1028,1188]
1807. PRE(sK87,sK89) | AB(sK87) <- (~2) [subsumption resolution 1805,1270]
1809. 3 <=> AB(sK87) [avatar definition]
1811. AB(sK87) <- (3) [avatar component clause 1809]
1813. 4 <=> PRE(sK87,sK89) [avatar definition]
1815. PRE(sK87,sK89) <- (4) [avatar component clause 1813]
1816. 3 | 4 | 2 [avatar split clause 1807,1269,1813,1809]
2435. ~OCCIN(X9,X11) | ~PRE(X9,X10) | PRE(X11,X10) [resolution 1130,963]
3547. ~PRE(sK87,X0) | PRE(sK88,X0) [resolution 2435,1132]
3582. PRE(sK88,sK89) <- (4) [resolution 3547,1815]
3665. sP1(sK89,sK88) <- (4) [resolution 3582,1034]
3703. EX(sK88,sK89) <- (4) [resolution 3665,1038]
3704. $false <- (4) [subsumption resolution 3703,1134]
3705. ~4 [avatar contradiction clause 3704]
3736. ~PD(sK87) <- (3) [resolution 1811,1008]
3737. $false <- (3) [subsumption resolution 3736,1155]
3738. ~3 [avatar contradiction clause 3737]
3739. $false [avatar sat refutation 1298,1816,3705,3738]
```

Proof of Theorem (t_{db} 62)

```
This went KO with both provers
```

Proof of Theorem (t_{db} 63)

```
This went KO with both provers
```

Proof of Theorem (t_{db} 64)

```
This went KO with both provers
```

Proof of Theorem (t_{db} 65)

```
12. ! [X0,X1] : (DQT(X0,X1) => ((NPED(X1) & AQ(X0)) | (PED(X1) & PQ(X0)) | (PD(X1) & tQ(X0)))) [input]
116. ! [X0] : ((PED(X0) | NPED(X0) | AS(X0)) <=> ED(X0)) [input]
118. ! [X0] : (((tQ(X0) | PQ(X0) | AQ(X0)) <=> Q(X0)) [input]
136. ~? [X0] : (Q(X0) & PD(X0)) [input]
```

```
137. ~? [X0] : (Q(X0) & ED(X0)) [input]
187. ! [X0,X1] : (SDEP(X0,X1) <=> (DQT(X0,X1) & ::(X0,sdcnt))) [input]
189. ! [X0,X1,X2] : ((X0 != X2 & SDEP(X1,X2) & SDEP(X0,X1)) => SDEP(X0,X2)) [input]
190. ~! [X0,X1,X2] : ((X0 != X2 & SDEP(X1,X2) & SDEP(X0,X1)) => SDEP(X0,X2)) [negated conjecture 189]
380. ! [X0,X1] : (((NPED(X1) & AQ(X0)) | (PED(X1) & PQ(X0)) | (PD(X1) & tQ(X0))) | ~DQT(X0,X1)) [ennf
     transformation 12]
381. ! [X0,X1] : ((NPED(X1) & AQ(X0)) | (PED(X1) & PQ(X0)) | (PD(X1) & tQ(X0)) | ~DQT(X0,X1)) [flattening 380]
543. ! [X0] : (~Q(X0) | ~PD(X0)) [ennf transformation 136]
544. ! [X0] : (~Q(X0) | ~ED(X0)) [ennf transformation 137]
563. ? [X0,X1,X2] : (~SDEP(X0,X2) & (X0 != X2 & SDEP(X1,X2) & SDEP(X0,X1))) [ennf transformation 190]
564. ? [X0,X1,X2] : (~SDEP(X0,X2) & X0 != X2 & SDEP(X1,X2) & SDEP(X0,X1)) [flattening 563]
691. ! [X0] : (((PED(X0) | NPED(X0) | AS(X0)) | ~ED(X0)) & (ED(X0) | (~PED(X0) & ~NPED(X0) & ~AS(X0)))) [nnf
     transformation 116]
692. ! [X0] : ((PED(X0) | NPED(X0) | AS(X0) | ~ED(X0)) & (ED(X0) | (~PED(X0) & ~NPED(X0) & ~AS(X0)))) [flattening
     691]
693. ! [X0] : (((tQ(X0) | PQ(X0) | AQ(X0)) | ~Q(X0)) & (Q(X0) | (~tQ(X0) & ~PQ(X0) & ~AQ(X0)))) [nnf
     transformation 118]
694. ! [X0] : ((tQ(X0) | PQ(X0) | AQ(X0) | ~Q(X0)) & (Q(X0) | (~tQ(X0) & ~PQ(X0) & ~AQ(X0)))) [flattening 693]
766. ! [X0,X1] : ((SDEP(X0,X1) | (~DQT(X0,X1) | ~::(X0,sdcnt))) & ((DQT(X0,X1) & ::(X0,sdcnt)) | ~SDEP(X0,X1))) [
     nnf transformation 187]
767. ! [X0,X1] : ((SDEP(X0,X1) | ~DQT(X0,X1) | ~::(X0,sdcnt)) & ((DQT(X0,X1) & ::(X0,sdcnt)) | ~SDEP(X0,X1))) [
     flattening 766]
768. ? [X0,X1,X2] : (~SDEP(X0,X2) & X0 != X2 & SDEP(X1,X2) & SDEP(X0,X1)) => (~SDEP(sK84,sK86) & sK84 != sK86 &
     SDEP(sK85,sK86) & SDEP(sK84,sK85)) [choice axiom]
769. ~SDEP(sK84,sK86) & sK84 != sK86 & SDEP(sK85,sK86) & SDEP(sK84,sK85) [skolemisation 564,768]
799. ~DQT(X0,X1) | PQ(X0) | tQ(X0) | AQ(X0) [cnf transformation 381]
806. ~DQT(X0,X1) | PED(X1) | PD(X1) | NPED(X1) [cnf transformation 381]
984. ~NPED(X0) | ED(X0) [cnf transformation 692]
985. ~PED(X0) | ED(X0) [cnf transformation 692]
987. ~AQ(X0) | Q(X0) [cnf transformation 694]
988. ~PQ(X0) | Q(X0) [cnf transformation 694]
989. ~tQ(X0) | Q(X0) [cnf transformation 694]
1007. ~Q(X0) | ~PD(X0) [cnf transformation 543]
1008. ~ED(X0) | ~Q(X0) [cnf transformation 544]
1124. ~SDEP(X0,X1) | DQT(X0,X1) [cnf transformation 767]
1126. SDEP(sK84,sK85) [cnf transformation 769]
1127. SDEP(sK85,sK86) [cnf transformation 769]
1195. DQT(sK84,sK85) [resolution 1124,1126]
1196. DQT(sK85,sK86) [resolution 1124,1127]
1221. 1 <=> PD(sK85) [avatar definition]
1222. ~PD(sK85) <- (~1) [avatar component clause 1221]
1244. 5 <=> PED(sK85) [avatar definition]
1246. PED(sK85) <- (5) [avatar component clause 1244]
1269. 9 <=> NPED(sK85) [avatar definition]
1271. NPED(sK85) <- (9) [avatar component clause 1269]
1282. 12 <=> AQ(sK85) [avatar definition]
1283. AQ(sK85) <- (12) [avatar component clause 1282]
1284. ~AQ(sK85) <- (~12) [avatar component clause 1282]
1298. 14 <=> PQ(sK85) [avatar definition]
1299. PQ(sK85) <- (14) [avatar component clause 1298]
1314. 16 <=> tQ(sK85) [avatar definition]
1315. tQ(sK85) <- (16) [avatar component clause 1314]
1513. PQ(sK85) | tQ(sK85) | AQ(sK85) [resolution 799,1196]
1525. ED(sK85) <- (9) [resolution 1271,984]
1560. Q(sK85) <- (12) [resolution 1283,987]
1567. ~Q(sK85) <- (9) [resolution 1525,1008]
1570. $false <- (9, 12) [subsumption resolution 1567,1560]
1571. ~9 | ~12 [avatar contradiction clause 1570]
1587. PED(sK85) | PD(sK85) | NPED(sK85) [resolution 806,1195]
1589. ED(sK85) <- (5) [resolution 1246,985]
1629. ~PD(sK85) <- (12) [resolution 1560,1007]
1631. ~1 | ~12 [avatar split clause 1629,1282,1221]
1647. ~Q(sK85) <- (5) [resolution 1589,1008]
1650. $false <- (5, 12) [subsumption resolution 1647,1560]
1651. ~5 | ~12 [avatar contradiction clause 1650]
1692. Q(sK85) <- (14) [resolution 1299,988]
1693. $false <- (5, 14) [subsumption resolution 1692,1647]
1694. ~5 | ~14 [avatar contradiction clause 1693]
1711. ED(sK85) <- (9) [resolution 1271,984]
1720. ~PD(sK85) <- (14) [resolution 1692,1007]
1723. ~Q(sK85) <- (9) [resolution 1711,1008]
1726. $false <- (9, 14) [subsumption resolution 1723,1692]
1727. ~9 | ~14 [avatar contradiction clause 1726]
1728. ~1 | ~14 [avatar split clause 1720,1298,1221]
1755. 20 <=> Q(sK85) [avatar definition]
1757. Q(sK85) <- (20) [avatar component clause 1755]
1759. 21 <=> ED(sK85) [avatar definition]
1760. ~ED(sK85) <- (~21) [avatar component clause 1759]
1761. ED(sK85) <- (21) [avatar component clause 1759]
1763. PQ(sK85) | tQ(sK85) <- (~12) [subsumption resolution 1513,1284]
1764. 16 | 14 | 12 [avatar split clause 1763,1282,1298,1314]
1768. ED(sK85) <- (5) [resolution 1246,985]
1769. 21 | ~5 [avatar split clause 1768,1244,1759]
1786. Q(sK85) <- (16) [resolution 1315,989]
1825. ~PD(sK85) <- (20) [resolution 1757,1007]
1827. ~1 | ~20 [avatar split clause 1825,1755,1221]
1845. ~Q(sK85) <- (21) [resolution 1761,1008]
1850. ~20 | ~21 [avatar split clause 1845,1759,1755]
1851. 20 | ~16 [avatar split clause 1786,1314,1755]
1854. PED(sK85) | NPED(sK85) <- (~1) [subsumption resolution 1587,1222]
1855. 9 | 5 | 1 [avatar split clause 1854,1221,1244,1269]
```

```
1880. ED(sK85) <- (9) [resolution 1271,984]
1881. $false <- (9, ~21) [subsumption resolution 1880,1760]
1882. ~9 | 21 [avatar contradiction clause 1881]
1883. $false [avatar sat refutation 1571,1631,1651,1694,1727,1728,1764,1769,1827,1850,1851,1855,1882]
```

Proof of Theorem (t_{db}66)

```
37. ! [X0,X1,X4] : ((PRE(X0,X4) & DQT(X0,X1)) => PRE(X1,X4)) [input]
80. ! [X0,X1] : (SD(X0,X1) <=> (! [X4] : (PRE(X0,X4) => exD(X0,X1,X4)) & ? [X4] : PRE(X0,X4))) [input]
87. ! [X0,X1] : (DQT(X0,X1) => SD(X0,X1)) [input]
122. ! [X0] : ((TR(X0) | PR(X0) | AR(X0)) <=> R(X0)) [input]
123. ! [X0] : (R(X0) => AB(X0)) [input]
128. ! [X0] : (T(X0) => TR(X0)) [input]
132. ~? [X0] : (Q(X0) & AB(X0)) [input]
160. ! [X0,X5] : (::(X0,X5) <=> ? [X4] : ::(X0,X5,X4)) [input]
169. ! [X0,X4] : (EX(X0,X4) <=> ((T(X4) & ~T(X0) & AB(X0)) | (P(X4,X0) & T(X0)) | PRE(X0,X4))) [input]
185. ! [X0,X4] : (::(X0,sdcnt,X4) <=> (? [X1] : (DQT(X0,X1) & ~S(X1) & ::(X1,idcnt,X4)) & PRE(X0,X4) & Q(X0))) [
       input]
194. ! [X0,X1] : (SDEP(X0,X1) <=> (DQT(X0,X1) & ::(X0,sdcnt))) [input]
196. ! [X0,X1] : (SDEP(X0,X1) => (! [X4] : (EX(X0,X4) => EX(X1,X4)) & ? [X4] : (EX(X1,X4) & EX(X0,X4)))) [input]
197. ~! [X0,X1] : (SDEP(X0,X1) => (! [X4] : (EX(X0,X4) => EX(X1,X4)) & ? [X4] : (EX(X1,X4) & EX(X0,X4)))) [negated
       conjecture 196]
210. ! [X0,X1,X2] : ((PRE(X0,X2) & DQT(X0,X1)) => PRE(X1,X2)) [rectify 37]
265. ! [X0,X1] : (SD(X0,X1) <=> (! [X2] : (PRE(X0,X2) => exD(X0,X1,X2)) & ? [X3] : PRE(X0,X3))) [rectify 80]
308. ! [X0,X1] : (::(X0,X1) <=> ? [X2] : ::(X0,X1,X2)) [rectify 160]
313. ! [X0,X1] : (EX(X0,X1) <=> ((T(X1) & ~T(X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1))) [rectify 169]
330. ! [X0,X1] : (::(X0,sdcnt,X1) <=> (? [X2] : (DQT(X0,X2) & ~S(X2) & ::(X2,idcnt,X1)) & PRE(X0,X1) & Q(X0))) [
       rectify 185]
344. ~! [X0,X1] : (SDEP(X0,X1) => (! [X2] : (EX(X0,X2) => EX(X1,X2)) & ? [X3] : (EX(X1,X3) & EX(X0,X3)))) [rectify
       197]
345. ! [X0,X1] : (SDEP(X0,X1) => (DQT(X0,X1) & ::(X0,sdcnt))) [unused predicate definition removal 194]
346. ! [X0,X1] : (::(X0,X1) => ? [X2] : ::(X0,X1,X2)) [unused predicate definition removal 308]
347. ! [X0] : ((TR(X0) | PR(X0) | AR(X0)) => R(X0)) [unused predicate definition removal 122]
350. ! [X0,X1] : (SD(X0,X1) => (! [X2] : (PRE(X0,X2) => exD(X0,X1,X2)) & ? [X3] : PRE(X0,X3))) [unused predicate
       definition removal 265]
427. ! [X0,X1,X2] : (PRE(X1,X2) | (~PRE(X0,X2) | ~DQT(X0,X1))) [ennf transformation 210]
428. ! [X0,X1,X2] : (PRE(X1,X2) | ~PRE(X0,X2) | ~DQT(X0,X1)) [flattening 427]
498. ! [X0,X1] : ((! [X2] : (exD(X0,X1,X2) | ~PRE(X0,X2)) & ? [X3] : PRE(X0,X3)) | ~SD(X0,X1)) [ennf
       transformation 350]
508. ! [X0,X1] : (SD(X0,X1) | ~DQT(X0,X1)) [ennf transformation 87]
543. ! [X0] : (R(X0) | (~TR(X0) & ~PR(X0) & ~AR(X0))) [ennf transformation 347]
544. ! [X0] : (AB(X0) | ~R(X0)) [ennf transformation 123]
548. ! [X0] : (TR(X0) | ~T(X0)) [ennf transformation 128]
552. ! [X0] : (~Q(X0) | ~AB(X0)) [ennf transformation 132]
571. ! [X0,X1] : (? [X2] : ::(X0,X1,X2) | ~::(X0,X1)) [ennf transformation 346]
578. ! [X0,X1] : ((DQT(X0,X1) & ::(X0,sdcnt)) | ~SDEP(X0,X1)) [ennf transformation 345]
579. ? [X0,X1] : ((? [X2] : (~EX(X1,X2) & EX(X0,X2)) | ! [X3] : (~EX(X1,X3) | ~EX(X0,X3))) & SDEP(X0,X1)) [ennf
       transformation 344]
582. ! [X1,X0] : (sP1(X1,X0) <=> ((T(X1) & ~T(X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1))) [predicate
       definition introduction]
583. ! [X0,X1] : (EX(X0,X1) <=> sP1(X1,X0)) [definition folding 313,582]
673. ! [X0] : (? [X3] : PRE(X0,X3) => PRE(X0,sK49(X0))) [choice axiom]
674. ! [X0,X1] : ((! [X2] : (exD(X0,X1,X2) | ~PRE(X0,X2)) & PRE(X0,sK49(X0))) | ~SD(X0,X1)) [skolemisation
       498,673]
712. ! [X1,X0] : (? [X2] : ::(X0,X1,X2) => ::(X0,X1,sK68(X0,X1))) [choice axiom]
713. ! [X0,X1] : (::(X0,X1,sK68(X0,X1)) | ~::(X0,X1)) [skolemisation 571,712]
716. ! [X1,X0] : ((sP1(X1,X0) | ((~T(X1) | T(X0) | ~AB(X0)) & (~P(X1,X0) | ~T(X0)) & ~PRE(X0,X1))) & (((T(X1) & ~T
       (X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1)) | ~sP1(X1,X0))) [nnf transformation 582]
717. ! [X1,X0] : ((sP1(X1,X0) | ((~T(X1) | T(X0) | ~AB(X0)) & (~P(X1,X0) | ~T(X0)) & ~PRE(X0,X1))) & ((T(X1) & ~T(
       X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1) | ~sP1(X1,X0))) [flattening 716]
718. ! [X0,X1] : ((sP1(X0,X1) | ((~T(X0) | T(X1) | ~AB(X1)) & (~P(X0,X1) | ~T(X1)) & ~PRE(X1,X0))) & ((T(X0) & ~T(
       X1) & AB(X1)) | (P(X0,X1) & T(X1)) | PRE(X1,X0) | ~sP1(X0,X1))) [rectify 717]
719. ! [X0,X1] : ((EX(X0,X1) | ~sP1(X1,X0)) & (sP1(X1,X0) | ~EX(X0,X1))) [nnf transformation 583]
766. ! [X0,X1] : ((::(X0,sdcnt,X1) | (! [X2] : (~DQT(X0,X2) | S(X2) | ~::(X2,idcnt,X1)) | ~PRE(X0,X1) | ~Q(X0))) &
       ((? [X2] : (DQT(X0,X2) & ~S(X2) & ::(X2,idcnt,X1)) & PRE(X0,X1) & Q(X0)) | ~::(X0,sdcnt,X1))) [nnf
       transformation 330]
767. ! [X0,X1] : ((::(X0,sdcnt,X1) | ! [X2] : (~DQT(X0,X2) | S(X2) | ~::(X2,idcnt,X1)) | ~PRE(X0,X1) | ~Q(X0)) &
       ((? [X2] : (DQT(X0,X2) & ~S(X2) & ::(X2,idcnt,X1)) & PRE(X0,X1) & Q(X0)) | ~::(X0,sdcnt,X1))) [flattening
       766]
768. ! [X0,X1] : ((::(X0,sdcnt,X1) | ! [X2] : (~DQT(X0,X2) | S(X2) | ~::(X2,idcnt,X1)) | ~PRE(X0,X1) | ~Q(X0)) &
       ((? [X3] : (DQT(X0,X3) & ~S(X3) & ::(X3,idcnt,X1)) & PRE(X0,X1) & Q(X0)) | ~::(X0,sdcnt,X1))) [rectify 767]
769. ! [X1,X0] : (? [X3] : (DQT(X0,X3) & ~S(X3) & ::(X3,idcnt,X1)) => (DQT(X0,sK82(X0,X1)) & ~S(sK82(X0,X1)) & ::(
       sK82(X0,X1),idcnt,X1))) [choice axiom]
770. ! [X0,X1] : ((::(X0,sdcnt,X1) | ! [X2] : (~DQT(X0,X2) | S(X2) | ~::(X2,idcnt,X1)) | ~PRE(X0,X1) | ~Q(X0)) &
       (((DQT(X0,sK82(X0,X1)) & ~S(sK82(X0,X1)) & ::(sK82(X0,X1),idcnt,X1)) & PRE(X0,X1) & Q(X0)) | ~::(X0,sdcnt,X1
       ))) [skolemisation 768,769]
784. ? [X0,X1] : ((? [X2] : (~EX(X1,X2) & EX(X0,X2)) | ! [X3] : (~EX(X1,X3) | ~EX(X0,X3))) & SDEP(X0,X1)) => ((? [
       X2] : (~EX(sK87,X2) & EX(sK86,X2)) | ! [X3] : (~EX(sK87,X3) | ~EX(sK86,X3))) & SDEP(sK86,sK87)) [choice
       axiom]
785. ? [X2] : (~EX(sK87,X2) & EX(sK86,X2)) => (~EX(sK87,sK88) & EX(sK86,sK88)) [choice axiom]
786. ((~EX(sK87,sK88) & EX(sK86,sK88)) | ! [X3] : (~EX(sK87,X3) | ~EX(sK86,X3))) & SDEP(sK86,sK87) [skolemisation
       579,785,784]
863. ~PRE(X0,X2) | PRE(X1,X2) | ~DQT(X0,X1) [cnf transformation 428]
948. ~SD(X0,X1) | PRE(X0,sK49(X0)) [cnf transformation 674]
959. ~DQT(X0,X1) | SD(X0,X1) [cnf transformation 508]
1008. ~TR(X0) | R(X0) [cnf transformation 543]
1009. ~R(X0) | AB(X0) [cnf transformation 544]
1015. ~T(X0) | TR(X0) [cnf transformation 548]
1019. ~Q(X0) | ~AB(X0) [cnf transformation 552]
1040. ::(X0,X1,sK68(X0,X1)) | ~::(X0,X1) [cnf transformation 713]
```

```
1044. ~sP1(X0,X1) | T(X1) | PRE(X1,X0) | AB(X1) [cnf transformation 718]
1050. ~PRE(X1,X0) | sP1(X0,X1) [cnf transformation 718]
1053. ~EX(X0,X1) | sP1(X1,X0) [cnf transformation 719]
1054. ~sP1(X1,X0) | EX(X0,X1) [cnf transformation 719]
1124. ~::(X0,sdcnt,X1) | Q(X0) [cnf transformation 770]
1143. ~SDEP(X0,X1) | ::(X0,sdcnt) [cnf transformation 578]
1144. ~SDEP(X0,X1) | DQT(X0,X1) [cnf transformation 578]
1145. SDEP(sK86,sK87) [cnf transformation 786]
1146. EX(sK86,sK88) | ~EX(sK87,X3) | ~EX(sK86,X3) [cnf transformation 786]
1147. ~EX(sK87,sK88) | ~EX(sK87,X3) | ~EX(sK86,X3) [cnf transformation 786]
1163. 1 <=> ! [X3] : (~EX(sK87,X3) | ~EX(sK86,X3)) [avatar definition]
1164. ~EX(sK87,X3) | ~EX(sK86,X3) <- (1) [avatar component clause 1163]
1166. 2 <=> EX(sK87,sK88) [avatar definition]
1168. ~EX(sK87,sK88) <- (~2) [avatar component clause 1166]
1169. 1 | ~2 [avatar split clause 1147,1166,1163]
1171. 3 <=> EX(sK86,sK88) [avatar definition]
1173. EX(sK86,sK88) <- (3) [avatar component clause 1171]
1174. 1 | 3 [avatar split clause 1146,1171,1163]
1224. ::(sK86,sdcnt) [resolution 1143,1145]
1225. DQT(sK86,sK87) [resolution 1144,1145]
1226. SD(sK86,sK87) [resolution 1225,959]
1319. PRE(sK86,sK49(sK86)) [resolution 948,1226]
1322. sP1(sK49(sK86),sK86) [resolution 1319,1050]
1328. EX(sK86,sK49(sK86)) [resolution 1322,1054]
1346. 13 <=> T(sK86) [avatar definition]
1348. T(sK86) <- (13) [avatar component clause 1346]
1660. PRE(X8,sK49(sK86)) | ~DQT(sK86,X8) [resolution 863,1319]
1721. ~::(X6,sdcnt) | Q(X6) [resolution 1040,1124]
1748. Q(sK86) [resolution 1721,1224]
2199. sP1(sK49(sK86),X8) | ~DQT(sK86,X8) [resolution 1660,1050]
2268. EX(X5,sK49(sK86)) | ~DQT(sK86,X5) [resolution 2199,1054]
2294. ~DQT(sK86,sK87) | ~EX(sK86,sK49(sK86)) <- (1) [resolution 2268,1164]
2296. ~EX(sK86,sK49(sK86)) <- (1) [subsumption resolution 2294,1225]
2297. $false <- (1) [subsumption resolution 2296,1328]
2298. ~1 [avatar contradiction clause 2297]
2299. sP1(sK88,sK86) <- (3) [resolution 1173,1053]
2325. T(sK86) | PRE(sK86,sK88) | AB(sK86) <- (3) [resolution 2299,1044]
2330. 21 <=> PRE(sK86,sK88) [avatar definition]
2332. PRE(sK86,sK88) <- (21) [avatar component clause 2330]
2344. 23 <=> AB(sK86) [avatar definition]
2345. ~AB(sK86) <- (~23) [avatar component clause 2344]
2347. 23 | 21 | 13 | ~3 [avatar split clause 2325,1171,1346,2330,2344]
2450. ~AB(sK86) [resolution 1748,1019]
2451. ~23 [avatar split clause 2450,2344]
2481. ~DQT(sK86,X2) | PRE(X2,sK88) <- (21) [resolution 2332,863]
2564. PRE(sK87,sK88) <- (21) [resolution 2481,1225]
2583. sP1(sK88,sK87) <- (21) [resolution 2564,1050]
2591. EX(sK87,sK88) <- (21) [resolution 2583,1054]
2592. $false <- (~2, 21) [subsumption resolution 2591,1168]
2593. 2 | ~21 [avatar contradiction clause 2592]
2596. TR(sK86) <- (13) [resolution 1348,1015]
2604. R(sK86) <- (13) [resolution 2596,1008]
2610. AB(sK86) <- (13) [resolution 2604,1009]
2611. $false <- (13, ~23) [subsumption resolution 2610,2345]
2612. ~13 | 23 [avatar contradiction clause 2611]
2613. $false [avatar sat refutation 1169,1174,2298,2347,2451,2593,2612]
```

Proof of Theorem (t_{db}67)

This went KO with both provers

Proof of Theorem (t_{db}68)

```
12. ! [X0,X1] : (DQT(X0,X1) => ((NPED(X1) & AQ(X0)) | (PED(X1) & PQ(X0)) | (PD(X1) & tQ(X0)))) [input]
13. ! [X0,X1,X2] : ((DQT(X0,X2) & DQT(X0,X1)) => X1 = X2) [input]
14. ! [X0] : (Q(X0) => ? [X1] : DQT(X0,X1)) [input]
114. ! [X0] : ((PED(X0) | NPED(X0) | AS(X0)) <=> ED(X0)) [input]
122. ! [X0] : ((TR(X0) | PR(X0) | AR(X0)) <=> R(X0)) [input]
123. ! [X0] : (R(X0) => AB(X0)) [input]
126. ! [X0] : (S(X0) => PR(X0)) [input]
130. ~? [X0] : (ED(X0) & AB(X0)) [input]
131. ~? [X0] : (PD(X0) & AB(X0)) [input]
132. ~? [X0] : (Q(X0) & AB(X0)) [input]
160. ! [X0,X5] : (::(X0,X5) <=> ? [X4] : ::(X0,X5,X4)) [input]
180. ! [X0,X4] : (::(X0,mten,X4) <=> (! [X5] : (PRE(X0,X5) => ? [X1] : ? [X3] : ? [X8] : (P(X8,X3) & SLC(X1,X8,X5)
     & SLC(X0,X3,X5) & M(X1))) & ? [X5] : (~AT(X5) & PRE(X0,X5)) & PRE(X0,X4) & ED(X0))) [input]
181. ! [X0,X4] : (((::(X0,cfbnd,X4) | ::(X0,site,X4)) <=> (! [X5] : (PRE(X0,X5) => ~? [X1] : ? [X3] : ? [X8] : (P(
     X8,X3) & SLC(X1,X8,X5) & SLC(X0,X3,X5) & M(X1))) & ? [X3] : SLC(X0,X3,X4) & PRE(X0,X4) & F(X0))) [input]
182. ! [X0,X4] : (::(X0,sreg,X4) <=> (EX(X0,X4) & S(X0))) [input]
183. ! [X0,X4] : (::(X0,imen,X4) <=> (::(X0,sreg,X4) | ::(X0,cfbnd,X4) | ::(X0,site,X4))) [input]
184. ! [X0,X4] : (::(X0,idcnt,X4) <=> (::(X0,imen,X4) | ::(X0,mten,X4))) [input]
185. ! [X0,X4] : (::(X0,sdcnt,X4) <=> (? [X1] : (DQT(X0,X1) & ~S(X1) & ::(X1,idcnt,X4)) & PRE(X0,X4) & Q(X0))) [
     input]
187. ! [X0,X4] : (::(X0,gdcnt,X4) <=> (! [X5] : (PRE(X0,X5) => ? [X1] : CONCR(X1,X0,X5)) & PRE(X0,X4))) [input]
188. ! [X0,X4] : (::(X0,cnt,X4) <=> (::(X0,gdcnt,X4) | ::(X0,sdcnt,X4) | ::(X0,idcnt,X4))) [input]
194. ! [X0,X1] : (SDEP(X0,X1) <=> (DQT(X0,X1) & ::(X0,sdcnt))) [input]
196. ! [X0,X4] : (::(X0,sdcnt,X4) <=> (? [X1] : (SDEP(X0,X1) & ~::(X1,sreg,X4) & ::(X1,idcnt,X4)) & ::(X0,cnt,X4))
     ) [input]
```

```
197. ~! [X0,X4] : (::(X0,sdcnt,X4) <=> (? [X1] : (SDEP(X0,X1) & ~::(X1,sreg,X4) & ::(X1,idcnt,X4)) & ::(X0,cnt,X4)
     )) [negated conjecture 196]
308. ! [X0,X1] : (::(X0,X1) <=> ? [X2] : ::(X0,X1,X2)) [rectify 160]
323. ! [X0,X1] : (::(X0,mten,X1) <=> (! [X2] : (PRE(X0,X2) => ? [X3] : ? [X4] : ? [X5] : (P(X5,X4) & SLC(X3,X5,X2)
     & SLC(X0,X4,X2) & M(X3))) & ? [X6] : (~AT(X6) & PRE(X0,X6)) & PRE(X0,X1) & ED(X0))) [rectify 180]
324. ! [X0,X1] : (::(X0,mten,X1) <=> (! [X2] : (PRE(X0,X2) => ? [X3,X4,X5] : (P(X5,X4) & SLC(X3,X5,X2) & SLC(X0,X4
     ,X2) & M(X3))) & ? [X6] : (~AT(X6) & PRE(X0,X6)) & PRE(X0,X1) & ED(X0))) [flattening 323]
325. ! [X0,X1] : ((::(X0,cfbnd,X1) | ::(X0,site,X1)) <=> (! [X2] : (PRE(X0,X2) => ~? [X3] : ? [X4] : ? [X5] : (P(
     X5,X4) & SLC(X3,X5,X2) & SLC(X0,X4,X2) & M(X3))) & ? [X6] : SLC(X0,X6,X1) & PRE(X0,X1) & F(X0))) [rectify
     181]
326. ! [X0,X1] : ((::(X0,cfbnd,X1) | ::(X0,site,X1)) <=> (! [X2] : (PRE(X0,X2) => ~? [X3,X4,X5] : (P(X5,X4) & SLC(
     X3,X5,X2) & SLC(X0,X4,X2) & M(X3))) & ? [X6] : SLC(X0,X6,X1) & PRE(X0,X1) & F(X0))) [flattening 325]
327. ! [X0,X1] : (::(X0,sreg,X1) <=> (EX(X0,X1) & S(X0))) [rectify 182]
328. ! [X0,X1] : (::(X0,imen,X1) <=> (::(X0,sreg,X1) | ::(X0,cfbnd,X1) | ::(X0,site,X1))) [rectify 183]
329. ! [X0,X1] : (::(X0,idcnt,X1) <=> (::(X0,imen,X1) | ::(X0,mten,X1))) [rectify 184]
330. ! [X0,X1] : (::(X0,sdcnt,X1) <=> (? [X2] : (DQT(X0,X2) & ~S(X2) & ::(X2,idcnt,X1)) & PRE(X0,X1) & Q(X0))) [
     rectify 185]
332. ! [X0,X1] : (::(X0,gdcnt,X1) <=> (! [X2] : (PRE(X0,X2) => ? [X3] : CONCR(X3,X0,X2)) & PRE(X0,X1))) [rectify
     187]
333. ! [X0,X1] : (::(X0,cnt,X1) <=> (::(X0,gdcnt,X1) | ::(X0,sdcnt,X1) | ::(X0,idcnt,X1))) [rectify 188]
344. ~! [X0,X1] : (::(X0,sdcnt,X1) <=> (? [X2] : (SDEP(X0,X2) & ~::(X2,sreg,X1) & ::(X2,idcnt,X1)) & ::(X0,cnt,X1)
     )) [rectify 197]
345. ! [X0] : ((TR(X0) | PR(X0) | AR(X0)) => R(X0)) [unused predicate definition removal 122]
390. ! [X0,X1] : (((NPED(X1) & AQ(X0)) | (PED(X1) & PQ(X0)) | (PD(X1) & tQ(X0))) | ~DQT(X0,X1)) [ennf
     transformation 12]
391. ! [X0,X1] : ((NPED(X1) & AQ(X0)) | (PED(X1) & PQ(X0)) | (PD(X1) & tQ(X0)) | ~DQT(X0,X1)) [flattening 390]
392. ! [X0,X1,X2] : (X1 = X2 | (~DQT(X0,X2) | ~DQT(X0,X1))) [ennf transformation 13]
393. ! [X0,X1,X2] : (X1 = X2 | ~DQT(X0,X2) | ~DQT(X0,X1)) [flattening 392]
394. ! [X0] : (? [X1] : DQT(X0,X1) | ~Q(X0)) [ennf transformation 14]
541. ! [X0] : (R(X0) | (~TR(X0) & ~PR(X0) & ~AR(X0))) [ennf transformation 345]
542. ! [X0] : (AB(X0) | ~R(X0)) [ennf transformation 123]
544. ! [X0] : (PR(X0) | ~S(X0)) [ennf transformation 126]
548. ! [X0] : (~ED(X0) | ~AB(X0)) [ennf transformation 130]
549. ! [X0] : (~PD(X0) | ~AB(X0)) [ennf transformation 131]
550. ! [X0] : (~Q(X0) | ~AB(X0)) [ennf transformation 132]
571. ! [X0,X1] : (::(X0,mten,X1) <=> (! [X2] : (? [X3,X4,X5] : (P(X5,X4) & SLC(X3,X5,X2) & SLC(X0,X4,X2) & M(X3))
     | ~PRE(X0,X2)) & ? [X6] : (~AT(X6) & PRE(X0,X6)) & PRE(X0,X1) & ED(X0))) [ennf transformation 324]
572. ! [X0,X1] : ((::(X0,cfbnd,X1) | ::(X0,site,X1)) <=> (! [X2] : (! [X3,X4,X5] : (~P(X5,X4) | ~SLC(X3,X5,X2) | ~
     SLC(X0,X4,X2) | ~M(X3)) | ~PRE(X0,X2)) & ? [X6] : SLC(X0,X6,X1) & PRE(X0,X1) & F(X0))) [ennf transformation
     326]
574. ! [X0,X1] : (::(X0,gdcnt,X1) <=> (! [X2] : (? [X3] : CONCR(X3,X0,X2) | ~PRE(X0,X2)) & PRE(X0,X1))) [ennf
     transformation 332]
575. ? [X0,X1] : (::(X0,sdcnt,X1) <~> (? [X2] : (SDEP(X0,X2) & ~::(X2,sreg,X1) & ::(X2,idcnt,X1)) & ::(X0,cnt,X1))
     ) [ennf transformation 344]
580. ! [X0,X1] : (sP2(X0,X1) <=> (! [X2] : (? [X3,X4,X5] : (P(X5,X4) & SLC(X3,X5,X2) & SLC(X0,X4,X2) & M(X3)) | ~
     PRE(X0,X2)) & ? [X6] : (~AT(X6) & PRE(X0,X6)) & PRE(X0,X1) & ED(X0))) [predicate definition introduction]
581. ! [X0,X1] : (::(X0,mten,X1) <=> sP2(X0,X1)) [definition folding 571,580]
582. ! [X0,X1] : (sP3(X0,X1) <=> (! [X2] : (! [X3,X4,X5] : (~P(X5,X4) | ~SLC(X3,X5,X2) | ~SLC(X0,X4,X2) | ~M(X3))
     | ~PRE(X0,X2)) & ? [X6] : SLC(X0,X6,X1) & PRE(X0,X1) & F(X0))) [predicate definition introduction]
583. ! [X0,X1] : ((::(X0,cfbnd,X1) | ::(X0,site,X1)) <=> sP3(X0,X1)) [definition folding 572,582]
603. ! [X0] : (? [X1] : DQT(X0,X1) => DQT(X0,sK8(X0))) [choice axiom]
604. ! [X0] : (DQT(X0,sK8(X0)) | ~Q(X0)) [skolemisation 394,603]
700. ! [X0] : (((PED(X0) | NPED(X0) | AS(X0)) | ~ED(X0)) & (ED(X0) | (~PED(X0) & ~NPED(X0) & ~AS(X0)))) [nnf
     transformation 114]
701. ! [X0] : ((PED(X0) | NPED(X0) | AS(X0) | ~ED(X0)) & (ED(X0) | (~PED(X0) & ~NPED(X0) & ~AS(X0)))) [flattening
     700]
708. ! [X0,X1] : ((::(X0,X1) | ! [X2] : ~::(X0,X1,X2)) & (? [X2] : ::(X0,X1,X2) | ~::(X0,X1))) [nnf transformation
     308]
709. ! [X0,X1] : ((::(X0,X1) | ! [X2] : ~::(X0,X1,X2)) & (? [X3] : ::(X0,X1,X3) | ~::(X0,X1))) [rectify 708]
710. ! [X1,X0] : (? [X3] : ::(X0,X1,X3) => ::(X0,X1,sK68(X0,X1))) [choice axiom]
711. ! [X0,X1] : ((::(X0,X1) | ! [X2] : ~::(X0,X1,X2)) & (::(X0,X1,sK68(X0,X1)) | ~::(X0,X1))) [skolemisation
     709,710]
741. ! [X0,X1] : ((sP2(X0,X1) | (? [X2] : (! [X3,X4,X5] : (~P(X5,X4) | ~SLC(X3,X5,X2) | ~SLC(X0,X4,X2) | ~M(X3)) &
     PRE(X0,X2)) | ! [X6] : (AT(X6) | ~PRE(X0,X6)) | ~PRE(X0,X1) | ~ED(X0))) & ((! [X2] : (? [X3,X4,X5] : (P(X5,
     X4) & SLC(X3,X5,X2) & SLC(X0,X4,X2) & M(X3)) | ~PRE(X0,X2)) & ? [X6] : (~AT(X6) & PRE(X0,X6)) & PRE(X0,X1) &
     ED(X0)) | ~sP2(X0,X1))) [nnf transformation 580]
742. ! [X0,X1] : ((sP2(X0,X1) | ? [X2] : (! [X3,X4,X5] : (~P(X5,X4) | ~SLC(X3,X5,X2) | ~SLC(X0,X4,X2) | ~M(X3)) &
     PRE(X0,X2)) | ! [X6] : (AT(X6) | ~PRE(X0,X6)) | ~PRE(X0,X1) | ~ED(X0)) & ((! [X2] : (? [X3,X4,X5] : (P(X5,X4
     ) & SLC(X3,X5,X2) & SLC(X0,X4,X2) & M(X3)) | ~PRE(X0,X2)) & ? [X6] : (~AT(X6) & PRE(X0,X6)) & PRE(X0,X1) &
     ED(X0)) | ~sP2(X0,X1))) [flattening 741]
743. ! [X0,X1] : ((sP2(X0,X1) | ? [X2] : (! [X3,X4,X5] : (~P(X5,X4) | ~SLC(X3,X5,X2) | ~SLC(X0,X4,X2) | ~M(X3)) &
     PRE(X0,X2)) | ! [X6] : (AT(X6) | ~PRE(X0,X6)) | ~PRE(X0,X1) | ~ED(X0)) & ((! [X7] : (? [X8,X9,X10] : (P(X10,
     X9) & SLC(X8,X10,X7) & SLC(X0,X9,X7) & M(X8)) | ~PRE(X0,X7)) & ? [X11] : (~AT(X11) & PRE(X0,X11)) & PRE(X0,
     X1) & ED(X0)) | ~sP2(X0,X1))) [rectify 742]
744. ! [X0] : (? [X2] : (! [X3,X4,X5] : (~P(X5,X4) | ~SLC(X3,X5,X2) | ~SLC(X0,X4,X2) | ~M(X3)) & PRE(X0,X2)) => (!
     [X5,X4,X3] : (~P(X5,X4) | ~SLC(X3,X5,sK72(X0)) | ~SLC(X0,X4,sK72(X0)) | ~M(X3)) & PRE(X0,sK72(X0)))) [
     choice axiom]
745. ! [X7,X0] : (? [X8,X9,X10] : (P(X10,X9) & SLC(X8,X10,X7) & SLC(X0,X9,X7) & M(X8)) => (P(sK75(X0,X7),sK74(X0,
     X7)) & SLC(sK73(X0,X7),sK75(X0,X7),X7) & SLC(X0,sK74(X0,X7),X7) & M(sK73(X0,X7)))) [choice axiom]
746. ! [X0] : (? [X11] : (~AT(X11) & PRE(X0,X11)) => (~AT(sK76(X0)) & PRE(X0,sK76(X0)))) [choice axiom]
747. ! [X0,X1] : ((sP2(X0,X1) | (! [X3,X4,X5] : (~P(X5,X4) | ~SLC(X3,X5,sK72(X0)) | ~SLC(X0,X4,sK72(X0)) | ~M(X3))
     & PRE(X0,sK72(X0))) | ! [X6] : (AT(X6) | ~PRE(X0,X6)) | ~PRE(X0,X1) | ~ED(X0)) & ((! [X7] : ((P(sK75(X0,X7)
     ,sK74(X0,X7)) & SLC(sK73(X0,X7),sK75(X0,X7),X7) & SLC(X0,sK74(X0,X7),X7) & M(sK73(X0,X7))) | ~PRE(X0,X7)) &
     (~AT(sK76(X0)) & PRE(X0,sK76(X0))) & PRE(X0,X1) & ED(X0)) | ~sP2(X0,X1))) [skolemisation 743,746,745,744]
748. ! [X0,X1] : ((::(X0,mten,X1) | ~sP2(X0,X1)) & (sP2(X0,X1) | ~::(X0,mten,X1))) [nnf transformation 581]
749. ! [X0,X1] : ((sP3(X0,X1) | (? [X2] : (? [X3,X4,X5] : (P(X5,X4) & SLC(X3,X5,X2) & SLC(X0,X4,X2) & M(X3)) & PRE
     (X0,X2)) | ! [X6] : ~SLC(X0,X6,X1) | ~PRE(X0,X1) | ~F(X0))) & ((! [X2] : (! [X3,X4,X5] : (~P(X5,X4) | ~SLC(
     X3,X5,X2) | ~SLC(X0,X4,X2) | ~M(X3)) | ~PRE(X0,X2)) & ? [X6] : SLC(X0,X6,X1) & PRE(X0,X1) & F(X0)) | ~sP3(X0
     ,X1))) [nnf transformation 582]
750. ! [X0,X1] : ((sP3(X0,X1) | ? [X2] : (? [X3,X4,X5] : (P(X5,X4) & SLC(X3,X5,X2) & SLC(X0,X4,X2) & M(X3)) & PRE(
     X0,X2)) | ! [X6] : ~SLC(X0,X6,X1) | ~PRE(X0,X1) | ~F(X0)) & ((! [X2] : (! [X3,X4,X5] : (~P(X5,X4) | ~SLC(X3,
```

X5,X2) | ~SLC(X0,X4,X2) | ~M(X3)) | ~PRE(X0,X2)) & ? [X6] : SLC(X0,X6,X1) & PRE(X0,X1) & F(X0)) | ~sP3(X0,X1))) [flattening 749]
751. ! [X0,X1] : ((sP3(X0,X1) | ? [X2] : (? [X3,X4,X5] : (P(X5,X4) & SLC(X3,X5,X2) & SLC(X0,X4,X2) & M(X3)) & PRE(X0,X2)) | ! [X6] : ~SLC(X0,X6,X1) | ~PRE(X0,X1) | ~F(X0)) & ((! [X7] : (! [X8,X9,X10] : (~P(X10,X9) | ~SLC(X8,X10,X7) | ~SLC(X0,X9,X7) | ~M(X8)) | ~PRE(X0,X7)) & ? [X11] : SLC(X0,X11,X1) & PRE(X0,X1) & F(X0)) | ~sP3(X0,X1))) [rectify 750]
752. ! [X0] : (? [X2] : (? [X3,X4,X5] : (P(X5,X4) & SLC(X3,X5,X2) & SLC(X0,X4,X2) & M(X3)) & PRE(X0,X2)) => (? [X5,X4,X3] : (P(X5,X4) & SLC(X3,X5,sK77(X0)) & SLC(X0,X4,sK77(X0)) & M(X3)) & PRE(X0,sK77(X0)))) [choice axiom]
753. ! [X0] : (? [X5,X4,X3] : (P(X5,X4) & SLC(X3,X5,sK77(X0)) & SLC(X0,X4,sK77(X0)) & M(X3)) => (P(sK80(X0),sK79(X0)) & SLC(sK78(X0),sK80(X0),sK77(X0)) & SLC(X0,sK79(X0),sK77(X0)) & M(sK78(X0)))) [choice axiom]
754. ! [X1,X0] : (? [X11] : SLC(X0,X11,X1) => SLC(X0,sK81(X0,X1),X1)) [choice axiom]
755. ! [X0,X1] : ((sP3(X0,X1) | ((P(sK80(X0),sK79(X0)) & SLC(sK78(X0),sK80(X0),sK77(X0)) & SLC(X0,sK79(X0),sK77(X0)) & M(sK78(X0))) & PRE(X0,sK77(X0))) | ! [X6] : ~SLC(X0,X6,X1) | ~PRE(X0,X1) | ~F(X0)) & ((! [X7] : (! [X8,X9,X10] : (~P(X10,X9) | ~SLC(X8,X10,X7) | ~SLC(X0,X9,X7) | ~M(X8)) | ~PRE(X0,X7)) & SLC(X0,sK81(X0,X1),X1) & PRE(X0,X1) & F(X0)) | ~sP3(X0,X1))) [skolemisation 751,754,753,752]
756. ! [X0,X1] : (((::(X0,cfbnd,X1) | ::(X0,site,X1)) | ~sP3(X0,X1)) & (sP3(X0,X1) | (~::(X0,cfbnd,X1) & ~::(X0,site,X1)))) [nnf transformation 583]
757. ! [X0,X1] : ((::(X0,cfbnd,X1) | ::(X0,site,X1) | ~sP3(X0,X1)) & (sP3(X0,X1) | (~::(X0,cfbnd,X1) & ~::(X0,site,X1)))) [flattening 756]
758. ! [X0,X1] : ((::(X0,sreg,X1) | (~EX(X0,X1) | ~S(X0))) & ((EX(X0,X1) & S(X0)) | ~::(X0,sreg,X1))) [nnf transformation 327]
759. ! [X0,X1] : ((::(X0,sreg,X1) | ~EX(X0,X1) | ~S(X0)) & ((EX(X0,X1) & S(X0)) | ~::(X0,sreg,X1))) [flattening 758]
760. ! [X0,X1] : ((::(X0,imen,X1) | (~::(X0,sreg,X1) & ~::(X0,cfbnd,X1) & ~::(X0,site,X1))) & ((::(X0,sreg,X1) | ::(X0,cfbnd,X1) | ::(X0,site,X1)) | ~::(X0,imen,X1))) [nnf transformation 328]
761. ! [X0,X1] : ((::(X0,imen,X1) | (~::(X0,sreg,X1) & ~::(X0,cfbnd,X1) & ~::(X0,site,X1))) & (::(X0,sreg,X1) | ::(X0,cfbnd,X1) | ::(X0,site,X1) | ~::(X0,imen,X1))) [flattening 760]
762. ! [X0,X1] : ((::(X0,idcnt,X1) | (~::(X0,imen,X1) & ~::(X0,mten,X1))) & ((::(X0,imen,X1) | ::(X0,mten,X1)) | ~::(X0,idcnt,X1))) [nnf transformation 329]
763. ! [X0,X1] : ((::(X0,idcnt,X1) | (~::(X0,imen,X1) & ~::(X0,mten,X1))) & (::(X0,imen,X1) | ::(X0,mten,X1) | ~::(X0,idcnt,X1))) [flattening 762]
764. ! [X0,X1] : ((::(X0,sdcnt,X1) | (! [X2] : (~DQT(X0,X2) | S(X2) | ~::(X2,idcnt,X1)) | ~PRE(X0,X1) | ~Q(X0))) & ((? [X2] : (DQT(X0,X2) & ~S(X2) & ::(X2,idcnt,X1)) & PRE(X0,X1) & Q(X0)) | ~::(X0,sdcnt,X1))) [nnf transformation 330]
765. ! [X0,X1] : ((::(X0,sdcnt,X1) | ! [X2] : (~DQT(X0,X2) | S(X2) | ~::(X2,idcnt,X1)) | ~PRE(X0,X1) | ~Q(X0)) & ((? [X2] : (DQT(X0,X2) & ~S(X2) & ::(X2,idcnt,X1)) & PRE(X0,X1) & Q(X0)) | ~::(X0,sdcnt,X1))) [flattening 764]
766. ! [X0,X1] : ((::(X0,sdcnt,X1) | ! [X2] : (~DQT(X0,X2) | S(X2) | ~::(X2,idcnt,X1)) | ~PRE(X0,X1) | ~Q(X0)) & ((? [X3] : (DQT(X0,X3) & ~S(X3) & ::(X3,idcnt,X1)) & PRE(X0,X1) & Q(X0)) | ~::(X0,sdcnt,X1))) [rectify 765]
767. ! [X1,X0] : (? [X3] : (DQT(X0,X3) & ~S(X3) & ::(X3,idcnt,X1)) => (DQT(X0,sK82(X0,X1)) & ~S(sK82(X0,X1)) & ::(sK82(X0,X1),idcnt,X1))) [choice axiom]
768. ! [X0,X1] : ((::(X0,sdcnt,X1) | ! [X2] : (~DQT(X0,X2) | S(X2) | ~::(X2,idcnt,X1)) | ~PRE(X0,X1) | ~Q(X0)) & (((DQT(X0,sK82(X0,X1)) & ~S(sK82(X0,X1)) & ::(sK82(X0,X1),idcnt,X1)) & PRE(X0,X1) & Q(X0)) | ~::(X0,sdcnt,X1))) [skolemisation 766,767]
774. ! [X0,X1] : ((::(X0,gdcnt,X1) | (? [X2] : (! [X3] : ~CONCR(X3,X0,X2) & PRE(X0,X2)) | ~PRE(X0,X1))) & ((! [X2] : (? [X3] : CONCR(X3,X0,X2) | ~PRE(X0,X2)) & PRE(X0,X1)) | ~::(X0,gdcnt,X1))) [nnf transformation 574]
775. ! [X0,X1] : ((::(X0,gdcnt,X1) | ? [X2] : (! [X3] : ~CONCR(X3,X0,X2) & PRE(X0,X2)) | ~PRE(X0,X1)) & ((! [X2] : (? [X3] : CONCR(X3,X0,X2) | ~PRE(X0,X2)) & PRE(X0,X1)) | ~::(X0,gdcnt,X1))) [flattening 774]
776. ! [X0,X1] : ((::(X0,gdcnt,X1) | ? [X2] : (! [X3] : ~CONCR(X3,X0,X2) & PRE(X0,X2)) | ~PRE(X0,X1)) & ((! [X4] : (? [X5] : CONCR(X5,X0,X4) | ~PRE(X0,X4)) & PRE(X0,X1)) | ~::(X0,gdcnt,X1))) [rectify 775]
777. ! [X0] : (? [X2] : (! [X3] : ~CONCR(X3,X0,X2) & PRE(X0,X2)) => (! [X3] : ~CONCR(X3,X0,sK84(X0)) & PRE(X0,sK84(X0)))) [choice axiom]
778. ! [X4,X0] : (? [X5] : CONCR(X5,X0,X4) => CONCR(sK85(X0,X4),X0,X4)) [choice axiom]
779. ! [X0,X1] : ((::(X0,gdcnt,X1) | (! [X3] : ~CONCR(X3,X0,sK84(X0)) & PRE(X0,sK84(X0)))) | ~PRE(X0,X1)) & ((! [X4] : (CONCR(sK85(X0,X4),X0,X4) | ~PRE(X0,X4)) & PRE(X0,X1)) | ~::(X0,gdcnt,X1))) [skolemisation 776,778,777]
780. ! [X0,X1] : ((::(X0,cnt,X1) | (~::(X0,gdcnt,X1) & ~::(X0,sdcnt,X1) & ~::(X0,idcnt,X1))) & ((::(X0,gdcnt,X1) | ::(X0,sdcnt,X1) | ::(X0,idcnt,X1)) | ~::(X0,cnt,X1))) [nnf transformation 333]
781. ! [X0,X1] : ((::(X0,cnt,X1) | (~::(X0,gdcnt,X1) & ~::(X0,sdcnt,X1) & ~::(X0,idcnt,X1))) & (::(X0,gdcnt,X1) | ::(X0,sdcnt,X1) | ::(X0,idcnt,X1) | ~::(X0,cnt,X1))) [flattening 780]
782. ! [X0,X1] : ((SDEP(X0,X1) | (~DQT(X0,X1) | ~::(X0,sdcnt))) & ((DQT(X0,X1) & ::(X0,sdcnt)) | ~SDEP(X0,X1))) [nnf transformation 194]
783. ! [X0,X1] : ((SDEP(X0,X1) | ~DQT(X0,X1) | ~::(X0,sdcnt)) & ((DQT(X0,X1) & ::(X0,sdcnt)) | ~SDEP(X0,X1))) [flattening 782]
784. ? [X0,X1] : (((! [X2] : (~SDEP(X0,X2) | ::(X2,sreg,X1) | ~::(X2,idcnt,X1)) | ~::(X0,cnt,X1)) | ~::(X0,sdcnt,X1)) & ((? [X2] : (SDEP(X0,X2) & ~::(X2,sreg,X1) & ::(X2,idcnt,X1)) & ::(X0,cnt,X1)) | ::(X0,sdcnt,X1))) [nnf transformation 575]
785. ? [X0,X1] : ((! [X2] : (~SDEP(X0,X2) | ::(X2,sreg,X1) | ~::(X2,idcnt,X1)) | ~::(X0,cnt,X1) | ~::(X0,sdcnt,X1)) & ((? [X2] : (SDEP(X0,X2) & ~::(X2,sreg,X1) & ::(X2,idcnt,X1)) & ::(X0,cnt,X1)) | ::(X0,sdcnt,X1))) [flattening 784]
786. ? [X0,X1] : ((! [X2] : (~SDEP(X0,X2) | ::(X2,sreg,X1) | ~::(X2,idcnt,X1)) | ~::(X0,cnt,X1) | ~::(X0,sdcnt,X1)) & ((? [X3] : (SDEP(X0,X3) & ~::(X3,sreg,X1) & ::(X3,idcnt,X1)) & ::(X0,cnt,X1)) | ::(X0,sdcnt,X1))) [rectify 785]
787. ? [X0,X1] : ((! [X2] : (~SDEP(X0,X2) | ::(X2,sreg,X1) | ~::(X2,idcnt,X1)) | ~::(X0,cnt,X1) | ~::(X0,sdcnt,X1)) & ((? [X3] : (SDEP(X0,X3) & ~::(X3,sreg,X1) & ::(X3,idcnt,X1)) & ::(X0,cnt,X1)) | ::(X0,sdcnt,X1))) => ((! [X2] : (~SDEP(sK86,X2) | ::(X2,sreg,sK87) | ~::(X2,idcnt,sK87)) | ~::(sK86,cnt,sK87) | ~::(sK86,sdcnt,sK87)) & ((? [X3] : (SDEP(sK86,X3) & ~::(X3,sreg,sK87) & ::(X3,idcnt,sK87)) & ::(sK86,cnt,sK87)) | ::(sK86,sdcnt,sK87))) [choice axiom]
788. ? [X3] : (SDEP(sK86,X3) & ~::(X3,sreg,sK87) & ::(X3,idcnt,sK87)) => (SDEP(sK86,sK88) & ~::(sK88,sreg,sK87) & ::(sK88,idcnt,sK87)) [choice axiom]
789. (! [X2] : (~SDEP(sK86,X2) | ::(X2,sreg,sK87) | ~::(X2,idcnt,sK87)) | ~::(sK86,cnt,sK87) | ~::(sK86,sdcnt,sK87)) & (((SDEP(sK86,sK88) & ~::(sK88,sreg,sK87) & ::(sK88,idcnt,sK87)) & ::(sK86,cnt,sK87)) | ::(sK86,sdcnt,sK87)) [skolemisation 786,788,787]
826. ~DQT(X0,X1) | PED(X1) | PD(X1) | NPED(X1) [cnf transformation 391]
827. ~DQT(X0,X2) | X1 = X2 | ~DQT(X0,X1) [cnf transformation 393]
828. DQT(X0,sK8(X0)) | ~Q(X0) [cnf transformation 604]
1001. ~NPED(X0) | ED(X0) [cnf transformation 701]
1002. ~PED(X0) | ED(X0) [cnf transformation 701]
1010. ~PR(X0) | R(X0) [cnf transformation 541]
1012. ~R(X0) | AB(X0) [cnf transformation 542]
1016. ~S(X0) | PR(X0) [cnf transformation 544]
1020. ~ED(X0) | ~AB(X0) [cnf transformation 548]

```
1021. ~AB(X0) | ~PD(X0) [cnf transformation 549]
1022. ~Q(X0) | ~AB(X0) [cnf transformation 550]
1043. ::(X0,X1,sK68(X0,X1)) | ~::(X0,X1) [cnf transformation 711]
1044. ~::(X0,X1,X2) | ::(X0,X1) [cnf transformation 711]
1095. ~sP2(X0,X1) | PRE(X0,X1) [cnf transformation 747]
1104. ~::(X0,mten,X1) | sP2(X0,X1) [cnf transformation 748]
1107. ~sP3(X0,X1) | PRE(X0,X1) [cnf transformation 755]
1115. ~::(X0,site,X1) | sP3(X0,X1) [cnf transformation 757]
1116. ~::(X0,cfbnd,X1) | sP3(X0,X1) [cnf transformation 757]
1118. ~::(X0,sreg,X1) | S(X0) [cnf transformation 759]
1121. ~::(X0,imen,X1) | ::(X0,cfbnd,X1) | ::(X0,site,X1) | ::(X0,sreg,X1) [cnf transformation 761]
1125. ~::(X0,idcnt,X1) | ::(X0,mten,X1) | ::(X0,imen,X1) [cnf transformation 763]
1128. ~::(X0,sdcnt,X1) | Q(X0) [cnf transformation 768]
1130. ::(sK82(X0,X1),idcnt,X1) | ~::(X0,sdcnt,X1) [cnf transformation 768]
1131. ~::(X0,sdcnt,X1) | ~S(sK82(X0,X1)) [cnf transformation 768]
1132. DQT(X0,sK82(X0,X1)) | ~::(X0,sdcnt,X1) [cnf transformation 768]
1133. ~::(X2,idcnt,X1) | ~DQT(X0,X2) | S(X2) | ::(X0,sdcnt,X1) | ~PRE(X0,X1) | ~Q(X0) [cnf transformation 768]
1139. ~::(X0,gdcnt,X1) | PRE(X0,X1) [cnf transformation 779]
1143. ~::(X0,cnt,X1) | ::(X0,sdcnt,X1) | ::(X0,idcnt,X1) | ::(X0,gdcnt,X1) [cnf transformation 781]
1145. ~::(X0,sdcnt,X1) | ::(X0,cnt,X1) [cnf transformation 781]
1147. ~SDEP(X0,X1) | ::(X0,sdcnt) [cnf transformation 783]
1148. ~SDEP(X0,X1) | DQT(X0,X1) [cnf transformation 783]
1149. ~::(X0,sdcnt) | ~DQT(X0,X1) | SDEP(X0,X1) [cnf transformation 783]
1150. ::(sK86,cnt,sK87) | ::(sK86,sdcnt,sK87) [cnf transformation 789]
1151. ::(sK88,idcnt,sK87) | ::(sK86,sdcnt,sK87) [cnf transformation 789]
1153. SDEP(sK86,sK88) | ::(sK86,sdcnt,sK87) [cnf transformation 789]
1154. ~SDEP(sK86,X2) | ::(X2,sreg,sK87) | ~::(X2,idcnt,sK87) | ~::(sK86,cnt,sK87) | ~::(sK86,sdcnt,sK87) [cnf
      transformation 789]
1169. ~SDEP(sK86,X2) | ::(X2,sreg,sK87) | ~::(X2,idcnt,sK87) | ~::(sK86,sdcnt,sK87) [subsumption resolution
      1154,1145]
1171. 1 <=> ::(sK86,sdcnt,sK87) [avatar definition]
1172. ::(sK86,sdcnt,sK87) <- (1) [avatar component clause 1171]
1173. ~::(sK86,sdcnt,sK87) <- (~1) [avatar component clause 1171]
1175. 2 <=> ! [X2] : (~SDEP(sK86,X2) | ~::(X2,idcnt,sK87) | ::(X2,sreg,sK87)) [avatar definition]
1176. ~::(X2,idcnt,sK87) | ~SDEP(sK86,X2) | ::(X2,sreg,sK87) <- (2) [avatar component clause 1175]
1177. ~1 | 2 [avatar split clause 1169,1175,1171]
1179. 3 <=> SDEP(sK86,sK88) [avatar definition]
1181. SDEP(sK86,sK88) <- (3) [avatar component clause 1179]
1182. 1 | 3 [avatar split clause 1153,1179,1171]
1189. 5 <=> ::(sK88,idcnt,sK87) [avatar definition]
1191. ::(sK88,idcnt,sK87) <- (5) [avatar component clause 1189]
1192. 1 | 5 [avatar split clause 1151,1189,1171]
1193. ::(sK86,cnt,sK87) [subsumption resolution 1150,1145]
1267. 6 <=> PD(sK88) [avatar definition]
1280. 8 <=> PED(sK88) [avatar definition]
1282. PED(sK88) <- (8) [avatar component clause 1280]
1295. 10 <=> NPED(sK88) [avatar definition]
1297. NPED(sK88) <- (10) [avatar component clause 1295]
1709. ~::(X6,sdcnt) | Q(X6) [resolution 1043,1128]
1754. 14 <=> S(sK88) [avatar definition]
1755. S(sK88) <- (14) [avatar component clause 1754]
1770. ~::(X16,sdcnt,X17) | sK82(X16,X17) = X18 | ~DQT(X16,X18) [resolution 1132,827]
2604. DQT(sK86,sK88) <- (3) [resolution 1181,1148]
2605. ::(sK86,sdcnt) <- (3) [resolution 1181,1147]
2613. PED(sK88) | PD(sK88) | NPED(sK88) <- (3) [resolution 2604,826]
2622. Q(sK86) <- (3) [resolution 2605,1709]
2977. ::(sK86,sdcnt,sK87) | ::(sK86,idcnt,sK87) | ::(sK86,gdcnt,sK87) [resolution 1143,1193]
2984. 32 <=> ::(sK86,gdcnt,sK87) [avatar definition]
2986. ::(sK86,gdcnt,sK87) <- (32) [avatar component clause 2984]
2988. 33 <=> ::(sK86,idcnt,sK87) [avatar definition]
2990. ::(sK86,idcnt,sK87) <- (33) [avatar component clause 2988]
3365. 47 <=> S(sK86) [avatar definition]
3366. S(sK86) <- (47) [avatar component clause 3365]
3367. ~S(sK86) <- (~47) [avatar component clause 3365]
3556. 52 <=> ED(sK88) [avatar definition]
3557. ED(sK88) <- (52) [avatar component clause 3556]
3573. 10 | 6 | 8 | ~3 [avatar split clause 2613,1179,1280,1267,1295]
3593. 59 <=> Q(sK86) [avatar definition]
3595. Q(sK86) <- (59) [avatar component clause 3593]
3606. 62 <=> SDEP(sK86,sK8(sK86)) [avatar definition]
3607. ~SDEP(sK86,sK8(sK86)) <- (~62) [avatar component clause 3606]
3608. SDEP(sK86,sK8(sK86)) <- (62) [avatar component clause 3606]
3627. 59 | ~3 [avatar split clause 2622,1179,3593]
3635. 66 <=> AB(sK86) [avatar definition]
3636. ~AB(sK86) <- (~66) [avatar component clause 3635]
3641. 67 <=> AB(sK88) [avatar definition]
3643. AB(sK88) <- (67) [avatar component clause 3641]
3657. ED(sK88) <- (10) [resolution 1297,1001]
3658. 52 | ~10 [avatar split clause 3657,1295,3556]
3679. ED(sK88) <- (8) [resolution 1282,1002]
3680. 52 | ~8 [avatar split clause 3679,1280,3556]
3684. ~S(sK82(sK86,sK87)) <- (1) [resolution 1172,1131]
3686. Q(sK86) <- (1) [resolution 1172,1128]
3687. ::(sK86,sdcnt) <- (1) [resolution 1172,1044]
3691. ::(sK82(X2,sK87),sreg,sK87) | ~SDEP(sK86,sK82(X2,sK87)) | ~::(X2,sdcnt,sK87) <- (2) [resolution 1176,1130]
3692. ~DQT(sK86,X0) | SDEP(sK86,X0) <- (1) [resolution 3687,1149]
3702. ~AB(sK86) <- (59) [resolution 3595,1022]
3703. ~66 | ~59 [avatar split clause 3702,3593,3635]
3728. ~PD(sK88) <- (67) [resolution 3643,1021]
4108. DQT(sK86,sK8(sK86)) <- (62) [resolution 3608,1148]
4580. SDEP(sK86,sK8(sK86)) | ~Q(sK86) <- (1) [resolution 3692,828]
```

```
5516. 107 <=> sP3(sK86,sK87) [avatar definition]
5517. ~sP3(sK86,sK87) <- (~107) [avatar component clause 5516]
5518. sP3(sK86,sK87) <- (107) [avatar component clause 5516]
5572. ~DQT(sK86,X0) | sK82(sK86,sK87) = X0 <- (1) [resolution 1770,1172]
5582. 112 <=> S(sK8(sK86)) [avatar definition]
5583. S(sK8(sK86)) <- (112) [avatar component clause 5582]
12656. sK8(sK86) = sK82(sK86,sK87) <- (1, 62) [resolution 5572,4108]
12657. sK8(sK86) = sK82(sK86,sK87) | ~Q(sK86) <- (1) [resolution 5572,828]
12667. ::(sK8(sK86),sreg,sK87) | ~SDEP(sK86,sK8(sK86)) | ~::(sK86,sdcnt,sK87) <- (1, 2, 62) [superposition
       3691,12656]
12675. ::(sK8(sK86),sreg,sK87) | ~::(sK86,sdcnt,sK87) <- (1, 2, 62) [subsumption resolution 12667,3608]
12676. ::(sK8(sK86),sreg,sK87) <- (1, 2, 62) [subsumption resolution 12675,1172]
12776. S(sK8(sK86)) <- (1, 2, 62) [resolution 12676,1118]
12782. 112 | ~1 | ~2 | ~62 [avatar split clause 12776,3606,1175,1171,5582]
12785. sK8(sK86) = sK82(sK86,sK87) <- (1, 59) [subsumption resolution 12657,3595]
12792. ~S(sK8(sK86)) <- (1, 59) [forward demodulation 3684,12785]
12793. $false <- (1, 59, 112) [subsumption resolution 12792,5583]
12794. ~1 | ~59 | ~112 [avatar contradiction clause 12793]
12820. ~Q(sK86) <- (1, ~62) [subsumption resolution 4580,3607]
12821. ~59 | ~1 | 62 [avatar split clause 12820,3606,1171,3593]
12850. 59 | ~1 [avatar split clause 3686,1171,3593]
12899. 32 | 33 | 1 [avatar split clause 2977,1171,2988,2984]
12901. ~6 | ~67 [avatar split clause 3728,3641,1267]
13243. DQT(sK86,sK88) <- (3) [resolution 1181,1148]
13245. PR(sK88) <- (14) [resolution 1755,1016]
13247. ~DQT(X0,sK88) | S(sK88) | ::(X0,sdcnt,sK87) | ~PRE(X0,sK87) | ~Q(X0) <- (5) [resolution 1191,1133]
13272. R(sK88) <- (14) [resolution 13245,1010]
13285. AB(sK88) <- (14) [resolution 13272,1012]
13303. 67 | ~14 [avatar split clause 13285,1754,3641]
13435. ::(sK86,mten,sK87) | ::(sK86,imen,sK87) <- (33) [resolution 2990,1125]
13440. 243 <=> ::(sK86,imen,sK87) [avatar definition]
13442. ::(sK86,imen,sK87) <- (243) [avatar component clause 13440]
13444. 244 <=> ::(sK86,mten,sK87) [avatar definition]
13446. ::(sK86,mten,sK87) <- (244) [avatar component clause 13444]
13447. 243 | 244 | ~33 [avatar split clause 13435,2988,13444,13440]
13469. ~AB(sK88) <- (52) [resolution 3557,1020]
13473. 245 <=> ! [X0] : (~DQT(X0,sK88) | ~Q(X0) | ~PRE(X0,sK87) | ::(X0,sdcnt,sK87)) [avatar definition]
13474. ::(X0,sdcnt,sK87) | ~Q(X0) | ~PRE(X0,sK87) | ~DQT(X0,sK88) <- (245) [avatar component clause 13473]
13475. 14 | 245 | ~5 [avatar split clause 13247,1189,13473,1754]
13476. ~67 | ~52 [avatar split clause 13469,3556,3641]
13479. ~Q(sK86) | ~PRE(sK86,sK87) | ~DQT(sK86,sK88) <- (~1, 245) [resolution 13474,1173]
13500. ~PRE(sK86,sK87) | ~DQT(sK86,sK88) <- (~1, 59, 245) [subsumption resolution 13479,3595]
13501. ~PRE(sK86,sK87) <- (~1, 3, 59, 245) [subsumption resolution 13500,13243]
13556. 246 <=> ::(sK86,sreg,sK87) [avatar definition]
13558. ::(sK86,sreg,sK87) <- (246) [avatar component clause 13556]
13560. 247 <=> ::(sK86,site,sK87) [avatar definition]
13562. ::(sK86,site,sK87) <- (247) [avatar component clause 13560]
13564. 248 <=> ::(sK86,cfbnd,sK87) [avatar definition]
13566. ::(sK86,cfbnd,sK87) <- (248) [avatar component clause 13564]
13574. sP2(sK86,sK87) <- (244) [resolution 13446,1104]
13689. PRE(sK86,sK87) <- (244) [resolution 13574,1095]
13701. $false <- (~1, 3, 59, 244, 245) [subsumption resolution 13689,13501]
13702. 1 | ~3 | ~59 | ~244 | ~245 [avatar contradiction clause 13701]
13703. ::(sK86,cfbnd,sK87) | ::(sK86,site,sK87) | ::(sK86,sreg,sK87) <- (243) [resolution 13442,1121]
14110. sP3(sK86,sK87) <- (247) [resolution 13562,1115]
14112. $false <- (~107, 247) [subsumption resolution 14110,5517]
14113. 107 | ~247 [avatar contradiction clause 14112]
14116. 246 | 247 | 248 | ~243 [avatar split clause 13703,13440,13564,13560,13556]
14143. S(sK86) <- (246) [resolution 13558,1118]
14145. $false <- (~47, 246) [subsumption resolution 14143,3367]
14146. 47 | ~246 [avatar contradiction clause 14145]
14199. sP3(sK86,sK87) <- (248) [resolution 13566,1116]
14201. $false <- (~107, 248) [subsumption resolution 14199,5517]
14202. 107 | ~248 [avatar contradiction clause 14201]
14208. PRE(sK86,sK87) <- (32) [resolution 2986,1139]
14210. $false <- (~1, 3, 32, 59, 245) [subsumption resolution 14208,13501]
14211. 1 | ~3 | ~32 | ~59 | ~245 [avatar contradiction clause 14210]
14214. PR(sK86) <- (47) [resolution 3366,1016]
14221. R(sK86) <- (47) [resolution 14214,1010]
14227. AB(sK86) <- (47) [resolution 14221,1012]
14228. $false <- (47, ~66) [subsumption resolution 14227,3636]
14229. ~47 | 66 [avatar contradiction clause 14228]
14235. PRE(sK86,sK87) <- (107) [resolution 5518,1107]
14248. $false <- (~1, 3, 59, 107, 245) [subsumption resolution 14235,13501]
14249. 1 | ~3 | ~59 | ~107 | ~245 [avatar contradiction clause 14248]
14251. $false [avatar sat refutation 1177,1182,1192,3573,3627,3658,3680,3703,12782,12794,12821,12850,12899,12901,
13303,13447,13475,13476,13702,14113,14116,14146,14202,14211,14229,14249]
```

Proof of Theorem (t_{db}69)

```
8. ! [X0] : ((PD(X0) | AB(X0)) => P(X0,X0)) [input]
34. ! [X0,X4] : (PRE(X0,X4) => (T(X4) & (Q(X0) | PD(X0) | ED(X0)))) [input]
122. ! [X0] : ((TR(X0) | PR(X0) | AR(X0)) <=> R(X0)) [input]
123. ! [X0] : (R(X0) => AB(X0)) [input]
128. ! [X0] : (T(X0) => TR(X0)) [input]
159. ! [X0] : (TM(X0) <=> ::(X0,treg,X0)) [input]
169. ! [X0,X4] : (EX(X0,X4) <=> ((T(X4) & ~T(X0) & AB(X0)) | (P(X4,X0) & T(X0)) | PRE(X0,X4))) [input]
174. ! [X0,X4] : (::(X0,proc,X4) <=> (~::(X0,pbnd,X4) & PRE(X0,X4) & PD(X0))) [input]
175. ! [X0,X4] : (::(X0,treg,X4) <=> (EX(X0,X4) & T(X0))) [input]
182. ! [X0,X4] : (::(X0,sreg,X4) <=> (EX(X0,X4) & S(X0))) [input]
```

```
191. ! [X0,X1,X4] : (PTC(X0,X1,X4) <=> (PC(X0,X1,X4) & ::(X1,proc,X4) & ~S(X0) & ::(X0,cnt,X4))) [input]
196. ! [X0,X1,X4] : (PTC(X0,X1,X4) => (TM(X4) & ::(X1,proc,X4) & ~::(X0,sreg,X4) & ::(X0,cnt,X4))) [input]
197. ~! [X0,X1,X4] : (PTC(X0,X1,X4) => (TM(X4) & ::(X1,proc,X4) & ~::(X0,sreg,X4) & ::(X0,cnt,X4))) [negated
       conjecture 196]
207. ! [X0,X1] : (PRE(X0,X1) => (T(X1) & (Q(X0) | PD(X0) | ED(X0)))) [rectify 34]
313. ! [X0,X1] : (EX(X0,X1) <=> ((T(X1) & ~T(X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1))) [rectify 169]
317. ! [X0,X1] : (::(X0,proc,X1) <=> (~::(X0,pbnd,X1) & PRE(X0,X1) & PD(X0))) [rectify 174]
318. ! [X0,X1] : (::(X0,treg,X1) <=> (EX(X0,X1) & T(X0))) [rectify 175]
327. ! [X0,X1] : (::(X0,sreg,X1) <=> (EX(X0,X1) & S(X0))) [rectify 182]
337. ! [X0,X1,X2] : (PTC(X0,X1,X2) <=> (PC(X0,X1,X2) & ::(X1,proc,X2) & ~S(X0) & ::(X0,cnt,X2))) [rectify 191]
344. ~! [X0,X1,X2] : (PTC(X0,X1,X2) => (TM(X2) & ::(X1,proc,X2) & ~::(X0,sreg,X2) & ::(X0,cnt,X2))) [rectify 197]
345. ! [X0,X1,X2] : (PTC(X0,X1,X2) => (PC(X0,X1,X2) & ::(X1,proc,X2) & ~S(X0) & ::(X0,cnt,X2))) [unused predicate
       definition removal 337]
346. ! [X0] : (::(X0,treg,X0) => TM(X0)) [unused predicate definition removal 159]
347. ! [X0] : ((TR(X0) | PR(X0) | AR(X0)) => R(X0)) [unused predicate definition removal 122]
385. ! [X0] : (P(X0,X0) | (~PD(X0) & ~AB(X0))) [ennf transformation 8]
422. ! [X0,X1] : ((T(X1) & (Q(X0) | PD(X0) | ED(X0))) | ~PRE(X0,X1)) [ennf transformation 207]
543. ! [X0] : (R(X0) | (~TR(X0) & ~PR(X0) & ~AR(X0))) [ennf transformation 347]
544. ! [X0] : (AB(X0) | ~R(X0)) [ennf transformation 123]
548. ! [X0] : (TR(X0) | ~T(X0)) [ennf transformation 128]
571. ! [X0] : (TM(X0) | ~::(X0,treg,X0)) [ennf transformation 346]
578. ! [X0,X1,X2] : ((PC(X0,X1,X2) & ::(X1,proc,X2) & ~S(X0) & ::(X0,cnt,X2)) | ~PTC(X0,X1,X2)) [ennf
       transformation 345]
579. ? [X0,X1,X2] : ((~TM(X2) | ~::(X1,proc,X2) | ::(X0,sreg,X2) | ~::(X0,cnt,X2)) & PTC(X0,X1,X2)) [ennf
       transformation 344]
582. ! [X1,X0] : (sP1(X1,X0) <=> ((T(X1) & ~T(X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1))) [predicate
       definition introduction]
583. ! [X0,X1] : (EX(X0,X1) <=> sP1(X1,X0)) [definition folding 313,582]
714. ! [X1,X0] : ((sP1(X1,X0) | ((~T(X1) | T(X0) | ~AB(X0)) & (~P(X1,X0) | ~T(X0)) & ~PRE(X0,X1))) & (((T(X1) & ~T
       (X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1)) | ~sP1(X1,X0))) [nnf transformation 582]
715. ! [X1,X0] : ((sP1(X1,X0) | ((~T(X1) | T(X0) | ~AB(X0)) & (~P(X1,X0) | ~T(X0)) & ~PRE(X0,X1))) & ((T(X1) & ~T(
       X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1) | ~sP1(X1,X0))) [flattening 714]
716. ! [X0,X1] : ((sP1(X0,X1) | ((~T(X0) | T(X1) | ~AB(X1)) & (~P(X0,X1) | ~T(X1)) & ~PRE(X1,X0))) & ((T(X0) & ~T(
       X1) & AB(X1)) | (P(X0,X1) & T(X1)) | PRE(X1,X0) | ~sP1(X0,X1))) [rectify 715]
717. ! [X0,X1] : ((EX(X0,X1) | ~sP1(X1,X0)) & (sP1(X1,X0) | ~EX(X0,X1))) [nnf transformation 583]
731. ! [X0,X1] : ((::(X0,proc,X1) | (::(X0,pbnd,X1) | ~PRE(X0,X1) | ~PD(X0))) & ((~::(X0,pbnd,X1) & PRE(X0,X1) &
       PD(X0)) | ~::(X0,proc,X1))) [nnf transformation 317]
732. ! [X0,X1] : ((::(X0,proc,X1) | ::(X0,pbnd,X1) | ~PRE(X0,X1) | ~PD(X0)) & ((~::(X0,pbnd,X1) & PRE(X0,X1) & PD(
       X0)) | ~::(X0,proc,X1))) [flattening 731]
733. ! [X0,X1] : ((::(X0,treg,X1) | (~EX(X0,X1) | ~T(X0))) & ((EX(X0,X1) & T(X0)) | ~::(X0,treg,X1))) [nnf
       transformation 318]
734. ! [X0,X1] : ((::(X0,treg,X1) | ~EX(X0,X1) | ~T(X0)) & ((EX(X0,X1) & T(X0)) | ~::(X0,treg,X1))) [flattening
       733]
758. ! [X0,X1] : ((::(X0,sreg,X1) | (~EX(X0,X1) | ~S(X0))) & ((EX(X0,X1) & S(X0)) | ~::(X0,sreg,X1))) [nnf
       transformation 327]
759. ! [X0,X1] : ((::(X0,sreg,X1) | ~EX(X0,X1) | ~S(X0)) & ((EX(X0,X1) & S(X0)) | ~::(X0,sreg,X1))) [flattening
       758]
782. ? [X0,X1,X2] : ((~TM(X2) | ~::(X1,proc,X2) | ::(X0,sreg,X2) | ~::(X0,cnt,X2)) & PTC(X0,X1,X2)) => ((~TM(sK87)
       | ~::(sK86,proc,sK87) | ::(sK85,sreg,sK87) | ~::(sK85,cnt,sK87)) & PTC(sK85,sK86,sK87)) [choice axiom]
783. (~TM(sK87) | ~::(sK86,proc,sK87) | ::(sK85,sreg,sK87) | ~::(sK85,cnt,sK87)) & PTC(sK85,sK86,sK87) [
       skolemisation 579,782]
805. P(X0,X0) | ~AB(X0) [cnf transformation 385]
857. ~PRE(X0,X1) | T(X1) [cnf transformation 422]
1005. ~TR(X0) | R(X0) [cnf transformation 543]
1006. ~R(X0) | AB(X0) [cnf transformation 544]
1012. ~T(X0) | TR(X0) [cnf transformation 548]
1037. ~::(X0,treg,X0) | TM(X0) [cnf transformation 571]
1048. ~P(X0,X1) | sP1(X0,X1) | ~T(X1) [cnf transformation 716]
1051. ~sP1(X1,X0) | EX(X0,X1) [cnf transformation 717]
1071. ~::(X0,proc,X1) | PRE(X0,X1) [cnf transformation 732]
1076. ::(X0,treg,X1) | ~EX(X0,X1) | ~T(X0) [cnf transformation 734]
1111. ~::(X0,sreg,X1) | S(X0) [cnf transformation 759]
1140. ~PTC(X0,X1,X2) | ::(X0,cnt,X2) [cnf transformation 578]
1141. ~PTC(X0,X1,X2) | ~S(X0) [cnf transformation 578]
1142. ~PTC(X0,X1,X2) | ::(X1,proc,X2) [cnf transformation 578]
1144. PTC(sK85,sK86,sK87) [cnf transformation 783]
1145. ~TM(sK87) | ~::(sK86,proc,sK87) | ::(sK85,sreg,sK87) | ~::(sK85,cnt,sK87) [cnf transformation 783]
1161. 1 <=> ::(sK85,cnt,sK87) [avatar definition]
1165. 2 <=> ::(sK85,sreg,sK87) [avatar definition]
1167. ::(sK85,sreg,sK87) <- (2) [avatar component clause 1165]
1169. 3 <=> ::(sK86,proc,sK87) [avatar definition]
1170. ::(sK86,proc,sK87) <- (3) [avatar component clause 1169]
1173. 4 <=> TM(sK87) [avatar definition]
1176. ~1 | 2 | ~3 | ~4 [avatar split clause 1145,1173,1169,1165,1161]
1226. ~S(sK85) [resolution 1141,1144]
1389. sP1(X1,X1) | ~T(X1) | ~AB(X1) [resolution 1048,805]
1413. ::(sK85,cnt,sK87) [resolution 1140,1144]
1416. 1 [avatar split clause 1413,1161]
1419. ::(sK86,proc,sK87) [resolution 1142,1144]
1422. 3 [avatar split clause 1419,1169]
1425. PRE(sK86,sK87) <- (3) [resolution 1170,1071]
1429. T(sK87) <- (3) [resolution 1425,857]
1432. TR(sK87) <- (3) [resolution 1429,1012]
1439. R(sK87) <- (3) [resolution 1432,1005]
1445. AB(sK87) <- (3) [resolution 1439,1006]
1635. ~EX(X0,X0) | ~T(X0) | TM(X0) [resolution 1076,1037]
1760. EX(X2,X2) | ~AB(X2) | ~T(X2) [resolution 1389,1051]
1971. ~AB(X1) | ~T(X1) | ~T(X1) | TM(X1) [resolution 1760,1635]
1972. ~T(X1) | ~AB(X1) | TM(X1) [duplicate literal removal 1971]
1990. ~AB(sK87) | TM(sK87) <- (3) [resolution 1972,1429]
2000. TM(sK87) <- (3) [subsumption resolution 1990,1445]
```

```
2003. 4 | ~3 [avatar split clause 2000,1169,1173]
2006. S(sK85) <- (2) [resolution 1167,1111]
2007. $false <- (2) [subsumption resolution 2006,1226]
2008. ~2 [avatar contradiction clause 2007]
2009. $false [avatar sat refutation 1176,1416,1422,2003,2008]
```

Proof of Theorem (t_{db}71)

```
This went KO with both provers
```

Proof of Theorem (t_{db}72)

```
70. ! [X0,X1,X4] : (PC(X0,X1,X4) => (T(X4) & PD(X1) & ED(X0))) [input]
137. ~? [X0] : (Q(X0) & ED(X0)) [input]
178. ! [X0,X4] : (::(X0,sdcnt,X4) <=> (? [X1] : (DQT(X0,X1) & ~S(X1) & ::(X1,idcnt,X4)) & PRE(X0,X4) & Q(X0))) [
       input]
184. ! [X0,X1,X4] : (PTC(X0,X1,X4) <=> (PC(X0,X1,X4) & ::(X1,proc,X4) & ~S(X0) & ::(X0,cnt,X4))) [input]
189. ! [X0,X4,X9] : ((PTC(X0,X9,X4) & ::(X0,sdcnt,X4)) => ? [X1] : ? [X10] : (PTC(X1,X9,X10) & SDEP(X0,X1) & ~::(
       X1,sreg,X10) & ::(X1,idcnt,X10) & tmP(X10,X4))) [input]
190. ~! [X0,X4,X9] : ((PTC(X0,X9,X4) & ::(X0,sdcnt,X4)) => ? [X1] : ? [X10] : (PTC(X1,X9,X10) & SDEP(X0,X1) & ~::(
       X1,sreg,X10) & ::(X1,idcnt,X10) & tmP(X10,X4))) [negated conjecture 189]
242. ! [X0,X1,X2] : (PC(X0,X1,X2) => (T(X2) & PD(X1) & ED(X0))) [rectify 70]
321. ! [X0,X1] : (::(X0,sdcnt,X1) <=> (? [X2] : (DQT(X0,X2) & ~S(X2) & ::(X2,idcnt,X1)) & PRE(X0,X1) & Q(X0))) [
       rectify 178]
328. ! [X0,X1,X2] : (PTC(X0,X1,X2) <=> (PC(X0,X1,X2) & ::(X1,proc,X2) & ~S(X0) & ::(X0,cnt,X2))) [rectify 184]
335. ~! [X0,X1,X2] : ((PTC(X0,X2,X1) & ::(X0,sdcnt,X1)) => ? [X3] : ? [X4] : (PTC(X3,X2,X4) & SDEP(X0,X3) & ~::(X3
       ,sreg,X4) & ::(X3,idcnt,X4) & tmP(X4,X1))) [rectify 190]
336. ~! [X0,X1,X2] : ((PTC(X0,X2,X1) & ::(X0,sdcnt,X1)) => ? [X3,X4] : (PTC(X3,X2,X4) & SDEP(X0,X3) & ~::(X3,sreg,
       X4) & ::(X3,idcnt,X4) & tmP(X4,X1))) [flattening 335]
356. ~! [X0,X1,X2] : ~(PTC(X0,X2,X1) & ::(X0,sdcnt,X1)) [pure predicate removal 336]
372. ! [X0,X1,X2] : (PTC(X0,X1,X2) => (PC(X0,X1,X2) & ::(X1,proc,X2) & ~S(X0) & ::(X0,cnt,X2))) [unused predicate
       definition removal 328]
472. ! [X0,X1,X2] : ((T(X2) & PD(X1) & ED(X0)) | ~PC(X0,X1,X2)) [ennf transformation 242]
550. ! [X0] : (~Q(X0) | ~ED(X0)) [ennf transformation 137]
569. ! [X0,X1,X2] : ((PC(X0,X1,X2) & ::(X1,proc,X2) & ~S(X0) & ::(X0,cnt,X2)) | ~PTC(X0,X1,X2)) [ennf
       transformation 372]
570. ? [X0,X1,X2] : (PTC(X0,X2,X1) & ::(X0,sdcnt,X1)) [ennf transformation 356]
754. ! [X0,X1] : ((::(X0,sdcnt,X1) | (! [X2] : (~DQT(X0,X2) | S(X2) | ~::(X2,idcnt,X1)) | ~PRE(X0,X1) | ~Q(X0))) &
       ((? [X2] : (DQT(X0,X2) & ~S(X2) & ::(X2,idcnt,X1)) & PRE(X0,X1) & Q(X0)) | ~::(X0,sdcnt,X1))) [nnf
       transformation 321]
755. ! [X0,X1] : ((::(X0,sdcnt,X1) | ! [X2] : (~DQT(X0,X2) | S(X2) | ~::(X2,idcnt,X1)) | ~PRE(X0,X1) | ~Q(X0)) &
       ((? [X2] : (DQT(X0,X2) & ~S(X2) & ::(X2,idcnt,X1)) & PRE(X0,X1) & Q(X0)) | ~::(X0,sdcnt,X1))) [flattening
       754]
756. ! [X0,X1] : ((::(X0,sdcnt,X1) | ! [X2] : (~DQT(X0,X2) | S(X2) | ~::(X2,idcnt,X1)) | ~PRE(X0,X1) | ~Q(X0)) &
       ((? [X3] : (DQT(X0,X3) & ~S(X3) & ::(X3,idcnt,X1)) & PRE(X0,X1) & Q(X0)) | ~::(X0,sdcnt,X1))) [rectify 755]
757. ! [X1,X0] : (? [X3] : (DQT(X0,X3) & ~S(X3) & ::(X3,idcnt,X1)) => (DQT(X0,sK80(X0,X1)) & ~S(sK80(X0,X1)) & ::(
       sK80(X0,X1),idcnt,X1))) [choice axiom]
758. ! [X0,X1] : ((::(X0,sdcnt,X1) | ! [X2] : (~DQT(X0,X2) | S(X2) | ~::(X2,idcnt,X1)) | ~PRE(X0,X1) | ~Q(X0)) &
       (((DQT(X0,sK80(X0,X1)) & ~S(sK80(X0,X1)) & ::(sK80(X0,X1),idcnt,X1)) & PRE(X0,X1) & Q(X0)) | ~::(X0,sdcnt,X1
       ))) [skolemisation 756,757]
772. ? [X0,X1,X2] : (PTC(X0,X2,X1) & ::(X0,sdcnt,X1)) => (PTC(sK84,sK86,sK85) & ::(sK84,sdcnt,sK85)) [choice axiom
       ]
773. PTC(sK84,sK86,sK85) & ::(sK84,sdcnt,sK85) [skolemisation 570,772]
918. ~PC(X0,X1,X2) | ED(X0) [cnf transformation 472]
1012. ~ED(X0) | ~Q(X0) [cnf transformation 550]
1108. ~::(X0,sdcnt,X1) | Q(X0) [cnf transformation 758]
1130. ~PTC(X0,X1,X2) | PC(X0,X1,X2) [cnf transformation 569]
1131. ::(sK84,sdcnt,sK85) [cnf transformation 773]
1132. PTC(sK84,sK86,sK85) [cnf transformation 773]
1194. Q(sK84) [resolution 1108,1131]
1411. PC(sK84,sK86,sK85) [resolution 1130,1132]
1436. ED(sK84) [resolution 1411,918]
1449. ~Q(sK84) [resolution 1436,1012]
1465. $false [subsumption resolution 1449,1194]
```

Proof of Theorem (t_{db}74)

```
This went KO with both provers
```

Proof of Theorem (t_{db}75)

```
6. ! [X0,X1] : (P(X0,X1) => (T(X0) <=> T(X1))) [input]
8. ! [X0] : ((PD(X0) | AB(X0)) => P(X0,X0)) [input]
34. ! [X0,X4] : (PRE(X0,X4) => (T(X4) & (Q(X0) | PD(X0) | ED(X0)))) [input]
124. ! [X0] : ((TR(X0) | PR(X0) | AR(X0)) <=> R(X0)) [input]
125. ! [X0] : (R(X0) => AB(X0)) [input]
130. ! [X0] : (T(X0) => TR(X0)) [input]
163. ! [X0,X1] : (oP(X0,X1) <=> (((T(X1) & T(X0)) | (PD(X1) & PD(X0))) & P(X0,X1))) [input]
164. ! [X0,X1] : (tmP(X0,X1) <=> (! [X2] : ((! [X4] : (EX(X2,X4) => EX(X0,X4)) & oP(X2,X1)) => oP(X2,X0)) & oP(X0,
       X1))) [input]
180. ! [X0,X4] : (::(X0,gdcnt,X4) <=> (! [X5] : (PRE(X0,X5) => ? [X1] : CONCR(X1,X0,X5)) & PRE(X0,X4))) [input]
189. ! [X0,X4] : (::(X0,gdcnt,X4) => ? [X1] : ? [X9] : (CONCR(X1,X0,X9) & tmP(X9,X4))) [input]
190. ~! [X0,X4] : (::(X0,gdcnt,X4) => ? [X1] : ? [X9] : (CONCR(X1,X0,X9) & tmP(X9,X4))) [negated conjecture 189]
200. ! [X0,X1] : (PRE(X0,X1) => (T(X1) & (Q(X0) | PD(X0) | ED(X0)))) [rectify 34]
305. ! [X0,X1] : (tmP(X0,X1) <=> (! [X2] : ((! [X3] : (EX(X2,X3) => EX(X0,X3)) & oP(X2,X1)) => oP(X2,X0)) & oP(X0,
       X1))) [rectify 164]
```

```
323. ! [X0,X1] : (::(X0,gdcnt,X1) <=> (! [X2] : (PRE(X0,X2) => ? [X3] : CONCR(X3,X0,X2)) & PRE(X0,X1))) [rectify
     180]
335. ~! [X0,X1] : (::(X0,gdcnt,X1) => ? [X2] : ? [X3] : (CONCR(X2,X0,X3) & tmP(X3,X1))) [rectify 190]
336. ~! [X0,X1] : (::(X0,gdcnt,X1) => ? [X2,X3] : (CONCR(X2,X0,X3) & tmP(X3,X1))) [flattening 335]
337. ! [X0] : ((TR(X0) | PR(X0) | AR(X0)) => R(X0)) [unused predicate definition removal 124]
373. ! [X0,X1] : ((T(X0) <=> T(X1)) | ~P(X0,X1)) [ennf transformation 6]
375. ! [X0] : (P(X0,X0) | (~PD(X0) & ~AB(X0))) [ennf transformation 8]
412. ! [X0,X1] : ((T(X1) & (Q(X0) | PD(X0) | ED(X0))) | ~PRE(X0,X1)) [ennf transformation 200]
534. ! [X0] : (R(X0) | (~TR(X0) & ~PR(X0) & ~AR(X0))) [ennf transformation 337]
535. ! [X0] : (AB(X0) | ~R(X0)) [ennf transformation 125]
539. ! [X0] : (TR(X0) | ~T(X0)) [ennf transformation 130]
559. ! [X0,X1] : (tmP(X0,X1) <=> (! [X2] : (oP(X2,X0) | (? [X3] : (~EX(X0,X3) & EX(X2,X3)) | ~oP(X2,X1))) & oP(X0,
     X1))) [ennf transformation 305]
560. ! [X0,X1] : (tmP(X0,X1) <=> (! [X2] : (oP(X2,X0) | ? [X3] : (~EX(X0,X3) & EX(X2,X3)) | ~oP(X2,X1)) & oP(X0,X1
     ))) [flattening 559]
564. ! [X0,X1] : (::(X0,gdcnt,X1) <=> (! [X2] : (? [X3] : CONCR(X3,X0,X2) | ~PRE(X0,X2)) & PRE(X0,X1))) [ennf
     transformation 323]
565. ? [X0,X1] : (! [X2,X3] : (~CONCR(X2,X0,X3) | ~tmP(X3,X1)) & ::(X0,gdcnt,X1)) [ennf transformation 336]
589. ! [X0,X1] : (((T(X0) | ~T(X1)) & (T(X1) | ~T(X0))) | ~P(X0,X1)) [nnf transformation 373]
703. ! [X0,X1] : ((oP(X0,X1) | (((~T(X1) | ~T(X0)) & (~PD(X1) | ~PD(X0))) | ~P(X0,X1))) & ((((T(X1) & T(X0)) | (PD
     (X1) & PD(X0))) & P(X0,X1)) | ~oP(X0,X1))) [nnf transformation 163]
704. ! [X0,X1] : ((oP(X0,X1) | (((~T(X1) | ~T(X0)) & (~PD(X1) | ~PD(X0))) | ~P(X0,X1)) & ((((T(X1) & T(X0)) | (PD(
     X1) & PD(X0))) & P(X0,X1)) | ~oP(X0,X1))) [flattening 703]
705. ! [X0,X1] : ((tmP(X0,X1) | (? [X2] : (~oP(X2,X0) & ! [X3] : (EX(X0,X3) | ~EX(X2,X3)) & oP(X2,X1)) | ~oP(X0,X1
     ))) & ((! [X2] : (oP(X2,X0) | ? [X3] : (~EX(X0,X3) & EX(X2,X3)) | ~oP(X2,X1)) & oP(X0,X1)) | ~tmP(X0,X1))) [
     nnf transformation 560]
706. ! [X0,X1] : ((tmP(X0,X1) | ? [X2] : (~oP(X2,X0) & ! [X3] : (EX(X0,X3) | ~EX(X2,X3)) & oP(X2,X1)) | ~oP(X0,X1)
     ) & ((! [X2] : (oP(X2,X0) | ? [X3] : (~EX(X0,X3) & EX(X2,X3)) | ~oP(X2,X1)) & oP(X0,X1)) | ~tmP(X0,X1))) [
     flattening 705]
707. ! [X0,X1] : ((tmP(X0,X1) | ? [X2] : (~oP(X2,X0) & ! [X3] : (EX(X0,X3) | ~EX(X2,X3)) & oP(X2,X1)) | ~oP(X0,X1)
     ) & ((! [X4] : (oP(X4,X0) | ? [X5] : (~EX(X0,X5) & EX(X4,X5)) | ~oP(X4,X1)) & oP(X0,X1)) | ~tmP(X0,X1))) [
     rectify 706]
708. ! [X1,X0] : (? [X2] : (~oP(X2,X0) & ! [X3] : (EX(X0,X3) | ~EX(X2,X3)) & oP(X2,X1)) => (~oP(sK67(X0,X1),X0) &
     ! [X3] : (EX(X0,X3) | ~EX(sK67(X0,X1),X3)) & oP(sK67(X0,X1),X1))) [choice axiom]
709. ! [X4,X0] : (? [X5] : (~EX(X0,X5) & EX(X4,X5)) => (~EX(X0,sK68(X0,X4)) & EX(X4,sK68(X0,X4)))) [choice axiom]
710. ! [X0,X1] : ((tmP(X0,X1) | (~oP(sK67(X0,X1),X0) & ! [X3] : (EX(X0,X3) | ~EX(sK67(X0,X1),X3)) & oP(sK67(X0,X1)
     ,X1)) | ~oP(X0,X1)) & ((! [X4] : (oP(X4,X0) | (~EX(X0,sK68(X0,X4)) & EX(X4,sK68(X0,X4)))) | ~oP(X4,X1)) & oP(
     X0,X1)) | ~tmP(X0,X1))) [skolemisation 707,709,708]
759. ! [X0,X1] : ((::(X0,gdcnt,X1) | (? [X2] : (! [X3] : ~CONCR(X3,X0,X2) & PRE(X0,X2)) | ~PRE(X0,X1))) & ((! [X2]
     : (? [X3] : CONCR(X3,X0,X2) | ~PRE(X0,X2)) & PRE(X0,X1)) | ~::(X0,gdcnt,X1))) [nnf transformation 564]
760. ! [X0,X1] : ((::(X0,gdcnt,X1) | ? [X2] : (! [X3] : ~CONCR(X3,X0,X2) & PRE(X0,X2)) | ~PRE(X0,X1)) & ((! [X2] :
     (? [X3] : CONCR(X3,X0,X2) | ~PRE(X0,X2)) & PRE(X0,X1)) | ~::(X0,gdcnt,X1))) [flattening 759]
761. ! [X0,X1] : ((::(X0,gdcnt,X1) | ? [X2] : (! [X3] : ~CONCR(X3,X0,X2) & PRE(X0,X2)) | ~PRE(X0,X1)) & ((! [X4] :
     (? [X5] : CONCR(X5,X0,X4) | ~PRE(X0,X4)) & PRE(X0,X1)) | ~::(X0,gdcnt,X1))) [rectify 760]
762. ! [X0] : (? [X2] : (! [X3] : ~CONCR(X3,X0,X2) & PRE(X0,X2)) => (! [X3] : ~CONCR(X3,X0,sK82(X0)) & PRE(X0,sK82
     (X0)))) [choice axiom]
763. ! [X4,X0] : (? [X5] : CONCR(X5,X0,X4) => CONCR(sK83(X0,X4),X0,X4)) [choice axiom]
764. ! [X0,X1] : ((::(X0,gdcnt,X1) | (! [X3] : ~CONCR(X3,X0,sK82(X0)) & PRE(X0,sK82(X0)))) | ~PRE(X0,X1)) & ((! [X4
     ] : (CONCR(sK83(X0,X4),X0,X4) | ~PRE(X0,X4)) & PRE(X0,X1)) | ~::(X0,gdcnt,X1))) [skolemisation 761,763,762]
767. ? [X0,X1] : (! [X2,X3] : (~CONCR(X2,X0,X3) | ~tmP(X3,X1)) & ::(X0,gdcnt,X1)) => (! [X3,X2] : (~CONCR(X2,sK84,
     X3) | ~tmP(X3,sK85)) & ::(sK84,gdcnt,sK85)) [choice axiom]
768. ! [X2,X3] : (~CONCR(X2,sK84,X3) | ~tmP(X3,sK85)) & ::(sK84,gdcnt,sK85) [skolemisation 565,767]
787. ~P(X0,X1) | ~T(X1) | T(X0) [cnf transformation 589]
790. P(X0,X0) | ~AB(X0) [cnf transformation 375]
842. ~PRE(X0,X1) | T(X1) [cnf transformation 412]
993. ~TR(X0) | R(X0) [cnf transformation 534]
994. ~R(X0) | AB(X0) [cnf transformation 535]
1000. ~T(X0) | TR(X0) [cnf transformation 539]
1040. oP(X0,X1) | ~T(X1) | ~T(X0) | ~P(X0,X1) [cnf transformation 704]
1044. oP(sK67(X0,X1),X1) | tmP(X0,X1) | ~oP(X0,X1) [cnf transformation 710]
1046. ~oP(sK67(X0,X1),X0) | tmP(X0,X1) | ~oP(X0,X1) [cnf transformation 710]
1114. ~::(X0,gdcnt,X1) | PRE(X0,X1) [cnf transformation 764]
1115. ~::(X0,gdcnt,X1) | ~PRE(X0,X4) | CONCR(sK83(X0,X4),X0,X4) [cnf transformation 764]
1122. ::(sK84,gdcnt,sK85) [cnf transformation 768]
1123. ~CONCR(X2,sK84,X3) | ~tmP(X3,sK85) [cnf transformation 768]
1131. ~P(X0,X1) | ~T(X1) | oP(X0,X1) [subsumption resolution 1040,787]
1248. PRE(sK84,sK85) [resolution 1114,1122]
1251. T(sK85) [resolution 1248,842]
1261. TR(sK85) [resolution 1251,1000]
1271. R(sK85) [resolution 1261,993]
1274. AB(sK85) [resolution 1271,994]
1373. oP(X1,X1) | ~T(X1) | ~AB(X1) [resolution 1131,790]
1957. tmP(X0,X0) | ~oP(X0,X0) | tmP(X0,X0) | ~oP(X0,X0) [resolution 1046,1044]
1958. ~oP(X0,X0) | tmP(X0,X0) [duplicate literal removal 1957]
1960. tmP(X1,X1) | ~T(X1) | ~AB(X1) [resolution 1958,1373]
2393. CONCR(sK83(sK84,X0),sK84,X0) | ~PRE(sK84,X0) [resolution 1115,1122]
2886. ~tmP(X0,sK85) | ~PRE(sK84,X0) [resolution 2393,1123]
3180. ~PRE(sK84,sK85) | ~T(sK85) | ~AB(sK85) [resolution 2886,1960]
3183. ~T(sK85) | ~AB(sK85) [subsumption resolution 3180,1248]
3184. ~AB(sK85) [subsumption resolution 3183,1251]
3185. $false [subsumption resolution 3184,1274]
```

Proof of Theorem (t_{db} 76)

This went KO with both provers

Proof of Theorem (t_{db} 77)

This went KO with both provers

Proof of Theorem (t_{db}78)

```
This went KO with both provers
```

Proof of Theorem (t_{db}86)

```
This went KO with both provers
```

Proof of Theorem (t_{db}87)

```
This went KO with both provers
```

Proof of Theorem (t_{db}88)

```
182. ! [X0,X4] : (::(X0,sreg,X4) <=> (EX(X0,X4) & S(X0))) [input]
196. ! [X0,X4,X6] : ((EX(X0,X6) & ::(X0,sreg,X4)) => ::(X0,sreg,X6)) [input]
197. ~! [X0,X4,X6] : ((EX(X0,X6) & ::(X0,sreg,X4)) => ::(X0,sreg,X6)) [negated conjecture 196]
327. ! [X0,X1] : (::(X0,sreg,X1) <=> (EX(X0,X1) & S(X0))) [rectify 182]
344. ~! [X0,X1,X2] : ((EX(X0,X2) & ::(X0,sreg,X1)) => ::(X0,sreg,X2)) [rectify 197]
575. ? [X0,X1,X2] : (~::(X0,sreg,X2) & (EX(X0,X2) & ::(X0,sreg,X1))) [ennf transformation 344]
576. ? [X0,X1,X2] : (~::(X0,sreg,X2) & EX(X0,X2) & ::(X0,sreg,X1)) [flattening 575]
755. ! [X0,X1] : ((::(X0,sreg,X1) | (~EX(X0,X1) | ~S(X0))) & ((EX(X0,X1) & S(X0)) | ~::(X0,sreg,X1))) [nnf
     transformation 327]
756. ! [X0,X1] : ((::(X0,sreg,X1) | ~EX(X0,X1) | ~S(X0)) & ((EX(X0,X1) & S(X0)) | ~::(X0,sreg,X1))) [flattening
     755]
779. ? [X0,X1,X2] : (~::(X0,sreg,X2) & EX(X0,X2) & ::(X0,sreg,X1)) => (~::(sK85,sreg,sK87) & EX(sK85,sK87) & ::(
     sK85,sreg,sK86)) [choice axiom]
780. ~::(sK85,sreg,sK87) & EX(sK85,sK87) & ::(sK85,sreg,sK86) [skolemisation 576,779]
1107. ~::(X0,sreg,X1) | S(X0) [cnf transformation 756]
1109. ::(X0,sreg,X1) | ~EX(X0,X1) | ~S(X0) [cnf transformation 756]
1136. ::(sK85,sreg,sK86) [cnf transformation 780]
1137. EX(sK85,sK87) [cnf transformation 780]
1138. ~::(sK85,sreg,sK87) [cnf transformation 780]
1201. S(sK85) [resolution 1107,1136]
1621. ~EX(sK85,sK87) | ~S(sK85) [resolution 1109,1138]
1625. ~S(sK85) [subsumption resolution 1621,1137]
1626. $false [subsumption resolution 1625,1201]
```

Proof of Theorem (t_{db}89)

```
This went KO with both provers
```

Proof of Theorem (t_{db}90)

```
114. ! [X0] : ((PED(X0) | NPED(X0) | AS(X0)) <=> ED(X0)) [input]
122. ! [X0] : ((TR(X0) | PR(X0) | AR(X0)) <=> R(X0)) [input]
123. ! [X0] : (R(X0) => AB(X0)) [input]
128. ! [X0] : (T(X0) => TR(X0)) [input]
130. ~? [X0] : (ED(X0) & AB(X0)) [input]
169. ! [X0,X4] : (EX(X0,X4) <=> ((T(X4) & ~T(X0) & AB(X0)) | (P(X4,X0) & T(X0)) | PRE(X0,X4))) [input]
186. ! [X0,X1,X4] : (CONCR(X0,X1,X4) <=> (exD(X1,X0,X4) & ~? [X3] : SLC(X1,X3,X4) & NPED(X1) & ::(X0,sdcnt,X4))) [
     input]
187. ! [X0,X4] : (::(X0,gdcnt,X4) <=> (! [X5] : (PRE(X0,X5) => ? [X1] : CONCR(X1,X0,X5)) & PRE(X0,X4))) [input]
196. ! [X0,X4,X6] : ((EX(X0,X6) & ::(X0,gdcnt,X4)) => ::(X0,gdcnt,X6)) [input]
197. ~! [X0,X4,X6] : ((EX(X0,X6) & ::(X0,gdcnt,X4)) => ::(X0,gdcnt,X6)) [negated conjecture 196]
313. ! [X0,X1] : (EX(X0,X1) <=> ((T(X1) & ~T(X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1))) [rectify 169]
331. ! [X0,X1,X2] : (CONCR(X0,X1,X2) <=> (exD(X1,X0,X2) & ~? [X3] : SLC(X1,X3,X2) & NPED(X1) & ::(X0,sdcnt,X2))) [
     rectify 186]
332. ! [X0,X1] : (::(X0,gdcnt,X1) <=> (! [X2] : (PRE(X0,X2) => ? [X3] : CONCR(X3,X0,X2)) & PRE(X0,X1))) [rectify
     187]
344. ~! [X0,X1,X2] : ((EX(X0,X2) & ::(X0,gdcnt,X1)) => ::(X0,gdcnt,X2)) [rectify 197]
345. ! [X0] : ((TR(X0) | PR(X0) | AR(X0)) => R(X0)) [unused predicate definition removal 122]
541. ! [X0] : (R(X0) | (~TR(X0) & ~PR(X0) & ~AR(X0))) [ennf transformation 345]
542. ! [X0] : (AB(X0) | ~R(X0)) [ennf transformation 123]
546. ! [X0] : (TR(X0) | ~T(X0)) [ennf transformation 128]
548. ! [X0] : (~ED(X0) | ~AB(X0)) [ennf transformation 130]
573. ! [X0,X1,X2] : (CONCR(X0,X1,X2) <=> (exD(X1,X0,X2) & ! [X3] : ~SLC(X1,X3,X2) & NPED(X1) & ::(X0,sdcnt,X2))) [
     ennf transformation 331]
574. ! [X0,X1] : (::(X0,gdcnt,X1) <=> (! [X2] : (? [X3] : CONCR(X3,X0,X2) | ~PRE(X0,X2)) & PRE(X0,X1))) [ennf
     transformation 332]
575. ? [X0,X1,X2] : (~::(X0,gdcnt,X2) & (EX(X0,X2) & ::(X0,gdcnt,X1))) [ennf transformation 344]
576. ? [X0,X1,X2] : (~::(X0,gdcnt,X2) & EX(X0,X2) & ::(X0,gdcnt,X1)) [flattening 575]
579. ! [X1,X0] : (sP1(X1,X0) <=> ((T(X1) & ~T(X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1))) [predicate
     definition introduction]
580. ! [X0,X1] : (EX(X0,X1) <=> sP1(X1,X0)) [definition folding 313,579]
701. ! [X0] : (((PED(X0) | NPED(X0) | AS(X0)) | ~ED(X0)) & (ED(X0) | (~PED(X0) & ~NPED(X0) & ~AS(X0)))) [nnf
     transformation 114]
702. ! [X0] : ((PED(X0) | NPED(X0) | AS(X0) | ~ED(X0)) & (ED(X0) | (~PED(X0) & ~NPED(X0) & ~AS(X0)))) [flattening
     701]
711. ! [X1,X0] : ((sP1(X1,X0) | ((~T(X1) | T(X0) | ~AB(X0)) & (~P(X1,X0) | ~T(X0)) & ~PRE(X0,X1))) & (((T(X1) & ~T
     (X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1)) | ~sP1(X1,X0))) [nnf transformation 579]
712. ! [X1,X0] : ((sP1(X1,X0) | ((~T(X1) | T(X0) | ~AB(X0)) & (~P(X1,X0) | ~T(X0)) & ~PRE(X0,X1))) & ((T(X1) & ~T(
     X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1) | ~sP1(X1,X0))) [flattening 711]
713. ! [X0,X1] : ((sP1(X0,X1) | ((~T(X0) | T(X1) | ~AB(X1)) & (~P(X0,X1) | ~T(X1)) & ~PRE(X1,X0))) & ((T(X0) & ~T(
     X1) & AB(X1)) | (P(X0,X1) & T(X1)) | PRE(X1,X0) | ~sP1(X0,X1))) [rectify 712]
714. ! [X0,X1] : ((EX(X0,X1) | ~sP1(X1,X0)) & (sP1(X1,X0) | ~EX(X0,X1))) [nnf transformation 580]
```

```
766. ! [X0,X1,X2] : ((CONCR(X0,X1,X2) | (~exD(X1,X0,X2) | ? [X3] : SLC(X1,X3,X2) | ~NPED(X1) | ~::(X0,sdcnt,X2))) & ((exD(X1,X0,X2) & ! [X3] : ~SLC(X1,X3,X2) & NPED(X1) & ::(X0,sdcnt,X2)) | ~CONCR(X0,X1,X2))) [nnf transformation 573]
767. ! [X0,X1,X2] : ((CONCR(X0,X1,X2) | ~exD(X1,X0,X2) | ? [X3] : SLC(X1,X3,X2) | ~NPED(X1) | ~::(X0,sdcnt,X2)) & ((exD(X1,X0,X2) & ! [X3] : ~SLC(X1,X3,X2) & NPED(X1) & ::(X0,sdcnt,X2)) | ~CONCR(X0,X1,X2))) [flattening 766]
768. ! [X0,X1,X2] : ((CONCR(X0,X1,X2) | ~exD(X1,X0,X2) | ? [X3] : SLC(X1,X3,X2) | ~NPED(X1) | ~::(X0,sdcnt,X2)) & ((exD(X1,X0,X2) & ! [X4] : ~SLC(X1,X4,X2) & NPED(X1) & ::(X0,sdcnt,X2)) | ~CONCR(X0,X1,X2))) [rectify 767]
769. ! [X2,X1] : (? [X3] : SLC(X1,X3,X2) => SLC(X1,sK82(X1,X2),X2)) [choice axiom]
770. ! [X0,X1,X2] : ((CONCR(X0,X1,X2) | ~exD(X1,X0,X2) | SLC(X1,sK82(X1,X2),X2) | ~NPED(X1) | ~::(X0,sdcnt,X2)) & ((exD(X1,X0,X2) & ! [X4] : ~SLC(X1,X4,X2) & NPED(X1) & ::(X0,sdcnt,X2)) | ~CONCR(X0,X1,X2))) [skolemisation 768,769]
771. ! [X0,X1] : ((::(X0,gdcnt,X1) | (? [X2] : (! [X3] : ~CONCR(X3,X0,X2) & PRE(X0,X2)) | ~PRE(X0,X1))) & ((! [X2] : (? [X3] : CONCR(X3,X0,X2) | ~PRE(X0,X2)) & PRE(X0,X1)) | ~::(X0,gdcnt,X1))) [nnf transformation 574]
772. ! [X0,X1] : ((::(X0,gdcnt,X1) | ? [X2] : (! [X3] : ~CONCR(X3,X0,X2) & PRE(X0,X2)) | ~PRE(X0,X1)) & ((! [X2] : (? [X3] : CONCR(X3,X0,X2) | ~PRE(X0,X2)) & PRE(X0,X1)) | ~::(X0,gdcnt,X1))) [flattening 771]
773. ! [X0,X1] : ((::(X0,gdcnt,X1) | ? [X2] : (! [X3] : ~CONCR(X3,X0,X2) & PRE(X0,X2)) | ~PRE(X0,X1)) & ((! [X4] : (? [X5] : CONCR(X5,X0,X4) | ~PRE(X0,X4)) & PRE(X0,X1)) | ~::(X0,gdcnt,X1))) [rectify 772]
774. ! [X0] : (? [X2] : (! [X3] : ~CONCR(X3,X0,X2) & PRE(X0,X2)) => (! [X3] : ~CONCR(X3,X0,sK83(X0)) & PRE(X0,sK83(X0)))) [choice axiom]
775. ! [X4,X0] : (? [X5] : CONCR(X5,X0,X4) => CONCR(sK84(X0,X4),X0,X4)) [choice axiom]
776. ! [X0,X1] : ((::(X0,gdcnt,X1) | (! [X3] : ~CONCR(X3,X0,sK83(X0)) & PRE(X0,sK83(X0))) | ~PRE(X0,X1)) & ((! [X4] : (CONCR(sK84(X0,X4),X0,X4) | ~PRE(X0,X4)) & PRE(X0,X1)) | ~::(X0,gdcnt,X1))) [skolemisation 773,775,774]
779. ? [X0,X1,X2] : (~::(X0,gdcnt,X2) & EX(X0,X2) & ::(X0,gdcnt,X1)) => (~::(sK85,gdcnt,sK87) & EX(sK85,sK87) & ::(sK85,gdcnt,sK86)) [choice axiom]
780. ~::(sK85,gdcnt,sK87) & EX(sK85,sK87) & ::(sK85,gdcnt,sK86) [skolemisation 576,779]
992. ~NPED(X0) | ED(X0) [cnf transformation 702]
1002. ~TR(X0) | R(X0) [cnf transformation 541]
1003. ~R(X0) | AB(X0) [cnf transformation 542]
1009. ~T(X0) | TR(X0) [cnf transformation 546]
1011. ~ED(X0) | ~AB(X0) [cnf transformation 548]
1037. ~sP1(X0,X1) | T(X1) | PRE(X1,X0) | AB(X1) [cnf transformation 713]
1046. ~EX(X0,X1) | sP1(X1,X0) [cnf transformation 714]
1124. ~CONCR(X0,X1,X2) | NPED(X1) [cnf transformation 770]
1128. ~::(X0,gdcnt,X1) | PRE(X0,X1) [cnf transformation 776]
1129. ~::(X0,gdcnt,X1) | ~PRE(X0,X4) | CONCR(sK84(X0,X4),X0,X4) [cnf transformation 776]
1130. ::(X0,gdcnt,X1) | PRE(X0,sK83(X0)) | ~PRE(X0,X1) [cnf transformation 776]
1131. ~CONCR(X3,X0,sK83(X0)) | ::(X0,gdcnt,X1) | ~PRE(X0,X1) [cnf transformation 776]
1136. ::(sK85,gdcnt,sK86) [cnf transformation 780]
1137. EX(sK85,sK87) [cnf transformation 780]
1138. ~::(sK85,gdcnt,sK87) [cnf transformation 780]
1191. sP1(sK87,sK85) [resolution 1046,1137]
1265. PRE(sK85,sK86) [resolution 1128,1136]
1273. 2 <=> T(sK85) [avatar definition]
1275. T(sK85) <- (2) [avatar component clause 1273]
1516. 5 <=> ED(sK85) [avatar definition]
1517. ~ED(sK85) <- (~5) [avatar component clause 1516]
1518. ED(sK85) <- (5) [avatar component clause 1516]
1573. 12 <=> NPED(sK85) [avatar definition]
1575. NPED(sK85) <- (12) [avatar component clause 1573]
1870. T(sK85) | PRE(sK85,sK87) | AB(sK85) [resolution 1037,1191]
2038. PRE(sK85,sK83(sK85)) | ~PRE(sK85,sK87) [resolution 1130,1138]
2066. 15 <=> PRE(sK85,sK87) [avatar definition]
2067. PRE(sK85,sK87) <- (15) [avatar component clause 2066]
2070. 16 <=> PRE(sK85,sK83(sK85)) [avatar definition]
2072. PRE(sK85,sK83(sK85)) <- (16) [avatar component clause 2070]
2073. ~15 | 16 [avatar split clause 2038,2070,2066]
2080. 17 <=> AB(sK85) [avatar definition]
2081. ~AB(sK85) <- (~17) [avatar component clause 2080]
2083. 17 | 15 | 2 [avatar split clause 1870,1273,2066,2080]
2090. TR(sK85) <- (2) [resolution 1275,1009]
2098. ~AB(sK85) <- (5) [resolution 1518,1011]
2099. ~17 | ~5 [avatar split clause 2098,1516,2080]
2119. R(sK85) <- (2) [resolution 2090,1002]
2120. AB(sK85) <- (2) [resolution 2119,1003]
2121. $false <- (2, ~17) [subsumption resolution 2120,2081]
2122. ~2 | 17 [avatar contradiction clause 2121]
2446. CONCR(sK84(sK85,X0),sK85,X0) | ~PRE(sK85,X0) [resolution 1129,1136]
3290. ~PRE(sK85,X4) | NPED(sK85) [resolution 2446,1124]
3291. ~PRE(sK85,sK83(sK85)) | ::(sK85,gdcnt,X5) | ~PRE(sK85,X5) [resolution 2446,1131]
3294. 59 <=> ! [X4] : ~PRE(sK85,X4) [avatar definition]
3295. ~PRE(sK85,X4) <- (59) [avatar component clause 3294]
3296. 12 | 59 [avatar split clause 3290,3294,1573]
3297. ::(sK85,gdcnt,X5) | ~PRE(sK85,X5) <- (16) [subsumption resolution 3291,2072]
3299. ED(sK85) <- (12) [resolution 1575,992]
3300. $false <- (~5, 12) [subsumption resolution 3299,1517]
3301. 5 | ~12 [avatar contradiction clause 3300]
3305. $false <- (59) [resolution 3295,1265]
3316. ~59 [avatar contradiction clause 3305]
3884. ~PRE(sK85,sK87) <- (16) [resolution 3297,1138]
3888. $false <- (15, 16) [subsumption resolution 3884,2067]
3889. ~15 | ~16 [avatar contradiction clause 3888]
3890. $false [avatar sat refutation 2073,2083,2099,2122,3296,3301,3316,3889]
```

Proof of Theorem (t_{ab} 102)

```
6. ! [X0,X1] : (P(X0,X1) => (T(X0) <=> T(X1))) [input]
34. ! [X0,X4] : (PRE(X0,X4) => (T(X4) & (Q(X0) | PD(X0) | ED(X0)))) [input]
169. ! [X0,X4] : (EX(X0,X4) <=> ((T(X4) & ~T(X0) & AB(X0)) | (P(X4,X0) & T(X0)) | PRE(X0,X4))) [input]
196. ! [X0,X4] : (EX(X0,X4) => T(X4)) [input]
```

```
197. ~! [X0,X4] : (EX(X0,X4) => T(X4)) [negated conjecture 196]
207. ! [X0,X1] : (PRE(X0,X1) => (T(X1) & (Q(X0) | PD(X0) | ED(X0)))) [rectify 34]
313. ! [X0,X1] : (EX(X0,X1) <=> ((T(X1) & ~T(X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1))) [rectify 169]
344. ~! [X0,X1] : (EX(X0,X1) => T(X1)) [rectify 197]
381. ! [X0,X1] : ((T(X0) <=> T(X1)) | ~P(X0,X1)) [ennf transformation 6]
420. ! [X0,X1] : ((T(X1) & (Q(X0) | PD(X0) | ED(X0))) | ~PRE(X0,X1)) [ennf transformation 207]
575. ? [X0,X1] : (~T(X1) & EX(X0,X1)) [ennf transformation 344]
578. ! [X1,X0] : (sP1(X1,X0) <=> ((T(X1) & ~T(X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1))) [predicate
     definition introduction]
579. ! [X0,X1] : (EX(X0,X1) <=> sP1(X1,X0)) [definition folding 313,578]
599. ! [X0,X1] : (((T(X0) | ~T(X1)) & (T(X1) | ~T(X0))) | ~P(X0,X1)) [nnf transformation 381]
710. ! [X1,X0] : ((sP1(X1,X0) | ((~T(X1) | T(X0) | ~AB(X0)) & (~P(X1,X0) | ~T(X0)) & ~PRE(X0,X1))) & (((T(X1) & ~T
     (X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1)) | ~sP1(X1,X0))) [nnf transformation 578]
711. ! [X1,X0] : ((sP1(X1,X0) | ((~T(X1) | T(X0) | ~AB(X0)) & (~P(X1,X0) | ~T(X0)) & ~PRE(X0,X1))) & ((T(X1) & ~T(
     X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1) | ~sP1(X1,X0))) [flattening 710]
712. ! [X0,X1] : ((sP1(X0,X1) | ((~T(X0) | T(X1) | ~AB(X1)) & (~P(X0,X1) | ~T(X1)) & ~PRE(X1,X0))) & ((T(X0) & ~T(
     X1) & AB(X1)) | (P(X0,X1) & T(X1)) | PRE(X1,X0) | ~sP1(X0,X1))) [rectify 711]
713. ! [X0,X1] : ((EX(X0,X1) | ~sP1(X1,X0)) & (sP1(X1,X0) | ~EX(X0,X1))) [nnf transformation 579]
778. ? [X0,X1] : (~T(X1) & EX(X0,X1)) => (~T(sK86) & EX(sK85,sK86)) [choice axiom]
779. ~T(sK86) & EX(sK85,sK86) [skolemisation 575,778]
798. ~P(X0,X1) | ~T(X1) | T(X0) [cnf transformation 599]
853. ~PRE(X0,X1) | T(X1) [cnf transformation 420]
1040. T(X0) | T(X1) | PRE(X1,X0) | ~sP1(X0,X1) [cnf transformation 712]
1041. T(X0) | P(X0,X1) | PRE(X1,X0) | ~sP1(X0,X1) [cnf transformation 712]
1045. ~EX(X0,X1) | sP1(X1,X0) [cnf transformation 713]
1135. EX(sK85,sK86) [cnf transformation 779]
1136. ~T(sK86) [cnf transformation 779]
1142. ~sP1(X0,X1) | P(X0,X1) | T(X0) [subsumption resolution 1041,853]
1143. ~sP1(X0,X1) | T(X1) | T(X0) [subsumption resolution 1040,853]
1190. sP1(sK86,sK85) [resolution 1045,1135]
1263. T(sK85) | T(sK86) [resolution 1143,1190]
1265. T(sK85) [subsumption resolution 1263,1136]
1390. P(sK86,sK85) | T(sK86) [resolution 1142,1190]
1392. P(sK86,sK85) [subsumption resolution 1390,1136]
1399. ~T(sK85) | T(sK86) [resolution 1392,798]
1414. T(sK86) [subsumption resolution 1399,1265]
1415. $false [subsumption resolution 1414,1136]
```

Proof of Theorem (t_{ab} 103)

```
6. ! [X0,X1] : (P(X0,X1) => (T(X0) <=> T(X1))) [input]
10. ! [X0,X1,X2] : ((P(X1,X2) & P(X0,X1)) => P(X0,X2)) [input]
34. ! [X0,X4] : (PRE(X0,X4) => (T(X4) & (Q(X0) | PD(X0) | ED(X0)))) [input]
35. ! [X0,X4,X3] : ((P(X3,X4) & PRE(X0,X4)) => PRE(X0,X3)) [input]
169. ! [X0,X4] : (EX(X0,X4) <=> ((T(X4) & ~T(X0) & AB(X0)) | (P(X4,X0) & T(X0)) | PRE(X0,X4))) [input]
196. ! [X0,X4,X5] : ((P(X5,X4) & EX(X0,X4)) => EX(X0,X5)) [input]
197. ~! [X0,X4,X5] : ((P(X5,X4) & EX(X0,X4)) => EX(X0,X5)) [negated conjecture 196]
207. ! [X0,X1] : (PRE(X0,X1) => (T(X1) & (Q(X0) | PD(X0) | ED(X0)))) [rectify 34]
208. ! [X0,X1,X2] : ((P(X2,X1) & PRE(X0,X1)) => PRE(X0,X2)) [rectify 35]
313. ! [X0,X1] : (EX(X0,X1) <=> ((T(X1) & ~T(X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1))) [rectify 169]
344. ~! [X0,X1,X2] : ((P(X2,X1) & EX(X0,X1)) => EX(X0,X2)) [rectify 197]
381. ! [X0,X1] : ((T(X0) <=> T(X1)) | ~P(X0,X1)) [ennf transformation 6]
386. ! [X0,X1,X2] : (P(X0,X2) | (~P(X1,X2) | ~P(X0,X1))) [ennf transformation 10]
387. ! [X0,X1,X2] : (P(X0,X2) | ~P(X1,X2) | ~P(X0,X1)) [flattening 386]
420. ! [X0,X1] : ((T(X1) & (Q(X0) | PD(X0) | ED(X0))) | ~PRE(X0,X1)) [ennf transformation 207]
421. ! [X0,X1,X2] : (PRE(X0,X2) | (~P(X2,X1) | ~PRE(X0,X1))) [ennf transformation 208]
422. ! [X0,X1,X2] : (PRE(X0,X2) | ~P(X2,X1) | ~PRE(X0,X1)) [flattening 421]
575. ? [X0,X1,X2] : (~EX(X0,X2) & (P(X2,X1) & EX(X0,X1))) [ennf transformation 344]
576. ? [X0,X1,X2] : (~EX(X0,X2) & P(X2,X1) & EX(X0,X1)) [flattening 575]
579. ! [X1,X0] : (sP1(X1,X0) <=> ((T(X1) & ~T(X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1))) [predicate
     definition introduction]
580. ! [X0,X1] : (EX(X0,X1) <=> sP1(X1,X0)) [definition folding 313,579]
600. ! [X0,X1] : (((T(X0) | ~T(X1)) & (T(X1) | ~T(X0))) | ~P(X0,X1)) [nnf transformation 381]
711. ! [X1,X0] : ((sP1(X1,X0) | ((~T(X1) | T(X0) | ~AB(X0)) & (~P(X1,X0) | ~T(X0)) & ~PRE(X0,X1))) & (((T(X1) & ~T
     (X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1)) | ~sP1(X1,X0))) [nnf transformation 579]
712. ! [X1,X0] : ((sP1(X1,X0) | ((~T(X1) | T(X0) | ~AB(X0)) & (~P(X1,X0) | ~T(X0)) & ~PRE(X0,X1))) & ((T(X1) & ~T(
     X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1) | ~sP1(X1,X0))) [flattening 711]
713. ! [X0,X1] : ((sP1(X0,X1) | ((~T(X0) | T(X1) | ~AB(X1)) & (~P(X0,X1) | ~T(X1)) & ~PRE(X1,X0))) & ((T(X0) & ~T(
     X1) & AB(X1)) | (P(X0,X1) & T(X1)) | PRE(X1,X0) | ~sP1(X0,X1))) [rectify 712]
714. ! [X0,X1] : ((EX(X0,X1) | ~sP1(X1,X0)) & (sP1(X1,X0) | ~EX(X0,X1))) [nnf transformation 580]
779. ? [X0,X1,X2] : (~EX(X0,X2) & P(X2,X1) & EX(X0,X1)) => (~EX(sK85,sK87) & P(sK87,sK86) & EX(sK85,sK86)) [choice
     axiom]
780. ~EX(sK85,sK87) & P(sK87,sK86) & EX(sK85,sK86) [skolemisation 576,779]
799. ~P(X0,X1) | ~T(X1) | T(X0) [cnf transformation 600]
805. ~P(X1,X2) | P(X0,X2) | ~P(X0,X1) [cnf transformation 387]
854. ~PRE(X0,X1) | T(X1) [cnf transformation 420]
855. ~PRE(X0,X1) | ~P(X2,X1) | PRE(X0,X2) [cnf transformation 422]
1037. ~sP1(X0,X1) | T(X1) | PRE(X1,X0) | AB(X1) [cnf transformation 713]
1038. ~sP1(X0,X1) | P(X0,X1) | PRE(X1,X0) | AB(X1) [cnf transformation 713]
1040. ~sP1(X0,X1) | P(X0,X1) | PRE(X1,X0) | ~T(X1) [cnf transformation 713]
1041. T(X0) | T(X1) | PRE(X1,X0) | ~sP1(X0,X1) [cnf transformation 713]
1043. ~PRE(X1,X0) | sP1(X0,X1) [cnf transformation 713]
1044. ~P(X0,X1) | sP1(X0,X1) | ~T(X1) [cnf transformation 713]
1045. sP1(X0,X1) | ~T(X0) | T(X1) | ~AB(X1) [cnf transformation 713]
1046. ~EX(X0,X1) | sP1(X1,X0) [cnf transformation 714]
1047. ~sP1(X1,X0) | EX(X0,X1) [cnf transformation 714]
1136. EX(sK85,sK86) [cnf transformation 780]
1137. P(sK87,sK86) [cnf transformation 780]
1138. ~EX(sK85,sK87) [cnf transformation 780]
1145. ~sP1(X0,X1) | T(X1) | T(X0) [subsumption resolution 1041,854]
```

```
1191. sP1(sK86,sK85) [resolution 1046,1136]
1231. 3 <=> T(sK86) [avatar definition]
1235. 4 <=> T(sK87) [avatar definition]
1236. T(sK87) <- (4) [avatar component clause 1235]
1239. ~T(sK86) | T(sK87) [resolution 799,1137]
1309. T(sK85) | T(sK86) [resolution 1145,1191]
1455. 7 <=> P(sK86,sK85) [avatar definition]
1456. ~P(sK86,sK85) <- (~7) [avatar component clause 1455]
1457. P(sK86,sK85) <- (7) [avatar component clause 1455]
1465. 9 <=> T(sK85) [avatar definition]
1466. ~T(sK85) <- (~9) [avatar component clause 1465]
1467. T(sK85) <- (9) [avatar component clause 1465]
1468. 3 | 9 [avatar split clause 1309,1465,1231]
1469. 4 | ~3 [avatar split clause 1239,1231,1235]
1653. EX(X5,X4) | T(X5) | ~AB(X5) | ~T(X4) [resolution 1045,1047]
1923. 13 <=> AB(sK85) [avatar definition]
1925. AB(sK85) <- (13) [avatar component clause 1923]
1927. 14 <=> PRE(sK85,sK86) [avatar definition]
1929. PRE(sK85,sK86) <- (14) [avatar component clause 1927]
2061. ~P(X1,sK86) | P(X1,sK85) <- (7) [resolution 1457,805]
2178. P(sK86,sK85) | PRE(sK85,sK86) | ~T(sK85) [resolution 1191,1040]
2179. P(sK86,sK85) | PRE(sK85,sK86) | AB(sK85) [resolution 1191,1038]
2180. T(sK85) | PRE(sK85,sK86) | AB(sK85) [resolution 1191,1037]
2364. P(sK87,sK85) <- (7) [resolution 2061,1137]
2391. sP1(sK87,sK85) | ~T(sK85) <- (7) [resolution 2364,1044]
2408. sP1(sK87,sK85) <- (7, 9) [subsumption resolution 2391,1467]
2633. EX(sK85,sK87) <- (7, 9) [resolution 2408,1047]
2634. $false <- (7, 9) [subsumption resolution 2633,1138]
2635. ~7 | ~9 [avatar contradiction clause 2634]
2637. P(sK86,sK85) | PRE(sK85,sK86) <- (9) [subsumption resolution 2178,1467]
2638. 14 | 7 | ~9 [avatar split clause 2637,1465,1455,1927]
2660. ~P(X0,sK86) | PRE(sK85,X0) <- (14) [resolution 1929,855]
2823. PRE(sK85,sK87) <- (14) [resolution 2660,1137]
2832. sP1(sK87,sK85) <- (14) [resolution 2823,1043]
2841. EX(sK85,sK87) <- (14) [resolution 2832,1047]
2842. $false <- (14) [subsumption resolution 2841,1138]
2843. ~14 [avatar contradiction clause 2842]
2846. PRE(sK85,sK86) | AB(sK85) <- (~7) [subsumption resolution 2179,1456]
2847. 13 | 14 | 7 [avatar split clause 2846,1455,1927,1923]
2848. PRE(sK85,sK86) | AB(sK85) <- (~9) [subsumption resolution 2180,1466]
2849. 13 | 14 | 9 [avatar split clause 2848,1465,1927,1923]
3517. T(sK85) | ~AB(sK85) | ~T(sK87) [resolution 1653,1138]
3519. ~AB(sK85) | ~T(sK87) <- (~9) [subsumption resolution 3517,1466]
3520. ~T(sK87) <- (~9, 13) [subsumption resolution 3519,1925]
3521. $false <- (4, ~9, 13) [subsumption resolution 3520,1236]
3522. ~4 | 9 | ~13 [avatar contradiction clause 3521]
3523. $false [avatar sat refutation 1468,1469,2635,2638,2843,2847,2849,3522]
```

Proof of Theorem (t_{db}104)

```
122. ! [X0] : ((TR(X0) | PR(X0) | AR(X0)) <=> R(X0)) [input]
123. ! [X0] : (R(X0) => AB(X0)) [input]
128. ! [X0] : (T(X0) => TR(X0)) [input]
169. ! [X0,X4] : (EX(X0,X4) <=> ((T(X4) & ~T(X0) & AB(X0)) | (P(X4,X0) & T(X0)) | PRE(X0,X4))) [input]
196. ! [X0,X4] : (~AB(X0) => (EX(X0,X4) <=> PRE(X0,X4))) [input]
197. ~! [X0,X4] : (~AB(X0) => (EX(X0,X4) <=> PRE(X0,X4))) [negated conjecture 196]
313. ! [X0,X1] : (EX(X0,X1) <=> ((T(X1) & ~T(X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1))) [rectify 169]
344. ~! [X0,X1] : (~AB(X0) => (EX(X0,X1) <=> PRE(X0,X1))) [rectify 197]
345. ! [X0] : ((TR(X0) | PR(X0) | AR(X0)) => R(X0)) [unused predicate definition removal 122]
541. ! [X0] : (R(X0) | (~TR(X0) & ~PR(X0) & ~AR(X0))) [ennf transformation 345]
542. ! [X0] : (AB(X0) | ~R(X0)) [ennf transformation 123]
546. ! [X0] : (TR(X0) | ~T(X0)) [ennf transformation 128]
575. ? [X0,X1] : ((EX(X0,X1) <~> PRE(X0,X1)) & ~AB(X0)) [ennf transformation 344]
578. ! [X1,X0] : (sP1(X1,X0) <=> ((T(X1) & ~T(X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1))) [predicate
       definition introduction]
579. ! [X0,X1] : (EX(X0,X1) <=> sP1(X1,X0)) [definition folding 313,578]
710. ! [X1,X0] : ((sP1(X1,X0) | ((~T(X1) | T(X0) | ~AB(X0)) & (~P(X1,X0) | ~T(X0)) & ~PRE(X0,X1))) & (((T(X1) & ~T
       (X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1)) | ~sP1(X1,X0))) [nnf transformation 578]
711. ! [X1,X0] : ((sP1(X1,X0) | ((~T(X1) | T(X0) | ~AB(X0)) & (~P(X1,X0) | ~T(X0)) & ~PRE(X0,X1))) & ((T(X1) & ~T(
       X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1) | ~sP1(X1,X0))) [flattening 710]
712. ! [X0,X1] : ((sP1(X0,X1) | ((~T(X0) | T(X1) | ~AB(X1)) & (~P(X0,X1) | ~T(X1)) & ~PRE(X1,X0))) & ((T(X0) & ~T(
       X1) & AB(X1)) | (P(X0,X1) & T(X1)) | PRE(X1,X0) | ~sP1(X0,X1))) [rectify 711]
713. ! [X0,X1] : ((EX(X0,X1) | ~sP1(X1,X0)) & (sP1(X1,X0) | ~EX(X0,X1))) [nnf transformation 579]
778. ? [X0,X1] : (((~PRE(X0,X1) | ~EX(X0,X1)) & (PRE(X0,X1) | EX(X0,X1))) & ~AB(X0)) [nnf transformation 575]
779. ? [X0,X1] : ((~PRE(X0,X1) | ~EX(X0,X1)) & (PRE(X0,X1) | EX(X0,X1)) & ~AB(X0)) [flattening 778]
780. ? [X0,X1] : ((~PRE(X0,X1) | ~EX(X0,X1)) & (PRE(X0,X1) | EX(X0,X1)) & ~AB(X0)) => ((~PRE(sK85,sK86) | ~EX(sK85
       ,sK86)) & (PRE(sK85,sK86) | EX(sK85,sK86)) & ~AB(sK85)) [choice axiom]
781. (~PRE(sK85,sK86) | ~EX(sK85,sK86)) & (PRE(sK85,sK86) | EX(sK85,sK86)) & ~AB(sK85) [skolemisation 779,780]
1003. ~TR(X0) | R(X0) [cnf transformation 541]
1004. ~R(X0) | AB(X0) [cnf transformation 542]
1010. ~T(X0) | TR(X0) [cnf transformation 546]
1038. ~sP1(X0,X1) | T(X1) | PRE(X1,X0) | AB(X1) [cnf transformation 712]
1044. ~PRE(X1,X0) | sP1(X0,X1) [cnf transformation 712]
1047. ~EX(X0,X1) | sP1(X1,X0) [cnf transformation 713]
1048. ~sP1(X1,X0) | EX(X0,X1) [cnf transformation 713]
1137. ~AB(sK85) [cnf transformation 781]
1138. PRE(sK85,sK86) | EX(sK85,sK86) [cnf transformation 781]
1139. ~PRE(sK85,sK86) | ~EX(sK85,sK86) [cnf transformation 781]
1155. 1 <=> EX(sK85,sK86) [avatar definition]
1156. EX(sK85,sK86) <- (1) [avatar component clause 1155]
```

```
1157. ~EX(sK85,sK86) <- (~1) [avatar component clause 1155]
1159. 2 <=> PRE(sK85,sK86) [avatar definition]
1160. PRE(sK85,sK86) <- (2) [avatar component clause 1159]
1161. ~PRE(sK85,sK86) <- (~2) [avatar component clause 1159]
1162. ~1 | ~2 [avatar split clause 1139,1159,1155]
1163. 1 | 2 [avatar split clause 1138,1159,1155]
1206. sP1(sK86,sK85) <- (2) [resolution 1044,1160]
1207. EX(sK85,sK86) <- (2) [resolution 1048,1206]
1208. $false <- (~1, 2) [subsumption resolution 1207,1157]
1209. 1 | ~2 [avatar contradiction clause 1208]
1210. sP1(sK86,sK85) <- (1) [resolution 1156,1047]
1292. 4 <=> T(sK85) [avatar definition]
1293. ~T(sK85) <- (~4) [avatar component clause 1292]
1294. T(sK85) <- (4) [avatar component clause 1292]
1296. TR(sK85) <- (4) [resolution 1294,1010]
1306. R(sK85) <- (4) [resolution 1296,1003]
1316. AB(sK85) <- (4) [resolution 1306,1004]
1317. $false <- (4) [subsumption resolution 1316,1137]
1318. ~4 [avatar contradiction clause 1317]
1833. T(sK85) | PRE(sK85,sK86) | AB(sK85) <- (1) [resolution 1038,1210]
1835. PRE(sK85,sK86) | AB(sK85) <- (1, ~4) [subsumption resolution 1833,1293]
1836. AB(sK85) <- (1, ~2, ~4) [subsumption resolution 1835,1161]
1837. $false <- (1, ~2, ~4) [subsumption resolution 1836,1137]
1838. ~1 | 2 | 4 [avatar contradiction clause 1837]
1839. $false [avatar sat refutation 1162,1163,1209,1318,1838]
```

Proof of Theorem (t_{db}105)

```
169. ! [X0,X4] : (EX(X0,X4) <=> ((T(X4) & ~T(X0) & AB(X0)) | (P(X4,X0) & T(X0)) | PRE(X0,X4))) [input]
180. ! [X0,X4] : (::(X0,mten,X4) <=> (! [X5] : (PRE(X0,X5) => ? [X1] : ? [X3] : ? [X8] : (P(X8,X3) & SLC(X1,X8,X5)
     & SLC(X0,X3,X5) & M(X1))) & ? [X5] : (~AT(X5) & PRE(X0,X5)) & PRE(X0,X4) & ED(X0))) [input]
181. ! [X0,X4] : ((::(X0,cfbnd,X4) | ::(X0,site,X4)) <=> (! [X5] : (PRE(X0,X5) => ~? [X1] : ? [X3] : ? [X8] : (P(
     X8,X3) & SLC(X1,X8,X5) & SLC(X0,X3,X5) & M(X1))) & ? [X3] : SLC(X0,X3,X4) & PRE(X0,X4) & F(X0))) [input]
182. ! [X0,X4] : (::(X0,sreg,X4) <=> (EX(X0,X4) & S(X0))) [input]
183. ! [X0,X4] : (::(X0,imen,X4) <=> (::(X0,sreg,X4) | ::(X0,cfbnd,X4) | ::(X0,site,X4))) [input]
184. ! [X0,X4] : (::(X0,idcnt,X4) <=> (::(X0,imen,X4) | ::(X0,mten,X4))) [input]
185. ! [X0,X4] : (::(X0,sdcnt,X4) <=> (? [X1] : (DQT(X0,X1) & ~S(X1) & ::(X1,idcnt,X4)) & PRE(X0,X4) & Q(X0))) [
     input]
187. ! [X0,X4] : (::(X0,gdcnt,X4) <=> (! [X5] : (PRE(X0,X5) => ? [X1] : CONCR(X1,X0,X5)) & PRE(X0,X4))) [input]
188. ! [X0,X4] : (::(X0,cnt,X4) <=> (::(X0,gdcnt,X4) | ::(X0,sdcnt,X4) | ::(X0,idcnt,X4))) [input]
196. ! [X0,X4] : (::(X0,cnt,X4) => EX(X0,X4)) [input]
197. ~! [X0,X4] : (::(X0,cnt,X4) => EX(X0,X4)) [negated conjecture 196]
313. ! [X0,X1] : (EX(X0,X1) <=> ((T(X1) & ~T(X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1))) [rectify 169]
323. ! [X0,X1] : (::(X0,mten,X1) <=> (! [X2] : (PRE(X0,X2) => ? [X3] : ? [X4] : ? [X5] : (P(X5,X4) & SLC(X3,X5,X2)
     & SLC(X0,X4,X2) & M(X3))) & ? [X6] : (~AT(X6) & PRE(X0,X6)) & PRE(X0,X1) & ED(X0))) [rectify 180]
324. ! [X0,X1] : (::(X0,mten,X1) <=> (! [X2] : (PRE(X0,X2) => ? [X3,X4,X5] : (P(X5,X4) & SLC(X3,X5,X2) & SLC(X0,X4
     ,X2) & M(X3))) & ? [X6] : (~AT(X6) & PRE(X0,X6)) & PRE(X0,X1) & ED(X0))) [flattening 323]
325. ! [X0,X1] : ((::(X0,cfbnd,X1) | ::(X0,site,X1)) <=> (! [X2] : (PRE(X0,X2) => ~? [X3] : ? [X4] : ? [X5] : (P(
     X5,X4) & SLC(X3,X5,X2) & SLC(X0,X4,X2) & M(X3))) & ? [X6] : SLC(X0,X6,X1) & PRE(X0,X1) & F(X0))) [rectify
     181]
326. ! [X0,X1] : ((::(X0,cfbnd,X1) | ::(X0,site,X1)) <=> (! [X2] : (PRE(X0,X2) => ~? [X3,X4,X5] : (P(X5,X4) & SLC(
     X3,X5,X2) & SLC(X0,X4,X2) & M(X3))) & ? [X6] : SLC(X0,X6,X1) & PRE(X0,X1) & F(X0))) [flattening 325]
327. ! [X0,X1] : (::(X0,sreg,X1) <=> (EX(X0,X1) & S(X0))) [rectify 182]
328. ! [X0,X1] : (::(X0,imen,X1) <=> (::(X0,sreg,X1) | ::(X0,cfbnd,X1) | ::(X0,site,X1))) [rectify 183]
329. ! [X0,X1] : (::(X0,idcnt,X1) <=> (::(X0,imen,X1) | ::(X0,mten,X1))) [rectify 184]
330. ! [X0,X1] : (::(X0,sdcnt,X1) <=> (? [X2] : (DQT(X0,X2) & ~S(X2) & ::(X2,idcnt,X1)) & PRE(X0,X1) & Q(X0))) [
     rectify 185]
332. ! [X0,X1] : (::(X0,gdcnt,X1) <=> (! [X2] : (PRE(X0,X2) => ? [X3] : CONCR(X3,X0,X2)) & PRE(X0,X1))) [rectify
     187]
333. ! [X0,X1] : (::(X0,cnt,X1) <=> (::(X0,gdcnt,X1) | ::(X0,sdcnt,X1) | ::(X0,idcnt,X1))) [rectify 188]
344. ~! [X0,X1] : (::(X0,cnt,X1) => EX(X0,X1)) [rectify 197]
571. ! [X0,X1] : (::(X0,mten,X1) <=> (! [X2] : (? [X3,X4,X5] : (P(X5,X4) & SLC(X3,X5,X2) & SLC(X0,X4,X2) & M(X3))
     | ~PRE(X0,X2)) & ? [X6] : (~AT(X6) & PRE(X0,X6)) & PRE(X0,X1) & ED(X0))) [ennf transformation 324]
572. ! [X0,X1] : ((::(X0,cfbnd,X1) | ::(X0,site,X1)) <=> (! [X2] : (! [X3,X4,X5] : (~P(X5,X4) | ~SLC(X3,X5,X2) | ~
     SLC(X0,X4,X2) | ~M(X3)) | ~PRE(X0,X2)) & ? [X6] : SLC(X0,X6,X1) & PRE(X0,X1) & F(X0))) [ennf transformation
     326]
574. ! [X0,X1] : (::(X0,gdcnt,X1) <=> (! [X2] : (? [X3] : CONCR(X3,X0,X2) | ~PRE(X0,X2)) & PRE(X0,X1))) [ennf
     transformation 332]
575. ? [X0,X1] : (~EX(X0,X1) & ::(X0,cnt,X1)) [ennf transformation 344]
578. ! [X1,X0] : (sP1(X1,X0) <=> ((T(X1) & ~T(X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1))) [predicate
     definition introduction]
579. ! [X0,X1] : (EX(X0,X1) <=> sP1(X1,X0)) [definition folding 313,578]
580. ! [X0,X1] : (sP2(X0,X1) <=> (! [X2] : (? [X3,X4,X5] : (P(X5,X4) & SLC(X3,X5,X2) & SLC(X0,X4,X2) & M(X3)) | ~
     PRE(X0,X2)) & ? [X6] : (~AT(X6) & PRE(X0,X6)) & PRE(X0,X1) & ED(X0))) [predicate definition introduction]
581. ! [X0,X1] : (::(X0,mten,X1) <=> sP2(X0,X1)) [definition folding 571,580]
582. ! [X0,X1] : (sP3(X0,X1) <=> (! [X2] : (! [X3,X4,X5] : (~P(X5,X4) | ~SLC(X3,X5,X2) | ~SLC(X0,X4,X2) | ~M(X3))
     | ~PRE(X0,X2)) & ? [X6] : SLC(X0,X6,X1) & PRE(X0,X1) & F(X0))) [predicate definition introduction]
583. ! [X0,X1] : ((::(X0,cfbnd,X1) | ::(X0,site,X1)) <=> sP3(X0,X1)) [definition folding 572,582]
710. ! [X1,X0] : ((sP1(X1,X0) | ((~T(X1) | T(X0) | ~AB(X0)) & (~P(X1,X0) | ~T(X0)) & ~PRE(X0,X1))) & (((T(X1) & ~T
     (X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1)) | ~sP1(X1,X0))) [nnf transformation 578]
711. ! [X1,X0] : ((sP1(X1,X0) | ((~T(X1) | T(X0) | ~AB(X0)) & (~P(X1,X0) | ~T(X0)) & ~PRE(X0,X1))) & ((T(X1) & ~T(
     X0) & AB(X0)) | (P(X1,X0) & T(X0)) | PRE(X0,X1) | ~sP1(X1,X0))) [flattening 710]
712. ! [X0,X1] : ((sP1(X0,X1) | ((~T(X0) | T(X1) | ~AB(X1)) & (~P(X0,X1) | ~T(X1)) & ~PRE(X1,X0))) & ((T(X0) & ~T(
     X1) & AB(X1)) | (P(X0,X1) & T(X1)) | PRE(X1,X0) | ~sP1(X0,X1))) [rectify 711]
713. ! [X0,X1] : ((EX(X0,X1) | ~sP1(X1,X0)) & (sP1(X1,X0) | ~EX(X0,X1))) [nnf transformation 579]
737. ! [X0,X1] : ((sP2(X0,X1) | (? [X2] : (! [X3,X4,X5] : (~P(X5,X4) | ~SLC(X3,X5,X2) | ~SLC(X0,X4,X2) | ~M(X3)) &
     PRE(X0,X2)) | ! [X6] : (AT(X6) | ~PRE(X0,X6)) | ~PRE(X0,X1) | ~ED(X0))) & ((! [X2] : (? [X3,X4,X5] : (P(X5,
     X4) & SLC(X3,X5,X2) & SLC(X0,X4,X2) & M(X3)) | ~PRE(X0,X2)) & ? [X6] : (~AT(X6) & PRE(X0,X6)) & PRE(X0,X1) &
     ED(X0)) | ~sP2(X0,X1))) [nnf transformation 580]
```

738. ! [X0,X1] : ((sP2(X0,X1) | ? [X2] : (! [X3,X4,X5] : (~P(X5,X4) | ~SLC(X3,X5,X2) | ~SLC(X0,X4,X2) | ~M(X3)) & PRE(X0,X2)) | ! [X6] : (AT(X6) | ~PRE(X0,X6)) | ~PRE(X0,X1) | ~ED(X0)) & ((! [X2] : (? [X3,X4,X5] : (P(X5,X4) & SLC(X3,X5,X2) & SLC(X0,X4,X2) & M(X3)) | ~PRE(X0,X2)) & ? [X6] : (~AT(X6) & PRE(X0,X6)) & PRE(X0,X1) & ED(X0)) | ~sP2(X0,X1))) [flattening 737]
739. ! [X0,X1] : ((sP2(X0,X1) | ? [X2] : (! [X3,X4,X5] : (~P(X5,X4) | ~SLC(X3,X5,X2) | ~SLC(X0,X4,X2) | ~M(X3)) & PRE(X0,X2)) | ! [X6] : (AT(X6) | ~PRE(X0,X6)) | ~PRE(X0,X1) | ~ED(X0)) & ((! [X7] : (? [X8,X9,X10] : (P(X10,X9) & SLC(X8,X10,X7) & SLC(X0,X9,X7) & M(X8)) | ~PRE(X0,X7)) & ? [X11] : (~AT(X11) & PRE(X0,X11)) & PRE(X0,X1) & ED(X0)) | ~sP2(X0,X1))) [rectify 738]
740. ! [X0] : (? [X2] : (! [X3,X4,X5] : (~P(X5,X4) | ~SLC(X3,X5,X2) | ~SLC(X0,X4,X2) | ~M(X3)) & PRE(X0,X2)) => (! [X5,X4,X3] : (~P(X5,X4) | ~SLC(X3,X5,sK71(X0)) | ~SLC(X0,X4,sK71(X0)) | ~M(X3)) & PRE(X0,sK71(X0)))) [choice axiom]
741. ! [X7,X0] : (? [X8,X9,X10] : (P(X10,X9) & SLC(X8,X10,X7) & SLC(X0,X9,X7) & M(X8)) => (P(sK74(X0,X7),sK73(X0,X7)) & SLC(sK72(X0,X7),sK74(X0,X7),X7) & SLC(X0,sK73(X0,X7),X7) & M(sK72(X0,X7)))) [choice axiom]
742. ! [X0] : (? [X11] : (~AT(X11) & PRE(X0,X11)) => (~AT(sK75(X0)) & PRE(X0,sK75(X0)))) [choice axiom]
743. ! [X0,X1] : ((sP2(X0,X1) | (! [X3,X4,X5] : (~P(X5,X4) | ~SLC(X3,X5,sK71(X0)) | ~SLC(X0,X4,sK71(X0)) | ~M(X3)) & PRE(X0,sK71(X0))) | ! [X6] : (AT(X6) | ~PRE(X0,X6)) | ~PRE(X0,X1) | ~ED(X0)) & ((! [X7] : ((P(sK74(X0,X7),sK73(X0,X7)) & SLC(sK72(X0,X7),sK74(X0,X7),X7) & SLC(X0,sK73(X0,X7),X7) & M(sK72(X0,X7))) | ~PRE(X0,X7)) & (~AT(sK75(X0)) & PRE(X0,sK75(X0)))) & PRE(X0,X1) & ED(X0)) | ~sP2(X0,X1))) [skolemisation 739,742,741,740]
744. ! [X0,X1] : ((::(X0,mten,X1) | ~sP2(X0,X1)) & (sP2(X0,X1) | ~::(X0,mten,X1))) [nnf transformation 581]
745. ! [X0,X1] : ((sP3(X0,X1) | (? [X2] : (? [X3,X4,X5] : (P(X5,X4) & SLC(X3,X5,X2) & SLC(X0,X4,X2) & M(X3)) & PRE(X0,X2)) | ! [X6] : (~SLC(X0,X6,X1) | ~PRE(X0,X1) | ~F(X0))) & ((! [X2] : (! [X3,X4,X5] : (~P(X5,X4) | ~SLC(X3,X5,X2) | ~SLC(X0,X4,X2) | ~M(X3)) | ~PRE(X0,X2)) & ? [X6] : SLC(X0,X6,X1) & PRE(X0,X1) & F(X0)) | ~sP3(X0,X1))) [nnf transformation 582]
746. ! [X0,X1] : ((sP3(X0,X1) | ? [X2] : (? [X3,X4,X5] : (P(X5,X4) & SLC(X3,X5,X2) & SLC(X0,X4,X2) & M(X3)) & PRE(X0,X2)) | ! [X6] : (~SLC(X0,X6,X1) | ~PRE(X0,X1) | ~F(X0)) & ((! [X2] : (! [X3,X4,X5] : (~P(X5,X4) | ~SLC(X3,X5,X2) | ~SLC(X0,X4,X2) | ~M(X3)) | ~PRE(X0,X2)) & ? [X6] : SLC(X0,X6,X1) & PRE(X0,X1) & F(X0)) | ~sP3(X0,X1))) [flattening 745]
747. ! [X0,X1] : ((sP3(X0,X1) | ? [X2] : (? [X3,X4,X5] : (P(X5,X4) & SLC(X3,X5,X2) & SLC(X0,X4,X2) & M(X3)) & PRE(X0,X2)) | ! [X6] : (~SLC(X0,X6,X1) | ~PRE(X0,X1) | ~F(X0)) & ((! [X7] : (! [X8,X9,X10] : (~P(X10,X9) | ~SLC(X8,X10,X7) | ~SLC(X0,X9,X7) | ~M(X8)) | ~PRE(X0,X7)) & ? [X11] : SLC(X0,X11,X1) & PRE(X0,X1) & F(X0)) | ~sP3(X0,X1))) [rectify 746]
748. ! [X0] : (? [X2] : (? [X3,X4,X5] : (P(X5,X4) & SLC(X3,X5,X2) & SLC(X0,X4,X2) & M(X3)) & PRE(X0,X2)) => (? [X5,X4,X3] : (P(X5,X4) & SLC(X3,X5,sK76(X0)) & SLC(X0,X4,sK76(X0)) & M(X3)) & PRE(X0,sK76(X0)))) [choice axiom]
749. ! [X0] : (? [X5,X4,X3] : (P(X5,X4) & SLC(X3,X5,sK76(X0)) & SLC(X0,X4,sK76(X0)) & M(X3)) => (P(sK79(X0),sK78(X0)) & SLC(sK77(X0),sK79(X0),sK76(X0)) & SLC(X0,sK78(X0),sK76(X0)) & M(sK77(X0)))) [choice axiom]
750. ! [X1,X0] : (? [X11] : SLC(X0,X11,X1) => SLC(X0,sK80(X0,X1),X1)) [choice axiom]
751. ! [X0,X1] : ((sP3(X0,X1) | ((P(sK79(X0),sK78(X0)) & SLC(sK77(X0),sK79(X0),sK76(X0)) & SLC(X0,sK78(X0),sK76(X0)) & M(sK77(X0))) & PRE(X0,sK76(X0))) | ! [X6] : ~SLC(X0,X6,X1) | ~PRE(X0,X1) | ~F(X0)) & ((! [X7] : (! [X8,X9,X10] : (~P(X10,X9) | ~SLC(X8,X10,X7) | ~SLC(X0,X9,X7) | ~M(X8)) | ~PRE(X0,X7)) & SLC(X0,sK80(X0,X1),X1) & PRE(X0,X1) & F(X0)) | ~sP3(X0,X1))) [skolemisation 747,750,749,748]
752. ! [X0,X1] : (((::(X0,cfbnd,X1) | ::(X0,site,X1)) | ~sP3(X0,X1)) & (sP3(X0,X1) | (~::(X0,cfbnd,X1) & ~::(X0,site,X1)))) [nnf transformation 583]
753. ! [X0,X1] : ((::(X0,cfbnd,X1) | ::(X0,site,X1) | ~sP3(X0,X1)) & (sP3(X0,X1) | (~::(X0,cfbnd,X1) & ~::(X0,site,X1)))) [flattening 752]
754. ! [X0,X1] : ((::(X0,sreg,X1) | (~EX(X0,X1) | ~S(X0))) & ((EX(X0,X1) & S(X0)) | ~::(X0,sreg,X1))) [nnf transformation 327]
755. ! [X0,X1] : ((::(X0,sreg,X1) | ~EX(X0,X1) | ~S(X0)) & ((EX(X0,X1) & S(X0)) | ~::(X0,sreg,X1))) [flattening 754]
756. ! [X0,X1] : ((::(X0,imen,X1) | (~::(X0,sreg,X1) & ~::(X0,cfbnd,X1) & ~::(X0,site,X1))) & ((::(X0,sreg,X1) | ::(X0,cfbnd,X1) | ::(X0,site,X1)) | ~::(X0,imen,X1))) [nnf transformation 328]
757. ! [X0,X1] : ((::(X0,imen,X1) | (~::(X0,sreg,X1) & ~::(X0,cfbnd,X1) & ~::(X0,site,X1))) & (::(X0,sreg,X1) | ::(X0,cfbnd,X1) | ::(X0,site,X1) | ~::(X0,imen,X1))) [flattening 756]
758. ! [X0,X1] : ((::(X0,idcnt,X1) | (~::(X0,imen,X1) & ~::(X0,mten,X1))) & ((::(X0,imen,X1) | ::(X0,mten,X1)) | ~::(X0,idcnt,X1))) [nnf transformation 329]
759. ! [X0,X1] : ((::(X0,idcnt,X1) | (~::(X0,imen,X1) & ~::(X0,mten,X1))) & (::(X0,imen,X1) | ::(X0,mten,X1) | ~::(X0,idcnt,X1))) [flattening 758]
760. ! [X0,X1] : ((::(X0,sdcnt,X1) | (! [X2] : (~DQT(X0,X2) | S(X2) | ~::(X2,idcnt,X1)) | ~PRE(X0,X1) | ~Q(X0))) & ((? [X2] : (DQT(X0,X2) & ~S(X2) & ::(X2,idcnt,X1)) & PRE(X0,X1) & Q(X0)) | ~::(X0,sdcnt,X1))) [nnf transformation 330]
761. ! [X0,X1] : ((::(X0,sdcnt,X1) | ! [X2] : (~DQT(X0,X2) | S(X2) | ~::(X2,idcnt,X1)) | ~PRE(X0,X1) | ~Q(X0)) & ((? [X2] : (DQT(X0,X2) & ~S(X2) & ::(X2,idcnt,X1)) & PRE(X0,X1) & Q(X0)) | ~::(X0,sdcnt,X1))) [flattening 760]
762. ! [X0,X1] : ((::(X0,sdcnt,X1) | ! [X2] : (~DQT(X0,X2) | S(X2) | ~::(X2,idcnt,X1)) | ~PRE(X0,X1) | ~Q(X0)) & ((? [X3] : (DQT(X0,X3) & ~S(X3) & ::(X3,idcnt,X1)) & PRE(X0,X1) & Q(X0)) | ~::(X0,sdcnt,X1))) [rectify 761]
763. ! [X1,X0] : (? [X3] : (DQT(X0,X3) & ~S(X3) & ::(X3,idcnt,X1)) => (DQT(X0,sK81(X0,X1)) & ~S(sK81(X0,X1)) & ::(sK81(X0,X1),idcnt,X1))) [choice axiom]
764. ! [X0,X1] : ((::(X0,sdcnt,X1) | ! [X2] : (~DQT(X0,X2) | S(X2) | ~::(X2,idcnt,X1)) | ~PRE(X0,X1) | ~Q(X0)) & (((DQT(X0,sK81(X0,X1)) & ~S(sK81(X0,X1)) & ::(sK81(X0,X1),idcnt,X1)) & PRE(X0,X1) & Q(X0)) | ~::(X0,sdcnt,X1))) [skolemisation 762,763]
770. ! [X0,X1] : ((::(X0,gdcnt,X1) | (? [X2] : (! [X3] : ~CONCR(X3,X0,X2) & PRE(X0,X2)) | ~PRE(X0,X1))) & ((! [X2] : (? [X3] : CONCR(X3,X0,X2) | ~PRE(X0,X2)) & PRE(X0,X1)) | ~::(X0,gdcnt,X1))) [nnf transformation 574]
771. ! [X0,X1] : ((::(X0,gdcnt,X1) | ? [X2] : (! [X3] : ~CONCR(X3,X0,X2) & PRE(X0,X2)) | ~PRE(X0,X1)) & ((! [X2] : (? [X3] : CONCR(X3,X0,X2) | ~PRE(X0,X2)) & PRE(X0,X1)) | ~::(X0,gdcnt,X1))) [flattening 770]
772. ! [X0,X1] : ((::(X0,gdcnt,X1) | ? [X2] : (! [X3] : ~CONCR(X3,X0,X2) & PRE(X0,X2)) | ~PRE(X0,X1)) & ((! [X4] : (? [X5] : CONCR(X5,X0,X4) | ~PRE(X0,X4)) & PRE(X0,X1)) | ~::(X0,gdcnt,X1))) [rectify 771]
773. ! [X0] : (? [X2] : (! [X3] : ~CONCR(X3,X0,X2) & PRE(X0,X2)) => (! [X3] : ~CONCR(X3,X0,sK83(X0)) & PRE(X0,sK83(X0)))) [choice axiom]
774. ! [X4,X0] : (? [X5] : CONCR(X5,X0,X4) => CONCR(sK84(X0,X4),X0,X4)) [choice axiom]
775. ! [X0,X1] : ((::(X0,gdcnt,X1) | (! [X3] : ~CONCR(X3,X0,sK83(X0)) & PRE(X0,sK83(X0))) | ~PRE(X0,X1)) & ((! [X4] : (CONCR(sK84(X0,X4),X0,X4) | ~PRE(X0,X4)) & PRE(X0,X1)) | ~::(X0,gdcnt,X1))) [skolemisation 772,774,773]
776. ! [X0,X1] : ((::(X0,cnt,X1) | (~::(X0,gdcnt,X1) & ~::(X0,sdcnt,X1) & ~::(X0,idcnt,X1))) & ((::(X0,gdcnt,X1) | ::(X0,sdcnt,X1) | ::(X0,idcnt,X1)) | ~::(X0,cnt,X1))) [nnf transformation 333]
777. ! [X0,X1] : ((::(X0,cnt,X1) | (~::(X0,gdcnt,X1) & ~::(X0,sdcnt,X1) & ~::(X0,idcnt,X1))) & (::(X0,gdcnt,X1) | ::(X0,sdcnt,X1) | ::(X0,idcnt,X1) | ~::(X0,cnt,X1))) [flattening 776]
778. ? [X0,X1] : (~EX(X0,X1) & ::(X0,cnt,X1)) => (~EX(sK85,sK86) & ::(sK85,cnt,sK86)) [choice axiom]
779. ~EX(sK85,sK86) & ::(sK85,cnt,sK86) [skolemisation 575,778]
1042. ~PRE(X1,X0) | sP1(X0,X1) [cnf transformation 712]
1046. ~sP1(X1,X0) | EX(X0,X1) [cnf transformation 713]
1083. ~sP2(X0,X1) | PRE(X0,X1) [cnf transformation 743]
1092. ~::(X0,mten,X1) | sP2(X0,X1) [cnf transformation 744]
1095. ~sP3(X0,X1) | PRE(X0,X1) [cnf transformation 751]

```
1103. ~::(X0,site,X1) | sP3(X0,X1) [cnf transformation 753]
1104. ~::(X0,cfbnd,X1) | sP3(X0,X1) [cnf transformation 753]
1107. ~::(X0,sreg,X1) | EX(X0,X1) [cnf transformation 755]
1109. ~::(X0,imen,X1) | ::(X0,cfbnd,X1) | ::(X0,site,X1) | ::(X0,sreg,X1) [cnf transformation 757]
1113. ~::(X0,idcnt,X1) | ::(X0,mten,X1) | ::(X0,imen,X1) [cnf transformation 759]
1117. ~::(X0,sdcnt,X1) | PRE(X0,X1) [cnf transformation 764]
1127. ~::(X0,gdcnt,X1) | PRE(X0,X1) [cnf transformation 775]
1131. ~::(X0,cnt,X1) | ::(X0,sdcnt,X1) | ::(X0,idcnt,X1) | ::(X0,gdcnt,X1) [cnf transformation 777]
1135. ::(sK85,cnt,sK86) [cnf transformation 779]
1136. ~EX(sK85,sK86) [cnf transformation 779]
2709. ::(sK85,sdcnt,sK86) | ::(sK85,idcnt,sK86) | ::(sK85,gdcnt,sK86) [resolution 1131,1135]
2713. 1 <=> ::(sK85,gdcnt,sK86) [avatar definition]
2715. ::(sK85,gdcnt,sK86) <- (1) [avatar component clause 2713]
2717. 2 <=> ::(sK85,idcnt,sK86) [avatar definition]
2719. ::(sK85,idcnt,sK86) <- (2) [avatar component clause 2717]
2721. 3 <=> ::(sK85,sdcnt,sK86) [avatar definition]
2723. ::(sK85,sdcnt,sK86) <- (3) [avatar component clause 2721]
2724. 1 | 2 | 3 [avatar split clause 2709,2721,2717,2713]
2728. 4 <=> ::(sK85,imen,sK86) [avatar definition]
2730. ::(sK85,imen,sK86) <- (4) [avatar component clause 2728]
2732. 5 <=> ::(sK85,mten,sK86) [avatar definition]
2734. ::(sK85,mten,sK86) <- (5) [avatar component clause 2732]
2744. 6 <=> ::(sK85,sreg,sK86) [avatar definition]
2746. ::(sK85,sreg,sK86) <- (6) [avatar component clause 2744]
2748. 7 <=> ::(sK85,site,sK86) [avatar definition]
2750. ::(sK85,site,sK86) <- (7) [avatar component clause 2748]
2752. 8 <=> ::(sK85,cfbnd,sK86) [avatar definition]
2754. ::(sK85,cfbnd,sK86) <- (8) [avatar component clause 2752]
2985. 26 <=> PRE(sK85,sK86) [avatar definition]
2986. PRE(sK85,sK86) <- (26) [avatar component clause 2985]
2987. ~PRE(sK85,sK86) <- (~26) [avatar component clause 2985]
3008. sP2(sK85,sK86) <- (5) [resolution 2734,1092]
3013. PRE(sK85,sK86) <- (5) [resolution 3008,1083]
3020. $false <- (5, ~26) [subsumption resolution 3013,2987]
3021. ~5 | 26 [avatar contradiction clause 3020]
3030. PRE(sK85,sK86) <- (3) [resolution 2723,1117]
3032. $false <- (3, ~26) [subsumption resolution 3030,2987]
3033. ~3 | 26 [avatar contradiction clause 3032]
3065. ::(sK85,cfbnd,sK86) | ::(sK85,site,sK86) | ::(sK85,sreg,sK86) <- (4) [resolution 2730,1109]
3073. EX(sK85,sK86) <- (6) [resolution 2746,1107]
3075. $false <- (6) [subsumption resolution 3073,1136]
3076. ~6 [avatar contradiction clause 3075]
3079. 6 | 7 | 8 | ~4 [avatar split clause 3065,2728,2752,2748,2744]
3088. sP3(sK85,sK86) <- (7) [resolution 2750,1103]
3093. PRE(sK85,sK86) <- (7) [resolution 3088,1095]
3097. 26 | ~7 [avatar split clause 3093,2748,2985]
3112. PRE(sK85,sK86) <- (1) [resolution 2715,1127]
3210. 26 | ~1 [avatar split clause 3112,2713,2985]
3213. ::(sK85,mten,sK86) | ::(sK85,imen,sK86) <- (2) [resolution 2719,1113]
3283. sP3(sK85,sK86) <- (8) [resolution 2754,1104]
3286. PRE(sK85,sK86) <- (8) [resolution 3283,1095]
3290. $false <- (8, ~26) [subsumption resolution 3286,2987]
3291. ~8 | 26 [avatar contradiction clause 3290]
3292. 4 | 5 | ~2 [avatar split clause 3213,2717,2732,2728]
3321. sP1(sK86,sK85) <- (26) [resolution 2986,1042]
3333. EX(sK85,sK86) <- (26) [resolution 3321,1046]
3334. $false <- (26) [subsumption resolution 3333,1136]
3335. ~26 [avatar contradiction clause 3334]
3336. $false [avatar sat refutation 2724,3021,3033,3076,3079,3097,3210,3291,3292,3335]
```

Proof of Theorem (t_{db} 106)

```
180. ! [X0,X4] : (::(X0,mten,X4) <=> (! [X5] : (PRE(X0,X5) => ? [X1] : ? [X3] : ? [X8] : (P(X8,X3) & SLC(X1,X8,X5)
       & SLC(X0,X3,X5) & M(X1))) & ? [X5] : (~AT(X5) & PRE(X0,X5)) & PRE(X0,X4) & ED(X0))) [input]
181. ! [X0,X4] : ((::(X0,cfbnd,X4) | ::(X0,site,X4)) <=> (! [X5] : (PRE(X0,X5) => ~? [X1] : ? [X3] : ? [X8] : (P(
       X8,X3) & SLC(X1,X8,X5) & SLC(X0,X3,X5) & M(X1))) & ? [X3] : SLC(X0,X3,X4) & PRE(X0,X4) & F(X0))) [input]
183. ! [X0,X4] : (::(X0,imen,X4) <=> (::(X0,sreg,X4) | ::(X0,cfbnd,X4) | ::(X0,site,X4))) [input]
184. ! [X0,X4] : (::(X0,idcnt,X4) <=> (::(X0,imen,X4) | ::(X0,mten,X4))) [input]
185. ! [X0,X4] : (::(X0,sdcnt,X4) <=> (? [X1] : (DQT(X0,X1) & ~S(X1) & ::(X1,idcnt,X4)) & PRE(X0,X4) & Q(X0))) [
       input]
187. ! [X0,X4] : (::(X0,gdcnt,X4) <=> (! [X5] : (PRE(X0,X5) => ? [X1] : CONCR(X1,X0,X5)) & PRE(X0,X4))) [input]
188. ! [X0,X4] : (::(X0,cnt,X4) <=> (::(X0,gdcnt,X4) | ::(X0,sdcnt,X4) | ::(X0,idcnt,X4))) [input]
196. ! [X0,X4] : ((~::(X0,sreg,X4) & ::(X0,cnt,X4)) => PRE(X0,X4)) [input]
197. ~! [X0,X4] : ((~::(X0,sreg,X4) & ::(X0,cnt,X4)) => PRE(X0,X4)) [negated conjecture 196]
323. ! [X0,X1] : (::(X0,mten,X1) <=> (! [X2] : (PRE(X0,X2) => ? [X3] : ? [X4] : ? [X5] : (P(X5,X4) & SLC(X3,X5,X2)
       & SLC(X0,X4,X2) & M(X3))) & ? [X6] : (~AT(X6) & PRE(X0,X6)) & PRE(X0,X1) & ED(X0))) [rectify 180]
324. ! [X0,X1] : (::(X0,mten,X1) <=> (! [X2] : (PRE(X0,X2) => ? [X3,X4,X5] : (P(X5,X4) & SLC(X3,X5,X2) & SLC(X0,X4
       ,X2) & M(X3))) & ? [X6] : (~AT(X6) & PRE(X0,X6)) & PRE(X0,X1) & ED(X0))) [flattening 323]
325. ! [X0,X1] : (((::(X0,cfbnd,X1) | ::(X0,site,X1)) <=> (! [X2] : (PRE(X0,X2) => ~? [X3] : ? [X4] : ? [X5] : (P(
       X5,X4) & SLC(X3,X5,X2) & SLC(X0,X4,X2) & M(X3))) & ? [X6] : SLC(X0,X6,X1) & PRE(X0,X1) & F(X0))) [rectify
       181]
326. ! [X0,X1] : (((::(X0,cfbnd,X1) | ::(X0,site,X1)) <=> (! [X2] : (PRE(X0,X2) => ~? [X3,X4,X5] : (P(X5,X4) & SLC(
       X3,X5,X2) & SLC(X0,X4,X2) & M(X3))) & ? [X6] : SLC(X0,X6,X1) & PRE(X0,X1) & F(X0))) [flattening 325]
328. ! [X0,X1] : (::(X0,imen,X1) <=> (::(X0,sreg,X1) | ::(X0,cfbnd,X1) | ::(X0,site,X1))) [rectify 183]
329. ! [X0,X1] : (::(X0,idcnt,X1) <=> (::(X0,imen,X1) | ::(X0,mten,X1))) [rectify 184]
330. ! [X0,X1] : (::(X0,sdcnt,X1) <=> (? [X2] : (DQT(X0,X2) & ~S(X2) & ::(X2,idcnt,X1)) & PRE(X0,X1) & Q(X0))) [
       rectify 185]
332. ! [X0,X1] : (::(X0,gdcnt,X1) <=> (! [X2] : (PRE(X0,X2) => ? [X3] : CONCR(X3,X0,X2)) & PRE(X0,X1))) [rectify
       187]
333. ! [X0,X1] : (::(X0,cnt,X1) <=> (::(X0,gdcnt,X1) | ::(X0,sdcnt,X1) | ::(X0,idcnt,X1))) [rectify 188]
```

```
344. ~! [X0,X1] : ((~::(X0,sreg,X1) & ::(X0,cnt,X1)) => PRE(X0,X1)) [rectify 197]
571. ! [X0,X1] : (::(X0,mten,X1) <=> (! [X2] : (? [X3,X4,X5] : (P(X5,X4) & SLC(X3,X5,X2) & SLC(X0,X4,X2) & M(X3)) | ~PRE(X0,X2)) & ? [X6] : (~AT(X6) & PRE(X0,X6)) & PRE(X0,X1) & ED(X0))) [ennf transformation 324]
572. ! [X0,X1] : ((::(X0,cfbnd,X1) | ::(X0,site,X1)) <=> (! [X2] : (! [X3,X4,X5] : (~P(X5,X4) | ~SLC(X3,X5,X2) | ~SLC(X0,X4,X2) | ~M(X3)) | ~PRE(X0,X2)) & ? [X6] : SLC(X0,X6,X1) & PRE(X0,X1) & F(X0))) [ennf transformation 326]
574. ! [X0,X1] : (::(X0,gdcnt,X1) <=> (! [X2] : (? [X3] : CONCR(X3,X0,X2) | ~PRE(X0,X2)) & PRE(X0,X1))) [ennf transformation 332]
575. ? [X0,X1] : (~PRE(X0,X1) & (~::(X0,sreg,X1) & ::(X0,cnt,X1))) [ennf transformation 344]
576. ? [X0,X1] : (~PRE(X0,X1) & ~::(X0,sreg,X1) & ::(X0,cnt,X1)) [flattening 575]
581. ! [X0,X1] : (sP2(X0,X1) <=> (! [X2] : (? [X3,X4,X5] : (P(X5,X4) & SLC(X3,X5,X2) & SLC(X0,X4,X2) & M(X3)) | ~PRE(X0,X2)) & ? [X6] : (~AT(X6) & PRE(X0,X6)) & PRE(X0,X1) & ED(X0))) [predicate definition introduction]
582. ! [X0,X1] : (::(X0,mten,X1) <=> sP2(X0,X1)) [definition folding 571,581]
583. ! [X0,X1] : (sP3(X0,X1) <=> (! [X2] : (! [X3,X4,X5] : (~P(X5,X4) | ~SLC(X3,X5,X2) | ~SLC(X0,X4,X2) | ~M(X3)) | ~PRE(X0,X2)) & ? [X6] : SLC(X0,X6,X1) & PRE(X0,X1) & F(X0))) [predicate definition introduction]
584. ! [X0,X1] : ((::(X0,cfbnd,X1) | ::(X0,site,X1)) <=> sP3(X0,X1)) [definition folding 572,583]
738. ! [X0,X1] : ((sP2(X0,X1) | (? [X2] : (! [X3,X4,X5] : (~P(X5,X4) | ~SLC(X3,X5,X2) | ~SLC(X0,X4,X2) | ~M(X3)) & PRE(X0,X2)) | ! [X6] : (AT(X6) | ~PRE(X0,X6)) | ~PRE(X0,X1) | ~ED(X0))) & ((! [X2] : (? [X3,X4,X5] : (P(X5,X4) & SLC(X3,X5,X2) & SLC(X0,X4,X2) & M(X3)) | ~PRE(X0,X2)) & ? [X6] : (~AT(X6) & PRE(X0,X6)) & PRE(X0,X1) & ED(X0)) | ~sP2(X0,X1))) [nnf transformation 581]
739. ! [X0,X1] : ((sP2(X0,X1) | ? [X2] : (! [X3,X4,X5] : (~P(X5,X4) | ~SLC(X3,X5,X2) | ~SLC(X0,X4,X2) | ~M(X3)) & PRE(X0,X2)) | ! [X6] : (AT(X6) | ~PRE(X0,X6)) | ~PRE(X0,X1) | ~ED(X0)) & ((! [X2] : (? [X3,X4,X5] : (P(X5,X4) & SLC(X3,X5,X2) & SLC(X0,X4,X2) & M(X3)) | ~PRE(X0,X2)) & ? [X6] : (~AT(X6) & PRE(X0,X6)) & PRE(X0,X1) & ED(X0)) | ~sP2(X0,X1))) [flattening 738]
740. ! [X0,X1] : ((sP2(X0,X1) | ? [X2] : (! [X3,X4,X5] : (~P(X5,X4) | ~SLC(X3,X5,X2) | ~SLC(X0,X4,X2) | ~M(X3)) & PRE(X0,X2)) | ! [X6] : (AT(X6) | ~PRE(X0,X6)) | ~PRE(X0,X1) | ~ED(X0)) & ((! [X7] : (? [X8,X9,X10] : (P(X10,X9) & SLC(X8,X10,X7) & SLC(X0,X9,X7) & M(X8)) | ~PRE(X0,X7)) & ? [X11] : (~AT(X11) & PRE(X0,X11)) & PRE(X0,X1) & ED(X0)) | ~sP2(X0,X1))) [rectify 739]
741. ! [X0] : (? [X2] : (! [X3,X4,X5] : (~P(X5,X4) | ~SLC(X3,X5,X2) | ~SLC(X0,X4,X2) | ~M(X3)) & PRE(X0,X2)) => (! [X5,X4,X3] : (~P(X5,X4) | ~SLC(X3,X5,sK71(X0)) | ~SLC(X0,X4,sK71(X0)) | ~M(X3)) & PRE(X0,sK71(X0)))) [choice axiom]
742. ! [X7,X0] : (? [X8,X9,X10] : (P(X10,X9) & SLC(X8,X10,X7) & SLC(X0,X9,X7) & M(X8)) => (P(sK74(X0,X7),sK73(X0,X7)) & SLC(sK72(X0,X7),sK74(X0,X7),X7) & SLC(X0,sK73(X0,X7),X7) & M(sK72(X0,X7)))) [choice axiom]
743. ! [X0] : (? [X11] : (~AT(X11) & PRE(X0,X11)) => (~AT(sK75(X0)) & PRE(X0,sK75(X0)))) [choice axiom]
744. ! [X0,X1] : ((sP2(X0,X1) | (! [X3,X4,X5] : (~P(X5,X4) | ~SLC(X3,X5,sK71(X0)) | ~SLC(X0,X4,sK71(X0)) | ~M(X3)) & PRE(X0,sK71(X0))) | ! [X6] : (AT(X6) | ~PRE(X0,X6)) | ~PRE(X0,X1) | ~ED(X0)) & ((! [X7] : ((P(sK74(X0,X7),sK73(X0,X7)) & SLC(sK72(X0,X7),sK74(X0,X7),X7) & SLC(X0,sK73(X0,X7),X7) & M(sK72(X0,X7))) | ~PRE(X0,X7)) & (~AT(sK75(X0)) & PRE(X0,sK75(X0))) & PRE(X0,X1) & ED(X0)) | ~sP2(X0,X1))) [skolemisation 740,743,742,741]
745. ! [X0,X1] : ((::(X0,mten,X1) | ~sP2(X0,X1)) & (sP2(X0,X1) | ~::(X0,mten,X1))) [nnf transformation 582]
746. ! [X0,X1] : ((sP3(X0,X1) | (? [X2] : (? [X3,X4,X5] : (P(X5,X4) & SLC(X3,X5,X2) & SLC(X0,X4,X2) & M(X3)) & PRE(X0,X2)) | ! [X6] : ~SLC(X0,X6,X1) | ~PRE(X0,X1) | ~F(X0))) & ((! [X2] : (! [X3,X4,X5] : (~P(X5,X4) | ~SLC(X3,X5,X2) | ~SLC(X0,X4,X2) | ~M(X3)) | ~PRE(X0,X2)) & ? [X6] : SLC(X0,X6,X1) & PRE(X0,X1) & F(X0)) | ~sP3(X0,X1))) [nnf transformation 583]
747. ! [X0,X1] : ((sP3(X0,X1) | ? [X2] : (? [X3,X4,X5] : (P(X5,X4) & SLC(X3,X5,X2) & SLC(X0,X4,X2) & M(X3)) & PRE(X0,X2)) | ! [X6] : ~SLC(X0,X6,X1) | ~PRE(X0,X1) | ~F(X0)) & ((! [X2] : (! [X3,X4,X5] : (~P(X5,X4) | ~SLC(X3,X5,X2) | ~SLC(X0,X4,X2) | ~M(X3)) | ~PRE(X0,X2)) & ? [X6] : SLC(X0,X6,X1) & PRE(X0,X1) & F(X0)) | ~sP3(X0,X1))) [flattening 746]
748. ! [X0,X1] : ((sP3(X0,X1) | ? [X2] : (? [X3,X4,X5] : (P(X5,X4) & SLC(X3,X5,X2) & SLC(X0,X4,X2) & M(X3)) & PRE(X0,X2)) | ! [X6] : ~SLC(X0,X6,X1) | ~PRE(X0,X1) | ~F(X0)) & ((! [X7] : (! [X8,X9,X10] : (~P(X10,X9) | ~SLC(X8,X10,X7) | ~SLC(X0,X9,X7) | ~M(X8)) | ~PRE(X0,X7)) & ? [X11] : SLC(X0,X11,X1) & PRE(X0,X1) & F(X0)) | ~sP3(X0,X1))) [rectify 747]
749. ! [X0] : (? [X2] : (? [X3,X4,X5] : (P(X5,X4) & SLC(X3,X5,X2) & SLC(X0,X4,X2) & M(X3)) & PRE(X0,X2)) => (? [X5,X4,X3] : (P(X5,X4) & SLC(X3,X5,sK76(X0)) & SLC(X0,X4,sK76(X0)) & M(X3)) & PRE(X0,sK76(X0)))) [choice axiom]
750. ! [X0] : (? [X5,X4,X3] : (P(X5,X4) & SLC(X3,X5,sK76(X0)) & SLC(X0,X4,sK76(X0)) & M(X3)) => (P(sK79(X0),sK78(X0)) & SLC(sK77(X0),sK79(X0),sK76(X0)) & SLC(X0,sK78(X0),sK76(X0)) & M(sK77(X0)))) [choice axiom]
751. ! [X1,X0] : (? [X11] : SLC(X0,X11,X1) => SLC(X0,sK80(X0,X1),X1)) [choice axiom]
752. ! [X0,X1] : ((sP3(X0,X1) | ((P(sK79(X0),sK78(X0)) & SLC(sK77(X0),sK79(X0),sK76(X0)) & SLC(X0,sK78(X0),sK76(X0)) & M(sK77(X0))) & PRE(X0,sK76(X0))) | ! [X6] : ~SLC(X0,X6,X1) | ~PRE(X0,X1) | ~F(X0)) & ((! [X7] : (! [X8,X9,X10] : (~P(X10,X9) | ~SLC(X8,X10,X7) | ~SLC(X0,X9,X7) | ~M(X8)) | ~PRE(X0,X7)) & SLC(X0,sK80(X0,X1),X1) & PRE(X0,X1) & F(X0)) | ~sP3(X0,X1))) [skolemisation 748,751,750,749]
753. ! [X0,X1] : (((::(X0,cfbnd,X1) | ::(X0,site,X1)) | ~sP3(X0,X1)) & (sP3(X0,X1) | (~::(X0,cfbnd,X1) & ~::(X0,site,X1)))) [nnf transformation 584]
754. ! [X0,X1] : ((::(X0,cfbnd,X1) | ::(X0,site,X1) | ~sP3(X0,X1)) & (sP3(X0,X1) | (~::(X0,cfbnd,X1) & ~::(X0,site,X1)))) [flattening 753]
757. ! [X0,X1] : ((::(X0,imen,X1) | (~::(X0,sreg,X1) & ~::(X0,cfbnd,X1) & ~::(X0,site,X1))) & ((::(X0,sreg,X1) | ::(X0,cfbnd,X1) | ::(X0,site,X1)) | ~::(X0,imen,X1))) [nnf transformation 328]
758. ! [X0,X1] : ((::(X0,imen,X1) | (~::(X0,sreg,X1) & ~::(X0,cfbnd,X1) & ~::(X0,site,X1))) & (::(X0,sreg,X1) | ::(X0,cfbnd,X1) | ::(X0,site,X1) | ~::(X0,imen,X1))) [flattening 757]
759. ! [X0,X1] : ((::(X0,idcnt,X1) | (~::(X0,imen,X1) & ~::(X0,mten,X1))) & ((::(X0,imen,X1) | ::(X0,mten,X1)) | ~::(X0,idcnt,X1))) [nnf transformation 329]
760. ! [X0,X1] : ((::(X0,idcnt,X1) | (~::(X0,imen,X1) & ~::(X0,mten,X1))) & (::(X0,imen,X1) | ::(X0,mten,X1) | ~::(X0,idcnt,X1))) [flattening 759]
761. ! [X0,X1] : ((::(X0,sdcnt,X1) | (! [X2] : (~DQT(X0,X2) | S(X2) | ~::(X2,idcnt,X1)) | ~PRE(X0,X1) | ~Q(X0))) & ((? [X2] : (DQT(X0,X2) & ~S(X2) & ::(X2,idcnt,X1)) & PRE(X0,X1) & Q(X0)) | ~::(X0,sdcnt,X1))) [nnf transformation 330]
762. ! [X0,X1] : ((::(X0,sdcnt,X1) | ! [X2] : (~DQT(X0,X2) | S(X2) | ~::(X2,idcnt,X1)) | ~PRE(X0,X1) | ~Q(X0)) & ((? [X2] : (DQT(X0,X2) & ~S(X2) & ::(X2,idcnt,X1)) & PRE(X0,X1) & Q(X0)) | ~::(X0,sdcnt,X1))) [flattening 761]
763. ! [X0,X1] : ((::(X0,sdcnt,X1) | ! [X2] : (~DQT(X0,X2) | S(X2) | ~::(X2,idcnt,X1)) | ~PRE(X0,X1) | ~Q(X0)) & ((? [X3] : (DQT(X0,X3) & ~S(X3) & ::(X3,idcnt,X1)) & PRE(X0,X1) & Q(X0)) | ~::(X0,sdcnt,X1))) [rectify 762]
764. ! [X1,X0] : (? [X3] : (DQT(X0,X3) & ~S(X3) & ::(X3,idcnt,X1)) => (DQT(X0,sK81(X0,X1)) & ~S(sK81(X0,X1)) & ::(sK81(X0,X1),idcnt,X1))) [choice axiom]
765. ! [X0,X1] : ((::(X0,sdcnt,X1) | ! [X2] : (~DQT(X0,X2) | S(X2) | ~::(X2,idcnt,X1)) | ~PRE(X0,X1) | ~Q(X0)) & (((DQT(X0,sK81(X0,X1)) & ~S(sK81(X0,X1)) & ::(sK81(X0,X1),idcnt,X1)) & PRE(X0,X1) & Q(X0)) | ~::(X0,sdcnt,X1))) [skolemisation 763,764]
771. ! [X0,X1] : ((::(X0,gdcnt,X1) | (? [X2] : (! [X3] : ~CONCR(X3,X0,X2) & PRE(X0,X2)) | ~PRE(X0,X1))) & ((! [X2] : (? [X3] : CONCR(X3,X0,X2) | ~PRE(X0,X2)) & PRE(X0,X1)) | ~::(X0,gdcnt,X1))) [nnf transformation 574]
772. ! [X0,X1] : ((::(X0,gdcnt,X1) | ? [X2] : (! [X3] : ~CONCR(X3,X0,X2) & PRE(X0,X2)) | ~PRE(X0,X1)) & ((! [X2] : (? [X3] : CONCR(X3,X0,X2) | ~PRE(X0,X2)) & PRE(X0,X1)) | ~::(X0,gdcnt,X1))) [flattening 771]
773. ! [X0,X1] : ((::(X0,gdcnt,X1) | ? [X2] : (! [X3] : ~CONCR(X3,X0,X2) & PRE(X0,X2)) | ~PRE(X0,X1)) & ((! [X4] : (? [X5] : CONCR(X5,X0,X4) | ~PRE(X0,X4)) & PRE(X0,X1)) | ~::(X0,gdcnt,X1))) [rectify 772]
```

```
774. ! [X0] : (? [X2] : (! [X3] : ~CONCR(X3,X0,X2) & PRE(X0,X2)) => (! [X3] : ~CONCR(X3,X0,sK83(X0)) & PRE(X0,sK83
     (X0)))) [choice axiom]
775. ! [X4,X0] : (? [X5] : CONCR(X5,X0,X4) => CONCR(sK84(X0,X4),X0,X4)) [choice axiom]
776. ! [X0,X1] : ((::(X0,gdcnt,X1) | (! [X3] : ~CONCR(X3,X0,sK83(X0)) & PRE(X0,sK83(X0))) | ~PRE(X0,X1)) & ((! [X4
     ] : (CONCR(sK84(X0,X4),X0,X4) | ~PRE(X0,X4)) & PRE(X0,X1)) | ~::(X0,gdcnt,X1))) [skolemisation 773,775,774]
777. ! [X0,X1] : ((::(X0,cnt,X1) | (~::(X0,gdcnt,X1) & ~::(X0,sdcnt,X1) & ~::(X0,idcnt,X1))) & ((::(X0,gdcnt,X1) |
      ::(X0,sdcnt,X1) | ::(X0,idcnt,X1)) | ~::(X0,cnt,X1))) [nnf transformation 333]
778. ! [X0,X1] : ((::(X0,cnt,X1) | (~::(X0,gdcnt,X1) & ~::(X0,sdcnt,X1) & ~::(X0,idcnt,X1))) & (::(X0,gdcnt,X1) |
     ::(X0,sdcnt,X1) | ::(X0,idcnt,X1) | ~::(X0,cnt,X1))) [flattening 777]
779. ? [X0,X1] : (~PRE(X0,X1) & ~::(X0,sreg,X1) & ::(X0,cnt,X1)) => (~PRE(sK85,sK86) & ~::(sK85,sreg,sK86) & ::(
     sK85,cnt,sK86)) [choice axiom]
780. ~PRE(sK85,sK86) & ~::(sK85,sreg,sK86) & ::(sK85,cnt,sK86) [skolemisation 576,779]
1084. ~sP2(X0,X1) | PRE(X0,X1) [cnf transformation 744]
1093. ~::(X0,mten,X1) | sP2(X0,X1) [cnf transformation 745]
1096. ~sP3(X0,X1) | PRE(X0,X1) [cnf transformation 752]
1104. ~::(X0,site,X1) | sP3(X0,X1) [cnf transformation 754]
1105. ~::(X0,cfbnd,X1) | sP3(X0,X1) [cnf transformation 754]
1110. ~::(X0,imen,X1) | ::(X0,cfbnd,X1) | ::(X0,site,X1) | ::(X0,sreg,X1) [cnf transformation 758]
1114. ~::(X0,idcnt,X1) | ::(X0,mten,X1) | ::(X0,imen,X1) [cnf transformation 760]
1118. ~::(X0,sdcnt,X1) | PRE(X0,X1) [cnf transformation 765]
1128. ~::(X0,gdcnt,X1) | PRE(X0,X1) [cnf transformation 776]
1132. ~::(X0,cnt,X1) | ::(X0,sdcnt,X1) | ::(X0,idcnt,X1) | ::(X0,gdcnt,X1) [cnf transformation 778]
1136. ::(sK85,cnt,sK86) [cnf transformation 780]
1137. ~::(sK85,sreg,sK86) [cnf transformation 780]
1138. ~PRE(sK85,sK86) [cnf transformation 780]
2715. ::(sK85,sdcnt,sK86) | ::(sK85,idcnt,sK86) | ::(sK85,gdcnt,sK86) [resolution 1132,1136]
2719. 3 <=> ::(sK85,gdcnt,sK86) [avatar definition]
2721. ::(sK85,gdcnt,sK86) <- (3) [avatar component clause 2719]
2723. 4 <=> ::(sK85,idcnt,sK86) [avatar definition]
2725. ::(sK85,idcnt,sK86) <- (4) [avatar component clause 2723]
2727. 5 <=> ::(sK85,sdcnt,sK86) [avatar definition]
2729. ::(sK85,sdcnt,sK86) <- (5) [avatar component clause 2727]
2730. 3 | 4 | 5 [avatar split clause 2715,2727,2723,2719]
2733. PRE(sK85,sK86) <- (3) [resolution 2721,1128]
2734. $false <- (3) [subsumption resolution 2733,1138]
2735. ~3 [avatar contradiction clause 2734]
2750. 6 <=> ::(sK85,imen,sK86) [avatar definition]
2752. ::(sK85,imen,sK86) <- (6) [avatar component clause 2750]
2754. 7 <=> ::(sK85,mten,sK86) [avatar definition]
2756. ::(sK85,mten,sK86) <- (7) [avatar component clause 2754]
2804. PRE(sK85,sK86) <- (5) [resolution 2729,1118]
2806. $false <- (5) [subsumption resolution 2804,1138]
2807. ~5 [avatar contradiction clause 2806]
2809. ::(sK85,mten,sK86) | ::(sK85,imen,sK86) <- (4) [resolution 2725,1114]
2839. 8 <=> ::(sK85,site,sK86) [avatar definition]
2841. ::(sK85,site,sK86) <- (8) [avatar component clause 2839]
2843. 9 <=> ::(sK85,cfbnd,sK86) [avatar definition]
2845. ::(sK85,cfbnd,sK86) <- (9) [avatar component clause 2843]
2847. 6 | 7 | ~4 [avatar split clause 2809,2723,2754,2750]
2880. sP2(sK85,sK86) <- (7) [resolution 2756,1093]
2883. PRE(sK85,sK86) <- (7) [resolution 2880,1084]
2890. $false <- (7) [subsumption resolution 2883,1138]
2891. ~7 [avatar contradiction clause 2890]
2892. ::(sK85,cfbnd,sK86) | ::(sK85,site,sK86) | ::(sK85,sreg,sK86) <- (6) [resolution 2752,1110]
2902. sP3(sK85,sK86) <- (8) [resolution 2841,1104]
2904. PRE(sK85,sK86) <- (8) [resolution 2902,1096]
2915. $false <- (8) [subsumption resolution 2904,1138]
2916. ~8 [avatar contradiction clause 2915]
2917. ::(sK85,cfbnd,sK86) | ::(sK85,site,sK86) <- (6) [subsumption resolution 2892,1137]
2918. 8 | 9 | ~6 [avatar split clause 2917,2750,2843,2839]
2923. sP3(sK85,sK86) <- (9) [resolution 2845,1105]
2930. PRE(sK85,sK86) <- (9) [resolution 2923,1096]
2933. $false <- (9) [subsumption resolution 2930,1138]
2934. ~9 [avatar contradiction clause 2933]
2935. $false [avatar sat refutation 2730,2735,2807,2847,2891,2916,2918,2934]
```

11.4 Validation of BFO-DOLCE mapping (non-theorems)

These are the axioms of BFO taxonomy and the definitions of the predicates defined in BFO, which we added to each set of axioms reported below in order to obtain the full syntactical descriptions of the counterexamples. Since the taxonomy and the definitions are constant among the counterexamples, we write them only once.

BFO taxonomy

```
% BFO 2020 Axiomatization, generated 2021/11/12
% Author: Alan Ruttenberg - alanruttenberg@gmail.com
% The most recent version of this file will always be in the GitHub repository https://github.com/bfo-ontology/bfo-2020
% This work is licensed under a Creative Commons "Attribution 4.0 International" license: https://creativecommons.org/licenses/by/4.0/

% Section: Universal Declaration

% role is a universal
UNI(role) # label("role-is-a-universal") .

% site is a universal
UNI(site) # label("site-is-a-universal") .

% object is a universal
UNI(obj) # label("object-is-a-universal") .

% history is a universal
UNI(hist) # label("history-is-a-universal") .

% process is a universal
UNI(proc) # label("process-is-a-universal") .

% quality is a universal
UNI(qlt) # label("quality-is-a-universal") .

% function is a universal
UNI(fnt) # label("function-is-a-universal") .

% fiat-line is a universal
UNI(fln) # label("fiat-line-is-a-universal") .

% occurrent is a universal
UNI(occ) # label("occurrent-is-a-universal") .

% continuant is a universal
UNI(cnt) # label("continuant-is-a-universal") .

% fiat-point is a universal
UNI(fpt) # label("fiat-point-is-a-universal") .

% disposition is a universal
UNI(disp) # label("disposition-is-a-universal") .

% fiat-surface is a universal
UNI(fsf) # label("fiat-surface-is-a-universal") .

% spatial-region is a universal
UNI(sreg) # label("spatial-region-is-a-universal") .

% material-entity is a universal
UNI(mten) # label("material-entity-is-a-universal") .

% temporal-region is a universal
UNI(treg) # label("temporal-region-is-a-universal") .

% fiat-object-part is a universal
UNI(fobj) # label("fiat-object-part-is-a-universal") .

% object-aggregate is a universal
UNI(objagg) # label("object-aggregate-is-a-universal") .

% process-boundary is a universal
UNI(pbnd) # label("process-boundary-is-a-universal") .

% temporal-instant is a universal
UNI(tinst) # label("temporal-instant-is-a-universal") .

% immaterial-entity is a universal
UNI(imen) # label("immaterial-entity-is-a-universal") .

% realizable-entity is a universal
UNI(rlzen) # label("realizable-entity-is-a-universal") .

% temporal-interval is a universal
UNI(tint) # label("temporal-interval-is-a-universal") .
```

```
% relational-quality is a universal
UNI(rqlt) # label("relational-quality-is-a-universal") .

% spatiotemporal-region is a universal
UNI(streg) # label("spatiotemporal-region-is-a-universal") .

% independent-continuant is a universal
UNI(idcnt) # label("independent-continuant-is-a-universal") .

% continuant-fiat-boundary is a universal
UNI(cfbnd) # label("continuant-fiat-boundary-is-a-universal") .

% one-dimensional-spatial-region is a universal
UNI(sreg1) # label("one-dimensional-spatial-region-is-a-universal") .

% two-dimensional-spatial-region is a universal
UNI(sreg2) # label("two-dimensional-spatial-region-is-a-universal") .

% one-dimensional-temporal-region is a universal
UNI(treg1) # label("one-dimensional-temporal-region-is-a-universal") .

% zero-dimensional-spatial-region is a universal
UNI(sreg0) # label("zero-dimensional-spatial-region-is-a-universal") .

% ADDED - Universals and particulars comprise the whole domain of discourse
all x (UNI(x) | PAR(x)) # label("aggiunta2-universals-and-particulars-comprise-the-whole-domain-of-discourse") .

% generically-dependent-continuant is a universal
UNI(gdcnt) # label("generically-dependent-continuant-is-a-universal") .

% three-dimensional-spatial-region is a universal
UNI(sreg3) # label("three-dimensional-spatial-region-is-a-universal") .

% zero-dimensional-temporal-region is a universal
UNI(treg0) # label("zero-dimensional-temporal-region-is-a-universal") .

% specifically-dependent-continuant is a universal
UNI(sdcnt) # label("specifically-dependent-continuant-is-a-universal") .

% universals are not particulars
-(exists x  (((UNI(x)) & (PAR(x))))) # label("universals-particulars-disjoint") .

% history is subclass of process
all t all x  ((::(x,hist,t)) -> (::(x,proc,t))) # label("history-isa-process") .

% process is subclass of occurrent
all t all x  ((::(x,proc,t)) -> (::(x,occ,t))) # label("process-isa-occurrent") .

% function is subclass of disposition
all t all x  ((::(x,fnt,t)) -> (::(x,disp,t))) # label("function-isa-disposition") .

% object is subclass of material-entity
all t all x  ((::(x,obj,t)) -> (::(x,mten,t))) # label("object-isa-material-entity") .

% role is subclass of realizable-entity
all t all x  ((::(x,role,t)) -> (::(x,rlzen,t))) # label("role-isa-realizable-entity") .

% site is subclass of immaterial-entity
all t all x  ((::(x,site,t)) -> (::(x,imen,t))) # label("site-isa-immaterial-entity") .

% If something is an instance of temporal region at t, then t is part of that temporal region
all ti all t  ((::(ti,treg,t)) -> (tmP(t,ti))) # label("time-once") .

% temporal-region is subclass of occurrent
all t all x  ((::(x,treg,t)) -> (::(x,occ,t))) # label("temporal-region-isa-occurrent") .

% disposition, role are mutually disjoint
((-(exists x exists t  (((::(x,disp,t)) & (::(x,role,t))))))) # label("disposition+role-are-mutually-disjoint") .

% process-boundary is subclass of occurrent
all t all x  ((::(x,pbnd,t)) -> (::(x,occ,t))) # label("process-boundary-isa-occurrent") .

% relational-quality is subclass of quality
all t all x  ((::(x,rqlt,t)) -> (::(x,qlt,t))) # label("relational-quality-isa-quality") .

% disposition is subclass of realizable-entity
all t all x  ((::(x,disp,t)) -> (::(x,rlzen,t))) # label("disposition-isa-realizable-entity") .

% continuant, occurrent are mutually disjoint
((-(exists x exists t  (((::(x,cnt,t)) & (::(x,occ,t))))))) # label("continuant+occurrent-are-mutually-disjoint")
      .

% spatiotemporal-region is subclass of occurrent
all t all x  ((::(x,streg,t)) -> (::(x,occ,t))) # label("spatiotemporal-region-isa-occurrent") .

% fiat-object-part is subclass of material-entity
all t all x  ((::(x,fobj,t)) -> (::(x,mten,t))) # label("fiat-object-part-isa-material-entity") .

% object-aggregate is subclass of material-entity
all t all x  ((::(x,objagg,t)) -> (::(x,mten,t))) # label("object-aggregate-isa-material-entity") .
```

```
% spatial-region is subclass of immaterial-entity
all t all x  ((::(x,sreg,t)) -> (::(x,imen,t))) # label("spatial-region-isa-immaterial-entity") .

% independent-continuant is subclass of continuant
all t all x  ((::(x,idcnt,t)) -> (::(x,cnt,t))) # label("independent-continuant-isa-continuant") .

% fiat-line is subclass of continuant-fiat-boundary
all t all x  ((::(x,fln,t)) -> (::(x,cfbnd,t))) # label("fiat-line-isa-continuant-fiat-boundary") .

% quality, realizable-entity are mutually disjoint
((-(exists x exists t  (((::(x,qlt,t)) & (::(x,rlzen,t)))))) # label("quality+realizable-entity-are-mutually-
      disjoint") .

% fiat-point is subclass of continuant-fiat-boundary
all t all x  ((::(x,fpt,t)) -> (::(x,cfbnd,t))) # label("fiat-point-isa-continuant-fiat-boundary") .

% fiat-surface is subclass of continuant-fiat-boundary
all t all x  ((::(x,fsf,t)) -> (::(x,cfbnd,t))) # label("fiat-surface-isa-continuant-fiat-boundary") .

% material-entity is subclass of independent-continuant
all t all x  ((::(x,mten,t)) -> (::(x,idcnt,t))) # label("material-entity-isa-independent-continuant") .

% immaterial-entity is subclass of independent-continuant
all t all x  ((::(x,imen,t)) -> (::(x,idcnt,t))) # label("immaterial-entity-isa-independent-continuant") .

% quality is subclass of specifically-dependent-continuant
all t all x  ((::(x,qlt,t)) -> (::(x,sdcnt,t))) # label("quality-isa-specifically-dependent-continuant") .

% continuant-fiat-boundary is subclass of immaterial-entity
all t all x  ((::(x,cfbnd,t)) -> (::(x,imen,t))) # label("continuant-fiat-boundary-isa-immaterial-entity") .

% material-entity, immaterial-entity are mutually disjoint
((-(exists x exists t  (((::(x,mten,t)) & (::(x,imen,t)))))) # label("material-entity+immaterial-entity-are-
      mutually-disjoint") .

% generically-dependent-continuant is subclass of continuant
all t all x  ((::(x,gdcnt,t)) -> (::(x,cnt,t))) # label("generically-dependent-continuant-isa-continuant") .

% specifically-dependent-continuant is subclass of continuant
all t all x  ((::(x,sdcnt,t)) -> (::(x,cnt,t))) # label("specifically-dependent-continuant-isa-continuant") .

% one-dimensional-spatial-region is subclass of spatial-region
all t all x  ((::(x,sreg1,t)) -> (::(x,sreg,t))) # label("one-dimensional-spatial-region-isa-spatial-region") .

% two-dimensional-spatial-region is subclass of spatial-region
all t all x  ((::(x,sreg2,t)) -> (::(x,sreg,t))) # label("two-dimensional-spatial-region-isa-spatial-region") .

% zero-dimensional-spatial-region is subclass of spatial-region
all t all x  ((::(x,sreg0,t)) -> (::(x,sreg,t))) # label("zero-dimensional-spatial-region-isa-spatial-region") .

% one-dimensional-temporal-region is subclass of temporal-region
all t all x  ((::(x,treg1,t)) -> (::(x,treg,t))) # label("one-dimensional-temporal-region-isa-temporal-region") .

% three-dimensional-spatial-region is subclass of spatial-region
all t all x  ((::(x,sreg3,t)) -> (::(x,sreg,t))) # label("three-dimensional-spatial-region-isa-spatial-region") .

% zero-dimensional-temporal-region is subclass of temporal-region
all t all x  ((::(x,treg0,t)) -> (::(x,treg,t))) # label("zero-dimensional-temporal-region-isa-temporal-region") .

% temporal-instant is subclass of zero-dimensional-temporal-region
all t all x  ((::(x,tinst,t)) -> (::(x,treg0,t))) # label("temporal-instant-isa-zero-dimensional-temporal-region")
      .

% temporal-interval is subclass of one-dimensional-temporal-region
all t all x  ((::(x,tint,t)) -> (::(x,treg1,t))) # label("temporal-interval-isa-one-dimensional-temporal-region")
      .

% Entity is either universal or particular, so not all are instantiated. Instead make a predicate 'entity'
      analogous to particular universal
all x  ((exists t  (((::(x,cnt,t)) | (::(x,occ,t))))) -> (ENT(x))) # label("entity-predicate") .

% realizable-entity is subclass of specifically-dependent-continuant
all t all x  ((::(x,rlzen,t)) -> (::(x,sdcnt,t))) # label("realizable-entity-isa-specifically-dependent-continuant
      ") .

% If something is a role at any time then as long as it exists it is a role.
all x  ((exists t  (::(x,role,t))) -> (all t  ((EX(x,t)) -> (::(x,role,t))))) # label("role-is-rigid") .

% If something is a site at any time then as long as it exists it is a site.
all x  ((exists t  (::(x,site,t))) -> (all t  ((EX(x,t)) -> (::(x,site,t))))) # label("site-is-rigid") .

% If something is a quality at any time then as long as it exists it is a quality.
all x  ((exists t  (::(x,qlt,t))) -> (all t  ((EX(x,t)) -> (::(x,qlt,t))))) # label("quality-is-rigid") .

% If something is a function at any time then as long as it exists it is a function.
all x  ((exists t  (::(x,fnt,t))) -> (all t  ((EX(x,t)) -> (::(x,fnt,t))))) # label("function-is-rigid") .

% one-dimensional-temporal-region, zero-dimensional-temporal-region are mutually disjoint
((-(exists x exists t  (((::(x,treg1,t)) & (::(x,treg0,t)))))) # label("one-dimensional-temporal-region+zero-
      dimensional-temporal-region-are-mutually-disjoint") .
```

```
% If something is a fiat-line at any time then as long as it exists it is a fiat-line.
all x ((exists t (::(x,fln,t))) -> (all t ((EX(x,t)) -> (::(x,fln,t))))) # label("fiat-line-is-rigid") .

% If something is a continuant at any time then as long as it exists it is a continuant.
all x ((exists t (::(x,cnt,t))) -> (all t ((EX(x,t)) -> (::(x,cnt,t))))) # label("continuant-is-rigid") .

% If something is a fiat-point at any time then as long as it exists it is a fiat-point.
all x ((exists t (::(x,fpt,t))) -> (all t ((EX(x,t)) -> (::(x,fpt,t))))) # label("fiat-point-is-rigid") .

% If something is a disposition at any time then as long as it exists it is a disposition.
all x ((exists t (::(x,disp,t))) -> (all t ((EX(x,t)) -> (::(x,disp,t))))) # label("disposition-is-rigid") .

% If something is a fiat-surface at any time then as long as it exists it is a fiat-surface.
all x ((exists t (::(x,fsf,t))) -> (all t ((EX(x,t)) -> (::(x,fsf,t))))) # label("fiat-surface-is-rigid") .

% If something is a spatial-region at any time then as long as it exists it is a spatial-region.
all x ((exists t (::(x,sreg,t))) -> (all t ((EX(x,t)) -> (::(x,sreg,t))))) # label("spatial-region-is-rigid") .

% If something is a material-entity at any time then as long as it exists it is a material-entity.
all x ((exists t (::(x,mten,t))) -> (all t ((EX(x,t)) -> (::(x,mten,t))))) # label("material-entity-is-rigid")
    .

% If something is a immaterial-entity at any time then as long as it exists it is a immaterial-entity.
all x ((exists t (::(x,imen,t))) -> (all t ((EX(x,t)) -> (::(x,imen,t))))) # label("immaterial-entity-is-rigid
    ") .

% If something is a realizable-entity at any time then as long as it exists it is a realizable-entity.
all x ((exists t (::(x,rlzen,t))) -> (all t ((EX(x,t)) -> (::(x,rlzen,t))))) # label("realizable-entity-is-
    rigid") .

% If something is a relational-quality at any time then as long as it exists it is a relational-quality.
all x ((exists t (::(x,rqlt,t))) -> (all t ((EX(x,t)) -> (::(x,rqlt,t))))) # label("relational-quality-is-rigid
    ") .

% If something is a independent-continuant at any time then as long as it exists it is a independent-continuant.
all x ((exists t (::(x,idcnt,t))) -> (all t ((EX(x,t)) -> (::(x,idcnt,t))))) # label("independent-continuant-is
    -rigid") .

% If something is a continuant-fiat-boundary at any time then as long as it exists it is a continuant-fiat-
    boundary.
all x ((exists t (::(x,cfbnd,t))) -> (all t ((EX(x,t)) -> (::(x,cfbnd,t))))) # label("continuant-fiat-boundary-
    is-rigid") .

% If something is a one-dimensional-spatial-region at any time then as long as it exists it is a one-dimensional-
    spatial-region.
all x ((exists t (::(x,sreg1,t))) -> (all t ((EX(x,t)) -> (::(x,sreg1,t))))) # label("one-dimensional-spatial-
    region-is-rigid") .

% If something is a two-dimensional-spatial-region at any time then as long as it exists it is a two-dimensional-
    spatial-region.
all x ((exists t (::(x,sreg2,t))) -> (all t ((EX(x,t)) -> (::(x,sreg2,t))))) # label("two-dimensional-spatial-
    region-is-rigid") .

% If something is a zero-dimensional-spatial-region at any time then as long as it exists it is a zero-dimensional
    -spatial-region.
all x ((exists t (::(x,sreg0,t))) -> (all t ((EX(x,t)) -> (::(x,sreg0,t))))) # label("zero-dimensional-spatial-
    region-is-rigid") .

% If something is a generically-dependent-continuant at any time then as long as it exists it is a generically-
    dependent-continuant.
all x ((exists t (::(x,gdcnt,t))) -> (all t ((EX(x,t)) -> (::(x,gdcnt,t))))) # label("generically-dependent-
    continuant-is-rigid") .

% If something is a three-dimensional-spatial-region at any time then as long as it exists it is a three-
    dimensional-spatial-region.
all x ((exists t (::(x,sreg3,t))) -> (all t ((EX(x,t)) -> (::(x,sreg3,t))))) # label("three-dimensional-spatial
    -region-is-rigid") .

% If something is a specifically-dependent-continuant at any time then as long as it exists it is a specifically-
    dependent-continuant.
all x ((exists t (::(x,sdcnt,t))) -> (all t ((EX(x,t)) -> (::(x,sdcnt,t))))) # label("specifically-dependent-
    continuant-is-rigid") .

% No occurrent changes type during its existence
all o ((exists t (::(o,occ,t))) -> (all u ((exists t (::(o,u,t))) -> (all t (((::(o,occ,t)) <-> (::(o,u,t))))
    )) # label("all-occurrent-types-are-rigid") .

% fiat-surface, fiat-line, fiat-point are mutually disjoint
((-(exists x exists t (((::(x,fsf,t)) & (::(x,fln,t)))))) & (-(exists x exists t (((::(x,fsf,t)) & (::(x,fpt,t))
    )))) & (-(exists x exists t (((::(x,fln,t)) & (::(x,fpt,t))))))) # label("fiat-surface+fiat-line+fiat-point
    -are-mutually-disjoint") .

% site, spatial-region, continuant-fiat-boundary are mutually disjoint
((-(exists x exists t (((::(x,site,t)) & (::(x,sreg,t)))))) & (-(exists x exists t (((::(x,site,t)) & (::(x,
    cfbnd,t)))))) & (-(exists x exists t (((::(x,sreg,t)) & (::(x,cfbnd,t))))))) # label("site+spatial-region+
    continuant-fiat-boundary-are-mutually-disjoint") .

% specifically-dependent-continuant, independent-continuant, generically-dependent-continuant are mutually
    disjoint
```

```
((-(exists x exists t  (((::(x,sdcnt,t)) & (::(x,idcnt,t)))))) & (-(exists x exists t  (((::(x,sdcnt,t)) & (::(x,
    gdcnt,t)))))) & (-(exists x exists t  (((::(x,idcnt,t)) & (::(x,gdcnt,t)))))) # label("specifically-
    dependent-continuant+independent-continuant+generically-dependent-continuant-are-mutually-disjoint") .

% process, spatiotemporal-region, process-boundary, temporal-region are mutually disjoint
((-(exists x exists t  (((::(x,proc,t)) & (::(x,streg,t)))))) & (-(exists x exists t  (((::(x,proc,t)) & (::(x,
    pbnd,t)))))) & (-(exists x exists t  (((::(x,proc,t)) & (::(x,treg,t)))))) & (-(exists x exists t  (((::(x,
    streg,t)) & (::(x,pbnd,t)))))) & (-(exists x exists t  (((::(x,streg,t)) & (::(x,treg,t)))))) & (-(exists x
    exists t  (((::(x,pbnd,t)) & (::(x,treg,t)))))) # label("process+spatiotemporal-region+process-boundary+
    temporal-region-are-mutually-disjoint") .

% continuant, material-entity, object, fiat-object-part, object-aggregate, site, immaterial-entity, continuant-
    fiat-boundary, fiat-surface, fiat-line, fiat-point, spatial-region, three-dimensional-spatial-region, two-
    dimensional-spatial-region, one-dimensional-spatial-region, zero-dimensional-spatial-region, independent-
    continuant, generically-dependent-continuant, specifically-dependent-continuant, quality, relational-quality
    , function, disposition, realizable-entity, role, occurrent, process, process-boundary, temporal-region,
    zero-dimensional-temporal-region, temporal-instant, one-dimensional-temporal-region, temporal-interval,
    history, spatiotemporal-region are all different

((-((cnt) = (mten))) & (-((cnt) = (obj))) & (-((cnt) = (fobj))) & (-((cnt) = (objagg))) & (-((cnt) = (site))) &
    (-((cnt) = (imen))) & (-((cnt) = (cfbnd))) & (-((cnt) = (fsf))) & (-((cnt) = (fln))) & (-((cnt) = (fpt))) &
    (-((cnt) = (sreg))) & (-((cnt) = (sreg3))) & (-((cnt) = (sreg2))) & (-((cnt) = (sreg1))) & (-((cnt) = (sreg0
    ))) & (-((cnt) = (idcnt))) & (-((cnt) = (gdcnt))) & (-((cnt) = (sdcnt))) & (-((cnt) = (qlt))) & (-((cnt) = (
    rqlt))) & (-((cnt) = (fnt))) & (-((cnt) = (disp))) & (-((cnt) = (rlzen))) & (-((cnt) = (role))) & (-((cnt) =
     (occ))) & (-((cnt) = (proc))) & (-((cnt) = (pbnd))) & (-((cnt) = (treg))) & (-((cnt) = (treg0))) & (-((cnt)
    = (tinst))) & (-((cnt) = (treg1))) & (-((cnt) = (tint))) & (-((cnt) = (hist))) & (-((cnt) = (streg))) &
    (-((mten) = (obj))) & (-((mten) = (fobj))) & (-((mten) = (objagg))) & (-((mten) = (site))) & (-((mten) = (
    imen))) & (-((mten) = (cfbnd))) & (-((mten) = (fsf))) & (-((mten) = (fln))) & (-((mten) = (fpt))) & (-((mten
    ) = (sreg))) & (-((mten) = (sreg3))) & (-((mten) = (sreg2))) & (-((mten) = (sreg1))) & (-((mten) = (sreg0)))
    & (-((mten) = (idcnt))) & (-((mten) = (gdcnt))) & (-((mten) = (sdcnt))) & (-((mten) = (qlt))) & (-((mten) =
     (rqlt))) & (-((mten) = (fnt))) & (-((mten) = (disp))) & (-((mten) = (rlzen))) & (-((mten) = (role))) & (-((
    mten) = (occ))) & (-((mten) = (proc))) & (-((mten) = (pbnd))) & (-((mten) = (treg))) & (-((mten) = (treg0)))
    & (-((mten) = (tinst))) & (-((mten) = (treg1))) & (-((mten) = (tint))) & (-((mten) = (hist))) & (-((mten) =
     (streg))) & (-((obj) = (fobj))) & (-((obj) = (objagg))) & (-((obj) = (site))) & (-((obj) = (imen))) & (-((
    obj) = (cfbnd))) & (-((obj) = (fsf))) & (-((obj) = (fln))) & (-((obj) = (fpt))) & (-((obj) = (sreg))) & (-((
    obj) = (sreg3))) & (-((obj) = (sreg2))) & (-((obj) = (sreg1))) & (-((obj) = (sreg0))) & (-((obj) = (idcnt)))
    & (-((obj) = (gdcnt))) & (-((obj) = (sdcnt))) & (-((obj) = (qlt))) & (-((obj) = (rqlt))) & (-((obj) = (fnt)
    )) & (-((obj) = (disp))) & (-((obj) = (rlzen))) & (-((obj) = (role))) & (-((obj) = (occ))) & (-((obj) = (
    proc))) & (-((obj) = (pbnd))) & (-((obj) = (treg))) & (-((obj) = (treg0))) & (-((obj) = (tinst))) & (-((obj)
     = (treg1))) &
(-((obj) = (tint))) & (-((obj) = (hist))) & (-((obj) = (streg))) & (-((fobj) = (objagg))) & (-((fobj) = (site))) &
    (-((fobj) = (imen))) & (-((fobj) = (cfbnd))) & (-((fobj) = (fsf))) & (-((fobj) = (fln))) & (-((fobj) = (fpt
    ))) & (-((fobj) = (sreg))) & (-((fobj) = (sreg3))) & (-((fobj) = (sreg2))) & (-((fobj) = (sreg1))) & (-((
    fobj) = (sreg0))) & (-((fobj) = (idcnt))) & (-((fobj) = (gdcnt))) & (-((fobj) = (sdcnt))) & (-((fobj) = (qlt
    ))) & (-((fobj) = (rqlt))) & (-((fobj) = (fnt))) & (-((fobj) = (disp))) & (-((fobj) = (rlzen))) & (-((fobj)
    = (role))) & (-((fobj) = (occ))) & (-((fobj) = (proc))) & (-((fobj) = (pbnd))) & (-((fobj) = (treg))) & (-((
    fobj) = (treg0))) & (-((fobj) = (tinst))) & (-((fobj) = (treg1))) & (-((fobj) = (tint))) & (-((fobj) = (hist
    ))) & (-((fobj) = (streg))) & (-((objagg) = (site))) & (-((objagg) = (imen))) & (-((objagg) = (cfbnd))) &
    (-((objagg) = (fsf))) & (-((objagg) = (fln))) & (-((objagg) = (fpt))) & (-((objagg) = (sreg))) & (-((objagg)
    = (sreg3))) & (-((objagg) = (sreg2))) & (-((objagg) = (sreg1))) & (-((objagg) = (sreg0))) & (-((objagg) = (
    idcnt))) & (-((objagg) = (gdcnt))) & (-((objagg) = (sdcnt))) & (-((objagg) = (qlt))) & (-((objagg) = (rqlt))
    ) & (-((objagg) = (fnt))) & (-((objagg) = (disp))) & (-((objagg) = (rlzen))) & (-((objagg) = (role))) & (-((
    objagg) = (occ))) & (-((objagg) = (proc))) & (-((objagg) = (pbnd))) & (-((objagg) = (treg))) & (-((objagg) =
     (treg0))) & (-((objagg) = (tinst))) & (-((objagg) = (treg1))) & (-((objagg) = (tint))) & (-((objagg) = (
    hist))) & (-((objagg) = (streg))) & (-((site) = (imen))) & (-((site) = (cfbnd))) & (-((site) = (fsf))) &
    (-((site) = (fln))) & (-((site) = (fpt))) & (-((site) = (sreg))) & (-((site) = (sreg3))) & (-((site) = (
    sreg2))) & (-((site) = (sreg1))) & (-((site) = (sreg0))) & (-((site) = (idcnt))) & (-((site) = (gdcnt))) &
    (-((site) = (sdcnt))) & (-((site) = (qlt))) & (-((site) = (rqlt))) & (-((site) = (fnt))) & (-((site) = (disp
    ))) & (-((site) = (rlzen))) & (-((site) = (role))) & (-((site) = (occ))) & (-((site) = (proc))) & (-((site)
    = (pbnd))) & (-((site) = (treg))) & (-((site) = (treg0))) & (-((site) = (tinst))) & (-((site) = (treg1))) &
    (-((site) = (tint))) & (-((site) = (hist))) & (-((site) = (streg))) & (-((imen) = (cfbnd))) & (-((imen) = (
    fsf))) & (-((imen) = (fln))) & (-((imen) = (fpt))) & (-((imen) = (sreg))) & (-((imen) = (sreg3))) & (-((imen
    ) = (sreg2))) & (-((imen) = (sreg1))) & (-((imen) = (sreg0))) & (-((imen) = (idcnt))) & (-((imen) = (gdcnt))
    ) & (-((imen) = (sdcnt))) & (-((imen) = (qlt))) & (-((imen) = (rqlt))) & (-((imen) = (fnt))) & (-((imen) = (
    disp))) & (-((imen) = (rlzen))) & (-((imen) = (role))) & (-((imen) = (occ))) & (-((imen) = (proc))) & (-((
    imen) = (pbnd))) & (-((imen) = (treg))) & (-((imen) = (treg0))) & (-((imen) = (tinst))) & (-((imen) = (treg1
    ))) & (-((imen) = (tint))) & (-((imen) = (hist))) & (-((imen) = (streg))) & (-((cfbnd) = (fsf))) & (-((cfbnd
    ) = (fln))) & (-((cfbnd) = (fpt))) & (-((cfbnd) = (sreg))) & (-((cfbnd) = (sreg3))) & (-((cfbnd) = (sreg2)))
    & (-((cfbnd) = (sreg1))) & (-((cfbnd) = (sreg0))) & (-((cfbnd) = (idcnt))) & (-((cfbnd) = (gdcnt))) & (-((
    cfbnd) = (sdcnt))) & (-((cfbnd) = (qlt))) & (-((cfbnd) = (rqlt))) & (-((cfbnd) = (fnt))) & (-((cfbnd) = (
    disp))) & (-((cfbnd) = (rlzen))) & (-((cfbnd) = (role))) & (-((cfbnd) = (occ))) & (-((cfbnd) = (proc))) &
    (-((cfbnd) = (pbnd))) & (-((cfbnd) = (treg))) & (-((cfbnd) = (treg0))) & (-((cfbnd) = (tinst))) & (-((cfbnd)
     = (treg1))) & (-((cfbnd) = (tint))) & (-((cfbnd) = (hist))) & (-((cfbnd) = (streg))) &

(-((fsf) = (fln))) & (-((fsf) = (fpt))) & (-((fsf) = (sreg))) & (-((fsf) = (sreg3))) & (-((fsf) = (sreg2))) & (-((
    fsf) = (sreg1))) & (-((fsf) = (sreg0))) & (-((fsf) = (idcnt))) & (-((fsf) = (gdcnt))) & (-((fsf) = (sdcnt)))
    & (-((fsf) = (qlt))) & (-((fsf) = (rqlt))) & (-((fsf) = (fnt))) & (-((fsf) = (disp))) & (-((fsf) = (rlzen))
    ) & (-((fsf) = (role))) & (-((fsf) = (occ))) & (-((fsf) = (proc))) & (-((fsf) = (pbnd))) & (-((fsf) = (treg
    ))) & (-((fsf) = (treg0))) & (-((fsf) = (tinst))) & (-((fsf) = (treg1))) & (-((fsf) = (tint))) & (-((fsf) = (
    hist))) & (-((fsf) = (streg))) & (-((fln) = (fpt))) & (-((fln) = (sreg))) & (-((fln) = (sreg3))) & (-((fln)
    = (sreg2))) & (-((fln) = (sreg1))) & (-((fln) = (sreg0))) & (-((fln) = (idcnt))) & (-((fln) = (gdcnt))) &
    (-((fln) = (sdcnt))) & (-((fln) = (qlt))) & (-((fln) = (rqlt))) & (-((fln) = (fnt))) & (-((fln) = (disp))) &
     (-((fln) = (rlzen))) & (-((fln) = (role))) & (-((fln) = (occ))) & (-((fln) = (proc))) & (-((fln) = (pbnd)))
    & (-((fln) = (treg))) & (-((fln) = (treg0))) & (-((fln) = (tinst))) & (-((fln) = (treg1))) & (-((fln) = (
    tint))) & (-((fln) = (hist))) & (-((fln) = (streg))) & (-((fpt) = (sreg))) & (-((fpt) = (sreg3))) & (-((fpt)
     = (sreg2))) & (-((fpt) = (sreg1))) & (-((fpt) = (sreg0))) & (-((fpt) = (idcnt))) & (-((fpt) = (gdcnt))) &
    (-((fpt) = (sdcnt))) & (-((fpt) = (qlt))) & (-((fpt) = (rqlt))) & (-((fpt) = (fnt))) & (-((fpt) = (disp))) &
     (-((fpt) = (rlzen))) & (-((fpt) = (role))) & (-((fpt) = (occ))) & (-((fpt) = (proc))) & (-((fpt) = (pbnd)))
    & (-((fpt) = (treg))) & (-((fpt) = (treg0))) & (-((fpt) = (tinst))) & (-((fpt) = (treg1))) & (-((fpt) = (
    tint))) & (-((fpt) = (hist))) & (-((fpt) = (streg))) & (-((sreg) = (sreg3))) & (-((sreg) = (sreg2))) & (-((
    sreg) = (sreg1))) & (-((sreg) = (sreg0))) & (-((sreg) = (idcnt))) & (-((sreg) = (gdcnt))) & (-((sreg) = (
```

```
sdcnt))) & (-((sreg) = (qlt))) & (-((sreg) = (rqlt))) & (-((sreg) = (fnt))) & (-((sreg) = (disp))) & (-((
sreg) = (rlzen))) & (-((sreg) = (role))) & (-((sreg) = (occ))) & (-((sreg) = (proc))) & (-((sreg) = (pbnd)))
 & (-((sreg) = (treg))) & (-((sreg) = (treg0))) & (-((sreg) = (tinst))) & (-((sreg) = (treg1))) & (-((sreg)
= (tint))) & (-((sreg) = (hist))) & (-((sreg) = (streg))) & (-((sreg3) = (sreg2))) & (-((sreg3) = (sreg1)))
&
(-((sreg3) = (sreg0))) & (-((sreg3) = (idcnt))) & (-((sreg3) = (gdcnt))) & (-((sreg3) = (sdcnt))) & (-((sreg3) = (
qlt))) & (-((sreg3) = (rqlt))) & (-((sreg3) = (fnt))) & (-((sreg3) = (disp))) & (-((sreg3) = (rlzen))) &
(-((sreg3) = (role))) & (-((sreg3) = (occ))) & (-((sreg3) = (proc))) & (-((sreg3) = (pbnd))) & (-((sreg3) =
(treg))) & (-((sreg3) = (treg0))) & (-((sreg3) = (tinst))) & (-((sreg3) = (treg1))) & (-((sreg3) = (tint)))
& (-((sreg3) = (hist))) & (-((sreg3) = (streg))) & (-((sreg2) = (sreg1))) & (-((sreg2) = (sreg0))) & (-((
sreg2) = (idcnt))) & (-((sreg2) = (gdcnt))) & (-((sreg2) = (sdcnt))) & (-((sreg2) = (qlt))) & (-((sreg2) = (
rqlt))) & (-((sreg2) = (fnt))) & (-((sreg2) = (disp))) & (-((sreg2) = (rlzen))) & (-((sreg2) = (role))) &
(-((sreg2) = (occ))) & (-((sreg2) = (proc))) & (-((sreg2) = (pbnd))) & (-((sreg2) = (treg))) & (-((sreg2) =
(treg0))) & (-((sreg2) = (tinst))) & (-((sreg2) = (treg1))) & (-((sreg2) = (tint))) & (-((sreg2) = (hist)))
& (-((sreg2) = (streg))) & (-((sreg1) = (sreg0))) & (-((sreg1) = (idcnt))) & (-((sreg1) = (gdcnt))) &

(-((sreg1) = (sdcnt))) & (-((sreg1) = (qlt))) & (-((sreg1) = (rqlt))) & (-((sreg1) = (fnt))) & (-((sreg1) = (disp
))) & (-((sreg1) = (rlzen))) & (-((sreg1) = (role))) & (-((sreg1) = (occ))) & (-((sreg1) = (proc))) & (-((
sreg1) = (pbnd))) & (-((sreg1) = (treg))) & (-((sreg1) = (treg0))) & (-((sreg1) = (tinst))) & (-((sreg1) =
(treg1))) & (-((sreg1) = (tint))) & (-((sreg1) = (hist))) & (-((sreg1) = (streg))) & (-((sreg0) = (idcnt)))
 & (-((sreg0) = (gdcnt))) & (-((sreg0) = (sdcnt))) & (-((sreg0) = (qlt))) & (-((sreg0) = (rqlt))) & (-((
sreg0) = (fnt))) & (-((sreg0) = (disp))) & (-((sreg0) = (rlzen))) & (-((sreg0) = (role))) & (-((sreg0) = (
occ))) & (-((sreg0) = (proc))) & (-((sreg0) = (pbnd))) & (-((sreg0) = (treg))) & (-((sreg0) = (treg0))) &
(-((sreg0) = (tinst))) & (-((sreg0) = (treg1))) & (-((sreg0) = (tint))) & (-((sreg0) = (hist))) & (-((sreg0
) = (streg))) & (-((idcnt) = (gdcnt))) & (-((idcnt) = (sdcnt))) & (-((idcnt) = (qlt))) & (-((idcnt) = (rqlt
))) & (-((idcnt) = (fnt))) & (-((idcnt) = (disp))) & (-((idcnt) = (rlzen))) & (-((idcnt) = (role))) & (-((
idcnt) = (occ))) & (-((idcnt) = (proc))) & (-((idcnt) = (pbnd))) & (-((idcnt) = (treg))) & (-((idcnt) = (
treg0))) & (-((idcnt) = (tinst))) & (-((idcnt) = (treg1))) & (-((idcnt) = (tint))) & (-((idcnt) = (hist)))
& (-((idcnt) = (streg))) & (-((gdcnt) = (sdcnt))) & (-((gdcnt) = (qlt))) & (-((gdcnt) = (rqlt))) & (-((
gdcnt) = (fnt))) & (-((gdcnt) = (disp))) & (-((gdcnt) = (rlzen))) & (-((gdcnt) = (role))) & (-((gdcnt) = (
occ))) & (-((gdcnt) = (proc))) & (-((gdcnt) = (pbnd))) & (-((gdcnt) = (treg))) & (-((gdcnt) = (treg0))) &
(-((gdcnt) = (tinst))) & (-((gdcnt) = (treg1))) & (-((gdcnt) = (tint))) & (-((gdcnt) = (hist))) & (-((gdcnt
) = (streg))) & (-((sdcnt) = (qlt))) & (-((sdcnt) = (rqlt))) & (-((sdcnt) = (fnt))) & (-((sdcnt) = (disp)))
 & (-((sdcnt) = (rlzen))) & (-((sdcnt) = (role))) & (-((sdcnt) = (occ))) & (-((sdcnt) = (proc))) & (-((
sdcnt) = (pbnd))) & (-((sdcnt) = (treg))) & (-((sdcnt) = (treg0))) & (-((sdcnt) = (tinst))) & (-((sdcnt) =
(treg1))) & (-((sdcnt) = (tint))) & (-((sdcnt) = (hist))) & (-((sdcnt) = (streg))) & (-((qlt) = (rqlt))) &
(-((qlt) = (fnt))) & (-((qlt) = (disp))) & (-((qlt) = (rlzen))) & (-((qlt) = (role))) & (-((qlt) = (occ)))
& (-((qlt) = (proc))) & (-((qlt) = (pbnd))) & (-((qlt) = (treg))) & (-((qlt) = (treg0))) & (-((qlt) = (
tinst))) & (-((qlt) = (treg1))) & (-((qlt) = (tint))) & (-((qlt) = (hist))) & (-((qlt) = (streg))) & (-((
rqlt) = (fnt))) & (-((rqlt) = (disp))) & (-((rqlt) = (rlzen))) & (-((rqlt) = (role))) & (-((rqlt) = (occ)))
 & (-((rqlt) = (proc))) & (-((rqlt) = (pbnd))) & (-((rqlt) = (treg))) & (-((rqlt) = (treg0))) & (-((rqlt) =
 (tinst))) & (-((rqlt) = (treg1))) & (-((rqlt) = (tint))) & (-((rqlt) = (hist))) &
(-((rqlt) = (streg))) & (-((fnt) = (disp))) & (-((fnt) = (rlzen))) & (-((fnt) = (role))) & (-((fnt) = (occ))) &
(-((fnt) = (proc))) & (-((fnt) = (pbnd))) & (-((fnt) = (treg))) & (-((fnt) = (treg0))) & (-((fnt) = (tinst))
) & (-((fnt) = (treg1))) & (-((fnt) = (tint))) & (-((fnt) = (hist))) & (-((fnt) = (streg))) & (-((disp) = (
rlzen))) & (-((disp) = (role))) & (-((disp) = (occ))) & (-((disp) = (proc))) & (-((disp) = (pbnd))) & (-((
disp) = (treg))) & (-((disp) = (treg0))) & (-((disp) = (tinst))) & (-((disp) = (treg1))) & (-((disp) = (tint
))) & (-((disp) = (hist))) & (-((disp) = (streg))) & (-((rlzen) = (role))) & (-((rlzen) = (occ))) & (-((
rlzen) = (proc))) & (-((rlzen) = (pbnd))) & (-((rlzen) = (treg))) & (-((rlzen) = (treg0))) & (-((rlzen) = (
tinst))) & (-((rlzen) = (treg1))) & (-((rlzen) = (tint))) & (-((rlzen) = (hist))) & (-((rlzen) = (streg))) &
 (-((role) = (occ))) & (-((role) = (proc))) & (-((role) = (pbnd))) & (-((role) = (treg))) & (-((role) = (
treg0))) & (-((role) = (tinst))) & (-((role) = (treg1))) & (-((role) = (tint))) & (-((role) = (hist))) &
(-((role) = (streg))) & (-((occ) = (proc))) & (-((occ) = (pbnd))) & (-((occ) = (treg))) & (-((occ) = (treg0)
)) & (-((occ) = (tinst))) & (-((occ) = (treg1))) & (-((occ) = (tint))) & (-((occ) = (hist))) & (-((occ) = (
streg))) & (-((proc) = (pbnd))) & (-((proc) = (treg))) & (-((proc) = (treg0))) & (-((proc) = (tinst))) &
(-((proc) = (treg1))) & (-((proc) = (tint))) & (-((proc) = (hist))) & (-((proc) = (streg))) & (-((pbnd) = (
treg))) & (-((pbnd) = (treg0))) & (-((pbnd) = (tinst))) & (-((pbnd) = (treg1))) & (-((pbnd) = (tint))) &
(-((pbnd) = (hist))) & (-((pbnd) = (streg))) & (-((treg) = (treg0))) & (-((treg) = (tinst))) & (-((treg) = (
treg1))) & (-((treg) = (tint))) & (-((treg) = (hist))) & (-((treg) = (streg))) & (-((treg0) = (tinst))) &
(-((treg0) = (treg1))) & (-((treg0) = (tint))) & (-((treg0) = (hist))) & (-((treg0) = (streg))) & (-((tinst)
 = (treg1))) & (-((tinst) = (tint))) & (-((tinst) = (hist))) & (-((tinst) = (streg))) & (-((treg1) = (tint))
) & (-((treg1) = (hist))) & (-((treg1) = (streg))) & (-((tint) = (hist))) & (-((tint) = (streg))) & (-((hist
) = (streg)))) # label("universals-all-different").

% zero-dimensional-spatial-region, one-dimensional-spatial-region, two-dimensional-spatial-region, three-
    dimensional-spatial-region are mutually disjoint

((-(exists x exists t  (((::(x,sreg0,t)) & (::(x,sreg1,t)))))) & (-(exists x exists t  (((::(x,sreg0,t)) & (::(x,
    sreg2,t)))))) & (-(exists x exists t  (((::(x,sreg0,t)) & (::(x,sreg3,t)))))) & (-(exists x exists t  (((::(
    x,sreg1,t)) & (::(x,sreg2,t)))))) & (-(exists x exists t  (((::(x,sreg1,t)) & (::(x,sreg3,t)))))) & (-(
    exists x exists t  (((::(x,sreg2,t)) & (::(x,sreg3,t))))))) # label("zero-dimensional-spatial-region+one-
    dimensional-spatial-region+two-dimensional-spatial-region+three-dimensional-spatial-region-are-mutually-
    disjoint").


%%%%%%%%%% UNIVERSAL CLOSURE

all x (UNI(x) <-> ((x = role) | (x = site) | (x = obj) | (x = hist) | (x = proc) | (x = qlt) | (x = fnt) | (x =
    fln) | (x = occ) | (x = cnt) | (x = fpt) | (x = disp) | (x = fsf) | (x = sreg) | (x = mten) | (x = treg) | (
    x = fobj) | (x = objagg) | (x = pbnd) | (x = tinst) | (x = imen) | (x = rlzen) | (x = tint) | (x = rqlt) | (
    x = streg) | (x = idcnt) | (x = cfbnd) | (x = sreg1) | (x = sreg2) | (x = treg1) | (x = sreg0) | (x = gdcnt)
     | (x = sreg3) | (x = treg0) | (x = sdcnt))).
% all x (UNI(x) <-> ((x = role) )).
% all x (UNI(x) <-> ( (x = sreg) | (x = mten) | (x = treg) | (x = fobj) | (x = objagg) | (x = pbnd) | (x = tinst)
    | (x = imen) | (x = rlzen) | (x = tint) | (x = rqlt) | (x = streg) | (x = idcnt) | (x = cfbnd) | (x = sreg1)
     | (x = sreg2) | (x = treg1) | (x = sreg0) | (x = gdcnt) | (x = sreg3) | (x = treg0) | (x = sdcnt))).
% all x (UNI(x) <-> ((x = role) | (x = site) | (x = obj) | (x = hist) | (x = proc) | (x = qlt) | (x = fnt) | (x =
    fln) | (x = occ) | (x = cnt) | (x = fpt) | (x = disp) | (x = fsf) )).

%%%%%%%%%% DEFINITIONS
```

```
% x proper continuant part of y means x is a continuant part of y but y is not continuant part of x
all x all y all t  ((cPP(x,y,t)) <-> (((cP(x,y,t)) & (-(cP(y,x,t)))))) # label("definition-of-proper-continuant-
    part-of") .

% A proper occurrent part of b means a is an occurrent part of b and a is not the same as b
all x all y  ((oPP(x,y)) <-> (((oP(x,y)) & (-((x) = (y)))))) # label("definition-of-proper-occurrent-part-of") .

% a inheres_in b =Def. a is a specifically dependent continuant and b is an independent continuant that is not a
    spatial region and a s-depends_on b.
all a all b  ((INH(a,b)) <-> (((SDEP(a,b)) & (exists t  (((::(a,sdcnt,t)) & (::(b,idcnt,t)) & (-(::(b,sreg,t))))))
    )) # label("inheres-in-definition") .

% a proper temporal part of b means a is a temporal part of b and b a is not the same as b
all x all y  ((tmPP(x,y)) <-> (((tmP(x,y)) & (-((x) = (y)))))) # label("definition-of-proper-temporal-part-of") .

% The temporal region during which a process occurs is the same as that which the spatiotemporal region the
    process occupies temporally projects onto
all p all t  ((TREG(p,t)) <-> (exists st  (((STREG(p,st)) & (TPROJ(st,t)))))) # label("process-occupies-temporal-
    region-same-as-st-region-occupies") .

% The first and last time points for an instant are the instant itself
all i  ((::(i,tinst,i)) <-> (((firstInstantOf(i,i)) & (lastInstantOf(i,i))))) # label("instant-first-and-last-
    instant-are-itself") .

% A fiat object part =def a proper part of an object
all f all t  ((::(f,fobj,t)) <-> (exists o  (((::(o,obj,t)) & (cPP(f,o,t)) & (-(::(f,imen,t))))))) # label("fiat-
    object-part-part-of-object") .

% i is an immaterial entity = Def. i is an independent continuant that has no material entities as parts.
all i all t  ((::(i,imen,t)) <-> (((::(i,idcnt,t)) & (-(exists m  (((::(m,mten,t)) & (cP(m,i,t)))))))) # label("
    immaterial-entity-definition") .

% DEFINITION: b is a relational quality = Def. b is a quality and there exists distinct c and d such that at all
    times t, b inheres in c if and only b specifically-depends-on.
all b  ((exists t  (::(b,rqlt,t))) <-> (((exists c exists d  (((-((c) = (d))) & (INH(b,c)) & (SDEP(b,d))))) & (
    exists t  (::(b,qlt,t)))))) # label("relational-quality-definition") .

% Definition of specifically dependent continuant.
all s  ((exists t  (::(s,sdcnt,t))) <-> (exists c exists t  (((::(s,cnt,t)) & (::(c,idcnt,t)) & (-(::(c,sreg,t)))
    & (SDEP(s,c)))))) # label("definition-of-specifically-dependent-continuant") .
```

Counter model to (t_{bd}7), (t_{bd}12), (t_{bd}14), (t_{bd}19), (t_{bd}21), (t_{bd}28), (t_{bd}30), (t_{bd}33), (t_{bd}34), (t_{bd}36), (t_{bd}42)

```
all X (PAR(X) <-> ((X = t1) | (X = t2) | (X = t12) | (X = s1) | (X = st1) | (X = st2) | (X = st12) | (X = pb1) | (
      X = p1) | (X = c1) | (X = c2) | (X = c3))).

(t1 != t2) & (t1 != t12) & (t2 != t12) &  (st1 != st2) & (st1 != st12) & (st2 != st12) & (c1 != c2) & (c1 != c3) &
      (c2 != c3) & (p1 != pb1).

(all X all Y all Z (::(X,Y,Z) <-> (X = t1 & Y = tinst & Z = t1 | X = t2 & Y = tinst & Z = t2 | X = t12 & Y = tint
      & Z = t1 | X = t12 & Y = tint & Z = t2 | X = t12 & Y = tint & Z = t12 | X = s1 & Y = sreg3 & Z = t1 | X = s1
       & Y = sreg3 & Z = t2 | X = s1 & Y = sreg3 & Z = t12 |  X = st1 & Y = streg & Z = t1 | X = st2 & Y = streg &
       Z = t2 | X = st12 & Y = streg & Z = t1 | X = st12 & Y = streg & Z = t2 | X = st12 & Y = streg & Z = t12 |
      X = pb1 & Y = pbnd & Z = t1 |  X = p1 & Y = proc & Z = t1 | X = p1 & Y = proc & Z = t2 | X = p1 & Y = proc &
       Z = t12 |  X = c1 & Y = site & Z = t1 | X = c1 & Y = site & Z = t2 | X = c1 & Y = site & Z = t12 | X = c2 &
       Y = site & Z = t1 | X = c2 & Y = site & Z = t2 |  X = c3 & Y = site & Z = t1 |  X = t1 & Y = treg0 & Z = t1
       | X = t2 & Y = treg0 & Z = t2 |  X = t12 & Y = treg1 & Z = t1 | X = t12 & Y = treg1 & Z = t2 | X = t12 & Y
      = treg1 & Z = t12 |  X = s1 & Y = sreg & Z = t1 | X = s1 & Y = sreg & Z = t2 | X = s1 & Y = sreg & Z = t12 |
       X = st1 & Y = occ & Z = t1 | X = st2 & Y = occ & Z = t2 | X = st12 & Y = occ & Z = t1 | X = st12 & Y = occ
       & Z = t2 | X = st12 & Y = occ & Z = t12 | X = pb1 & Y = occ & Z = t1 | X = p1 & Y = occ & Z = t1 | X = p1 &
       Y = occ & Z = t2 | X = p1 & Y = occ & Z = t12 | X = t1 & Y = occ & Z = t1 | X = t2 & Y = occ & Z = t2 | X =
       t12 & Y = occ & Z = t1 | X = t12 & Y = occ & Z = t2 | X = t12 & Y = occ & Z = t12 |  X = c1 & Y = imen & Z
      = t1 | X = c1 & Y = imen & Z = t2 | X = c1 & Y = imen & Z = t12 | X = c2 & Y = imen & Z = t1 | X = c2 & Y =
       imen & Z = t2 |  X = c3 & Y = imen & Z = t1 |  X = s1 & Y = imen & Z = t1 | X = s1 & Y = imen & Z = t2 | X =
       s1 & Y = imen & Z = t12 |  X = t1 & Y = treg & Z = t1 | X = t2 & Y = treg & Z = t2 | X = t12 & Y = treg & Z
      = t1 | X = t12 & Y = treg & Z = t2 | X = t12 & Y = treg & Z = t12 | X = c1 & Y = idcnt & Z = t1 | X = c1 &
      Y = idcnt & Z = t2 | X = c1 & Y = idcnt & Z = t12 | X = c2 & Y = idcnt & Z = t1 | X = c2 & Y = idcnt & Z =
      t2 | X = s1 & Y = idcnt & Z = t1 | X = s1 & Y = idcnt & Z = t2 | X = s1 & Y = idcnt & Z = t12 | X = c3 & Y =
       idcnt & Z = t1 | X = c1 & Y = cnt & Z = t1 | X = c1 & Y = cnt & Z = t2 | X = c1 & Y = cnt & Z = t12 | X =
      c2 & Y = cnt & Z = t1 | X = c2 & Y = cnt & Z = t2 | X = c3 & Y = cnt & Z = t1 | X = s1 & Y = cnt & Z = t1 |
      X = s1 & Y = cnt & Z = t2 | X = s1 & Y = cnt & Z = t12))).
all X all T(- (::(X,hist,T))).

all X all Y (EX(X,Y) <-> ((X = t1 & Y = t1) | (X = t2 & Y = t2) | (X = t12 & Y = t1) | (X = t12 & Y = t2) | (X =
      t12 & Y = t12) | (X = s1 & Y = t1) | (X = s1 & Y = t2) | (X = s1 & Y = t12) | (X = st1 & Y = t1) | (X = st2
      & Y = t2)  | (X = st12 & Y = t1) | (X = st12 & Y = t2) | (X = st12 & Y = t12) | (X = pb1 & Y = t1) | (X = p1
       & Y = t1) | (X = p1 & Y = t2) | (X = p1 & Y = t12) | (X = c1 & Y = t1) | (X = c1 & Y = t2) | (X = c1 & Y =
      t12) | (X = c2 & Y = t1) | (X = c2 & Y = t2) | (X = c3 & Y = t1))).

all X all Y all Z (cP(X,Y,Z) <-> ((X = s1 & Y = s1 & Z = t1) | (X = s1 & Y = s1 & Z = t2) | (X = s1 & Y = s1 & Z =
      t12) | (X = c1 & Y = c1 & Z = t1) | (X = c1 & Y = c1 & Z = t2) | (X = c1 & Y = c1 & Z = t12) | (X = c2 & Y
      = c2 & Z = t1) | (X = c2 & Y = c2 & Z = t2) | (X = c3 & Y = c3 & Z = t1))).

all X all Y (oP(X,Y) <-> ((X = t1 & Y = t1) | (X = t2 & Y = t2) | (X = t12 & Y = t12) | (X = t1 & Y = t12) | (X =
      t2 & Y = t12) | (X = st1 & Y = st1) | (X = st2 & Y = st2) | (X = st12 & Y = st12) | (X = st1 & Y = st12) | (
      X = st2 & Y = st12) | (X = pb1 & Y = pb1) | (X = p1 & Y = p1) | (X = pb1 & Y = p1))).

all X all Y all Z (-(mP(X,Y,Z))).

all X all Y (tmP(X,Y) <-> ((X = t1 & Y = t1) | (X = t2 & Y = t2) | (X = t12 & Y = t12) | (X = t1 & Y = t12) | (X =
      t2 & Y = t12) | (X = st1 & Y = st1) | (X = st2 & Y = st2) | (X = st12 & Y = st12) | (X = st1 & Y = st12) |
      (X = st2 & Y = st12) | (X = pb1 & Y = pb1) | (X = p1 & Y = p1) | (X = pb1 & Y = p1))).

all X all Y (STREG(X,Y) <-> ((X = pb1 & Y = st1) | (X = p1 & Y = st12))).

all X all Y all Z (SREG(X,Y,Z) <-> ((X = c1 & Y = s1 & Z = t1) | (X = c1 & Y = s1 & Z = t2) | (X = c1 & Y = s1 & Z
      = t12) | (X = c2 & Y = s1 & Z = t1) | (X = c2 & Y = s1 & Z = t2) | (X = c3 & Y = s1 & Z = t1))).

all X all Y (TREG(X,Y) <-> ((X = pb1 & Y = t1) | (X = p1 & Y = t12))).

all X all Y (TPROJ(X,Y) <-> ((X = st1 & Y = t1) | (X = st2 & Y = t2) | (X = st12 & Y = t12))).

all X all Y all Z (SPROJ(X,Y,Z) <-> ((X = st1 & Y = s1 & Z = t1) | (X = st2 & Y = s1 & Z = t2) | (X = st12 & Y =
      s1 & Z = t1) | (X = st12 & Y = s1 & Z = t2) | (X = st12 & Y = s1 & Z = t12))).

all X all Y (-(OCCIN(X,Y))).

all X all Y all Z (LOC(X,Y,Z) <-> ((X = c1 & Y = c1 & Z = t1) | (X = c1 & Y = c1 & Z = t2) | (X = c1 & Y = c1 & Z
      = t12) | (X = c2 & Y = c2 & Z = t1) | (X = c2 & Y = c2 & Z = t2) | (X = c3 & Y = c3 & Z = t1))).

all X all Y (-(SDEP(X,Y))).
all X all Y all T (-(CONCR(X,Y,T))).
all X all Y all Z (-(GDEP(X,Y,Z))).

all X all Y all Z (PTC(X,Y,Z) <-> ((X = c1 & Y = p1 & Z = t1) | (X = c1 & Y = p1 & Z = t2) | (X = c1 & Y = p1 & Z
      = t12) | (X = c2 & Y = p1 & Z = t1) | (X = c2 & Y = p1 & Z = t2))).


all X all Y (-(REAL(X,Y))).
all X all Y (-(HIST(X,Y))).
all X all Y ((PREC(X,Y)) <-> (X = t1 & Y = t2) | (X = t1 & Y = st2) | (X = st1 & Y = st2) | (X = st1 & Y = t2) | (
      X = pb1 & Y = st2) | (X = pb1 & Y = t2)).
all X all Y (FINST(X,Y) <-> ((X = t1 & Y = t1) | (X = t2 & Y = t2) | (X = t1 & Y = t12))).
all X all Y (LINST(X,Y) <-> ((X = t1 & Y = t1) | (X = t2 & Y = t2) | (X = t2 & Y = t12))).
all X all Y all T (-(MBAS(X,Y,T))).
```

Counter model to (t_{bd}32), (t_{bd}45)

```
all X (PAR(X) <-> ((X = t1) | (X = t2) | (X = t12) | (X = s1) | (X = st1) | (X = st2) | (X = st12) | (X = c1) | (X
      = c2) | (X = c12))).

(t1 != t2) & (t1 != t12) & (t2 != t12) & (st1 != st2) & (st1 != st12) & (st2 != st12) & (c1 != c2) & (c1 != c12) &
      (c2 != c12).

(all X all Y all Z (::(X,Y,Z) <-> (X = t1 & Y = tinst & Z = t1 | X = t2 & Y = tinst & Z = t2 | X = t12 & Y = tint
      & Z = t1 | X = t12 & Y = tint & Z = t2 | X = t12 & Y = tint & Z = t12 | X = st1 & Y = streg & Z = t1 | X =
       st2 & Y = streg & Z = t2 | X = st12 & Y = streg & Z = t1 | X = st12 & Y = streg & Z = t2 | X = st12 & Y =
       streg & Z = t12 | X = c1 & Y = site & Z = t1 | X = c1 & Y = site & Z = t2 | X = c1 & Y = site & Z = t12 | X
      = c2 & Y = site & Z = t1 | X = c2 & Y = site & Z = t2 | X = c2 & Y = site & Z = t12 | X = c12 & Y = site & Z
       = t1 | X = c12 & Y = site & Z = t2 | X = c12 & Y = site & Z = t12 | X = s1 & Y = sreg3 & Z = t1 | X = s1 &
      Y = sreg3 & Z = t2 | X = s1 & Y = sreg3 & Z = t12 | X = t1 & Y = treg0 & Z = t1 | X = t2 & Y = treg0 & Z =
      t2 | X = t12 & Y = treg1 & Z = t1 | X = t12 & Y = treg1 & Z = t2 | X = t12 & Y = treg1 & Z = t12 | X = t1 &
      Y = treg & Z = t1 | X = t2 & Y = treg & Z = t2 | X = t12 & Y = treg & Z = t1 | X = t12 & Y = treg & Z = t2 |
       X = t12 & Y = treg & Z = t12 | X = t1 & Y = occ & Z = t1 | X = t2 & Y = occ & Z = t2 | X = t12 & Y = occ &
      Z = t1 | X = t12 & Y = occ & Z = t2 | X = t12 & Y = occ & Z = t12 | X = st1 & Y = occ & Z = t1 | X = st2 & Y
       = occ & Z = t2 | X = st12 & Y = occ & Z = t1 | X = st12 & Y = occ & Z = t2 | X = st12 & Y = occ & Z = t12 |
       X = s1 & Y = sreg & Z = t1 | X = s1 & Y = sreg & Z = t2 | X = s1 & Y = sreg & Z = t12 | X = c1 & Y = imen &
       Z = t1 | X = c1 & Y = imen & Z = t2 | X = c1 & Y = imen & Z = t12 | X = c2 & Y = imen & Z = t1 | X = c2 & Y
       = imen & Z = t2 | X = c2 & Y = imen & Z = t12 | X = c12 & Y = imen & Z = t1 | X = c12 & Y = imen & Z = t2 |
       X = c12 & Y = imen & Z = t12 | X = s1 & Y = imen & Z = t1 | X = s1 & Y = imen & Z = t2 | X = s1 & Y = imen
      & Z = t12 | X = c1 & Y = idcnt & Z = t1 | X = c1 & Y = idcnt & Z = t2 | X = c1 & Y = idcnt & Z = t12 | X =
      c2 & Y = idcnt & Z = t1 | X = c2 & Y = idcnt & Z = t2 | X = c2 & Y = idcnt & Z = t12 | X = c12 & Y = idcnt &
      Z = t1 | X = c12 & Y = idcnt & Z = t2 | X = c12 & Y = idcnt & Z = t12 | X = s1 & Y = idcnt & Z = t1 | X =
      s1 & Y = idcnt & Z = t2 | X = s1 & Y = idcnt & Z = t12 | X = c1 & Y = cnt & Z = t1 | X = c1 & Y = cnt & Z =
      t2 | X = c1 & Y = cnt & Z = t12 | X = c2 & Y = cnt & Z = t1 | X = c2 & Y = cnt & Z = t2 | X = c2 & Y = cnt &
       Z = t12 | X = c12 & Y = cnt & Z = t1 | X = c12 & Y = cnt & Z = t2 | X = c12 & Y = cnt & Z = t12 | X = s1 &
      Y = cnt & Z = t1 | X = s1 & Y = cnt & Z = t2 | X = s1 & Y = cnt & Z = t12))).

all X all T(-(::(X,hist,T))).

all X all Y (EX(X,Y) <-> ( (X = t1 & Y = t1) | (X = t2 & Y = t2) | (X = t12 & Y = t1) | (X = t12 & Y = t2) | (X =
      t12 & Y = t12) | (X = s1 & Y = t1) | (X = s1 & Y = t2) | (X = s1 & Y = t12) | (X = st1 & Y = t1) | (X = st2
      & Y = t2) | (X = st12 & Y = t1) | (X = st12 & Y = t2) | (X = st12 & Y = t12) | (X = c12 & Y = t1) | (X = c12
       & Y = t2) | (X = c12 & Y = t12) | (X = c1 & Y = t1) | (X = c1 & Y = t2) | (X = c1 & Y = t12) | (X = c2 & Y
      = t1) | (X = c2 & Y = t2) | (X = c2 & Y = t12))).

all X all Y all Z (cP(X,Y,Z) <-> ( (X = s1 & Y = s1 & Z = t1) | (X = s1 & Y = s1 & Z = t2) | (X = s1 & Y = s1 & Z
      = t12) | (X = c1 & Y = c1 & Z = t1) | (X = c1 & Y = c1 & Z = t2) | (X = c1 & Y = c1 & Z = t12) | (X = c2 & Y
      = c2 & Z = t1) | (X = c2 & Y = c2 & Z = t2) | (X = c2 & Y = c2 & Z = t12) | (X = c12 & Y = c12 & Z = t1) |
      (X = c12 & Y = c12 & Z = t2) | (X = c12 & Y = c12 & Z = t12) | (X = c1 & Y = c12 & Z = t1) | (X = c1 & Y =
      c12 & Z = t2) | (X = c1 & Y = c12 & Z = t12) | (X = c2 & Y = c12 & Z = t1) | (X = c2 & Y = c12 & Z = t2) | (
      X = c2 & Y = c12 & Z = t12))).

all X all Y (oP(X,Y) <-> ( (X = t1 & Y = t1) | (X = t2 & Y = t2) | (X = t12 & Y = t12) | (X = t1 & Y = t12) | (X =
       t2 & Y = t12) | (X = st1 & Y = st1) | (X = st2 & Y = st2) | (X = st12 & Y = st12) | (X = st1 & Y = st12) |
      (X = st2 & Y = st12))).

all X all Y all Z (-(mP(X,Y,Z))).

all X all Y (tmP(X,Y) <-> ( (X = t1 & Y = t1) | (X = t2 & Y = t2) | (X = t12 & Y = t12) | (X = t1 & Y = t12) | (X
      = t2 & Y = t12) | (X = st1 & Y = st1) | (X = st2 & Y = st2) | (X = st12 & Y = st12) | (X = st1 & Y = st12) |
       (X = st2 & Y = st12))).

all X all Y (-(STREG(X,Y))).

all X all Y all Z (SREG(X,Y,Z) <-> ( (X = c12 & Y = s1 & Z = t1) | (X = c12 & Y = s1 & Z = t2) | (X = c1 & Y = s1
      & Z = t1) | (X = c1 & Y = s1 & Z = t2) | (X = c2 & Y = s1 & Z = t1) | (X = c2 & Y = s1 & Z = t2))).

all X all Y (-(TREG(X,Y))).

all X all Y (TPROJ(X,Y) <-> ((X = st1 & Y = t1) | (X = st2 & Y = t2) | (X = st12 & Y = t12))).

all X all Y all Z (SPROJ(X,Y,Z) <-> ( (X = st1 & Y = s1 & Z = t1) | (X = st2 & Y = s1 & Z = t2) | (X = st12 & Y =
      s1 & Z = t1) | (X = st12 & Y = s1 & Z = t2) | (X = st12 & Y = s1 & Z = t12))).

all X all Y (-(OCCIN(X,Y))).

all X all Y all Z (LOC(X,Y,Z) <-> ( (X = c1 & Y = c1 & Z = t1) | (X = c1 & Y = c1 & Z = t2) | (X = c1 & Y = c1 & Z
       = t12) | (X = c2 & Y = c2 & Z = t1) | (X = c2 & Y = c2 & Z = t2) | (X = c2 & Y = c2 & Z = t12) | (X = c12 &
      Y = c12 & Z = t1) | (X = c12 & Y = c12 & Z = t2) | (X = c12 & Y = c12 & Z = t12) | (X = c1 & Y = c12 & Z =
      t1) | (X = c1 & Y = c12 & Z = t2) | (X = c1 & Y = c12 & Z = t12) | (X = c2 & Y = c12 & Z = t1) | (X = c2 & Y
      = c12 & Z = t2) | (X = c2 & Y = c12 & Z = t12))).

all X all Y (-(SDEP(X,Y))).
all X all Y all T (-(CONCR(X,Y,T))).
all X all Y all Z (-(GDEP(X,Y,Z))).

all X all Y all Z (-(PTC(X,Y,Z))).

all X all Y (-(REAL(X,Y))).
all X all Y (-(HIST(X,Y))).
all X all Y (PREC(X,Y) <-> ((X = t1 & Y = t2) | (X = t1 & Y = st2) | (X = st1 & Y = t2) | (X = st1 & Y = st2))).
all X all Y (FINST(X,Y) <-> ((X = t1 & Y = t1) | (X = t2 & Y = t2) | (X = t1 & Y = t12))).
all X all Y (LINST(X,Y) <-> ((X = t1 & Y = t1) | (X = t2 & Y = t2) | (X = t2 & Y = t12))).
all X all Y all Z (-(MBAS(X,Y,Z))).
```

Counter model to (t_{bd}22), (t_{bd}24), (t_{bd}26), (t_{bd}35), (t_{bd}39)

```
all X (PAR(X) <-> ((X = t1) | (X = t2) | (X = t12) | (X = s1) | (X = st1) | (X = st2) | (X = st12) | (X = c1) | (
      X = c2) | (X = sd1) |  (X = sd2) | (X = gd1))).

(t1 != t2) & (t1 != t12) & (t2 != t12) & (st1 != st2) & (st1 != st12) & (st2 != st12) & (c1 != c2) & (sd1 != sd2).

(all X all Y all Z (::(X,Y,Z) <-> (X = t1 & Y = tinst & Z = t1 | X = t1 & Y = treg0 & Z = t1 | X = t1 & Y = treg &
       Z = t1 | X = t1 & Y = occ & Z = t1 | X = t2 & Y = tinst & Z = t2 | X = t2 & Y = treg0 & Z = t2 | X = t2 & Y
       = treg & Z = t2 | X = t2 & Y = occ & Z = t2 | X = t12 & Y = tint & Z = t1 | X = t12 & Y = treg1 & Z = t1 |
      X = t12 & Y = treg & Z = t1 | X = t12 & Y = occ & Z = t1 | X = t12 & Y = tint & Z = t2 | X = t12 & Y = treg1
       & Z = t2 | X = t12 & Y = treg & Z = t2 | X = t12 & Y = occ & Z = t2 | X = t12 & Y = tint & Z = t12 | X =
       t12 & Y = treg1 & Z = t12 | X = t12 & Y = treg & Z = t12 | X = t12 & Y = occ & Z = t12 | X = s1 & Y = sreg3
      & Z = t1 | X = s1 & Y = sreg & Z = t1 |  X = s1 & Y = imen & Z = t1 | X = s1 & Y = idcnt & Z = t1 | X = s1 &
       Y = cnt & Z = t1 | X = s1 & Y = sreg3 & Z = t2 | X = s1 & Y = sreg & Z = t2 |  X = s1 & Y = imen & Z = t2 |
       X = s1 & Y = idcnt & Z = t2 | X = s1 & Y = cnt & Z = t2 | X = s1 & Y = sreg3 & Z = t12 | X = s1 & Y = sreg
      & Z = t12 |  X = s1 & Y = imen & Z = t12 | X = s1 & Y = idcnt & Z = t12 | X = s1 & Y = cnt & Z = t12 | X =
       st1 & Y = streg & Z = t1 | X = st1 & Y = occ & Z = t1 | X = st2 & Y = streg & Z = t2 | X = st2 & Y = occ & Z
       = t2 | X = st12 & Y = streg & Z = t1 | X = st12 & Y = occ & Z = t1 | X = st12 & Y = streg & Z = t2 | X =
       st12 & Y = occ & Z = t2 | X = st12 & Y = streg & Z = t12 | X = st12 & Y = occ & Z = t12 | X = c1 & Y = site
      & Z = t1 | X = c1 & Y = imen & Z = t1 | X = c1 & Y = idcnt & Z = t1 | X = c1 & Y = cnt & Z = t1 | X = c2 & Y
       = site & Z = t1 | X = c2 & Y = imen & Z = t1 | X = c2 & Y = idcnt & Z = t1 | X = c2 & Y = cnt & Z = t1 | X
       = c2 & Y = site & Z = t2 | X = c2 & Y = imen & Z = t2 | X = c2 & Y = idcnt & Z = t2 | X = c2 & Y = cnt & Z =
       t2 | X = c2 & Y = site & Z = t12 | X = c2 & Y = imen & Z = t12 | X = c2 & Y = idcnt & Z = t12 | X = c2 & Y
      = cnt & Z = t12 | X = sd1 & Y = qlt & Z = t1 | X = sd1 & Y = sdcnt & Z = t1 | X = sd1 & Y = cnt & Z = t1 | X
       = sd2 & Y = qlt & Z = t1 | X = sd2 & Y = sdcnt & Z = t1 | X = sd2 & Y = cnt & Z = t1 | X = sd2 & Y = qlt &
      Z = t2 | X = sd2 & Y = sdcnt & Z = t2 | X = sd2 & Y = cnt & Z = t2 | X = gd1 & Y = gdcnt & Z = t1 | X = gd1
      & Y = cnt & Z = t1 | X = gd1 & Y = gdcnt & Z = t2 | X = gd1 & Y = cnt & Z = t2 | X = gd1 & Y = gdcnt & Z =
       t12 | X = gd1 & Y = cnt & Z = t12))).

all X all T(- (::(X,rqlt,T))).

all X all Y (EX(X,Y) <-> ((X = t1 & Y = t1) | (X = t2 & Y = t2) | (X = t12 & Y = t1) | (X = t12 & Y = t2) | (X =
       t12 & Y = t12) | (X = s1 & Y = t1) | (X = s1 & Y = t2) | (X = s1 & Y = t12) | (X = st1 & Y = t1) | (X = st2
      & Y = t2) | (X = st12 & Y = t1) | (X = st12 & Y = t2) | (X = st12 & Y = t12) | (X = c1 & Y = t1) | (X = c2 &
       Y = t1) | (X = c2 & Y = t2) | (X = c2 & Y = t12) | (X = sd1 & Y = t1) | (X = sd2 & Y = t1) | (X = sd2 & Y =
       t2) | (X = gd1 & Y = t1) | (X = gd1 & Y = t2) | (X = gd1 & Y = t12))).

all X all Y all Z (cP(X,Y,Z) <-> ((X = s1 & Y = s1 & Z = t1) | (X = s1 & Y = s1 & Z = t2) | (X = s1 & Y = s1 & Z =
       t12) | (X = c1 & Y = c1 & Z = t1) | (X = c2 & Y = c2 & Z = t1) | (X = c2 & Y = c2 & Z = t2) | (X = c2 & Y =
       c2 & Z = t12))).

all X all Y (oP(X,Y) <-> ((X = t1 & Y = t1) | (X = t2 & Y = t2) | (X = t12 & Y = t12) | (X = t1 & Y = t12) | (X =
       t2 & Y = t12) | (X = st1 & Y = st1) | (X = st2 & Y = st2) | (X = st12 & Y = st12) | (X = st1 & Y = st12) | (
      X = st2 & Y = st12))).

all X all Y all Z (-(mP(X,Y,Z))).

all X all Y (tmP(X,Y) <-> ((X = t1 & Y = t1) | (X = t2 & Y = t2) | (X = t12 & Y = t12) | (X = t1 & Y = t12) | (X =
       t2 & Y = t12) | (X = st1 & Y = st1) | (X = st2 & Y = st2) | (X = st12 & Y = st12) | (X = st1 & Y = st12) |
      (X = st2 & Y = st12))).

all X all Y (-(STREG(X,Y))).

all X all Y all Z (SREG(X,Y,Z) <-> ((X = c1 & Y = s1 & Z = t1) | (X = c2 & Y = s1 & Z = t1) | (X = c2 & Y = s1 & Z
       = t2) | (X = c2 & Y = s1 & Z = t12))).

all X all Y (-(TREG(X,Y))).

all X all Y (TPROJ(X,Y) <-> ((X = st1 & Y = t1) | (X = st2 & Y = t2) | (X = st12 & Y = t12))).

all X all Y all Z (SPROJ(X,Y,Z) <-> ((X = st1 & Y = s1 & Z = t1) | (X = st2 & Y = s1 & Z = t2) | (X = st12 & Y =
       s1 & Z = t1) | (X = st12 & Y = s1 & Z = t2) | (X = st12 & Y = s1 & Z = t12))).

all X all Y (-(OCCIN(X,Y))).

all X all Y all Z (LOC(X,Y,Z) <-> ((X = c1 & Y = c1 & Z = t1) | (X = c2 & Y = c2 & Z = t1) | (X = c2 & Y = c2 & Z
       = t2) | (X = c2 & Y = c2 & Z = t12))).

all X all Y (SDEP(X,Y) <-> ((X = sd1 & Y = c1) | (X = sd2 & Y = c2))).
all X all Y all Z (CONCR(X,Y,Z) <-> ((X = sd1 & Y = gd1 & Z = t1) | (X = sd2 & Y = gd1 & Z = t2))).
all X all Y all Z (GDEP(X,Y,Z) <-> ((X = gd1 & Y = c1 & Z = t1) | (X = gd1 & Y = c2 & Z = t2) | (X = gd1 & Y = c2
       & Z = t12))).

all X all Y all Z (-(PTC(X,Y,Z))).
all X all Y (-(REAL(X,Y))).
all X all Y (-(HIST(X,Y))).
all X all Y (PREC(X,Y) <-> ((X = t1 & Y = t2) | (X = t1 & Y = st2) | (X = st1 & Y = t2) | (X = st1 & Y = st2))).
all X all Y (FINST(X,Y) <->  ((X = t1 & Y = t1) | (X = t2 & Y = t2) | (X = t1 & Y = t12))).
all X all Y (LINST(X,Y) <->  ((X = t1 & Y = t1) | (X = t2 & Y = t2) | (X = t2 & Y = t12))).
all X all Y all Z (-(MBAS(X,Y,Z))).
```

Counter model to (t_{bd}37)

```
all X (PAR(X) <-> ((X = i1) | (X = i2) | (X = t1) | (X = t2) | (X = t3) | (X = sti1) | (X = sti2) | (X = stt1) | (
      X = stt2) | (X = stt3) | (X = s1) | (X = c1) | (X= pb1) | (X= p1))).

(i1 != i2) & (t1 != t2) & (t1 != t3) & (t2 != t3) & (sti1 != sti2) & (sti1 != stt1) & (sti1 != stt2) & (sti1 !=
      stt3) & (sti2 != stt1) & (sti2 != stt2) & (sti2 != stt3) & (stt1 != stt2) & (stt1 != stt3) & (stt2 != stt3).

(all X all Y all Z (::(X,Y,Z) <-> (X = i1 & Y = tinst & Z = i1 | X = i1 & Y = treg0 & Z = i1 | X = i1 & Y = treg &
       Z = i1 | X = i1 & Y = occ & Z = i1 | X = i2 & Y = tinst & Z = i2 | X = i2 & Y = treg0 & Z = i2 | X = i2 & Y
       = treg & Z = i2 | X = i2 & Y = occ & Z = i2 | X = t1 & Y = tint & Z = t1 | X = t1 & Y = treg1 & Z = t1 | X
      = t1 & Y = treg & Z = t1 | X = t1 & Y = occ & Z = t1 | X = t2 & Y = tint & Z = i1 | X = t2 & Y = treg1 & Z =
       i1 | X = t2 & Y = treg & Z = i1 | X = t2 & Y = occ & Z = i1 | X = t2 & Y = tint & Z = i2 | X = t2 & Y =
      treg1 & Z = i2 | X = t2 & Y = treg & Z = i2 | X = t2 & Y = occ & Z = i2 | X = t2 & Y = tint & Z = t1 | X =
      t2 & Y = treg1 & Z = t1 | X = t2 & Y = treg & Z = t1 | X = t2 & Y = occ & Z = t1 | X = t2 & Y = tint & Z =
      t2 | X = t2 & Y = treg1 & Z = t2 | X = t2 & Y = treg & Z = t2 | X = t2 & Y = occ & Z = t2 | X = t2 & Y =
      tint & Z = t3 | X = t2 & Y = treg1 & Z = t3 | X = t2 & Y = treg & Z = t3 | X = t2 & Y = occ & Z = t3 | X =
      t3 & Y = tint & Z = i1 | X = t3 & Y = treg1 & Z = i1 | X = t3 & Y = treg & Z = i1 | X = t3 & Y = occ & Z =
      i1 | X = t3 & Y = tint & Z = t1 | X = t3 & Y = treg1 & Z = t1 | X = t3 & Y = treg & Z = t1 | X = t3 & Y =
      occ & Z = t1 | X = t3 & Y = tint & Z = t3 | X = t3 & Y = treg1 & Z = t3 | X = t2 & Y = treg & Z = t3 | X =
      t3 & Y = occ & Z = t3 | X = sti1 & Y = streg & Z = i1 | X = sti1 & Y = occ & Z = i1 | X = sti2 & Y = streg &
       Z = i2 | X = sti2 & Y = occ & Z = i2 | X = stt1 & Y = streg & Z = t1 | X = stt1 & Y = occ & Z = t1 | X =
      stt2 & Y = streg & Z = i1 | X = stt2 & Y = occ & Z = i1 | X = stt2 & Y = streg & Z = i2 | X = stt2 & Y = occ
       & Z = i2 | X = stt2 & Y = streg & Z = t1 | X = stt2 & Y = occ & Z = t1 | X = stt2 & Y = streg & Z = t2 | X
      = stt2 & Y = occ & Z = t2 | X = stt2 & Y = streg & Z = t3 | X = stt2 & Y = occ & Z = t3 | X = stt3 & Y =
      streg & Z = i1 | X = stt3 & Y = occ & Z = i1 | X = stt3 & Y = streg & Z = t1 | X = stt3 & Y = occ & Z = t1 |
       X = stt3 & Y = streg & Z = t3 | X = stt3 & Y = occ & Z = t3 | X = s1 & Y = sreg3 & Z = t1 | X = s1 & Y =
      sreg & Z = t1 |  X = s1 & Y = imen & Z = t1 | X = s1 & Y = idcnt & Z = t1 | X = s1 & Y = cnt & Z = t1 | X =
      s1 & Y = sreg3 & Z = t2 | X = s1 & Y = sreg & Z = t2 |  X = s1 & Y = imen & Z = t2 | X = s1 & Y = idcnt & Z
      = t2 | X = s1 & Y = cnt & Z = t2 | X = s1 & Y = sreg3 & Z = t3 | X = s1 & Y = sreg & Z = t3 |  X = s1 & Y =
      imen & Z = t3 | X = s1 & Y = idcnt & Z = t3 | X = s1 & Y = cnt & Z = t3 | X = s1 & Y = sreg3 & Z = i1 | X =
      s1 & Y = sreg & Z = i1 |  X = s1 & Y = imen & Z = i1 | X = s1 & Y = idcnt & Z = i1 | X = s1 & Y = cnt & Z =
      i1 | X = s1 & Y = sreg3 & Z = i2 | X = s1 & Y = sreg & Z = i2 |  X = s1 & Y = imen & Z = i2 | X = s1 & Y =
      idcnt & Z = i2 | X = s1 & Y = cnt & Z = i2 | X = c1 & Y = site & Z = t1 | X = c1 & Y = imen & Z = t1 | X =
      c1 & Y = idcnt & Z = t1 | X = c1 & Y = cnt & Z = t1 | X = c1 & Y = site & Z = t2 | X = c1 & Y = imen & Z =
      t2 | X = c1 & Y = idcnt & Z = t2 | X = c1 & Y = cnt & Z = t2 | X = c1 & Y = site & Z = t3 | X = c1 & Y =
      imen & Z = t3 | X = c1 & Y = idcnt & Z = t3 | X = c1 & Y = cnt & Z = t3 | X = c1 & Y = site & Z = i1 | X =
      c1 & Y = imen & Z = i1 | X = c1 & Y = idcnt & Z = i1 | X = c1 & Y = cnt & Z = i1 | X = c1 & Y = site & Z =
      i2 | X = c1 & Y = imen & Z = i2 | X = c1 & Y = idcnt & Z = i2 | X = c1 & Y = cnt & Z = i2 | X = pb1 & Y =
      pbnd & Z = t1 | X = pb1 & Y = occ & Z = t1 | X = p1 & Y = proc & Z = t1 | X = p1 & Y = occ & Z = t1 | X = p1
      & Y = proc & Z = t3 | X = p1 & Y = occ & Z = t3 | X = p1 & Y = proc & Z = i1 | X = p1 & Y = occ & Z = i1)))
      .

all X all T(- (::(X, hist ,T))).

all X all Y (EX(X,Y) <-> ((X = i1 & Y = i1) | (X = i2 & Y = i2) | (X = t1 & Y = t1) | (X = t2 & Y = i1) | (X = t2
      & Y = i2) | (X = t2 & Y = t1) | (X = t2 & Y = t2) | (X = t2 & Y = t3) | (X = t3 & Y = i1) | (X = t3 & Y = t1
      ) | (X = t3 & Y = t3) | (X = sti1 & Y = i1) | (X = sti2 & Y = i2) | (X = stt1 & Y = t1) | (X = stt2 & Y = i1
      ) | (X = stt2 & Y = i2) | (X = stt2 & Y = t1) | (X = stt2 & Y = t2) | (X = stt2 & Y = t3) | (X = stt3 & Y =
      i1) | (X = stt3 & Y = t1) | (X = stt3 & Y = t3) | (X = s1 & Y = i1) | (X = s1 & Y = i2) | (X = s1 & Y = t1)
      | (X = s1 & Y = t2) | (X = s1 & Y = t3) | (X = c1 & Y = i1) | (X = c1 & Y = i2) | (X = c1 & Y = t1) | (X =
      c1 & Y = t2) | (X = c1 & Y = t3) | (X = pb1 & Y = i1) | (X = p1 & Y = i1) | (X = p1 & Y = t1) | (X = p1 & Y
      = t3))).

all X all Y all Z (cP(X,Y,Z) <-> ((X = s1 & Y = s1 & Z = i1) | (X = s1 & Y = s1 & Z = i2) | (X = s1 & Y = s1 & Z =
       t1) | (X = s1 & Y = s1 & Z = t2) | (X = s1 & Y = s1 & Z = t3)  | (X = c1 & Y = c1 & Z = i1) | (X = c1 & Y =
       c1 & Z = i2) | (X = c1 & Y = c1 & Z = t1) | (X = c1 & Y = c1 & Z = t2) | (X = c1 & Y = c1 & Z = t3))).

all X all Y (oP(X,Y) <-> ((X = t1 & Y = t1) | (X = t2 & Y = t2) | (X = t3 & Y = t3) | (X = i1 & Y = i1) | (X = i2
      & Y = i2) | (X = i1 & Y = t2) | (X = i2 & Y = t2) | (X = t1 & Y = t2) | (X = t3 & Y = t2) | (X = i1 & Y = t3
      ) | (X = t1 & Y = t3) | (X = sti1 & Y = sti1) | (X = sti2 & Y = sti2) | (X = stt1 & Y = stt1) | (X = stt2 &
      Y = stt2) | (X = stt3 & Y = stt3) | (X = sti1 & Y = stt2) | (X = sti2 & Y = stt2) | (X = stt1 & Y = stt2) |
      (X = stt3 & Y = stt2) | (X = sti1 & Y = stt3) | (X = stt1 & Y = stt3) | (X = pb1 & Y = pb1) | (X = p1 & Y =
      p1) | (X = pb1 & Y = p1))).

all X all Y all Z (-(mP(X,Y,Z))).

all X all Y (tmP(X,Y) <-> ((X = t1 & Y = t1) | (X = t2 & Y = t2) | (X = t3 & Y = t3) | (X = i1 & Y = i1) | (X = i2
      & Y = i2) | (X = i1 & Y = t2) | (X = i2 & Y = t2) | (X = t1 & Y = t2) | (X = t3 & Y = t2) | (X = i1 & Y =
      t3) | (X = t1 & Y = t3) | (X = sti1 & Y = sti1) | (X = sti2 & Y = sti2) | (X = stt1 & Y = stt1) | (X = stt2
      & Y = stt2) | (X = stt3 & Y = stt3) | (X = sti1 & Y = stt2) | (X = sti2 & Y = stt2) | (X = stt1 & Y = stt2)
      | (X = stt3 & Y = stt2) | (X = sti1 & Y = stt3) | (X = stt1 & Y = stt3) | (X = pb1 & Y = pb1) | (X = p1 & Y
      = p1) | (X = pb1 & Y = p1))).

all X all Y (STREG(X,Y) <-> ((X = pb1 & Y = sti1) | (X = p1 & Y = stt3))).

all X all Y all Z (SREG(X,Y,Z) <-> ((X = c1 & Y = s1 & Z = t1) | (X = c1 & Y = s1 & Z = t2) | (X = c1 & Y = s1 & Z
      = t3) | (X = c1 & Y = s1 & Z = i1) | (X = c1 & Y = s1 & Z = i2))).

all X all Y (TREG(X,Y) <-> ((X = pb1 & Y = i1) | (X = p1 & Y = t3))).

all X all Y (TPROJ(X,Y) <-> ((X = sti1 & Y = i1) | (X = sti2 & Y = i2) |  (X = stt1 & Y = t1) | (X = stt2 & Y = t2
      ) | (X = stt3 & Y = t3))).

all X all Y all Z (SPROJ(X,Y,Z) <-> ((X = sti1 & Y = s1 & Z = ti1) | (X = sti2 & Y = s1 & Z = ti2) | (X = stt1 & Y
      = s1 & Z = t1) | (X = stt2 & Y = s1 & Z = t1) | (X = stt2 & Y = s1 & Z = t2) | (X = stt2 & Y = s1 & Z = t3)
      | (X = stt2 & Y = s1 & Z = i1) | (X = stt2 & Y = s1 & Z = i2) | (X = stt3 & Y = s1 & Z = t1) | (X = stt3 &
      Y = s1 & Z = t3) | (X = stt3 & Y = s1 & Z = i1))).

all X all Y (-(OCCIN(X,Y))).
```

```
all X all Y all Z (LOC(X,Y,Z) <-> ((X = c1 & Y = c1 & Z = t1) | (X = c1 & Y = c1 & Z = t2) | (X = c1 & Y = c1 & Z
      = t3) | (X = c1 & Y = c1 & Z = i1) | (X = c1 & Y = c1 & Z = i2))).

all X all Y (-(SDEP(X,Y))).
all X all Y all Z (-(CONCR(X,Y,Z))).
all X all Y all Z (-(GDEP(X,Y,Z))).

all X all Y all Z (PTC(X,Y,Z) <-> ((X = c1 & Y = p1 & Z = i1) | (X = c1 & Y = p1 & Z = t1) | (X = c1 & Y = p1 & Z
      = t3))).
all X all Y (-(REAL(X,Y))).
all X all Y (-(HIST(X,Y))).
all X all Y (PREC(X,Y) <-> ((X = i1 & Y = t1) | (X = i1 & Y = i2) | (X = t1 & Y = i2) | (X = t3 & Y = i2) | (X =
       sti1 & Y = t1) | (X = sti1 & Y = i2) | (X = sti1 & Y = stt1) | (X = sti1 & Y = sti2) | (X = stt1 & Y = i2) |
       (X = stt1 & Y = sti2) | (X = stt3 & Y = i2) | (X = stt3 & Y = sti2) | (X = pb1 & Y = t1) | (X = pb1 & Y =
       stt1) | (X = pb1 & Y = i2) | (X = pb1 & Y = sti2) | (X = p1 & Y = i2) | (X = p1 & Y = sti2))).
all X all Y (FINST(X,Y) <->  ((X = i1 & Y = i1) | (X = i2 & Y = i2) | (X = i1 & Y = t1) | (X = i1 & Y = t2) | (X =
       i1 & Y = t3))).
all X all Y (LINST(X,Y) <->  ((X = i1 & Y = i1) | (X = i2 & Y = i2) | (X = i2 & Y = t1) | (X = i2 & Y = t2) | (X =
       i2 & Y = t3))).
all X all Y all Z (-(MBAS(X,Y,Z))).
```

Counter model to (t_{bd}40)

```
all X (PAR(X) <-> ((X = t1) | (X = t2) | (X = t12) | (X = st1) | (X = st2) | (X = st12) | (X = pb1) | (X = pb2) |
      (X = p1) | (X = s1) | (X = c1) | (X = sd1) | (X = gd1))).

(t1 != t2) & (t1 != t12) & (t2 != t12) & (st1 != st2) & (st1 != st12) & (st2 != st12) & (pb1 != pb2).

(all X all Y all Z (::(X,Y,Z) <-> (X = t1 & Y = tinst & Z = t1 | X = t1 & Y = treg0 & Z = t1 | X = t1 & Y = treg &
       Z = t1 | X = t1 & Y = occ & Z = t1 | X = t2 & Y = tinst & Z = t2 | X = t2 & Y = treg0 & Z = t2 | X = t2 & Y
       = treg & Z = t2 | X = t2 & Y = occ & Z = t2 | X = t12 & Y = tint & Z = t1 | X = t12 & Y = treg1 & Z = t1 |
      X = t12 & Y = treg & Z = t1 | X = t12 & Y = occ & Z = t1 | X = t12 & Y = tint & Z = t2 | X = t12 & Y = treg1
       & Z = t2 | X = t12 & Y = treg & Z = t2 | X = t12 & Y = occ & Z = t2 | X = t12 & Y = tint & Z = t12 | X =
      t12 & Y = treg1 & Z = t12 | X = t12 & Y = treg & Z = t12 | X = t12 & Y = occ & Z = t12 | X = st1 & Y = streg
       & Z = t1 | X = st1 & Y = occ & Z = t1 | X = st2 & Y = streg & Z = t2 | X = st2 & Y = occ & Z = t2 | X =
      st12 & Y = streg & Z = t1 | X = st12 & Y = occ & Z = t1 | X = st12 & Y = streg & Z = t2 | X = st12 & Y = occ
       & Z = t2 | X = st12 & Y = streg & Z = t12 | X = st12 & Y = occ & Z = t12 | X = pb1 & Y = pbnd & Z = t1 | X
      = pb1 & Y = occ & Z = t1 | X = pb2 & Y = pbnd & Z = t2 | X = pb2 & Y = occ & Z = t2 | X = p1 & Y = proc & Z
      = t1 | X = p1 & Y = occ & Z = t1 | X = p1 & Y = proc & Z = t2 | X = p1 & Y = occ & Z = t2 | X = p1 & Y =
      proc & Z = t12 | X = p1 & Y = occ & Z = t12 | X = s1 & Y = sreg3 & Z = t1 | X = s1 & Y = sreg & Z = t1 |  X
      = s1 & Y = imen & Z = t1 | X = s1 & Y = idcnt & Z = t1 | X = s1 & Y = cnt & Z = t1 | X = s1 & Y = sreg3 & Z
      = t2 | X = s1 & Y = sreg & Z = t2 |  X = s1 & Y = imen & Z = t2 | X = s1 & Y = idcnt & Z = t2 | X = s1 & Y =
       cnt & Z = t2 | X = s1 & Y = sreg3 & Z = t12 | X = s1 & Y = sreg & Z = t12 |  X = s1 & Y = imen & Z = t12 |
      X = s1 & Y = idcnt & Z = t12 | X = s1 & Y = cnt & Z = t12 | X = c1 & Y = site & Z = t1 | X = c1 & Y = imen &
       Z = t1 | X = c1 & Y = idcnt & Z = t1 | X = c1 & Y = cnt & Z = t1 | X = c1 & Y = site & Z = t2 | X = c1 & Y
      = imen & Z = t2 | X = c1 & Y = idcnt & Z = t2 | X = c1 & Y = cnt & Z = t2 | X = c1 & Y = site & Z = t12 | X
      = c1 & Y = imen & Z = t12 | X = c1 & Y = idcnt & Z = t12 | X = c1 & Y = cnt & Z = t12 | X = sd1 & Y = qlt &
      Z = t1 | X = sd1 & Y = sdcnt & Z = t1 | X = sd1 & Y = cnt & Z = t1 | X = gd1 & Y = gdcnt & Z = t1 | X = gd1
      & Y = cnt & Z = t1 | X = gd1 & Y = gdcnt & Z = t2 | X = gd1 & Y = cnt & Z = t2 | X = gd1 & Y = gdcnt & Z =
      t12 | X = gd1 & Y = cnt & Z = t12))).

all X all T(-(::(X,rqlt,T))).

all X all Y (EX(X,Y) <-> ((X = t1 & Y = t1) | (X = t2 & Y = t2) | (X = t12 & Y = t1) | (X = t12 & Y = t2) | (X =
      t12 & Y = t12) | (X = st1 & Y = t1) | (X = st2 & Y = t2) | (X = st12 & Y = t1) | (X = st12 & Y = t2) | (X =
      st12 & Y = t12) | (X = pb1 & Y = t1) | (X = pb2 & Y = t2) | (X = p1 & Y = t1) | (X = p1 & Y = t2) | (X = p1
      & Y = t12) | (X = s1 & Y = t1) | (X = s1 & Y = t2) | (X = s1 & Y = t12) | (X = c1 & Y = t1) | (X = c1 & Y =
      t2) | (X = c1 & Y = t12) | (X = sd1 & Y = t1) | (X = gd1 & Y = t1) | (X = gd1 & Y = t2) | (X = gd1 & Y = t12
      ))).

all X all Y all Z (cP(X,Y,Z) <-> ((X = s1 & Y = s1 & Z = t1) | (X = s1 & Y = s1 & Z = t2) | (X = s1 & Y = s1 & Z =
       t12) | (X = c1 & Y = c1 & Z = t1) | (X = c1 & Y = c1 & Z = t2) | (X = c1 & Y = c1 & Z = t12))).

all X all Y (oP(X,Y) <-> ((X = t1 & Y = t1) | (X = t2 & Y = t2) | (X = t12 & Y = t12) | (X = t1 & Y = t12) | (X =
      t2 & Y = t12) | (X = st1 & Y = st1) | (X = st2 & Y = st2) | (X = st12 & Y = st12) | (X = st1 & Y = st12) | (
      X = st2 & Y = st12) | (X = pb1 & Y = pb1) | (X = pb2 & Y = pb2) | (X = p1 & Y = p1) | (X = pb1 & Y = p1) | (
      X = pb2 & Y = p1))).

all X all Y all Z (-(mP(X,Y,Z))).

all X all Y (tmP(X,Y) <-> ((X = t1 & Y = t1) | (X = t2 & Y = t2) | (X = t12 & Y = t12) | (X = t1 & Y = t12) | (X =
       t2 & Y = t12) | (X = st1 & Y = st1) | (X = st2 & Y = st2) | (X = st12 & Y = st12) | (X = st1 & Y = st12) |
      (X = st2 & Y = st12) | (X = pb1 & Y = pb1) | (X = pb2 & Y = pb2) | (X = p1 & Y = p1) | (X = pb1 & Y = p1) |
      (X = pb2 & Y = p1))).

all X all Y (STREG(X,Y) <-> ((X = pb1 & Y = st1) | (X = pb2 & Y = st2) | (X = p1 & Y = st12))).

all X all Y all Z (SREG(X,Y,Z) <-> ((X = c1 & Y = s1 & Z = t1) | (X = c1 & Y = s1 & Z = t2) | (X = c1 & Y = s1 & Z
       = t12))).

all X all Y (TREG(X,Y) <-> ((X = pb1 & Y = t1) | (X = pb2 & Y = t2) | (X = p1 & Y = t12))).

all X all Y (TPROJ(X,Y) <-> ((X = st1 & Y = t1) | (X = st2 & Y = t2) | (X = st12 & Y = t12))).

all X all Y all Z (SPROJ(X,Y,Z) <-> ((X = st1 & Y = s1 & Z = t1) | (X = st2 & Y = s1 & Z = t2) | (X = st12 & Y =
      s1 & Z = t1) | (X = st12 & Y = s1 & Z = t2) | (X = st12 & Y = s1 & Z = t12))).

all X all Y (-(OCCIN(X,Y))).
```

```
all X all Y all Z (LOC(X,Y,Z) <-> ((X = c1 & Y = c1 & Z = t1) | (X = c1 & Y = c1 & Z = t2) | (X = c1 & Y = c1 & Z
      = t12))).

all X all Y (SDEP(X,Y) <-> ((X = sd1 & Y = c1))).
all X all Y all Z (CONCR(X,Y,Z) <-> ((X = sd1 & Y = gd1 & Z = t1) | (X = p1 & Y = gd1 & Z = t2))).
all X all Y all Z (GDEP(X,Y,Z) <-> (X = gd1 & Y = c1 & Z = t1)).

all X all Y all Z (PTC(X,Y,Z) <-> ((X = c1 & Y = p1 & Z = t1) | (X = c1 & Y = p1 & Z = t2) | (X = c1 & Y = p1 & Z
      = t12))).
all X all Y (-(REAL(X,Y))).
all X all Y (-(HIST(X,Y))).
all X all Y (PREC(X,Y) <-> ((X = t1 & Y = t2) | (X = t1 & Y = st2) | (X = t1 & Y = pb2) | (X = st1 & Y = t2) | (X
      = st1 & Y = st2) | (X = st1 & Y = pb2) | (X = pb1 & Y = pb2) | (X = pb1 & Y = t2) |  (X = pb1 & Y = st2))).
all X all Y (FINST(X,Y) <->  ((X = t1 & Y = t1) | (X = t2 & Y = t2) | (X = t1 & Y = t12))).
all X all Y (LINST(X,Y) <->  ((X = t1 & Y = t1) | (X = t2 & Y = t2) | (X = t2 & Y = t12))).
all X all Y all Z (-(MBAS(X,Y,Z))).
```

Counter model to (t_{bd}47), (t_{bd}51)

```
all X (PAR(X) <-> ((X = t1) | (X = t2) | (X = t12) | (X = st1) | (X = st2) | (X = st3) | (X = st12) | (X = st123)
      | (X = pb1) | (X = p1) | (X = p2) | (X = s1) | (X = s2) | (X = s12) | (X = c1) | (X = c2))).

(t1 != t2) & (t1 != t12) & (t2 != t12) & (st1 != st2) & (st1 != st3) & (st1 != st12) & (st1 != st123) & (st2 !=
      st3) & (st2 != st12) & (st2 != st123) & (st3 != st12) & (st3 != st123) & (st12 != st123) & (p1 != p2) & (s1
      != s2) & (s1 != s12) & (s2 != s12) & (c1 != c2).

(all X all Y all Z (::(X,Y,Z) <-> (X = t1 & Y = tinst & Z = t1 | X = t1 & Y = treg0 & Z = t1 | X = t1 & Y = treg &
      Z = t1 | X = t1 & Y = occ & Z = t1 | X = t2 & Y = tinst & Z = t2 | X = t2 & Y = treg0 & Z = t2 | X = t2 & Y
      = treg & Z = t2 | X = t2 & Y = occ & Z = t2 | X = t12 & Y = tint & Z = t1 | X = t12 & Y = treg1 & Z = t1 |
     X = t12 & Y = treg & Z = t1 | X = t12 & Y = occ & Z = t1 | X = t12 & Y = tint & Z = t2 | X = t12 & Y = treg1
      & Z = t2 | X = t12 & Y = treg & Z = t2 | X = t12 & Y = occ & Z = t2 | X = t12 & Y = tint & Z = t12 | X =
      t12 & Y = treg1 & Z = t12 | X = t12 & Y = treg & Z = t12 | X = t12 & Y = occ & Z = t12 | X = st1 & Y = streg
      & Z = t1 | X = st1 & Y = occ & Z = t1 | X = st2 & Y = streg & Z = t2 | X = st2 & Y = occ & Z = t2 | X = st3
      & Y = streg & Z = t1 | X = st3 & Y = occ & Z = t1 | X = st3 & Y = streg & Z = t2 | X = st3 & Y = occ & Z =
      t2 | X = st3 & Y = streg & Z = t12 | X = st3 & Y = occ & Z = t12 | X = st12 & Y = streg & Z = t1 | X = st12
      & Y = occ & Z = t1 | X = st12 & Y = streg & Z = t2 | X = st12 & Y = occ & Z = t2 | X = st12 & Y = streg & Z
      = t12 | X = st12 & Y = occ & Z = t12 | X = st123 & Y = streg & Z = t1 | X = st123 & Y = occ & Z = t1 | X =
      st123 & Y = streg & Z = t2 | X = st123 & Y = occ & Z = t2 | X = st123 & Y = streg & Z = t12 | X = st123 & Y
      = occ & Z = t12 | X = pb1 & Y = pbnd & Z = t1 | X = pb1 & Y = occ & Z = t1 | X = p1 & Y = proc & Z = t1 | X
      = p1 & Y = occ & Z = t1 | X = p1 & Y = proc & Z = t2 | X = p1 & Y = occ & Z = t2 | X = p1 & Y = proc & Z =
      t12 | X = p1 & Y = occ & Z = t12 | X = p2 & Y = proc & Z = t1 | X = p2 & Y = occ & Z = t1 | X = p2 & Y =
      proc & Z = t2 | X = p2 & Y = occ & Z = t2 | X = p2 & Y = proc & Z = t12 | X = p2 & Y = occ & Z = t12 | X =
      s1 & Y = sreg3 & Z = t1 | X = s1 & Y = sreg & Z = t1 | X = s1 & Y = imen & Z = t1 | X = s1 & Y = idcnt & Z
      = t1 | X = s1 & Y = cnt & Z = t1 | X = s1 & Y = sreg3 & Z = t2 | X = s1 & Y = sreg & Z = t2 | X = s1 & Y =
      imen & Z = t2 | X = s1 & Y = idcnt & Z = t2 | X = s1 & Y = cnt & Z = t2 | X = s1 & Y = sreg3 & Z = t12 | X =
       s1 & Y = sreg & Z = t12 | X = s1 & Y = imen & Z = t12 | X = s1 & Y = idcnt & Z = t12 | X = s1 & Y = cnt &
      Z = t12 | X = s2 & Y = sreg3 & Z = t1 | X = s2 & Y = sreg & Z = t1 | X = s2 & Y = imen & Z = t1 | X = s2 &
      Y = idcnt & Z = t1 | X = s2 & Y = cnt & Z = t1 | X = s2 & Y = sreg3 & Z = t2 | X = s2 & Y = sreg & Z = t2 |
       X = s2 & Y = imen & Z = t2 | X = s2 & Y = idcnt & Z = t2 | X = s2 & Y = cnt & Z = t2 | X = s2 & Y = sreg3 &
      Z = t12 | X = s2 & Y = sreg & Z = t12 | X = s2 & Y = imen & Z = t12 | X = s2 & Y = idcnt & Z = t12 | X =
      s2 & Y = cnt & Z = t12 | X = s12 & Y = sreg3 & Z = t1 | X = s12 & Y = sreg & Z = t1 | X = s12 & Y = imen &
      Z = t1 | X = s12 & Y = idcnt & Z = t1 | X = s12 & Y = cnt & Z = t1 | X = s12 & Y = sreg3 & Z = t2 | X = s12
      & Y = sreg & Z = t2 | X = s12 & Y = imen & Z = t2 | X = s12 & Y = idcnt & Z = t2 | X = s12 & Y = cnt & Z =
      t2 | X = s12 & Y = sreg3 & Z = t12 | X = s12 & Y = sreg & Z = t12 | X = s12 & Y = imen & Z = t12 | X = s12
      & Y = idcnt & Z = t12 | X = s12 & Y = cnt & Z = t12 | X = c1 & Y = site & Z = t1 | X = c1 & Y = imen & Z =
      t1 | X = c1 & Y = idcnt & Z = t1 | X = c1 & Y = cnt & Z = t1 | X = c1 & Y = site & Z = t2 | X = c1 & Y =
      imen & Z = t2 | X = c1 & Y = idcnt & Z = t2 | X = c1 & Y = cnt & Z = t2 | X = c1 & Y = site & Z = t12 | X =
      c1 & Y = imen & Z = t12 | X = c1 & Y = idcnt & Z = t12 | X = c1 & Y = cnt & Z = t12 | X = c2 & Y = site & Z
      = t1 | X = c2 & Y = imen & Z = t1 | X = c2 & Y = idcnt & Z = t1 | X = c2 & Y = cnt & Z = t1 | X = c2 & Y =
      site & Z = t2 | X = c2 & Y = imen & Z = t2 | X = c2 & Y = idcnt & Z = t2 | X = c2 & Y = cnt & Z = t2 | X =
      c2 & Y = site & Z = t12 | X = c2 & Y = imen & Z = t12 | X = c2 & Y = idcnt & Z = t12 | X = c2 & Y = cnt & Z
      = t12))).

all X all Y (EX(X,Y) <-> ((X = t1 & Y = t1) | (X = t2 & Y = t2) | (X = t12 & Y = t1) | (X = t12 & Y = t2) | (X =
      t12 & Y = t12) | (X = st1 & Y = t1) | (X = st2 & Y = t2) | (X = st3 & Y = t1) | (X = st3 & Y = t2) | (X =
      st3 & Y = t12) | (X = st12 & Y = t1) | (X = st12 & Y = t2) | (X = st12 & Y = t12) | (X = st123 & Y = t1) | (
      X = st123 & Y = t2) | (X = st123 & Y = t12) | (X = pb1 & Y = t1) | (X = p1 & Y = t1) | (X = p1 & Y = t2) | (
      X = p1 & Y = t12) | (X = p2 & Y = t1) | (X = p2 & Y = t2) | (X = p2 & Y = t12) | (X = s1 & Y = t1) | (X = s1
       & Y = t2) | (X = s1 & Y = t12) | (X = s2 & Y = t1) | (X = s2 & Y = t2) | (X = s2 & Y = t12) | (X = s12 & Y
      = t1) | (X = s12 & Y = t2) | (X = s12 & Y = t12) | (X = c1 & Y = t1) | (X = c1 & Y = t2) | (X = c1 & Y = t12
      ) | (X = c2 & Y = t1) | (X = c2 & Y = t2) | (X = c2 & Y = t12))).

all X all Y all Z (cP(X,Y,Z) <-> ((X = s1 & Y = s1 & Z = t1) | (X = s1 & Y = s1 & Z = t2) | (X = s1 & Y = s1 & Z =
      t12) | (X = s2 & Y = s2 & Z = t1) | (X = s2 & Y = s2 & Z = t2) | (X = s2 & Y = s2 & Z = t12) | (X = s12 & Y
      = s12 & Z = t1) | (X = s12 & Y = s12 & Z = t2) | (X = s12 & Y = s12 & Z = t12) | (X = s1 & Y = s12 & Z = t1
      ) | (X = s1 & Y = s12 & Z = t2) | (X = s1 & Y = s12 & Z = t12) | (X = s2 & Y = s12 & Z = t1) | (X = s2 & Y =
      s12 & Z = t2) | (X = s2 & Y = s12 & Z = t12) | (X = c1 & Y = c1 & Z = t1) | (X = c1 & Y = c1 & Z = t2) | (X
      = c1 & Y = c1 & Z = t12) | (X = c2 & Y = c2 & Z = t1) | (X = c2 & Y = c2 & Z = t2) | (X = c2 & Y = c2 & Z =
      t12))).

all X all Y (oP(X,Y) <-> ((X = t1 & Y = t1) | (X = t2 & Y = t2) | (X = t12 & Y = t12) | (X = t1 & Y = t12) | (X =
      t2 & Y = t12) | (X = st1 & Y = st1) | (X = st2 & Y = st2) | (X = st3 & Y = st3) | (X = st12 & Y = st12) | (X
      = st123 & Y = st123) | (X = st1 & Y = st12) | (X = st2 & Y = st12) | (X = st1 & Y = st123) | (X = st2 & Y =
      st123) | (X = st3 & Y = st123) | (X = st12 & Y = st123) | (X = pb1 & Y = pb1) | (X = p1 & Y = p1) | (X = p2
      & Y = p2) | (X = pb1 & Y = p1) | (X = pb1 & Y = p2) | (X = p1 & Y = p2))).

all X all Y all Z (-(mP(X,Y,Z))).

all X all Y (tmP(X,Y) <-> ((X = t1 & Y = t1) | (X = t2 & Y = t2) | (X = t12 & Y = t12) | (X = t1 & Y = t12) | (X =
      t2 & Y = t12) | (X = st1 & Y = st1) | (X = st2 & Y = st2) | (X = st3 & Y = st3) | (X = st12 & Y = st12) | (
      X = st123 & Y = st123) | (X = st1 & Y = st12) | (X = st2 & Y = st12) | (X = st1 & Y = st123) | (X = st2 & Y
      = st123) | (X = st3 & Y = st123) | (X = st12 & Y = st123) | (X = pb1 & Y = pb1) | (X = p1 & Y = p1) | (X =
      p2 & Y = p2) | (X = pb1 & Y = p1) | (X = pb1 & Y = p2) | (X = p1 & Y = p2))).

all X all Y (STREG(X,Y) <-> ((X = pb1 & Y = st1) | (X = p1 & Y = st12) | (X = p2 & Y = st123))).

all X all Y all Z (SREG(X,Y,Z) <-> ((X = c1 & Y = s1 & Z = t1) | (X = c1 & Y = s1 & Z = t2) | (X = c1 & Y = s1 & Z
      = t12) | (X = c2 & Y = s12 & Z = t1) | (X = c2 & Y = s12 & Z = t2) | (X = c2 & Y = s12 & Z = t12))).

all X all Y (TREG(X,Y) <-> ((X = pb1 & Y = t1) | (X = p1 & Y = t12) | (X = p2 & Y = t12))).

all X all Y (TPROJ(X,Y) <-> ((X = st1 & Y = t1) | (X = st2 & Y = t2) | (X = st3 & Y = t12) | (X = st12 & Y = t12)
      | (X = st123 & Y = t12))).

all X all Y all Z (SPROJ(X,Y,Z) <-> ((X = st1 & Y = s1 & Z = t1) | (X = st2 & Y = s1 & Z = t2) | (X = st12 & Y =
      s1 & Z = t1) | (X = st12 & Y = s1 & Z = t2) | (X = st12 & Y = s1 & Z = t12) | (X = st3 & Y = s2 & Z = t1) |
      (X = st3 & Y = s2 & Z = t2) | (X = st3 & Y = s2 & Z = t12) | (X = st123 & Y = s12 & Z = t1) | (X = st123 & Y
      = s12 & Z = t2))).
```

```
all X all Y (-(OCCIN(X,Y))).

all X all Y all Z (LOC(X,Y,Z) <-> ((X = c1 & Y = c1 & Z = t1) | (X = c1 & Y = c1 & Z = t2) | (X = c1 & Y = c1 & Z
      = t12) | (X = c2 & Y = c2 & Z = t1) | (X = c2 & Y = c2 & Z = t2) | (X = c2 & Y = c2 & Z = t12))).

all X all Y (-(SDEP(X,Y))).
all X all Y all Z (-(CONCR(X,Y,Z))).
all X all Y all Z (-(GDEP(X,Y,Z))).

all X all Y all Z (PTC(X,Y,Z) <-> ((X = c1 & Y = p1 & Z = t1) | (X = c1 & Y = p1 & Z = t2) | (X = c1 & Y = p1 & Z
      = t12) | (X = c2 & Y = p2 & Z = t1) | (X = c2 & Y = p2 & Z = t2) | (X = c2 & Y = p2 & Z = t12))).
all X all Y (-(REAL(X,Y))).
all X all Y (-(HIST(X,Y))).
all X all Y (PREC(X,Y) <-> ((X = t1 & Y = t2) | (X = t1 & Y = st2) | (X = st1 & Y = t2) | (X = st1 & Y = st2) | (X
      = pb1 & Y = t2) | (X = pb1 & Y = st2))).
all X all Y (FINST(X,Y) <-> ((X = t1 & Y = t1) | (X = t2 & Y = t2) | (X = t1 & Y = t12))).
all X all Y (LINST(X,Y) <-> ((X = t1 & Y = t1) | (X = t2 & Y = t2) | (X = t2 & Y = t12))).
all X all Y all Z (-(MBAS(X,Y,Z))).
```

11.5 Validation of BFO-DOLCE mapping (theorems)

Proof of Theorem (t_{bd}1)

```
270. ! [X6] : (TR(X6) <=> T(X6)) [input]
272. ! [X6] : (PR(X6) <=> (? [X7] : (::(X7,X6) & Q(X7)) | S(X6))) [input]
273. ! [X6] : (R(X6) <=> (PR(X6) | TR(X6))) [input]
274. ! [X6] : (AB(X6) <=> R(X6)) [input]
276. ! [X6,X7] : (P(X6,X7) <=> ((ISA(X6,X7) & ~S(X7) & ~S(X6) & PR(X7) & PR(X6)) | (? [X1] : cP(X6,X7,X1) & S(X7)
       & S(X6)) | (oP(X6,X7) & ((T(X7) & T(X6)) | (PD(X7) & PD(X6)))))) [input]
292. ! [X6,X7] : (P(X6,X7) => ((PD(X7) & PD(X6)) | (AB(X7) & AB(X6)))) [input]
293. ~! [X6,X7] : (P(X6,X7) => ((PD(X7) & PD(X6)) | (AB(X7) & AB(X6)))) [negated conjecture 292]
719. ! [X0] : (TR(X0) <=> T(X0)) [rectify 270]
721. ! [X0] : (PR(X0) <=> (? [X1] : (::(X1,X0) & Q(X1)) | S(X0))) [rectify 272]
722. ! [X0] : (R(X0) <=> (PR(X0) | TR(X0))) [rectify 273]
723. ! [X0] : (AB(X0) <=> R(X0)) [rectify 274]
725. ! [X0,X1] : (P(X0,X1) <=> ((ISA(X0,X1) & ~S(X1) & ~S(X0) & PR(X1) & PR(X0)) | (? [X2] : cP(X0,X1,X2) & S(X1)
       & S(X0)) | (oP(X0,X1) & ((T(X1) & T(X0)) | (PD(X1) & PD(X0)))))) [rectify 276]
741. ~! [X0,X1] : (P(X0,X1) => ((PD(X1) & PD(X0)) | (AB(X1) & AB(X0)))) [rectify 293]
742. ! [X0,X1] : (P(X0,X1) => ((ISA(X0,X1) & ~S(X1) & ~S(X0) & PR(X1) & PR(X0)) | (? [X2] : cP(X0,X1,X2) & S(X1) &
       S(X0)) | (oP(X0,X1) & ((T(X1) & T(X0)) | (PD(X1) & PD(X0)))))) [unused predicate definition removal 725]
743. ! [X0] : (R(X0) => AB(X0)) [unused predicate definition removal 723]
744. ! [X0] : ((PR(X0) | TR(X0)) => R(X0)) [unused predicate definition removal 722]
745. ! [X0] : (T(X0) => TR(X0)) [unused predicate definition removal 719]
1047. ! [X0] : (TR(X0) | ~T(X0)) [ennf transformation 745]
1048. ! [X0] : (R(X0) | (~PR(X0) & ~TR(X0))) [ennf transformation 744]
1049. ! [X0] : (AB(X0) | ~R(X0)) [ennf transformation 743]
1050. ! [X0,X1] : (((ISA(X0,X1) & ~S(X1) & ~S(X0) & PR(X1) & PR(X0)) | (? [X2] : cP(X0,X1,X2) & S(X1) & S(X0)) | (
       oP(X0,X1) & ((T(X1) & T(X0)) | (PD(X1) & PD(X0))))) | ~P(X0,X1)) [ennf transformation 742]
1051. ! [X0,X1] : ((ISA(X0,X1) & ~S(X1) & ~S(X0) & PR(X1) & PR(X0)) | (? [X2] : cP(X0,X1,X2) & S(X1) & S(X0)) | (
       oP(X0,X1) & ((T(X1) & T(X0)) | (PD(X1) & PD(X0)))) | ~P(X0,X1)) [flattening 1050]
1052. ? [X0,X1] : (((~PD(X1) | ~PD(X0)) & (~AB(X1) | ~AB(X0))) & P(X0,X1)) [ennf transformation 741]
1053. ? [X0,X1] : ((~PD(X1) | ~PD(X0)) & (~AB(X1) | ~AB(X0)) & P(X0,X1)) [flattening 1052]
1069. ! [X1,X0] : ((oP(X0,X1) & ((T(X1) & T(X0)) | (PD(X1) & PD(X0)))) | ~sP9(X1,X0)) [predicate definition
       introduction]
1070. ! [X1,X0] : ((ISA(X0,X1) & ~S(X1) & ~S(X0) & PR(X1) & PR(X0)) | ~sP10(X1,X0)) [predicate definition
       introduction]
1071. ! [X0,X1] : (sP10(X1,X0) | (? [X2] : cP(X0,X1,X2) & S(X1) & S(X0)) | sP9(X1,X0) | ~P(X0,X1)) [definition
       folding 1051,1070,1069]
1410. ! [X0] : ((PR(X0) | (! [X1] : (~::(X1,X0) | ~Q(X1)) & ~S(X0))) & ((? [X1] : (::(X1,X0) & Q(X1)) | S(X0)) | ~
       PR(X0))) [nnf transformation 721]
1411. ! [X0] : ((PR(X0) | (! [X1] : (~::(X1,X0) | ~Q(X1)) & ~S(X0))) & (? [X1] : (::(X1,X0) & Q(X1)) | S(X0) | ~PR
       (X0))) [flattening 1410]
1412. ! [X0] : ((PR(X0) | (! [X1] : (~::(X1,X0) | ~Q(X1)) & ~S(X0))) & (? [X2] : (::(X2,X0) & Q(X2)) | S(X0) | ~PR
       (X0))) [rectify 1411]
1413. ! [X0] : (? [X2] : (::(X2,X0) & Q(X2)) => (::(sK161(X0),X0) & Q(sK161(X0)))) [choice axiom]
1414. ! [X0] : ((PR(X0) | (! [X1] : (~::(X1,X0) | ~Q(X1)) & ~S(X0))) & ((::(sK161(X0),X0) & Q(sK161(X0))) | S(X0)
       | ~PR(X0))) [skolemisation 1412,1413]
1415. ! [X1,X0] : ((ISA(X0,X1) & ~S(X1) & ~S(X0) & PR(X1) & PR(X0)) | ~sP10(X1,X0)) [nnf transformation 1070]
1416. ! [X0,X1] : ((ISA(X1,X0) & ~S(X0) & ~S(X1) & PR(X0) & PR(X1)) | ~sP10(X0,X1)) [rectify 1415]
1417. ! [X1,X0] : ((oP(X0,X1) & ((T(X1) & T(X0)) | (PD(X1) & PD(X0)))) | ~sP9(X1,X0)) [nnf transformation 1069]
1418. ! [X0,X1] : ((oP(X1,X0) & ((T(X0) & T(X1)) | (PD(X0) & PD(X1)))) | ~sP9(X0,X1)) [rectify 1417]
1419. ! [X1,X0] : (? [X2] : cP(X0,X1,X2) => cP(X0,X1,sK162(X0,X1))) [choice axiom]
1420. ! [X0,X1] : (sP10(X1,X0) | (cP(X0,X1,sK162(X0,X1)) & S(X1) & S(X0)) | sP9(X1,X0) | ~P(X0,X1)) [skolemisation
       1071,1419]
1421. ? [X0,X1] : ((~PD(X1) | ~PD(X0)) & (~AB(X1) | ~AB(X0)) & P(X0,X1)) => ((~PD(sK164) | ~PD(sK163)) & (~AB(
       sK164) | ~AB(sK163)) & P(sK163,sK164)) [choice axiom]
1422. (~PD(sK164) | ~PD(sK163)) & (~AB(sK164) | ~AB(sK163)) & P(sK163,sK164) [skolemisation 1053,1421]
2329. ~T(X0) | TR(X0) [cnf transformation 1047]
2334. ~S(X0) | PR(X0) [cnf transformation 1414]
2336. ~TR(X0) | R(X0) [cnf transformation 1048]
2337. ~PR(X0) | R(X0) [cnf transformation 1048]
2338. ~R(X0) | AB(X0) [cnf transformation 1049]
2339. ~sP10(X0,X1) | PR(X1) [cnf transformation 1416]
2340. ~sP10(X0,X1) | PR(X0) [cnf transformation 1416]
2344. ~sP9(X0,X1) | PD(X1) | T(X1) [cnf transformation 1418]
2345. ~sP9(X0,X1) | PD(X0) | T(X1) [cnf transformation 1418]
2346. ~sP9(X0,X1) | PD(X1) | T(X0) [cnf transformation 1418]
2347. ~sP9(X0,X1) | PD(X0) | T(X0) [cnf transformation 1418]
2349. ~P(X0,X1) | S(X0) | sP9(X1,X0) | sP10(X1,X0) [cnf transformation 1420]
2350. ~P(X0,X1) | S(X1) | sP9(X1,X0) | sP10(X1,X0) [cnf transformation 1420]
2352. P(sK163,sK164) [cnf transformation 1422]
2353. ~AB(sK164) | ~AB(sK163) [cnf transformation 1422]
2354. ~PD(sK164) | ~PD(sK163) [cnf transformation 1422]
2403. 1 <=> PD(sK163) [avatar definition]
2405. ~PD(sK163) <- (~1) [avatar component clause 2403]
2407. 2 <=> PD(sK164) [avatar definition]
2410. ~1 | ~2 [avatar split clause 2354,2407,2403]
2412. 3 <=> AB(sK163) [avatar definition]
2414. ~AB(sK163) <- (~3) [avatar component clause 2412]
2416. 4 <=> AB(sK164) [avatar definition]
2418. ~AB(sK164) <- (~4) [avatar component clause 2416]
2419. ~3 | ~4 [avatar split clause 2353,2416,2412]
4062. S(sK163) | sP9(sK164,sK163) | sP10(sK164,sK163) [resolution 2349,2352]
4064. 58 <=> sP10(sK164,sK163) [avatar definition]
4066. sP10(sK164,sK163) <- (58) [avatar component clause 4064]
4068. 59 <=> sP9(sK164,sK163) [avatar definition]
4070. sP9(sK164,sK163) <- (59) [avatar component clause 4068]
4072. 60 <=> S(sK163) [avatar definition]
4074. S(sK163) <- (60) [avatar component clause 4072]
```

```
4075. 58 | 59 | 60 [avatar split clause 4062,4072,4068,4064]
4079. PR(sK164) <- (58) [resolution 4066,2340]
4080. PR(sK163) <- (58) [resolution 4066,2339]
4082. S(sK164) | sP9(sK164,sK163) | sP10(sK164,sK163) [resolution 2350,2352]
4088. R(sK164) <- (58) [resolution 4079,2337]
4169. R(sK163) <- (58) [resolution 4080,2337]
4175. AB(sK164) <- (58) [resolution 4088,2338]
4178. 4 | ~58 [avatar split clause 4175,4064,2416]
4181. AB(sK163) <- (58) [resolution 4169,2338]
4182. $false <- (~3, 58) [subsumption resolution 4181,2414]
4183. 3 | ~58 [avatar contradiction clause 4182]
4185. 75 <=> S(sK164) [avatar definition]
4187. S(sK164) <- (75) [avatar component clause 4185]
4188. 58 | 59 | 75 [avatar split clause 4082,4185,4068,4064]
4203. PD(sK164) | T(sK164) <- (59) [resolution 4070,2347]
4204. PD(sK163) | T(sK164) <- (59) [resolution 4070,2346]
4205. PD(sK164) | T(sK163) <- (59) [resolution 4070,2345]
4206. PD(sK163) | T(sK163) <- (59) [resolution 4070,2344]
4209. 76 <=> T(sK164) [avatar definition]
4211. T(sK164) <- (76) [avatar component clause 4209]
4212. 76 | 2 | ~59 [avatar split clause 4203,4068,2407,4209]
4213. T(sK164) <- (~1, 59) [subsumption resolution 4204,2405]
4214. 76 | 1 | ~59 [avatar split clause 4213,4068,2403,4209]
4216. 77 <=> T(sK163) [avatar definition]
4218. T(sK163) <- (77) [avatar component clause 4216]
4219. 77 | 2 | ~59 [avatar split clause 4205,4068,2407,4216]
4220. T(sK163) <- (~1, 59) [subsumption resolution 4206,2405]
4221. 77 | 1 | ~59 [avatar split clause 4220,4068,2403,4216]
4224. PR(sK163) <- (60) [resolution 4074,2334]
4231. PR(sK164) <- (75) [resolution 4187,2334]
4238. R(sK163) <- (60) [resolution 4224,2337]
4249. R(sK164) <- (75) [resolution 4231,2337]
4250. AB(sK163) <- (60) [resolution 4238,2338]
4253. 3 | ~60 [avatar split clause 4250,4072,2412]
4254. AB(sK164) <- (75) [resolution 4249,2338]
4257. 4 | ~75 [avatar split clause 4254,4185,2416]
4276. TR(sK164) <- (76) [resolution 4211,2329]
4279. TR(sK163) <- (77) [resolution 4218,2329]
4280. R(sK164) <- (76) [resolution 4276,2336]
4293. R(sK163) <- (77) [resolution 4279,2336]
4298. AB(sK163) <- (77) [resolution 4293,2338]
4301. 3 | ~77 [avatar split clause 4298,4216,2412]
4312. AB(sK164) <- (76) [resolution 4280,2338]
4313. $false <- (~4, 76) [subsumption resolution 4312,2418]
4314. 4 | ~76 [avatar contradiction clause 4313]
4315. $false [avatar sat refutation 2410,2419,4075,4178,4183,4188,4212,4214,4219,4221,4253,4257,4301,4314]
```

Proof of Theorem ($t_{bd}2$)

```
31. ! [X17] : ! [X16] : ! [X1] : (::(X17,X16,X1) => (::(X1,treg,X1) & UNI(X16) & PAR(X17))) [input]
34. ! [X0] : ! [X1] : (? [X16] : (::(X1,treg,X1) & ::(X0,X16,X1) & UNI(X16)) <=> (EX(X0,X1) & ::(X1,treg,X1) & PAR
    (X0))) [input]
53. ! [X17] : ! [X1] : (::(X17,imen,X1) <=> (~? [X11] : (cP(X11,X17,X1) & ::(X11,mten,X1)) & ::(X17,idcnt,X1))) [
    input]
54. ! [X28] : (? [X1] : ::(X28,idcnt,X1) <=> (~? [X29] : ? [X1] : (GDEP(X28,X29,X1) | SDEP(X28,X29)) & ? [X1] :
    ::(X28,cnt,X1))) [input]
69. ! [X2] : ! [X3] : (oP(X2,X3) => (? [X1] : ::(X2,treg,X1) <=> ? [X1] : ::(X3,treg,X1))) [input]
201. ~? [X6] : (PAR(X6) & UNI(X6)) [input]
207. ! [X1] : ! [X6] : (::(X6,treg,X1) => ::(X6,occ,X1)) [input]
210. ~? [X6] : ? [X1] : (::(X6,occ,X1) & ::(X6,cnt,X1)) [input]
214. ! [X1] : ! [X6] : (::(X6,sreg,X1) => ::(X6,imen,X1)) [input]
234. ! [X6] : (? [X1] : ::(X6,cnt,X1) => ! [X1] : (EX(X6,X1) => ::(X6,cnt,X1))) [input]
257. ! [X6,X16] : (::(X6,X16) <=> ? [X1] : ::(X6,X16,X1)) [input]
269. ! [X6] : (T(X6) <=> ::(X6,treg)) [input]
271. ! [X6] : (S(X6) <=> ::(X6,sreg)) [input]
272. ! [X6] : (PR(X6) <=> (? [X7] : (::(X7,X6) & Q(X7)) | S(X6))) [input]
276. ! [X6,X7] : (P(X6,X7) <=> ((ISA(X6,X7) & ~S(X7) & ~S(X6) & PR(X7) & PR(X6)) | (? [X1] : cP(X6,X7,X1) & S(X7)
    & S(X6)) | (oP(X6,X7) & ((T(X7) & T(X6)) | (PD(X7) & PD(X6)))))) [input]
292. ! [X6,X7] : (P(X6,X7) => (T(X6) <=> T(X7))) [input]
293. ~! [X6,X7] : (P(X6,X7) => (T(X6) <=> T(X7))) [negated conjecture 292]
352. ! [X0] : ! [X1] : ! [X2] : (::(X0,X1,X2) => (::(X2,treg,X2) & UNI(X1) & PAR(X0))) [rectify 31]
353. ! [X0,X1,X2] : (::(X0,X1,X2) => (::(X2,treg,X2) & UNI(X1) & PAR(X0))) [flattening 352]
357. ! [X0] : ! [X1] : (? [X2] : (::(X1,treg,X1) & ::(X0,X2,X1) & UNI(X2)) <=> (EX(X0,X1) & ::(X1,treg,X1) & PAR(
    X0))) [rectify 34]
358. ! [X0,X1] : (? [X2] : (::(X1,treg,X1) & ::(X0,X2,X1) & UNI(X2)) <=> (EX(X0,X1) & ::(X1,treg,X1) & PAR(X0))) [
    flattening 357]
393. ! [X0] : ! [X1] : (::(X0,imen,X1) <=> (~? [X2] : (cP(X2,X0,X1) & ::(X2,mten,X1)) & ::(X0,idcnt,X1))) [rectify
    53]
394. ! [X0,X1] : (::(X0,imen,X1) <=> (~? [X2] : (cP(X2,X0,X1) & ::(X2,mten,X1)) & ::(X0,idcnt,X1))) [flattening
    393]
395. ! [X0] : (? [X1] : ::(X0,idcnt,X1) <=> (~? [X2] : ? [X3] : (GDEP(X0,X2,X3) | SDEP(X0,X2)) & ? [X4] : ::(X0,
    cnt,X4))) [rectify 54]
396. ! [X0] : (? [X1] : ::(X0,idcnt,X1) <=> (~? [X2,X3] : (GDEP(X0,X2,X3) | SDEP(X0,X2)) & ? [X4] : ::(X0,cnt,X4))
    ) [flattening 395]
423. ! [X0] : ! [X1] : (oP(X0,X1) => (? [X2] : ::(X0,treg,X2) <=> ? [X3] : ::(X1,treg,X3))) [rectify 69]
424. ! [X0,X1] : (oP(X0,X1) => (? [X2] : ::(X0,treg,X2) <=> ? [X3] : ::(X1,treg,X3))) [flattening 423]
617. ~? [X0] : (PAR(X0) & UNI(X0)) [rectify 201]
628. ! [X0] : ! [X1] : (::(X1,treg,X0) => ::(X1,occ,X0)) [rectify 207]
629. ! [X0,X1] : (::(X1,treg,X0) => ::(X1,occ,X0)) [flattening 628]
634. ~? [X0] : ? [X1] : (::(X0,occ,X1) & ::(X0,cnt,X1)) [rectify 210]
```

635. ~? [X0,X1] : (::(X0,occ,X1) & ::(X0,cnt,X1)) [flattening 634]
642. ! [X0] : ! [X1] : (::(X1,sreg,X0) => ::(X1,imen,X0)) [rectify 214]
643. ! [X0,X1] : (::(X1,sreg,X0) => ::(X1,imen,X0)) [flattening 642]
679. ! [X0] : (? [X1] : ::(X0,cnt,X1) => ! [X2] : (EX(X0,X2) => ::(X0,cnt,X2))) [rectify 234]
705. ! [X0,X1] : (::(X0,X1) <=> ? [X2] : ::(X0,X1,X2)) [rectify 257]
718. ! [X0] : (T(X0) <=> ::(X0,treg)) [rectify 269]
720. ! [X0] : (S(X0) <=> ::(X0,sreg)) [rectify 271]
721. ! [X0] : (PR(X0) <=> (? [X1] : (::(X1,X0) & Q(X1)) | S(X0))) [rectify 272]
725. ! [X0,X1] : (P(X0,X1) <=> ((ISA(X0,X1) & ~S(X1) & ~S(X0) & PR(X1) & PR(X0)) | (? [X2] : cP(X0,X1,X2) & S(X1) & S(X0)) | (oP(X0,X1) & ((T(X1) & T(X0)) | (PD(X1) & PD(X0)))))) [rectify 276]
741. ~! [X0,X1] : (P(X0,X1) => (T(X0) <=> T(X1))) [rectify 293]
742. ! [X0,X1] : (P(X0,X1) => ((ISA(X0,X1) & ~S(X1) & ~S(X0) & PR(X1) & PR(X0)) | (? [X2] : cP(X0,X1,X2) & S(X1) & S(X0)) | (oP(X0,X1) & ((T(X1) & T(X0)) | (PD(X1) & PD(X0)))))) [unused predicate definition removal 725]
743. ! [X0] : (PR(X0) => (? [X1] : (::(X1,X0) & Q(X1)) | S(X0))) [unused predicate definition removal 721]
802. ! [X0,X1,X2] : (((::(X2,treg,X2) & UNI(X1) & PAR(X0)) | ~::(X0,X1,X2)) [ennf transformation 353]
828. ! [X0,X1] : (::(X0,imen,X1) <=> (! [X2] : (~cP(X2,X0,X1) | ~::(X2,mten,X1)) & ::(X0,idcnt,X1))) [ennf transformation 394]
829. ! [X0] : (? [X1] : ::(X0,idcnt,X1) <=> (! [X2,X3] : (~GDEP(X0,X2,X3) & ~SDEP(X0,X2)) & ? [X4] : ::(X0,cnt,X4))) [ennf transformation 396]
851. ! [X0,X1] : ((? [X2] : ::(X0,treg,X2) <=> ? [X3] : ::(X1,treg,X3)) | ~oP(X0,X1)) [ennf transformation 424]
996. ! [X0] : (~PAR(X0) | ~UNI(X0)) [ennf transformation 617]
1002. ! [X0,X1] : (::(X1,occ,X0) | ~::(X1,treg,X0)) [ennf transformation 629]
1005. ! [X0,X1] : (~::(X0,occ,X1) | ~::(X0,cnt,X1)) [ennf transformation 635]
1009. ! [X0,X1] : (::(X1,imen,X0) | ~::(X1,sreg,X0)) [ennf transformation 643]
1028. ! [X0] : (! [X2] : (::(X0,cnt,X2) | ~EX(X0,X2)) | ! [X1] : ~::(X0,cnt,X1)) [ennf transformation 679]
1052. ! [X0] : ((? [X1] : (::(X1,X0) & Q(X1)) | S(X0)) | ~PR(X0)) [ennf transformation 743]
1053. ! [X0] : (? [X1] : (::(X1,X0) & Q(X1)) | S(X0) | ~PR(X0)) [flattening 1052]
1054. ! [X0,X1] : (((ISA(X0,X1) & ~S(X1) & ~S(X0) & PR(X1) & PR(X0)) | (? [X2] : cP(X0,X1,X2) & S(X1) & S(X0)) | (oP(X0,X1) & ((T(X1) & T(X0)) | (PD(X1) & PD(X0))))) | ~P(X0,X1)) [ennf transformation 742]
1055. ! [X0,X1] : ((ISA(X0,X1) & ~S(X1) & ~S(X0) & PR(X1) & PR(X0)) | (? [X2] : cP(X0,X1,X2) & S(X1) & S(X0)) | (oP(X0,X1) & ((T(X1) & T(X0)) | (PD(X1) & PD(X0)))) | ~P(X0,X1)) [flattening 1054]
1056. ? [X0,X1] : ((T(X0) <~> T(X1)) & P(X0,X1)) [ennf transformation 741]
1072. ! [X1,X0] : ((oP(X0,X1) & ((T(X1) & T(X0)) | (PD(X1) & PD(X0)))) | ~sP9(X1,X0)) [predicate definition introduction]
1073. ! [X1,X0] : ((ISA(X0,X1) & ~S(X1) & ~S(X0) & PR(X1) & PR(X0)) | ~sP10(X1,X0)) [predicate definition introduction]
1074. ! [X0,X1] : (sP10(X1,X0) | (? [X2] : cP(X0,X1,X2) & S(X1) & S(X0)) | sP9(X1,X0) | ~P(X0,X1)) [definition folding 1055,1073,1072]
1101. ! [X0,X1] : ((? [X2] : (::(X1,treg,X1) & ::(X0,X2,X1) & UNI(X2)) | (~EX(X0,X1) | ~::(X1,treg,X1) | ~PAR(X0))) & ((EX(X0,X1) & ::(X1,treg,X1) & PAR(X0)) | ! [X2] : (~::(X1,treg,X1) | ~::(X0,X2,X1) | ~UNI(X2)))) [nnf transformation 358]
1102. ! [X0,X1] : ((? [X2] : (::(X1,treg,X1) & ::(X0,X2,X1) & UNI(X2)) | ~EX(X0,X1) | ~::(X1,treg,X1) | ~PAR(X0)) & ((EX(X0,X1) & ::(X1,treg,X1) & PAR(X0)) | ! [X2] : (~::(X1,treg,X1) | ~::(X0,X2,X1) | ~UNI(X2)))) [flattening 1101]
1103. ! [X0,X1] : ((? [X2] : (::(X1,treg,X1) & ::(X0,X2,X1) & UNI(X2)) | ~EX(X0,X1) | ~::(X1,treg,X1) | ~PAR(X0)) & ((EX(X0,X1) & ::(X1,treg,X1) & PAR(X0)) | ! [X3] : (~::(X1,treg,X1) | ~::(X0,X3,X1) | ~UNI(X3)))) [rectify 1102]
1104. ! [X1,X0] : (? [X2] : (::(X1,treg,X1) & ::(X0,X2,X1) & UNI(X2)) => (::(X1,treg,X1) & ::(X0,sK21(X0,X1),X1) & UNI(sK21(X0,X1)))) [choice axiom]
1105. ! [X0,X1] : (((::(X1,treg,X1) & ::(X0,sK21(X0,X1),X1) & UNI(sK21(X0,X1))) | ~EX(X0,X1) | ~::(X1,treg,X1) | ~PAR(X0)) & ((EX(X0,X1) & ::(X1,treg,X1) & PAR(X0)) | ! [X3] : (~::(X1,treg,X1) | ~::(X0,X3,X1) | ~UNI(X3)))) [skolemisation 1103,1104]
1130. ! [X0,X1] : (((::(X0,imen,X1) | (? [X2] : (cP(X2,X0,X1) & ::(X2,mten,X1)) | ~::(X0,idcnt,X1))) & ((! [X2] : (~cP(X2,X0,X1) | ~::(X2,mten,X1)) & ::(X0,idcnt,X1)) | ~::(X0,imen,X1))) [nnf transformation 828]
1131. ! [X0,X1] : (((::(X0,imen,X1) | ? [X2] : (cP(X2,X0,X1) & ::(X2,mten,X1)) | ~::(X0,idcnt,X1)) & ((! [X2] : (~cP(X2,X0,X1) | ~::(X2,mten,X1)) & ::(X0,idcnt,X1)) | ~::(X0,imen,X1))) [flattening 1130]
1132. ! [X0,X1] : (((::(X0,imen,X1) | ? [X2] : (cP(X2,X0,X1) & ::(X2,mten,X1)) | ~::(X0,idcnt,X1)) & ((! [X3] : (~cP(X3,X0,X1) | ~::(X3,mten,X1)) & ::(X0,idcnt,X1)) | ~::(X0,imen,X1))) [rectify 1131]
1133. ! [X1,X0] : (? [X2] : (cP(X2,X0,X1) & ::(X2,mten,X1)) => (cP(sK35(X0,X1),X0,X1) & ::(sK35(X0,X1),mten,X1))) [choice axiom]
1134. ! [X0,X1] : (((::(X0,imen,X1) | (cP(sK35(X0,X1),X0,X1) & ::(sK35(X0,X1),mten,X1)) | ~::(X0,idcnt,X1)) & ((! [X3] : (~cP(X3,X0,X1) | ~::(X3,mten,X1)) & ::(X0,idcnt,X1)) | ~::(X0,imen,X1))) [skolemisation 1132,1133]
1135. ! [X0] : ((? [X1] : ::(X0,idcnt,X1) | (? [X2,X3] : (GDEP(X0,X2,X3) | SDEP(X0,X2)) | ! [X4] : ~::(X0,cnt,X4))) & ((! [X2,X3] : (~GDEP(X0,X2,X3) & ~SDEP(X0,X2)) & ? [X4] : ::(X0,cnt,X4)) | ! [X1] : ~::(X0,idcnt,X1))) [nnf transformation 829]
1136. ! [X0] : ((? [X1] : ::(X0,idcnt,X1) | ? [X2,X3] : (GDEP(X0,X2,X3) | SDEP(X0,X2)) | ! [X4] : ~::(X0,cnt,X4)) & ((! [X2,X3] : (~GDEP(X0,X2,X3) & ~SDEP(X0,X2)) & ? [X4] : ::(X0,cnt,X4)) | ! [X1] : ~::(X0,idcnt,X1))) [flattening 1135]
1137. ! [X0] : ((? [X1] : ::(X0,idcnt,X1) | ? [X2,X3] : (GDEP(X0,X2,X3) | SDEP(X0,X2)) | ! [X4] : ~::(X0,cnt,X4)) & ((! [X5,X6] : (~GDEP(X0,X5,X6) & ~SDEP(X0,X5)) & ? [X7] : ::(X0,cnt,X7)) | ! [X8] : ~::(X0,idcnt,X8))) [rectify 1136]
1138. ! [X0] : (? [X1] : ::(X0,idcnt,X1) => ::(X0,idcnt,sK36(X0))) [choice axiom]
1139. ! [X0] : (? [X2,X3] : (GDEP(X0,X2,X3) | SDEP(X0,X2)) => (GDEP(X0,sK37(X0),sK38(X0)) | SDEP(X0,sK37(X0)))) [choice axiom]
1140. ! [X0] : (? [X7] : ::(X0,cnt,X7) => ::(X0,cnt,sK39(X0))) [choice axiom]
1141. ! [X0] : (((::(X0,idcnt,sK36(X0)) | (GDEP(X0,sK37(X0),sK38(X0)) | SDEP(X0,sK37(X0))) | ! [X4] : ~::(X0,cnt,X4)) & ((! [X5,X6] : (~GDEP(X0,X5,X6) & ~SDEP(X0,X5)) & ::(X0,cnt,sK39(X0))) | ! [X8] : ~::(X0,idcnt,X8))) [skolemisation 1137,1140,1139,1138]
1172. ! [X0,X1] : (((? [X2] : ::(X0,treg,X2) | ! [X3] : ~::(X1,treg,X3)) & (? [X3] : ::(X1,treg,X3) | ! [X2] : ~::(X0,treg,X2))) | ~oP(X0,X1)) [nnf transformation 851]
1173. ! [X0,X1] : (((? [X2] : ::(X0,treg,X2) | ! [X3] : ~::(X1,treg,X3)) & (? [X4] : ::(X1,treg,X4) | ! [X5] : ~::(X0,treg,X5))) | ~oP(X0,X1)) [rectify 1172]
1174. ! [X0] : (? [X2] : ::(X0,treg,X2) => ::(X0,treg,sK57(X0))) [choice axiom]
1175. ! [X1] : (? [X4] : ::(X1,treg,X4) => ::(X1,treg,sK58(X1))) [choice axiom]
1176. ! [X0,X1] : (((::(X0,treg,sK57(X0)) | ! [X3] : ~::(X1,treg,X3)) & (::(X1,treg,sK58(X1)) | ! [X5] : ~::(X0,treg,X5))) | ~oP(X0,X1)) [skolemisation 1173,1175,1174]
1382. ! [X0] : (! [X1] : (::(X0,cnt,X1) | ~EX(X0,X1)) | ! [X2] : ~::(X0,cnt,X2)) [rectify 1028]
1400. ! [X0,X1] : (((::(X0,X1) | ! [X2] : ~::(X0,X1,X2)) & (? [X2] : ::(X0,X1,X2) | ~::(X0,X1))) [nnf transformation 705]
1401. ! [X0,X1] : (((::(X0,X1) | ! [X2] : ~::(X0,X1,X2)) & (? [X3] : ::(X0,X1,X3) | ~::(X0,X1))) [rectify 1400]
1402. ! [X1,X0] : (? [X3] : ::(X0,X1,X3) => ::(X0,X1,sK160(X0,X1))) [choice axiom]

```
1403. ! [X0,X1] : ((::(X0,X1) | ! [X2] : ~::(X0,X1,X2)) & (::(X0,X1,sK160(X0,X1)) | ~::(X0,X1))) [skolemisation
      1401,1402]
1406. ! [X0] : ((T(X0) | ~::(X0,treg)) & (::(X0,treg) | ~T(X0))) [nnf transformation 718]
1407. ! [X0] : ((S(X0) | ~::(X0,sreg)) & (::(X0,sreg) | ~S(X0))) [nnf transformation 720]
1408. ! [X0] : (? [X1] : (::(X1,X0) & Q(X1)) => (::(sK161(X0),X0) & Q(sK161(X0)))) [choice axiom]
1409. ! [X0] : ((::(sK161(X0),X0) & Q(sK161(X0))) | S(X0) | ~PR(X0)) [skolemisation 1053,1408]
1410. ! [X1,X0] : ((ISA(X0,X1) & ~S(X1) & ~S(X0) & PR(X1) & PR(X0)) | ~sP10(X1,X0)) [nnf transformation 1073]
1411. ! [X0,X1] : ((ISA(X1,X0) & ~S(X0) & ~S(X1) & PR(X0) & PR(X1)) | ~sP10(X0,X1)) [rectify 1410]
1412. ! [X1,X0] : ((oP(X0,X1) & ((T(X1) & T(X0)) | (PD(X1) & PD(X0)))) | ~sP9(X1,X0)) [nnf transformation 1072]
1413. ! [X0,X1] : ((oP(X1,X0) & ((T(X0) & T(X1)) | (PD(X0) & PD(X1)))) | ~sP9(X0,X1)) [rectify 1412]
1414. ! [X1,X0] : (? [X2] : cP(X0,X1,X2) => cP(X0,X1,sK162(X0,X1))) [choice axiom]
1415. ! [X0,X1] : (sP10(X1,X0) | (cP(X0,X1,sK162(X0,X1)) & S(X1) & S(X0)) | sP9(X1,X0) | ~P(X0,X1)) [skolemisation
      1074,1414]
1416. ? [X0,X1] : (((~T(X1) | ~T(X0)) & (T(X1) | T(X0))) & P(X0,X1)) [nnf transformation 1056]
1417. ? [X0,X1] : (((~T(X1) | ~T(X0)) & (T(X1) | T(X0)) & P(X0,X1)) [flattening 1416]
1418. ? [X0,X1] : (((~T(X1) | ~T(X0)) & (T(X1) | T(X0)) & P(X0,X1)) => ((~T(sK164) | ~T(sK163)) & (T(sK164) | T(
      sK163)) & P(sK163,sK164)) [choice axiom]
1419. (~T(sK164) | ~T(sK163)) & (T(sK164) | T(sK163)) & P(sK163,sK164) [skolemisation 1417,1418]
1469. ~::(X0,X1,X2) | PAR(X0) [cnf transformation 802]
1470. ~::(X0,X1,X2) | UNI(X1) [cnf transformation 802]
1471. ~::(X0,X1,X2) | ::(X2,treg,X2) [cnf transformation 802]
1480. EX(X0,X1) | ~::(X1,treg,X1) | ~::(X0,X3,X1) | ~UNI(X3) [cnf transformation 1105]
1525. ~::(X0,imen,X1) | ::(X0,idcnt,X1) [cnf transformation 1134]
1529. ~::(X0,idcnt,X8) | ::(X0,cnt,sK39(X0)) [cnf transformation 1141]
1566. ~::(X0,treg,X5) | ::(X1,treg,sK58(X1)) | ~oP(X0,X1) [cnf transformation 1176]
1567. ~::(X1,treg,X3) | ::(X0,treg,sK57(X0)) | ~oP(X0,X1) [cnf transformation 1176]
1861. ~UNI(X0) | ~PAR(X0) [cnf transformation 996]
1867. ~::(X1,treg,X0) | ::(X1,occ,X0) [cnf transformation 1002]
1870. ~::(X0,occ,X1) | ~::(X0,cnt,X1) [cnf transformation 1005]
1874. ~::(X1,sreg,X0) | ::(X1,imen,X0) [cnf transformation 1009]
1893. ~::(X0,cnt,X2) | ~EX(X0,X1) | ::(X0,cnt,X1) [cnf transformation 1382]
2310. ::(X0,X1,sK160(X0,X1)) | ~::(X0,X1) [cnf transformation 1403]
2311. ~::(X0,X1,X2) | ::(X0,X1) [cnf transformation 1403]
2320. ::(X0,treg) | ~T(X0) [cnf transformation 1406]
2321. ~::(X0,treg) | T(X0) [cnf transformation 1406]
2322. ::(X0,sreg) | ~S(X0) [cnf transformation 1407]
2325. ::(sK161(X0),X0) | S(X0) | ~PR(X0) [cnf transformation 1409]
2326. ~sP10(X0,X1) | PR(X1) [cnf transformation 1411]
2327. ~sP10(X0,X1) | PR(X0) [cnf transformation 1411]
2328. ~sP10(X0,X1) | ~S(X1) [cnf transformation 1411]
2329. ~sP10(X0,X1) | ~S(X0) [cnf transformation 1411]
2335. ~sP9(X0,X1) | oP(X1,X0) [cnf transformation 1413]
2336. ~P(X0,X1) | S(X0) | sP9(X1,X0) | sP10(X1,X0) [cnf transformation 1415]
2337. ~P(X0,X1) | S(X1) | sP9(X1,X0) | sP10(X1,X0) [cnf transformation 1415]
2339. P(sK163,sK164) [cnf transformation 1419]
2340. T(sK164) | T(sK163) [cnf transformation 1419]
2341. ~T(sK164) | ~T(sK163) [cnf transformation 1419]
2380. EX(X0,X1) | ~::(X1,treg,X1) | ~::(X0,X3,X1) [subsumption resolution 1480,1470]
2381. ~::(X0,X3,X1) | EX(X0,X1) [subsumption resolution 2380,1471]
2390. 1 <=> T(sK163) [avatar definition]
2391. T(sK163) <- (1) [avatar component clause 2390]
2392. ~T(sK163) <- (~1) [avatar component clause 2390]
2394. 2 <=> T(sK164) [avatar definition]
2395. T(sK164) <- (2) [avatar component clause 2394]
2396. ~T(sK164) <- (~2) [avatar component clause 2394]
2397. ~1 | ~2 [avatar split clause 2341,2394,2390]
2398. 1 | 2 [avatar split clause 2340,2394,2390]
3198. EX(X2,sK160(X2,X3)) | ~::(X2,X3) [resolution 2310,2381]
3200. ~::(X6,X7) | UNI(X7) [resolution 2310,1470]
3201. ~::(X8,X9) | PAR(X8) [resolution 2310,1469]
3202. ::(X10,cnt,sK39(X10)) | ~::(X10,idcnt) [resolution 2310,1529]
3209. ::(X18,imen,sK160(X18,sreg)) | ~::(X18,sreg) [resolution 2310,1874]
3216. ::(X25,occ,sK160(X25,treg)) | ~::(X25,treg) [resolution 2310,1867]
3227. ::(X36,idcnt,sK160(X36,imen)) | ~::(X36,imen) [resolution 2310,1525]
3254. ~::(X63,cnt,sK160(X63,occ)) | ~::(X63,occ) [resolution 2310,1870]
3308. ~PR(X5) | S(X5) | UNI(X5) [resolution 3200,2325]
3315. ~T(X1) | PAR(X1) [resolution 3201,2320]
3356. PAR(sK164) <- (2) [resolution 3315,2395]
3796. ::(X4,cnt,X5) | ~EX(X4,X5) | ~::(X4,cnt) [resolution 1893,2310]
3911. 27 <=> sP10(sK164,sK163) [avatar definition]
3912. ~sP10(sK164,sK163) <- (~27) [avatar component clause 3911]
3913. sP10(sK164,sK163) <- (27) [avatar component clause 3911]
3915. 28 <=> sP9(sK164,sK163) [avatar definition]
3917. sP9(sK164,sK163) <- (28) [avatar component clause 3915]
3919. 29 <=> S(sK163) [avatar definition]
3920. ~S(sK163) <- (~29) [avatar component clause 3919]
3921. S(sK163) <- (29) [avatar component clause 3919]
3924. ~S(sK164) <- (27) [resolution 3913,2329]
3925. ~S(sK163) <- (27) [resolution 3913,2328]
3926. PR(sK164) <- (27) [resolution 3913,2327]
3928. ~29 | ~27 [avatar split clause 3925,3911,3919]
4028. ~::(X4,treg) | ~oP(X4,X3) | ::(X3,treg,sK58(X3)) [resolution 1566,2310]
4033. ~::(X4,treg) | ~oP(X3,X4) | ::(X3,treg,sK57(X3)) [resolution 1567,2310]
6940. ~::(X8,idcnt) | ::(X8,cnt) [resolution 3202,2311]
7663. ~::(X9,sreg) | ::(X9,imen) [resolution 3209,2311]
7724. ~::(X20,treg) | ::(X20,occ) [resolution 3216,2311]
7793. ~::(X16,imen) | ::(X16,idcnt) [resolution 3227,2311]
8139. S(sK164) | UNI(sK164) <- (27) [resolution 3308,3926]
8141. UNI(sK164) <- (27) [subsumption resolution 8139,3924]
8145. ~PAR(sK164) <- (27) [resolution 8141,1861]
8146. $false <- (2, 27) [subsumption resolution 8145,3356]
```

```
8147. ~2 | ~27 [avatar contradiction clause 8146]
8149. 176 <=> S(sK164) [avatar definition]
8151. S(sK164) <- (176) [avatar component clause 8149]
8203. PAR(sK163) <- (1) [resolution 2391,3315]
8208. PR(sK163) <- (27) [resolution 3913,2326]
8220. S(sK163) | UNI(sK163) <- (27) [resolution 8208,3308]
8221. UNI(sK163) <- (27, ~29) [subsumption resolution 8220,3920]
8235. ~PAR(sK163) <- (27, ~29) [resolution 8221,1861]
8236. $false <- (1, 27, ~29) [subsumption resolution 8235,8203]
8237. ~1 | ~27 | 29 [avatar contradiction clause 8236]
8245. S(sK164) | sP9(sK164,sK163) | sP10(sK164,sK163) [resolution 2339,2337]
8246. S(sK163) | sP9(sK164,sK163) | sP10(sK164,sK163) [resolution 2339,2336]
8251. oP(sK163,sK164) <- (28) [resolution 3917,2335]
9708. ~EX(X5,sK160(X5,occ)) | ~::(X5,cnt) | ~::(X5,occ) [resolution 3796,3254]
9721. ~::(X5,occ) | ~::(X5,cnt) [subsumption resolution 9708,3198]
10931. ~oP(X0,X1) | ::(X1,treg,sK58(X1)) | ~T(X0) [resolution 4028,2320]
10953. ~oP(X0,X1) | ::(X0,treg,sK57(X0)) | ~T(X1) [resolution 4033,2320]
34563. ::(X0,imen) | ~S(X0) [resolution 7663,2322]
34840. ::(X0,occ) | ~T(X0) [resolution 7724,2320]
36060. ::(X0,idcnt) | ~S(X0) [resolution 7793,34563]
36179. ::(X0,cnt) | ~S(X0) [resolution 36060,6940]
38438. ~::(X1,cnt) | ~T(X1) [resolution 9721,34840]
44727. ~S(X0) | ~T(X0) [resolution 38438,36179]
75743. ::(sK163,treg,sK57(sK163)) | ~T(sK164) <- (28) [resolution 10953,8251]
75784. ::(sK163,treg,sK57(sK163)) <- (2, 28) [subsumption resolution 75743,2395]
76120. ::(sK163,treg) <- (2, 28) [resolution 75784,2311]
76130. T(sK163) <- (2, 28) [resolution 76120,2321]
76141. $false <- (~1, 2, 28) [subsumption resolution 76130,2392]
76142. 1 | ~2 | ~28 [avatar contradiction clause 76141]
76153. S(sK163) | sP9(sK164,sK163) <- (~27) [subsumption resolution 8246,3912]
76154. 28 | 29 | 27 [avatar split clause 76153,3911,3919,3915]
76155. S(sK164) | sP9(sK164,sK163) <- (~27) [subsumption resolution 8245,3912]
76156. 28 | 176 | 27 [avatar split clause 76155,3911,8149,3915]
76157. ~T(sK163) <- (29) [resolution 3921,44727]
76161. ~T(sK164) <- (176) [resolution 8151,44727]
76165. $false <- (2, 176) [subsumption resolution 76161,2395]
76166. ~2 | ~176 [avatar contradiction clause 76165]
76167. ~1 | ~29 [avatar split clause 76157,3919,2390]
76174. oP(sK163,sK164) <- (28) [resolution 3917,2335]
76185. ::(sK164,treg,sK58(sK164)) | ~T(sK163) <- (28) [resolution 76174,10931]
76187. ::(sK164,treg,sK58(sK164)) <- (1, 28) [subsumption resolution 76185,2391]
76567. ::(sK164,treg) <- (1, 28) [resolution 76187,2311]
76571. T(sK164) <- (1, 28) [resolution 76567,2321]
76582. $false <- (1, ~2, 28) [subsumption resolution 76571,2396]
76583. ~1 | 2 | ~28 [avatar contradiction clause 76582]
76584. $false [avatar sat refutation 2397,2398,3928,8147,8237,76142,76154,76156,76166,76167,76583]
```

Proof of Theorem ($t_{bd}3$)

```
31. ! [X17] : ! [X16] : ! [X1] : (::(X17,X16,X1) => (::(X1,treg,X1) & UNI(X16) & PAR(X17))) [input]
34. ! [X0] : ! [X1] : (? [X16] : (::(X1,treg,X1) & ::(X0,X16,X1) & UNI(X16)) <=> (EX(X0,X1) & ::(X1,treg,X1) & PAR
    (X0))) [input]
53. ! [X17] : ! [X1] : (::(X17,imen,X1) <=> (~? [X11] : (cP(X11,X17,X1) & ::(X11,mten,X1)) & ::(X17,idcnt,X1))) [
    input]
54. ! [X28] : (? [X1] : ::(X28,idcnt,X1) <=> (~? [X29] : ? [X1] : (GDEP(X28,X29,X1) | SDEP(X28,X29)) & ? [X1] :
    ::(X28,cnt,X1))) [input]
59. ! [X0] : (? [X1] : ::(X0,occ,X1) => oP(X0,X0)) [input]
64. ! [X0] : ! [X8] : (oP(X0,X8) => (? [X1] : ::(X8,occ,X1) & ? [X1] : ::(X0,occ,X1))) [input]
210. ~? [X6] : ? [X1] : (::(X6,occ,X1) & ::(X6,cnt,X1)) [input]
214. ! [X1] : ! [X6] : (::(X6,sreg,X1) => ::(X6,imen,X1)) [input]
234. ! [X6] : (? [X1] : ::(X6,cnt,X1) => ! [X1] : (EX(X6,X1) => ::(X6,cnt,X1))) [input]
257. ! [X6,X16] : (::(X6,X16) <=> ? [X1] : ::(X6,X16,X1)) [input]
271. ! [X6] : (S(X6) <=> ::(X6,sreg)) [input]
276. ! [X6,X7] : (P(X6,X7) <=> ((ISA(X6,X7) & ~S(X7) & ~S(X6) & PR(X7) & PR(X6)) | (? [X1] : cP(X6,X7,X1) & S(X7)
    & S(X6)) | (oP(X6,X7) & ((T(X7) & T(X6)) | (PD(X7) & PD(X6)))))) [input]
292. ! [X6,X7] : (P(X6,X7) => (S(X6) <=> S(X7))) [input]
293. ~! [X6,X7] : (P(X6,X7) => (S(X6) <=> S(X7))) [negated conjecture 292]
352. ! [X0] : ! [X1] : ! [X2] : (::(X0,X1,X2) => (::(X2,treg,X2) & UNI(X1) & PAR(X0))) [rectify 31]
353. ! [X0,X1,X2] : (::(X0,X1,X2) => (::(X2,treg,X2) & UNI(X1) & PAR(X0))) [flattening 352]
357. ! [X0] : ! [X1] : (? [X2] : (::(X1,treg,X1) & ::(X0,X2,X1) & UNI(X2)) <=> (EX(X0,X1) & ::(X1,treg,X1) & PAR(
    X0))) [rectify 34]
358. ! [X0,X1] : (? [X2] : (::(X1,treg,X1) & ::(X0,X2,X1) & UNI(X2)) <=> (EX(X0,X1) & ::(X1,treg,X1) & PAR(X0))) [
    flattening 357]
393. ! [X0] : ! [X1] : (::(X0,imen,X1) <=> (~? [X2] : (cP(X2,X0,X1) & ::(X2,mten,X1)) & ::(X0,idcnt,X1))) [rectify
    53]
394. ! [X0,X1] : (::(X0,imen,X1) <=> (~? [X2] : (cP(X2,X0,X1) & ::(X2,mten,X1)) & ::(X0,idcnt,X1))) [flattening
    393]
395. ! [X0] : (? [X1] : ::(X0,idcnt,X1) <=> (~? [X2] : ? [X3] : (GDEP(X0,X2,X3) | SDEP(X0,X2)) & ? [X4] : ::(X0,
    cnt,X4))) [rectify 54]
396. ! [X0] : (? [X1] : ::(X0,idcnt,X1) <=> (~? [X2,X3] : (GDEP(X0,X2,X3) | SDEP(X0,X2)) & ? [X4] : ::(X0,cnt,X4))
    ) [flattening 395]
413. ! [X0] : ! [X1] : (oP(X0,X1) => (? [X2] : ::(X1,occ,X2) & ? [X3] : ::(X0,occ,X3))) [rectify 64]
414. ! [X0,X1] : (oP(X0,X1) => (? [X2] : ::(X1,occ,X2) & ? [X3] : ::(X0,occ,X3))) [flattening 413]
634. ~? [X0] : ? [X1] : (::(X0,occ,X1) & ::(X0,cnt,X1)) [rectify 210]
635. ~? [X0,X1] : (::(X0,occ,X1) & ::(X0,cnt,X1)) [flattening 634]
642. ! [X0] : ! [X1] : (::(X1,sreg,X0) => ::(X1,imen,X0)) [rectify 214]
643. ! [X0,X1] : (::(X1,sreg,X0) => ::(X1,imen,X0)) [flattening 642]
679. ! [X0] : (? [X1] : ::(X0,cnt,X1) => ! [X2] : (EX(X0,X2) => ::(X0,cnt,X2))) [rectify 234]
705. ! [X0,X1] : (::(X0,X1) <=> ? [X2] : ::(X0,X1,X2)) [rectify 257]
720. ! [X0] : (S(X0) <=> ::(X0,sreg)) [rectify 271]
```

```
725. ! [X0,X1] : (P(X0,X1) <=> ((ISA(X0,X1) & ~S(X1) & ~S(X0) & PR(X1) & PR(X0)) | (? [X2] : cP(X0,X1,X2) & S(X1)
       & S(X0)) | (oP(X0,X1) & ((T(X1) & T(X0)) | (PD(X1) & PD(X0)))))) [rectify 276]
741. ~! [X0,X1] : (P(X0,X1) => (S(X0) <=> S(X1))) [rectify 293]
742. ! [X0,X1] : (P(X0,X1) => ((ISA(X0,X1) & ~S(X1) & ~S(X0) & PR(X1) & PR(X0)) | (? [X2] : cP(X0,X1,X2) & S(X1) &
       S(X0)) | (oP(X0,X1) & ((T(X1) & T(X0)) | (PD(X1) & PD(X0)))))) [unused predicate definition removal 725]
803. ! [X0,X1,X2] : ((::(X2,treg,X2) & UNI(X1) & PAR(X0)) | ~::(X0,X1,X2)) [ennf transformation 353]
829. ! [X0,X1] : (::(X0,imen,X1) <=> (! [X2] : (~cP(X2,X0,X1) | ~::(X2,mten,X1)) & ::(X0,idcnt,X1))) [ennf
       transformation 394]
830. ! [X0] : (? [X1] : ::(X0,idcnt,X1) <=> (! [X2,X3] : (~GDEP(X0,X2,X3) & ~SDEP(X0,X2)) & ? [X4] : ::(X0,cnt,X4)
       )) [ennf transformation 396]
838. ! [X0] : (oP(X0,X0) | ! [X1] : ~::(X0,occ,X1)) [ennf transformation 59]
846. ! [X0,X1] : ((? [X2] : ::(X1,occ,X2) & ? [X3] : ::(X0,occ,X3)) | ~oP(X0,X1)) [ennf transformation 414]
1006. ! [X0,X1] : (~::(X0,occ,X1) | ~::(X0,cnt,X1)) [ennf transformation 635]
1010. ! [X0,X1] : (::(X1,imen,X0) | ~::(X1,sreg,X0)) [ennf transformation 643]
1029. ! [X0] : (! [X2] : (::(X0,cnt,X2) | ~EX(X0,X2)) | ! [X1] : ~::(X0,cnt,X1)) [ennf transformation 679]
1056. ! [X0,X1] : (((ISA(X0,X1) & ~S(X1) & ~S(X0) & PR(X1) & PR(X0)) | (? [X2] : cP(X0,X1,X2) & S(X1) & S(X0)) | (
       oP(X0,X1) & ((T(X1) & T(X0)) | (PD(X1) & PD(X0))))) | ~P(X0,X1)) [ennf transformation 742]
1057. ! [X0,X1] : ((ISA(X0,X1) & ~S(X1) & ~S(X0) & PR(X1) & PR(X0)) | (? [X2] : cP(X0,X1,X2) & S(X1) & S(X0)) | (
       oP(X0,X1) & ((T(X1) & T(X0)) | (PD(X1) & PD(X0)))) | ~P(X0,X1)) [flattening 1056]
1058. ? [X0,X1] : ((S(X0) <~> S(X1)) & P(X0,X1)) [ennf transformation 741]
1074. ! [X1,X0] : ((oP(X0,X1) & ((T(X1) & T(X0)) | (PD(X1) & PD(X0)))) | ~sP9(X1,X0)) [predicate definition
       introduction]
1075. ! [X1,X0] : ((ISA(X0,X1) & ~S(X1) & ~S(X0) & PR(X1) & PR(X0)) | ~sP10(X1,X0)) [predicate definition
       introduction]
1076. ! [X0,X1] : (sP10(X1,X0) | (? [X2] : cP(X0,X1,X2) & S(X1) & S(X0)) | sP9(X1,X0) | ~P(X0,X1)) [definition
       folding 1057,1075,1074]
1103. ! [X0,X1] : ((? [X2] : (::(X1,treg,X1) & ::(X0,X2,X1) & UNI(X2)) | (~EX(X0,X1) | ~::(X1,treg,X1) | ~PAR(X0))
       ) & ((EX(X0,X1) & ::(X1,treg,X1) & PAR(X0)) | ! [X2] : (~::(X1,treg,X1) | ~::(X0,X2,X1) | ~UNI(X2)))) [nnf
       transformation 358]
1104. ! [X0,X1] : ((? [X2] : (::(X1,treg,X1) & ::(X0,X2,X1) & UNI(X2)) | ~EX(X0,X1) | ~::(X1,treg,X1) | ~PAR(X0))
       & ((EX(X0,X1) & ::(X1,treg,X1) & PAR(X0)) | ! [X2] : (~::(X1,treg,X1) | ~::(X0,X2,X1) | ~UNI(X2)))) [
       flattening 1103]
1105. ! [X0,X1] : ((? [X2] : (::(X1,treg,X1) & ::(X0,X2,X1) & UNI(X2)) | ~EX(X0,X1) | ~::(X1,treg,X1) | ~PAR(X0))
       & ((EX(X0,X1) & ::(X1,treg,X1) & PAR(X0)) | ! [X3] : (~::(X1,treg,X1) | ~::(X0,X3,X1) | ~UNI(X3)))) [rectify
       1104]
1106. ! [X1,X0] : (? [X2] : (::(X1,treg,X1) & ::(X0,X2,X1) & UNI(X2)) => (::(X1,treg,X1) & ::(X0,sK21(X0,X1),X1) &
       UNI(sK21(X0,X1)))) [choice axiom]
1107. ! [X0,X1] : (((::(X1,treg,X1) & ::(X0,sK21(X0,X1),X1) & UNI(sK21(X0,X1))) | ~EX(X0,X1) | ~::(X1,treg,X1) | ~
       PAR(X0)) & ((EX(X0,X1) & ::(X1,treg,X1) & PAR(X0)) | ! [X3] : (~::(X1,treg,X1) | ~::(X0,X3,X1) | ~UNI(X3))))
       [skolemisation 1105,1106]
1132. ! [X0,X1] : ((::(X0,imen,X1) | (? [X2] : (cP(X2,X0,X1) & ::(X2,mten,X1)) | ~::(X0,idcnt,X1))) & ((! [X2] :
       (~cP(X2,X0,X1) | ~::(X2,mten,X1)) & ::(X0,idcnt,X1)) | ~::(X0,imen,X1))) [nnf transformation 829]
1133. ! [X0,X1] : (((::(X0,imen,X1) | ? [X2] : (cP(X2,X0,X1) & ::(X2,mten,X1)) | ~::(X0,idcnt,X1)) & ((! [X2] : (~
       cP(X2,X0,X1) | ~::(X2,mten,X1)) & ::(X0,idcnt,X1)) | ~::(X0,imen,X1))) [flattening 1132]
1134. ! [X0,X1] : (((::(X0,imen,X1) | ? [X2] : (cP(X2,X0,X1) & ::(X2,mten,X1)) | ~::(X0,idcnt,X1)) & ((! [X3] : (~
       cP(X3,X0,X1) | ~::(X3,mten,X1)) & ::(X0,idcnt,X1)) | ~::(X0,imen,X1))) [rectify 1133]
1135. ! [X1,X0] : (? [X2] : (cP(X2,X0,X1) & ::(X2,mten,X1)) => (cP(sK35(X0,X1),X0,X1) & ::(sK35(X0,X1),mten,X1)))
       [choice axiom]
1136. ! [X0,X1] : (((::(X0,imen,X1) | (cP(sK35(X0,X1),X0,X1) & ::(sK35(X0,X1),mten,X1)) | ~::(X0,idcnt,X1)) & ((! [
       X3] : (~cP(X3,X0,X1) | ~::(X3,mten,X1)) & ::(X0,idcnt,X1)) | ~::(X0,imen,X1))) [skolemisation 1134,1135]
1137. ! [X0] : ((? [X1] : ::(X0,idcnt,X1) | (? [X2,X3] : (GDEP(X0,X2,X3) | SDEP(X0,X2)) | ! [X4] : ~::(X0,cnt,X4))
       ) & ((! [X2,X3] : (~GDEP(X0,X2,X3) & ~SDEP(X0,X2)) & ? [X4] : ::(X0,cnt,X4)) | ! [X1] : ~::(X0,idcnt,X1))) [
       nnf transformation 830]
1138. ! [X0] : ((? [X1] : ::(X0,idcnt,X1) | ? [X2,X3] : (GDEP(X0,X2,X3) | SDEP(X0,X2)) | ! [X4] : ~::(X0,cnt,X4))
       & ((! [X2,X3] : (~GDEP(X0,X2,X3) & ~SDEP(X0,X2)) & ? [X4] : ::(X0,cnt,X4)) | ! [X1] : ~::(X0,idcnt,X1))) [
       flattening 1137]
1139. ! [X0] : ((? [X1] : ::(X0,idcnt,X1) | ? [X2,X3] : (GDEP(X0,X2,X3) | SDEP(X0,X2)) | ! [X4] : ~::(X0,cnt,X4))
       & ((! [X5,X6] : (~GDEP(X0,X5,X6) & ~SDEP(X0,X5)) & ? [X7] : ::(X0,cnt,X7)) | ! [X8] : ~::(X0,idcnt,X8))) [
       rectify 1138]
1140. ! [X0] : (? [X1] : ::(X0,idcnt,X1) => ::(X0,idcnt,sK36(X0))) [choice axiom]
1141. ! [X0] : (? [X2,X3] : (GDEP(X0,X2,X3) | SDEP(X0,X2)) => (GDEP(X0,sK37(X0),sK38(X0)) | SDEP(X0,sK37(X0)))) [
       choice axiom]
1142. ! [X0] : (? [X7] : ::(X0,cnt,X7) => ::(X0,cnt,sK39(X0))) [choice axiom]
1143. ! [X0] : ((::(X0,idcnt,sK36(X0)) | (GDEP(X0,sK37(X0),sK38(X0)) | SDEP(X0,sK37(X0))) | ! [X4] : ~::(X0,cnt,X4
       )) & ((! [X5,X6] : (~GDEP(X0,X5,X6) & ~SDEP(X0,X5)) & ::(X0,cnt,sK39(X0))) | ! [X8] : ~::(X0,idcnt,X8))) [
       skolemisation 1139,1142,1141,1140]
1159. ! [X1] : (? [X2] : ::(X1,occ,X2) => ::(X1,occ,sK48(X1))) [choice axiom]
1160. ! [X0] : (? [X3] : ::(X0,occ,X3) => ::(X0,occ,sK49(X0))) [choice axiom]
1161. ! [X0,X1] : ((::(X1,occ,sK48(X1)) & ::(X0,occ,sK49(X0))) | ~oP(X0,X1)) [skolemisation 846,1160,1159]
1384. ! [X0] : (! [X1] : (::(X0,cnt,X1) | ~EX(X0,X1)) | ! [X2] : ~::(X0,cnt,X2)) [rectify 1029]
1402. ! [X0,X1] : ((::(X0,X1) | ! [X2] : ~::(X0,X1,X2)) & (? [X2] : ::(X0,X1,X2) | ~::(X0,X1))) [nnf
       transformation 705]
1403. ! [X0,X1] : ((::(X0,X1) | ! [X2] : ~::(X0,X1,X2)) & (? [X3] : ::(X0,X1,X3) | ~::(X0,X1))) [rectify 1402]
1404. ! [X1,X0] : (? [X3] : ::(X0,X1,X3) => ::(X0,X1,sK160(X0,X1))) [choice axiom]
1405. ! [X0,X1] : ((::(X0,X1) | ! [X2] : ~::(X0,X1,X2)) & (::(X0,X1,sK160(X0,X1)) | ~::(X0,X1))) [skolemisation
       1403,1404]
1408. ! [X0] : ((S(X0) | ~::(X0,sreg)) & (::(X0,sreg) | ~S(X0))) [nnf transformation 720]
1411. ! [X1,X0] : ((ISA(X0,X1) & ~S(X1) & ~S(X0) & PR(X1) & PR(X0)) | ~sP10(X1,X0)) [nnf transformation 1075]
1412. ! [X0,X1] : ((ISA(X1,X0) & ~S(X0) & ~S(X1) & PR(X0) & PR(X1)) | ~sP10(X0,X1)) [rectify 1411]
1413. ! [X1,X0] : ((oP(X0,X1) & ((T(X1) & T(X0)) | (PD(X1) & PD(X0)))) | ~sP9(X1,X0)) [nnf transformation 1074]
1414. ! [X0,X1] : ((oP(X1,X0) & ((T(X0) & T(X1)) | (PD(X0) & PD(X1)))) | ~sP9(X0,X1)) [rectify 1413]
1415. ! [X1,X0] : (? [X2] : cP(X0,X1,X2) => cP(X0,X1,sK162(X0,X1))) [choice axiom]
1416. ! [X0,X1] : (sP10(X1,X0) | (cP(X0,X1,sK162(X0,X1)) & S(X1) & S(X0)) | sP9(X1,X0) | ~P(X0,X1)) [skolemisation
       1076,1415]
1417. ? [X0,X1] : (((~S(X1) | ~S(X0)) & (S(X1) | S(X0))) & P(X0,X1)) [nnf transformation 1058]
1418. ? [X0,X1] : ((~S(X1) | ~S(X0)) & (S(X1) | S(X0)) & P(X0,X1)) [flattening 1417]
1419. ? [X0,X1] : ((~S(X1) | ~S(X0)) & (S(X1) | S(X0)) & P(X0,X1)) => ((~S(sK164) | ~S(sK163)) & (S(sK164) | S(
       sK163)) & P(sK163,sK164)) [choice axiom]
1420. (~S(sK164) | ~S(sK163)) & (S(sK164) | S(sK163)) & P(sK163,sK164) [skolemisation 1418,1419]
1471. ~::(X0,X1,X2) | UNI(X1) [cnf transformation 803]
1472. ~::(X0,X1,X2) | ::(X2,treg,X2) [cnf transformation 803]
```

```
1481. EX(X0,X1) | ~::(X1,treg,X1) | ~::(X0,X3,X1) | ~UNI(X3) [cnf transformation 1107]
1526. ~::(X0,imen,X1) | ::(X0,idcnt,X1) [cnf transformation 1136]
1530. ~::(X0,idcnt,X8) | ::(X0,cnt,sK39(X0)) [cnf transformation 1143]
1553. ~::(X0,occ,X1) | oP(X0,X0) [cnf transformation 838]
1558. ~oP(X0,X1) | ::(X0,occ,sK49(X0)) [cnf transformation 1161]
1559. ~oP(X0,X1) | ::(X1,occ,sK48(X1)) [cnf transformation 1161]
1871. ~::(X0,occ,X1) | ~::(X0,cnt,X1) [cnf transformation 1006]
1875. ~::(X1,sreg,X0) | ::(X1,imen,X0) [cnf transformation 1010]
1894. ~::(X0,cnt,X2) | ~EX(X0,X1) | ::(X0,cnt,X1) [cnf transformation 1384]
2311. ::(X0,X1,sK160(X0,X1)) | ~::(X0,X1) [cnf transformation 1405]
2312. ~::(X0,X1,X2) | ::(X0,X1) [cnf transformation 1405]
2322. ::(X0,sreg) | ~S(X0) [cnf transformation 1408]
2328. ~sP10(X0,X1) | ~S(X1) [cnf transformation 1412]
2329. ~sP10(X0,X1) | ~S(X0) [cnf transformation 1412]
2335. ~sP9(X0,X1) | oP(X1,X0) [cnf transformation 1414]
2336. ~P(X0,X1) | S(X0) | sP9(X1,X0) | sP10(X1,X0) [cnf transformation 1416]
2337. ~P(X0,X1) | S(X1) | sP9(X1,X0) | sP10(X1,X0) [cnf transformation 1416]
2339. P(sK163,sK164) [cnf transformation 1420]
2340. S(sK164) | S(sK163) [cnf transformation 1420]
2341. ~S(sK164) | ~S(sK163) [cnf transformation 1420]
2380. EX(X0,X1) | ~::(X1,treg,X1) | ~::(X0,X3,X1) [subsumption resolution 1481,1471]
2381. ~::(X0,X3,X1) | EX(X0,X1) [subsumption resolution 2380,1472]
2390. 1 <=> S(sK163) [avatar definition]
2394. 2 <=> S(sK164) [avatar definition]
2397. ~1 | ~2 [avatar split clause 2341,2394,2390]
2398. 1 | 2 [avatar split clause 2340,2394,2390]
3191. ::(X10,cnt,sK39(X10)) | ~::(X10,idcnt) [resolution 2311,1530]
3198. ::(X18,imen,sK160(X18,sreg)) | ~::(X18,sreg) [resolution 2311,1875]
3216. ::(X36,idcnt,sK160(X36,imen)) | ~::(X36,imen) [resolution 2311,1526]
3780. ::(X4,cnt,X5) | ~EX(X4,X5) | ~::(X4,cnt) [resolution 1894,2311]
3910. S(sK163) | sP9(sK164,sK163) | sP10(sK164,sK163) [resolution 2336,2339]
3913. 27 <=> sP10(sK164,sK163) [avatar definition]
3915. sP10(sK164,sK163) <- (27) [avatar component clause 3913]
3917. 28 <=> sP9(sK164,sK163) [avatar definition]
3919. sP9(sK164,sK163) <- (28) [avatar component clause 3917]
3921. 27 | 28 | 1 [avatar split clause 3910,2390,3917,3913]
3923. S(sK164) | sP9(sK164,sK163) | sP10(sK164,sK163) [resolution 2337,2339]
3926. 27 | 28 | 2 [avatar split clause 3923,2394,3917,3913]
3932. ~S(sK164) <- (27) [resolution 3915,2329]
3933. ~S(sK163) <- (27) [resolution 3915,2328]
3938. ~1 | ~27 [avatar split clause 3933,3913,2390]
3939. ~2 | ~27 [avatar split clause 3932,3913,2394]
3955. oP(sK163,sK164) <- (28) [resolution 3919,2335]
3995. ::(sK164,occ,sK48(sK164)) <- (28) [resolution 3955,1559]
3996. ::(sK163,occ,sK49(sK163)) <- (28) [resolution 3955,1558]
4020. ~::(sK164,cnt,sK48(sK164)) <- (28) [resolution 3995,1871]
4029. EX(sK164,sK48(sK164)) <- (28) [resolution 3995,2381]
4073. oP(sK163,sK163) <- (28) [resolution 3996,1553]
4102. ::(sK163,occ,sK48(sK163)) <- (28) [resolution 4073,1559]
4383. ~::(sK163,cnt,sK48(sK163)) <- (28) [resolution 4102,1871]
4392. EX(sK163,sK48(sK163)) <- (28) [resolution 4102,2381]
6867. ~::(X8,idcnt) | ::(X8,cnt) [resolution 3191,2312]
7533. ~::(X9,sreg) | ::(X9,imen) [resolution 3198,2312]
7652. ~::(X16,imen) | ::(X16,idcnt) [resolution 3216,2312]
9610. ~EX(sK163,sK48(sK163)) | ~::(sK163,cnt) <- (28) [resolution 3780,4383]
9615. ~EX(sK164,sK48(sK164)) | ~::(sK164,cnt) <- (28) [resolution 3780,4020]
9634. ~::(sK163,cnt) <- (28) [subsumption resolution 9610,4392]
9635. ~::(sK164,cnt) <- (28) [subsumption resolution 9615,4029]
34126. ::(X0,imen) | ~S(X0) [resolution 7533,2322]
35433. ::(X0,idcnt) | ~S(X0) [resolution 7652,34126]
35581. ::(X0,cnt) | ~S(X0) [resolution 35433,6867]
35604. ~S(sK163) <- (28) [resolution 35581,9634]
35605. ~S(sK164) <- (28) [resolution 35581,9635]
35608. ~1 | ~28 [avatar split clause 35604,3917,2390]
35609. ~2 | ~28 [avatar split clause 35605,3917,2394]
35610. $false [avatar sat refutation 2397,2398,3921,3926,3938,3939,35608,35609]
```

Proof of Theorem (t_{bd}4)

```
1. ! [X0] : ! [X1] : (::(X0,idcnt,X1) => cP(X0,X0,X1)) [input]
53. ! [X17] : ! [X1] : (::(X17,imen,X1) <=> (~? [X11] : (cP(X11,X17,X1) & ::(X11,mten,X1)) & ::(X17,idcnt,X1))) [
      input]
146. ! [X0] : ! [X8] : (tmP(X0,X8) => oP(X0,X8)) [input]
148. ! [X0] : (? [X1] : ::(X0,occ,X1) => tmP(X0,X0)) [input]
152. ! [X0] : (? [X1] : ::(X0,treg,X1) => tmP(X0,X0)) [input]
203. ! [X1] : ! [X6] : (::(X6,proc,X1) => ::(X6,occ,X1)) [input]
208. ! [X1] : ! [X6] : (::(X6,pbnd,X1) => ::(X6,occ,X1)) [input]
214. ! [X1] : ! [X6] : (::(X6,sreg,X1) => ::(X6,imen,X1)) [input]
257. ! [X6,X16] : (::(X6,X16) <=> ? [X1] : ::(X6,X16,X1)) [input]
258. ! [X6,X7] : (ISA(X6,X7) <=> ! [X13] : ! [X1] : (::(X13,X6,X1) => ::(X13,X7,X1))) [input]
262. ! [X6] : (PD(X6) <=> (::(X6,pbnd) | ::(X6,proc))) [input]
269. ! [X6] : (T(X6) <=> ::(X6,treg)) [input]
270. ! [X6] : (TR(X6) <=> T(X6)) [input]
271. ! [X6] : (S(X6) <=> ::(X6,sreg)) [input]
273. ! [X6] : (R(X6) <=> (PR(X6) | TR(X6))) [input]
274. ! [X6] : (AB(X6) <=> R(X6)) [input]
276. ! [X6,X7] : (P(X6,X7) <=> ((ISA(X6,X7) & ~S(X7) & ~S(X6) & PR(X7) & PR(X6)) | (? [X1] : cP(X6,X7,X1) & S(X7)
      & S(X6)) | (oP(X6,X7) & ((T(X7) & T(X6)) | (PD(X7) & PD(X6)))))) [input]
292. ! [X6] : ((PD(X6) | AB(X6)) => P(X6,X6)) [input]
293. ~! [X6] : ((PD(X6) | AB(X6)) => P(X6,X6)) [negated conjecture 292]
```

```
294. ! [X0,X1] : (::(X0,idcnt,X1) => cP(X0,X0,X1)) [flattening 1]
393. ! [X0] : ! [X1] : (::(X0,imen,X1) <=> (~? [X2] : (cP(X2,X0,X1) & ::(X2,mten,X1)) & ::(X0,idcnt,X1))) [rectify
     53]
394. ! [X0,X1] : (::(X0,imen,X1) <=> (~? [X2] : (cP(X2,X0,X1) & ::(X2,mten,X1)) & ::(X0,idcnt,X1))) [flattening
     393]
571. ! [X0] : ! [X1] : (tmP(X0,X1) => oP(X0,X1)) [rectify 146]
572. ! [X0,X1] : (tmP(X0,X1) => oP(X0,X1)) [flattening 571]
620. ! [X0] : ! [X1] : (::(X1,proc,X0) => ::(X1,occ,X0)) [rectify 203]
621. ! [X0,X1] : (::(X1,proc,X0) => ::(X1,occ,X0)) [flattening 620]
630. ! [X0] : ! [X1] : (::(X1,pbnd,X0) => ::(X1,occ,X0)) [rectify 208]
631. ! [X0,X1] : (::(X1,pbnd,X0) => ::(X1,occ,X0)) [flattening 630]
642. ! [X0] : ! [X1] : (::(X1,sreg,X0) => ::(X1,imen,X0)) [rectify 214]
643. ! [X0,X1] : (::(X1,sreg,X0) => ::(X1,imen,X0)) [flattening 642]
705. ! [X0,X1] : (::(X0,X1) <=> ? [X2] : ::(X0,X1,X2)) [rectify 257]
706. ! [X0,X1] : (ISA(X0,X1) <=> ! [X2] : ! [X3] : (::(X2,X0,X3) => ::(X2,X1,X3))) [rectify 258]
707. ! [X0,X1] : (ISA(X0,X1) <=> ! [X2,X3] : (::(X2,X0,X3) => ::(X2,X1,X3))) [flattening 706]
711. ! [X0] : (PD(X0) <=> (::(X0,pbnd) | ::(X0,proc))) [rectify 262]
718. ! [X0] : (T(X0) <=> ::(X0,treg)) [rectify 269]
719. ! [X0] : (TR(X0) <=> T(X0)) [rectify 270]
720. ! [X0] : (S(X0) <=> ::(X0,sreg)) [rectify 271]
722. ! [X0] : (R(X0) <=> (PR(X0) | TR(X0))) [rectify 273]
723. ! [X0] : (AB(X0) <=> R(X0)) [rectify 274]
725. ! [X0,X1] : (P(X0,X1) <=> ((ISA(X0,X1) & ~S(X1) & ~S(X0) & PR(X1) & PR(X0)) | (? [X2] : cP(X0,X1,X2) & S(X1)
     & S(X0)) | (oP(X0,X1) & ((T(X1) & T(X0)) | (PD(X1) & PD(X0)))))) [rectify 276]
741. ~! [X0] : ((PD(X0) | AB(X0)) => P(X0,X0)) [rectify 293]
742. ! [X0,X1] : (((ISA(X0,X1) & ~S(X1) & ~S(X0) & PR(X1) & PR(X0)) | (? [X2] : cP(X0,X1,X2) & S(X1) & S(X0)) | (
     oP(X0,X1) & ((T(X1) & T(X0)) | (PD(X1) & PD(X0))))) => P(X0,X1)) [unused predicate definition removal 725]
743. ! [X0] : (AB(X0) => R(X0)) [unused predicate definition removal 723]
744. ! [X0] : (R(X0) => (PR(X0) | TR(X0))) [unused predicate definition removal 722]
745. ! [X0] : (TR(X0) => T(X0)) [unused predicate definition removal 719]
746. ! [X0,X1] : (! [X2,X3] : (::(X2,X0,X3) => ::(X2,X1,X3)) => ISA(X0,X1)) [unused predicate definition removal
     707]
752. ! [X0,X1] : (cP(X0,X0,X1) | ~::(X0,idcnt,X1)) [ennf transformation 294]
827. ! [X0,X1] : (::(X0,imen,X1) <=> (! [X2] : (~cP(X2,X0,X1) | ~::(X2,mten,X1)) & ::(X0,idcnt,X1))) [ennf
     transformation 394]
960. ! [X0,X1] : (oP(X0,X1) | ~tmP(X0,X1)) [ennf transformation 572]
962. ! [X0] : (tmP(X0,X0) | ! [X1] : ~::(X0,occ,X1)) [ennf transformation 148]
968. ! [X0] : (tmP(X0,X0) | ! [X1] : ~::(X0,treg,X1)) [ennf transformation 152]
997. ! [X0,X1] : (::(X1,occ,X0) | ~::(X1,proc,X0)) [ennf transformation 621]
1002. ! [X0,X1] : (::(X1,occ,X0) | ~::(X1,pbnd,X0)) [ennf transformation 631]
1008. ! [X0,X1] : (::(X1,imen,X0) | ~::(X1,sreg,X0)) [ennf transformation 643]
1046. ! [X0,X1] : (ISA(X0,X1) | ? [X2,X3] : (~::(X2,X1,X3) & ::(X2,X0,X3))) [ennf transformation 746]
1047. ! [X0] : (T(X0) | ~TR(X0)) [ennf transformation 745]
1048. ! [X0] : ((PR(X0) | TR(X0)) | ~R(X0)) [ennf transformation 744]
1049. ! [X0] : (PR(X0) | TR(X0) | ~R(X0)) [flattening 1048]
1050. ! [X0] : (R(X0) | ~AB(X0)) [ennf transformation 743]
1051. ! [X0,X1] : (P(X0,X1) | ((~ISA(X0,X1) | S(X1) | S(X0) | ~PR(X1) | ~PR(X0)) & (! [X2] : ~cP(X0,X1,X2) | ~S(X1
     ) | ~S(X0)) & (~oP(X0,X1) | ((~T(X1) | ~T(X0)) & (~PD(X1) | ~PD(X0)))))) [ennf transformation 742]
1052. ? [X0] : (~P(X0,X0) & (PD(X0) | AB(X0))) [ennf transformation 741]
1123. ! [X0,X1] : ((::(X0,imen,X1) | (? [X2] : (cP(X2,X0,X1) & ::(X2,mten,X1)) | ~::(X0,idcnt,X1))) & ((! [X2] :
     (~cP(X2,X0,X1) | ~::(X2,mten,X1)) & ::(X0,idcnt,X1)) | ~::(X0,imen,X1))) [nnf transformation 827]
1124. ! [X0,X1] : ((::(X0,imen,X1) | ? [X2] : (cP(X2,X0,X1) & ::(X2,mten,X1)) | ~::(X0,idcnt,X1)) & ((! [X2] : (~
     cP(X2,X0,X1) | ~::(X2,mten,X1)) & ::(X0,idcnt,X1)) | ~::(X0,imen,X1))) [flattening 1123]
1125. ! [X0,X1] : ((::(X0,imen,X1) | ? [X2] : (cP(X2,X0,X1) & ::(X2,mten,X1)) | ~::(X0,idcnt,X1)) & ((! [X3] : (~
     cP(X3,X0,X1) | ~::(X3,mten,X1)) & ::(X0,idcnt,X1)) | ~::(X0,imen,X1))) [rectify 1124]
1126. ! [X1,X0] : (? [X2] : (cP(X2,X0,X1) & ::(X2,mten,X1)) => (cP(sK33(X0,X1),X0,X1) & ::(sK33(X0,X1),mten,X1)))
     [choice axiom]
1127. ! [X0,X1] : ((::(X0,imen,X1) | (cP(sK33(X0,X1),X0,X1) & ::(sK33(X0,X1),mten,X1)) | ~::(X0,idcnt,X1)) & ((! [
     X3] : (~cP(X3,X0,X1) | ~::(X3,mten,X1)) & ::(X0,idcnt,X1)) | ~::(X0,imen,X1))) [skolemisation 1125,1126]
1393. ! [X0,X1] : (((::(X0,X1) | ! [X2] : ~::(X0,X1,X2)) & (? [X2] : ::(X0,X1,X2) | ~::(X0,X1))) [nnf
     transformation 705]
1394. ! [X0,X1] : (((::(X0,X1) | ! [X2] : ~::(X0,X1,X2)) & (? [X3] : ::(X0,X1,X3) | ~::(X0,X1))) [rectify 1393]
1395. ! [X1,X0] : (? [X3] : ::(X0,X1,X3) => ::(X0,X1,sK158(X0,X1))) [choice axiom]
1396. ! [X0,X1] : (((::(X0,X1) | ! [X2] : ~::(X0,X1,X2)) & (::(X0,X1,sK158(X0,X1)) | ~::(X0,X1))) [skolemisation
     1394,1395]
1397. ! [X1,X0] : (? [X2,X3] : (~::(X2,X1,X3) & ::(X2,X0,X3)) => (~::(sK159(X0,X1),X1,sK160(X0,X1)) & ::(sK159(X0,
     X1),X0,sK160(X0,X1)))) [choice axiom]
1398. ! [X0,X1] : (ISA(X0,X1) | (~::(sK159(X0,X1),X1,sK160(X0,X1)) & ::(sK159(X0,X1),X0,sK160(X0,X1)))) [
     skolemisation 1046,1397]
1401. ! [X0] : ((PD(X0) | (~::(X0,pbnd) & ~::(X0,proc))) & ((::(X0,pbnd) | ::(X0,proc)) | ~PD(X0))) [nnf
     transformation 711]
1402. ! [X0] : ((PD(X0) | (~::(X0,pbnd) & ~::(X0,proc))) & (::(X0,pbnd) | ::(X0,proc) | ~PD(X0))) [flattening
     1401]
1406. ! [X0] : ((T(X0) | ~::(X0,treg)) & (::(X0,treg) | ~T(X0))) [nnf transformation 718]
1407. ! [X0] : ((S(X0) | ~::(X0,sreg)) & (::(X0,sreg) | ~S(X0))) [nnf transformation 720]
1413. ? [X0] : (~P(X0,X0) & (PD(X0) | AB(X0))) => (~P(sK162,sK162) & (PD(sK162) | AB(sK162))) [choice axiom]
1414. ~P(sK162,sK162) & (PD(sK162) | AB(sK162)) [skolemisation 1052,1413]
1415. ~::(X0,idcnt,X1) | cP(X0,X0,X1) [cnf transformation 752]
1520. ~::(X0,imen,X1) | ::(X0,idcnt,X1) [cnf transformation 1127]
1783. ~tmP(X0,X1) | oP(X0,X1) [cnf transformation 960]
1785. ~::(X0,occ,X1) | tmP(X0,X0) [cnf transformation 962]
1789. ~::(X0,treg,X1) | tmP(X0,X0) [cnf transformation 968]
1858. ~::(X1,proc,X0) | ::(X1,occ,X0) [cnf transformation 997]
1863. ~::(X1,pbnd,X0) | ::(X1,occ,X0) [cnf transformation 1002]
1869. ~::(X1,sreg,X0) | ::(X1,imen,X0) [cnf transformation 1008]
2305. ::(X0,X1,sK158(X0,X1)) | ~::(X0,X1) [cnf transformation 1396]
2306. ~::(X0,X1,X2) | ::(X0,X1) [cnf transformation 1396]
2307. ::(sK159(X0,X1),X0,sK160(X0,X1)) | ISA(X0,X1) [cnf transformation 1398]
2308. ~::(sK159(X0,X1),X1,sK160(X0,X1)) | ISA(X0,X1) [cnf transformation 1398]
2312. ::(X0,pbnd) | ::(X0,proc) | ~PD(X0) [cnf transformation 1402]
2320. ::(X0,treg) | ~T(X0) [cnf transformation 1406]
```

```
2322. ~TR(X0) | T(X0) [cnf transformation 1047]
2323. ::(X0,sreg) | ~S(X0) [cnf transformation 1407]
2329. ~R(X0) | TR(X0) | PR(X0) [cnf transformation 1049]
2330. ~AB(X0) | R(X0) [cnf transformation 1050]
2331. ~oP(X0,X1) | P(X0,X1) | ~PD(X1) | ~PD(X0) [cnf transformation 1051]
2332. ~oP(X0,X1) | P(X0,X1) | ~T(X1) | ~T(X0) [cnf transformation 1051]
2333. ~cP(X0,X1,X2) | P(X0,X1) | ~S(X1) | ~S(X0) [cnf transformation 1051]
2334. ~ISA(X0,X1) | P(X0,X1) | S(X1) | S(X0) | ~PR(X1) | ~PR(X0) [cnf transformation 1051]
2335. PD(sK162) | AB(sK162) [cnf transformation 1414]
2336. ~P(sK162,sK162) [cnf transformation 1414]
2385. 1 <=> AB(sK162) [avatar definition]
2387. AB(sK162) <- (1) [avatar component clause 2385]
2389. 2 <=> PD(sK162) [avatar definition]
2392. 1 | 2 [avatar split clause 2335,2389,2385]
3288. cP(X12,X12,sK158(X12,idcnt)) | ~::(X12,idcnt) [resolution 2305,1415]
3293. ::(X18,imen,sK158(X18,sreg)) | ~::(X18,sreg) [resolution 2305,1869]
3302. ~::(X27,treg) | tmP(X27,X27) [resolution 2305,1789]
3311. ::(X36,idcnt,sK158(X36,imen)) | ~::(X36,imen) [resolution 2305,1520]
3331. ::(X56,occ,sK158(X56,proc)) | ~::(X56,proc) [resolution 2305,1858]
3343. ::(X68,occ,sK158(X68,pbnd)) | ~::(X68,pbnd) [resolution 2305,1863]
4239. ISA(X0,X0) | ISA(X0,X0) [resolution 2308,2307]
4240. ISA(X0,X0) [duplicate literal removal 4239]
4654. tmP(X0,X0) | ~T(X0) [resolution 3302,2320]
4668. oP(X2,X2) | ~T(X2) [resolution 4654,1783]
4688. ~T(X9) | P(X9,X9) | ~T(X9) | ~T(X9) [resolution 4668,2332]
4689. P(X9,X9) | ~T(X9) [duplicate literal removal 4688]
4698. ~T(sK162) [resolution 4689,2336]
4799. P(X2,X2) | S(X2) | S(X2) | ~PR(X2) | ~PR(X2) [resolution 2334,4240]
4800. P(X2,X2) | S(X2) | ~PR(X2) [duplicate literal removal 4799]
8035. ~::(X32,idcnt) | P(X32,X32) | ~S(X32) | ~S(X32) [resolution 3288,2333]
8041. ~::(X32,idcnt) | P(X32,X32) | ~S(X32) [duplicate literal removal 8035]
8091. ~::(X8,sreg) | ::(X8,imen) [resolution 3293,2306]
8246. ~::(X15,imen) | ::(X15,idcnt) [resolution 3311,2306]
8348. ~::(X13,proc) | tmP(X13,X13) [resolution 3331,1785]
8437. ~::(X13,pbnd) | tmP(X13,X13) [resolution 3343,1785]
22989. ::(X0,imen) | ~S(X0) [resolution 8091,2323]
24365. ::(X0,idcnt) | ~S(X0) [resolution 8246,22989]
25813. tmP(X0,X0) | ::(X0,proc) | ~PD(X0) [resolution 8437,2312]
25817. tmP(X0,X0) | ~PD(X0) [subsumption resolution 25813,8348]
25840. oP(X2,X2) | ~PD(X2) [resolution 25817,1783]
25898. ~PD(X8) | P(X8,X8) | ~PD(X8) | ~PD(X8) [resolution 25840,2331]
25926. P(X8,X8) | ~PD(X8) [duplicate literal removal 25898]
25948. ~PD(sK162) [resolution 25926,2336]
25951. ~2 [avatar split clause 25948,2389]
25952. R(sK162) <- (1) [resolution 2387,2330]
25953. TR(sK162) | PR(sK162) <- (1) [resolution 25952,2329]
25955. 811 <=> PR(sK162) [avatar definition]
25957. PR(sK162) <- (811) [avatar component clause 25955]
25959. 812 <=> TR(sK162) [avatar definition]
25961. TR(sK162) <- (812) [avatar component clause 25959]
25962. 811 | 812 | ~1 [avatar split clause 25953,2385,25959,25955]
25969. 814 <=> S(sK162) [avatar definition]
25971. S(sK162) <- (814) [avatar component clause 25969]
26000. T(sK162) <- (812) [resolution 25961,2322]
26001. $false <- (812) [subsumption resolution 26000,4698]
26002. ~812 [avatar contradiction clause 26001]
31799. S(sK162) | ~PR(sK162) [resolution 4800,2336]
31800. S(sK162) <- (811) [subsumption resolution 31799,25957]
31801. 814 | ~811 [avatar split clause 31800,25955,25969]
36595. P(X0,X0) | ~S(X0) | ~S(X0) [resolution 8041,24365]
36622. P(X0,X0) | ~S(X0) [duplicate literal removal 36595]
44332. ~S(sK162) [resolution 36622,2336]
44333. $false <- (814) [subsumption resolution 44332,25971]
44334. ~814 [avatar contradiction clause 44333]
44335. $false [avatar sat refutation 2392,25951,25962,26002,31801,44334]
```

Proof of Theorem ($t_{bd}5$)

```
This went KO with both provers
```

Proof of Theorem ($t_{bd}6$)

```
30. ! [X17] : ! [X1] : (EX(X17,X1) => (::(X1,treg,X1) & PAR(X1) & PAR(X17))) [input]
31. ! [X17] : ! [X16] : ! [X1] : (::(X17,X16,X1) => (::(X1,treg,X1) & UNI(X16) & PAR(X17))) [input]
34. ! [X0] : ! [X1] : (? [X16] : (::(X1,treg,X1) & ::(X0,X16,X1) & UNI(X16)) <=> (EX(X0,X1) & ::(X1,treg,X1) & PAR
    (X0))) [input]
53. ! [X17] : ! [X1] : (::(X17,imen,X1) <=> (~? [X11] : (cP(X11,X17,X1) & ::(X11,mten,X1)) & ::(X17,idcnt,X1))) [
    input]
54. ! [X28] : (? [X1] : ::(X28,idcnt,X1) <=> (~? [X29] : ? [X1] : (GDEP(X28,X29,X1) | SDEP(X28,X29)) & ? [X1] :
    ::(X28,cnt,X1))) [input]
62. ! [X0] : ! [X8] : ! [X9] : ((oP(X8,X9) & oP(X0,X8)) => oP(X0,X9)) [input]
64. ! [X0] : ! [X8] : (oP(X0,X8) => (? [X1] : ::(X8,occ,X1) & ? [X1] : ::(X0,occ,X1))) [input]
69. ! [X2] : ! [X3] : (oP(X2,X3) => (? [X1] : ::(X2,treg,X1) <=> ? [X1] : ::(X3,treg,X1))) [input]
191. UNI(idcnt) [input]
201. ~? [X6] : (PAR(X6) & UNI(X6)) [input]
203. ! [X1] : ! [X6] : (::(X6,proc,X1) => ::(X6,occ,X1)) [input]
207. ! [X1] : ! [X6] : (::(X6,treg,X1) => ::(X6,occ,X1)) [input]
208. ! [X1] : ! [X6] : (::(X6,pbnd,X1) => ::(X6,occ,X1)) [input]
210. ~? [X6] : ? [X1] : (::(X6,occ,X1) & ::(X6,cnt,X1)) [input]
```

```
214. ! [X1] : ! [X6] : (::(X6,sreg,X1) => ::(X6,imen,X1)) [input]
234. ! [X6] : (? [X1] : ::(X6,cnt,X1) => ! [X1] : (EX(X6,X1) => ::(X6,cnt,X1))) [input]
257. ! [X6,X16] : (::(X6,X16) <=> ? [X1] : ::(X6,X16,X1)) [input]
262. ! [X6] : (PD(X6) <=> (::(X6,pbnd) | ::(X6,proc))) [input]
269. ! [X6] : (T(X6) <=> ::(X6,treg)) [input]
271. ! [X6] : (S(X6) <=> ::(X6,sreg)) [input]
276. ! [X6,X7] : (P(X6,X7) <=> ((! [X6] : (EX(X6,X1) => cP(X6,X7,X1)) & S(X7) & S(X6)) | (oP(X6,X7) & ((T(X7) & T(
     X6)) | (PD(X7) & PD(X6)))))) [input]
291. ! [X6,X7,X13] : ((P(X7,X13) & P(X6,X7)) => P(X6,X13)) [input]
292. ~! [X6,X7,X13] : ((P(X7,X13) & P(X6,X7)) => P(X6,X13)) [negated conjecture 291]
349. ! [X0] : ! [X1] : (EX(X0,X1) => (::(X1,treg,X1) & PAR(X1) & PAR(X0))) [rectify 30]
350. ! [X0,X1] : (EX(X0,X1) => (::(X1,treg,X1) & PAR(X1) & PAR(X0))) [flattening 349]
351. ! [X0] : ! [X1] : ! [X2] : (::(X0,X1,X2) => (::(X2,treg,X2) & UNI(X1) & PAR(X0))) [rectify 31]
352. ! [X0,X1,X2] : (::(X0,X1,X2) => (::(X2,treg,X2) & UNI(X1) & PAR(X0))) [flattening 351]
356. ! [X0] : ! [X1] : (? [X2] : (::(X1,treg,X1) & ::(X0,X2,X1) & UNI(X2)) <=> EX(X0,X1) & ::(X1,treg,X1) & PAR(
     X0))) [rectify 34]
357. ! [X0,X1] : (? [X2] : (::(X1,treg,X1) & ::(X0,X2,X1) & UNI(X2)) <=> EX(X0,X1) & ::(X1,treg,X1) & PAR(X0))) [
     flattening 356]
392. ! [X0] : ! [X1] : (::(X0,imen,X1) <=> (~? [X2] : (cP(X2,X0,X1) & ::(X2,mten,X1)) & ::(X0,idcnt,X1))) [rectify
      53]
393. ! [X0,X1] : (::(X0,imen,X1) <=> (~? [X2] : (cP(X2,X0,X1) & ::(X2,mten,X1)) & ::(X0,idcnt,X1))) [flattening
     392]
394. ! [X0] : (? [X1] : ::(X0,idcnt,X1) <=> (~? [X2] : ? [X3] : (GDEP(X0,X2,X3) | SDEP(X0,X2)) & ? [X4] : ::(X0,
     cnt,X4))) [rectify 54]
395. ! [X0] : (? [X1] : ::(X0,idcnt,X1) <=> (~? [X2,X3] : (GDEP(X0,X2,X3) | SDEP(X0,X2)) & ? [X4] : ::(X0,cnt,X4))
     ) [flattening 394]
408. ! [X0] : ! [X1] : ! [X2] : ((oP(X1,X2) & oP(X0,X1)) => oP(X0,X2)) [rectify 62]
409. ! [X0,X1,X2] : ((oP(X1,X2) & oP(X0,X1)) => oP(X0,X2)) [flattening 408]
412. ! [X0] : ! [X1] : (oP(X0,X1) => (? [X2] : ::(X1,occ,X2) & ? [X3] : ::(X0,occ,X3))) [rectify 64]
413. ! [X0,X1] : (oP(X0,X1) => (? [X2] : ::(X1,occ,X2) & ? [X3] : ::(X0,occ,X3))) [flattening 412]
422. ! [X0] : ! [X1] : (oP(X0,X1) => (? [X2] : ::(X0,treg,X2) <=> ? [X3] : ::(X1,treg,X3))) [rectify 69]
423. ! [X0,X1] : (oP(X0,X1) => (? [X2] : ::(X0,treg,X2) <=> ? [X3] : ::(X1,treg,X3))) [flattening 422]
616. ~? [X0] : (PAR(X0) & UNI(X0)) [rectify 201]
619. ! [X0] : ! [X1] : (::(X1,proc,X0) => ::(X1,occ,X0)) [rectify 203]
620. ! [X0,X1] : (::(X1,proc,X0) => ::(X1,occ,X0)) [flattening 619]
627. ! [X0] : ! [X1] : (::(X1,treg,X0) => ::(X1,occ,X0)) [rectify 207]
628. ! [X0,X1] : (::(X1,treg,X0) => ::(X1,occ,X0)) [flattening 627]
629. ! [X0] : ! [X1] : (::(X1,pbnd,X0) => ::(X1,occ,X0)) [rectify 208]
630. ! [X0,X1] : (::(X1,pbnd,X0) => ::(X1,occ,X0)) [flattening 629]
633. ~? [X0] : ? [X1] : (::(X0,occ,X1) & ::(X0,cnt,X1)) [rectify 210]
634. ~? [X0,X1] : (::(X0,occ,X1) & ::(X0,cnt,X1)) [flattening 633]
641. ! [X0] : ! [X1] : (::(X1,sreg,X0) => ::(X1,imen,X0)) [rectify 214]
642. ! [X0,X1] : (::(X1,sreg,X0) => ::(X1,imen,X0)) [flattening 641]
678. ! [X0] : (? [X1] : ::(X0,cnt,X1) => ! [X2] : (EX(X0,X2) => ::(X0,cnt,X2))) [rectify 234]
704. ! [X0,X1] : (::(X0,X1) <=> ? [X2] : ::(X0,X1,X2)) [rectify 257]
710. ! [X0] : (PD(X0) <=> (::(X0,pbnd) | ::(X0,proc))) [rectify 262]
717. ! [X0] : (T(X0) <=> ::(X0,treg)) [rectify 269]
719. ! [X0] : (S(X0) <=> ::(X0,sreg)) [rectify 271]
724. ! [X0,X1] : (P(X0,X1) <=> ((! [X2] : (EX(X2,X3) => cP(X2,X1,X3)) & S(X1) & S(X0)) | (oP(X0,X1) & ((T(X1) & T(
     X0)) | (PD(X1) & PD(X0)))))) [rectify 276]
725. ! [X3] : ! [X0,X1] : (P(X0,X1) <=> ((! [X2] : (EX(X2,X3) => cP(X2,X1,X3)) & S(X1) & S(X0)) | (oP(X0,X1) & ((T
     (X1) & T(X0)) | (PD(X1) & PD(X0)))))) [closure 724]
726. ! [X3,X0,X1] : (P(X0,X1) <=> ((! [X2] : (EX(X2,X3) => cP(X2,X1,X3)) & S(X1) & S(X0)) | (oP(X0,X1) & ((T(X1) &
     T(X0)) | (PD(X1) & PD(X0)))))) [flattening 725]
741. ~! [X0,X1,X2] : ((P(X1,X2) & P(X0,X1)) => P(X0,X2)) [rectify 292]
795. ! [X0,X1] : ((::(X1,treg,X1) & PAR(X1) & PAR(X0)) | ~EX(X0,X1)) [ennf transformation 350]
796. ! [X0,X1,X2] : ((::(X2,treg,X2) & UNI(X1) & PAR(X0)) | ~::(X0,X1,X2)) [ennf transformation 352]
822. ! [X0,X1] : (::(X0,imen,X1) <=> (! [X2] : (~cP(X2,X0,X1) | ~::(X2,mten,X1)) & ::(X0,idcnt,X1))) [ennf
     transformation 393]
823. ! [X0] : (? [X1] : ::(X0,idcnt,X1) <=> (! [X2,X3] : (~GDEP(X0,X2,X3) & ~SDEP(X0,X2)) & ? [X4] : ::(X0,cnt,X4)
     )) [ennf transformation 395]
836. ! [X0,X1,X2] : (oP(X0,X2) | (~oP(X1,X2) | ~oP(X0,X1))) [ennf transformation 409]
837. ! [X0,X1,X2] : (oP(X0,X2) | ~oP(X1,X2) | ~oP(X0,X1)) [flattening 836]
839. ! [X0,X1] : ((? [X2] : ::(X1,occ,X2) & ? [X3] : ::(X0,occ,X3)) | ~oP(X0,X1)) [ennf transformation 413]
845. ! [X0,X1] : ((? [X2] : ::(X0,treg,X2) <=> ? [X3] : ::(X1,treg,X3)) | ~oP(X0,X1)) [ennf transformation 423]
990. ! [X0] : (~PAR(X0) | ~UNI(X0)) [ennf transformation 616]
992. ! [X0,X1] : (::(X1,occ,X0) | ~::(X1,proc,X0)) [ennf transformation 620]
996. ! [X0,X1] : (::(X1,occ,X0) | ~::(X1,treg,X0)) [ennf transformation 628]
997. ! [X0,X1] : (::(X1,occ,X0) | ~::(X1,pbnd,X0)) [ennf transformation 630]
999. ! [X0,X1] : (~::(X0,occ,X1) | ~::(X0,cnt,X1)) [ennf transformation 634]
1003. ! [X0,X1] : (::(X1,imen,X0) | ~::(X1,sreg,X0)) [ennf transformation 642]
1022. ! [X0] : (! [X2] : (::(X0,cnt,X2) | ~EX(X0,X2)) | ! [X1] : ~::(X0,cnt,X1)) [ennf transformation 678]
1041. ! [X3,X0,X1] : (P(X0,X1) <=> ((! [X2] : (cP(X2,X1,X3) | ~EX(X2,X3)) & S(X1) & S(X0)) | (oP(X0,X1) & ((T(X1)
      & T(X0)) | (PD(X1) & PD(X0)))))) [ennf transformation 726]
1042. ? [X0,X1,X2] : (~P(X0,X2) & (P(X1,X2) & P(X0,X1))) [ennf transformation 741]
1043. ? [X0,X1,X2] : (~P(X0,X2) & P(X1,X2) & P(X0,X1)) [flattening 1042]
1059. ! [X1,X0] : (sP9(X1,X0) <=> (oP(X0,X1) & ((T(X1) & T(X0)) | (PD(X1) & PD(X0))))) [predicate definition
      introduction]
1060. ! [X3,X0,X1] : (P(X0,X1) <=> ((! [X2] : (cP(X2,X1,X3) | ~EX(X2,X3)) & S(X1) & S(X0)) | sP9(X1,X0))) [
      definition folding 1041,1059]
1087. ! [X0,X1] : ((? [X2] : (::(X1,treg,X1) & ::(X0,X2,X1) & UNI(X2)) | (~EX(X0,X1) | ~::(X1,treg,X1) | ~PAR(X0))
      ) & ((EX(X0,X1) & ::(X1,treg,X1) & PAR(X0)) | ! [X2] : (~::(X1,treg,X1) | ~::(X0,X2,X1) | ~UNI(X2)))) [nnf
      transformation 357]
1088. ! [X0,X1] : ((? [X2] : (::(X1,treg,X1) & ::(X0,X2,X1) & UNI(X2)) | ~EX(X0,X1) | ~::(X1,treg,X1) | ~PAR(X0))
      & ((EX(X0,X1) & ::(X1,treg,X1) & PAR(X0)) | ! [X2] : (~::(X1,treg,X1) | ~::(X0,X2,X1) | ~UNI(X2)))) [
      flattening 1087]
1089. ! [X0,X1] : ((? [X2] : (::(X1,treg,X1) & ::(X0,X2,X1) & UNI(X2)) | ~EX(X0,X1) | ~::(X1,treg,X1) | ~PAR(X0))
      & ((EX(X0,X1) & ::(X1,treg,X1) & PAR(X0)) | ! [X3] : (~::(X1,treg,X1) | ~::(X0,X3,X1) | ~UNI(X3)))) [rectify
      1088]
1090. ! [X1,X0] : (? [X2] : (::(X1,treg,X1) & ::(X0,X2,X1) & UNI(X2)) => (::(X1,treg,X1) & ::(X0,sK20(X0,X1),X1) &
      UNI(sK20(X0,X1)))) [choice axiom]
```

```
1091. ! [X0,X1] : (((::(X1,treg,X1) & ::(X0,sK20(X0,X1),X1) & UNI(sK20(X0,X1))) | ~EX(X0,X1) | ~::(X1,treg,X1) | ~PAR(X0)) & ((EX(X0,X1) & ::(X1,treg,X1) & PAR(X0)) | ! [X3] : (~::(X1,treg,X1) | ~::(X0,X3,X1) | ~UNI(X3)))) [skolemisation 1089,1090]
1116. ! [X0,X1] : ((::(X0,imen,X1) | (? [X2] : (cP(X2,X0,X1) & ::(X2,mten,X1)) | ~::(X0,idcnt,X1))) & ((! [X2] : (~cP(X2,X0,X1) | ~::(X2,mten,X1)) & ::(X0,idcnt,X1)) | ~::(X0,imen,X1))) [nnf transformation 822]
1117. ! [X0,X1] : ((::(X0,imen,X1) | ? [X2] : (cP(X2,X0,X1) & ::(X2,mten,X1)) | ~::(X0,idcnt,X1)) & ((! [X2] : (~cP(X2,X0,X1) | ~::(X2,mten,X1)) & ::(X0,idcnt,X1)) | ~::(X0,imen,X1))) [flattening 1116]
1118. ! [X0,X1] : ((::(X0,imen,X1) | ? [X2] : (cP(X2,X0,X1) & ::(X2,mten,X1)) | ~::(X0,idcnt,X1)) & ((! [X3] : (~cP(X3,X0,X1) | ~::(X3,mten,X1)) & ::(X0,idcnt,X1)) | ~::(X0,imen,X1))) [rectify 1117]
1119. ! [X1,X0] : (? [X2] : (cP(X2,X0,X1) & ::(X2,mten,X1)) => (cP(sK34(X0,X1),X0,X1) & ::(sK34(X0,X1),mten,X1))) [choice axiom]
1120. ! [X0,X1] : ((::(X0,imen,X1) | (cP(sK34(X0,X1),X0,X1) & ::(sK34(X0,X1),mten,X1)) | ~::(X0,idcnt,X1)) & ((! [X3] : (~cP(X3,X0,X1) | ~::(X3,mten,X1)) & ::(X0,idcnt,X1)) | ~::(X0,imen,X1))) [skolemisation 1118,1119]
1121. ! [X0] : ((? [X1] : ::(X0,idcnt,X1) | (? [X2,X3] : (GDEP(X0,X2,X3) | SDEP(X0,X2)) | ! [X4] : ~::(X0,cnt,X4)) & ((! [X2,X3] : (~GDEP(X0,X2,X3) & ~SDEP(X0,X2)) & ? [X4] : ::(X0,cnt,X4)) | ! [X1] : ~::(X0,idcnt,X1))) [nnf transformation 823]
1122. ! [X0] : ((? [X1] : ::(X0,idcnt,X1) | ? [X2,X3] : (GDEP(X0,X2,X3) | SDEP(X0,X2)) | ! [X4] : ~::(X0,cnt,X4)) & ((! [X2,X3] : (~GDEP(X0,X2,X3) & ~SDEP(X0,X2)) & ? [X4] : ::(X0,cnt,X4)) | ! [X1] : ~::(X0,idcnt,X1))) [flattening 1121]
1123. ! [X0] : ((? [X1] : ::(X0,idcnt,X1) | ? [X2,X3] : (GDEP(X0,X2,X3) | SDEP(X0,X2)) | ! [X4] : ~::(X0,cnt,X4)) & ((! [X5,X6] : (~GDEP(X0,X5,X6) & ~SDEP(X0,X5)) & ? [X7] : ::(X0,cnt,X7)) | ! [X8] : ~::(X0,idcnt,X8))) [rectify 1122]
1124. ! [X0] : (? [X1] : ::(X0,idcnt,X1) => ::(X0,idcnt,sK35(X0))) [choice axiom]
1125. ! [X0] : (? [X2,X3] : (GDEP(X0,X2,X3) | SDEP(X0,X2)) => (GDEP(X0,sK36(X0),sK37(X0)) | SDEP(X0,sK36(X0)))) [choice axiom]
1126. ! [X0] : (? [X7] : ::(X0,cnt,X7) => ::(X0,cnt,sK38(X0))) [choice axiom]
1127. ! [X0] : ((::(X0,idcnt,sK35(X0)) | (GDEP(X0,sK36(X0),sK37(X0)) | SDEP(X0,sK36(X0))) | ! [X4] : ~::(X0,cnt,X4)) & ((! [X5,X6] : (~GDEP(X0,X5,X6) & ~SDEP(X0,X5)) & ::(X0,cnt,sK38(X0))) | ! [X8] : ~::(X0,idcnt,X8))) [skolemisation 1123,1126,1125,1124]
1143. ! [X1] : (? [X2] : ::(X1,occ,X2) => ::(X1,occ,sK47(X1))) [choice axiom]
1144. ! [X0] : (? [X3] : ::(X0,occ,X3) => ::(X0,occ,sK48(X0))) [choice axiom]
1145. ! [X0,X1] : ((::(X1,occ,sK47(X1)) & ::(X0,occ,sK48(X0))) | ~oP(X0,X1)) [skolemisation 839,1144,1143]
1158. ! [X0,X1] : (((? [X2] : ::(X0,treg,X2) | ! [X3] : ~::(X1,treg,X3)) & (? [X3] : ::(X1,treg,X3) | ! [X2] : ~::(X0,treg,X2))) | ~oP(X0,X1)) [nnf transformation 845]
1159. ! [X0,X1] : (((? [X2] : ::(X0,treg,X2) | ! [X3] : ~::(X1,treg,X3)) & (? [X4] : ::(X1,treg,X4) | ! [X5] : ~::(X0,treg,X5))) | ~oP(X0,X1)) [rectify 1158]
1160. ! [X0] : (? [X2] : ::(X0,treg,X2) => ::(X0,treg,sK56(X0))) [choice axiom]
1161. ! [X1] : (? [X4] : ::(X1,treg,X4) => ::(X1,treg,sK57(X1))) [choice axiom]
1162. ! [X0,X1] : (((::(X0,treg,sK56(X0)) | ! [X3] : ~::(X1,treg,X3)) & (::(X1,treg,sK57(X1)) | ! [X5] : ~::(X0,treg,X5))) | ~oP(X0,X1)) [skolemisation 1159,1161,1160]
1368. ! [X0] : (! [X1] : (::(X0,cnt,X1) | ~EX(X0,X1)) | ! [X2] : ~::(X0,cnt,X2)) [rectify 1022]
1386. ! [X0,X1] : ((::(X0,X1) | ! [X2] : ~::(X0,X1,X2)) & (? [X2] : ::(X0,X1,X2) | ~::(X0,X1))) [nnf transformation 704]
1387. ! [X0,X1] : ((::(X0,X1) | ! [X2] : ~::(X0,X1,X2)) & (? [X3] : ::(X0,X1,X3) | ~::(X0,X1))) [rectify 1386]
1388. ! [X1,X0] : (? [X3] : ::(X0,X1,X3) => ::(X0,X1,sK159(X0,X1))) [choice axiom]
1389. ! [X0,X1] : ((::(X0,X1) | ! [X2] : ~::(X0,X1,X2)) & (::(X0,X1,sK159(X0,X1)) | ~::(X0,X1))) [skolemisation 1387,1388]
1392. ! [X0] : ((PD(X0) | (~::(X0,pbnd) & ~::(X0,proc))) & ((::(X0,pbnd) | ::(X0,proc)) | ~PD(X0))) [nnf transformation 710]
1393. ! [X0] : ((PD(X0) | (~::(X0,pbnd) & ~::(X0,proc))) & (::(X0,pbnd) | ::(X0,proc) | ~PD(X0))) [flattening 1392]
1394. ! [X0] : ((T(X0) | ~::(X0,treg)) & (::(X0,treg) | ~T(X0))) [nnf transformation 717]
1395. ! [X0] : ((S(X0) | ~::(X0,sreg)) & (::(X0,sreg) | ~S(X0))) [nnf transformation 719]
1396. ! [X1,X0] : ((sP9(X1,X0) | (~oP(X0,X1) | ((~T(X1) | ~T(X0)) & (~PD(X1) | ~PD(X0))))) & ((oP(X0,X1) & ((T(X1) & T(X0)) | (PD(X1) & PD(X0)))) | ~sP9(X1,X0))) [nnf transformation 1059]
1397. ! [X1,X0] : ((sP9(X1,X0) | ~oP(X0,X1) | ((~T(X1) | ~T(X0)) & (~PD(X1) | ~PD(X0)))) & ((oP(X0,X1) & ((T(X1) & T(X0)) | (PD(X1) & PD(X0)))) | ~sP9(X1,X0))) [flattening 1396]
1398. ! [X0,X1] : ((sP9(X0,X1) | ~oP(X1,X0) | ((~T(X0) | ~T(X1)) & (~PD(X0) | ~PD(X1)))) & ((oP(X1,X0) & ((T(X0) & T(X1)) | (PD(X0) & PD(X1)))) | ~sP9(X0,X1))) [rectify 1397]
1399. ! [X3,X0,X1] : ((P(X0,X1) | ((? [X2] : (~cP(X2,X1,X3) & EX(X2,X3)) | ~S(X1) | ~S(X0)) & ~sP9(X1,X0))) & (((! [X2] : (cP(X2,X1,X3) | ~EX(X2,X3)) & S(X1) & S(X0)) | sP9(X1,X0)) | ~P(X0,X1))) [nnf transformation 1060]
1400. ! [X3,X0,X1] : ((P(X0,X1) | ((? [X2] : (~cP(X2,X1,X3) & EX(X2,X3)) | ~S(X1) | ~S(X0)) & ~sP9(X1,X0))) & ((! [X2] : (cP(X2,X1,X3) | ~EX(X2,X3)) & S(X1) & S(X0)) | sP9(X1,X0) | ~P(X0,X1))) [flattening 1399]
1401. ! [X0,X1,X2] : ((P(X1,X2) | ((? [X3] : (~cP(X3,X2,X0) & EX(X3,X0)) | ~S(X2) | ~S(X1)) & ~sP9(X2,X1))) & ((! [X4] : (cP(X4,X2,X0) | ~EX(X4,X0)) & S(X2) & S(X1)) | sP9(X2,X1) | ~P(X1,X2))) [rectify 1400]
1402. ! [X2,X0] : (? [X3] : (~cP(X3,X2,X0) & EX(X3,X0)) => (~cP(sK160(X0,X2),X2,X0) & EX(sK160(X0,X2),X0))) [choice axiom]
1403. ! [X0,X1,X2] : ((P(X1,X2) | (((~cP(sK160(X0,X2),X2,X0) & EX(sK160(X0,X2),X0)) | ~S(X2) | ~S(X1)) & ~sP9(X2,X1))) & ((! [X4] : (cP(X4,X2,X0) | ~EX(X4,X0)) & S(X2) & S(X1)) | sP9(X2,X1) | ~P(X1,X2))) [skolemisation 1401,1402]
1404. ? [X0,X1,X2] : (~P(X0,X2) & P(X1,X2) & P(X0,X1)) => (~P(sK161,sK163) & P(sK162,sK163) & P(sK161,sK162)) [choice axiom]
1405. ~P(sK161,sK163) & P(sK162,sK163) & P(sK161,sK162) [skolemisation 1043,1404]
1453. ~EX(X0,X1) | PAR(X1) [cnf transformation 795]
1456. ~::(X0,X1,X2) | UNI(X1) [cnf transformation 796]
1457. ~::(X0,X1,X2) | ::(X2,treg,X2) [cnf transformation 796]
1466. EX(X0,X1) | ~::(X1,treg,X1) | ~::(X0,X3,X1) | ~UNI(X3) [cnf transformation 1091]
1511. ~::(X0,imen,X1) | ::(X0,idcnt,X1) [cnf transformation 1120]
1515. ~::(X0,idcnt,X8) | ::(X0,cnt,sK38(X0)) [cnf transformation 1127]
1541. ~oP(X1,X2) | oP(X0,X2) | ~oP(X0,X1) [cnf transformation 837]
1544. ~oP(X0,X1) | ::(X1,occ,sK47(X1)) [cnf transformation 1145]
1552. ~::(X0,treg,X5) | ::(X1,treg,sK57(X1)) | ~oP(X0,X1) [cnf transformation 1162]
1553. ~::(X1,treg,X3) | ::(X0,treg,sK56(X0)) | ~oP(X0,X1) [cnf transformation 1162]
1837. UNI(idcnt) [cnf transformation 191]
1847. ~UNI(X0) | ~PAR(X0) [cnf transformation 990]
1849. ~::(X1,proc,X0) | ::(X1,occ,X0) [cnf transformation 992]
1853. ~::(X1,treg,X0) | ::(X1,occ,X0) [cnf transformation 996]
1854. ~::(X1,pbnd,X0) | ::(X1,occ,X0) [cnf transformation 997]
1856. ~::(X0,occ,X1) | ~::(X0,cnt,X1) [cnf transformation 999]
1860. ~::(X1,sreg,X0) | ::(X1,imen,X0) [cnf transformation 1003]
1879. ~::(X0,cnt,X2) | ~EX(X0,X1) | ::(X0,cnt,X1) [cnf transformation 1368]
```

```
2296. ::(X0,X1,sK159(X0,X1)) | ~::(X0,X1) [cnf transformation 1389]
2297. ~::(X0,X1,X2) | ::(X0,X1) [cnf transformation 1389]
2301. ::(X0,pbnd) | ::(X0,proc) | ~PD(X0) [cnf transformation 1393]
2304. ::(X0,treg) | ~T(X0) [cnf transformation 1394]
2305. ~::(X0,treg) | T(X0) [cnf transformation 1394]
2306. ::(X0,sreg) | ~S(X0) [cnf transformation 1395]
2308. ~sP9(X0,X1) | PD(X1) | T(X1) [cnf transformation 1398]
2309. ~sP9(X0,X1) | PD(X0) | T(X1) [cnf transformation 1398]
2310. ~sP9(X0,X1) | PD(X1) | T(X0) [cnf transformation 1398]
2312. ~sP9(X0,X1) | oP(X1,X0) [cnf transformation 1398]
2313. ~oP(X1,X0) | sP9(X0,X1) | ~PD(X0) | ~PD(X1) [cnf transformation 1398]
2314. ~oP(X1,X0) | sP9(X0,X1) | ~T(X0) | ~T(X1) [cnf transformation 1398]
2315. ~P(X1,X2) | sP9(X2,X1) | S(X1) [cnf transformation 1403]
2316. ~P(X1,X2) | sP9(X2,X1) | S(X2) [cnf transformation 1403]
2318. ~sP9(X2,X1) | P(X1,X2) [cnf transformation 1403]
2319. EX(sK160(X0,X2),X0) | P(X1,X2) | ~S(X2) | ~S(X1) [cnf transformation 1403]
2321. P(sK161,sK162) [cnf transformation 1405]
2322. P(sK162,sK163) [cnf transformation 1405]
2323. ~P(sK161,sK163) [cnf transformation 1405]
2362. EX(X0,X1) | ~::(X1,treg,X1) | ~::(X0,X3,X1) [subsumption resolution 1466,1456]
2363. ~::(X0,X3,X1) | EX(X0,X1) [subsumption resolution 2362,1457]
2372. ~PAR(idcnt) [resolution 1847,1837]
2823. 1 <=> S(sK161) [avatar definition]
2827. 2 <=> sP9(sK162,sK161) [avatar definition]
2829. sP9(sK162,sK161) <- (2) [avatar component clause 2827]
2832. 3 <=> S(sK162) [avatar definition]
2834. S(sK162) <- (3) [avatar component clause 2832]
2836. 4 <=> sP9(sK163,sK162) [avatar definition]
2838. sP9(sK163,sK162) <- (4) [avatar component clause 2836]
2840. sP9(sK162,sK161) | S(sK162) [resolution 2316,2321]
2841. sP9(sK163,sK162) | S(sK163) [resolution 2316,2322]
2843. 5 <=> S(sK163) [avatar definition]
2845. S(sK163) <- (5) [avatar component clause 2843]
2846. 5 | 4 [avatar split clause 2841,2836,2843]
2848. PD(sK161) | T(sK162) <- (2) [resolution 2829,2310]
2854. 6 <=> T(sK162) [avatar definition]
2858. 7 <=> PD(sK162) [avatar definition]
2863. 8 <=> PD(sK161) [avatar definition]
2866. 6 | 8 | ~2 [avatar split clause 2848,2827,2863,2854]
2868. 9 <=> T(sK161) [avatar definition]
2870. T(sK161) <- (9) [avatar component clause 2868]
2887. 10 <=> T(sK163) [avatar definition]
2891. 11 <=> PD(sK163) [avatar definition]
3197. EX(X2,sK159(X2,X3)) | ~::(X2,X3) [resolution 2296,2363]
3201. ::(X10,cnt,sK38(X10)) | ~::(X10,idcnt) [resolution 2296,1515]
3208. ::(X18,imen,sK159(X18,sreg)) | ~::(X18,sreg) [resolution 2296,1860]
3215. ::(X25,occ,sK159(X25,treg)) | ~::(X25,treg) [resolution 2296,1853]
3226. ::(X36,idcnt,sK159(X36,imen)) | ~::(X36,imen) [resolution 2296,1511]
3246. ::(X56,occ,sK159(X56,proc)) | ~::(X56,proc) [resolution 2296,1849]
3253. ~::(X63,cnt,sK159(X63,occ)) | ~::(X63,occ) [resolution 2296,1856]
3258. ::(X68,occ,sK159(X68,pbnd)) | ~::(X68,pbnd) [resolution 2296,1854]
3923. ::(X4,cnt,X5) | ~EX(X4,X5) | ~::(X4,cnt) [resolution 1879,2296]
4071. ~::(X4,treg) | ~oP(X4,X3) | ::(X3,treg,sK57(X3)) [resolution 1552,2296]
4076. ~::(X4,treg) | ~oP(X3,X4) | ::(X3,treg,sK56(X3)) [resolution 1553,2296]
4157. P(X3,X4) | ~S(X4) | ~S(X3) | PAR(X5) [resolution 2319,1453]
4166. 45 <=> ! [X1,X0] : (P(X0,X1) | ~S(X0) | ~S(X1)) [avatar definition]
4167. P(X0,X1) | ~S(X0) | ~S(X1) <- (45) [avatar component clause 4166]
4170. 46 <=> ! [X5] : PAR(X5) [avatar definition]
4171. PAR(X5) <- (46) [avatar component clause 4170]
4172. 46 | 45 [avatar split clause 4157,4166,4170]
4283. sP9(sK162,sK161) | S(sK161) [resolution 2321,2315]
4285. sP9(sK163,sK162) | S(sK162) [resolution 2322,2315]
4336. oP(sK161,sK162) <- (2) [resolution 2829,2312]
4342. ::(sK162,occ,sK47(sK162)) <- (2) [resolution 4336,1544]
4517. ~::(sK162,cnt,sK47(sK162)) <- (2) [resolution 4342,1856]
4522. EX(sK162,sK47(sK162)) <- (2) [resolution 4342,2363]
6717. ~S(sK161) | ~S(sK163) <- (45) [resolution 4167,2323]
6953. ~::(X7,idcnt) | ::(X7,cnt) [resolution 3201,2297]
7698. ~::(X8,sreg) | ::(X8,imen) [resolution 3208,2297]
7779. ~::(X19,treg) | ::(X19,occ) [resolution 3215,2297]
7840. ~::(X15,imen) | ::(X15,idcnt) [resolution 3226,2297]
7908. ~::(X19,proc) | ::(X19,occ) [resolution 3246,2297]
8025. ~::(X19,pbnd) | ::(X19,occ) [resolution 3258,2297]
9632. ~EX(X5,sK159(X5,occ)) | ~::(X5,cnt) | ~::(X5,occ) [resolution 3923,3253]
9649. ~EX(sK162,sK47(sK162)) | ~::(sK162,cnt) <- (2) [resolution 3923,4517]
9654. ~::(X5,occ) | ~::(X5,cnt) [subsumption resolution 9632,3197]
9669. ~::(sK162,cnt) <- (2) [subsumption resolution 9649,4522]
10565. ~oP(X0,X1) | ::(X1,treg,sK57(X1)) | ~T(X0) [resolution 4071,2304]
10582. ~oP(X0,X1) | ::(X0,treg,sK56(X0)) | ~T(X1) [resolution 4076,2304]
20011. ::(X0,imen) | ~S(X0) [resolution 7698,2306]
20192. ::(X0,occ) | ~T(X0) [resolution 7779,2304]
21050. ::(X0,idcnt) | ~S(X0) [resolution 7840,20011]
21188. ::(X0,cnt) | ~S(X0) [resolution 21050,6953]
21235. ~S(sK162) <- (2) [resolution 21188,9669]
21236. $false <- (2, 3) [subsumption resolution 21235,2834]
21237. ~2 | ~3 [avatar contradiction clause 21236]
21243. ~S(sK161) <- (5, 45) [subsumption resolution 6717,2845]
21244. ~1 | ~5 | ~45 [avatar split clause 21243,4166,2843,2823]
21251. 1 | 2 [avatar split clause 4283,2827,2823]
21253. 3 | 4 [avatar split clause 4285,2836,2832]
21254. 3 | 2 [avatar split clause 2840,2827,2832]
```

```
21276. $false <- (46) [resolution 4171,2372]
21384. ~46 [avatar contradiction clause 21276]
21394. PD(sK163) | T(sK162) <- (4) [resolution 2838,2309]
21395. PD(sK162) | T(sK162) <- (4) [resolution 2838,2308]
21397. oP(sK162,sK163) <- (4) [resolution 2838,2312]
21462. ~oP(X0,sK162) | oP(X0,sK163) <- (4) [resolution 21397,1541]
23723. ::(X0,occ) | ::(X0,proc) | ~PD(X0) [resolution 8025,2301]
23725. ::(X0,occ) | ~PD(X0) [subsumption resolution 23723,7908]
27248. ~::(X0,cnt) | ~PD(X0) [resolution 9654,23725]
27249. ~::(X1,cnt) | ~T(X1) [resolution 9654,20192]
28646. ~S(X0) | ~PD(X0) [resolution 27248,21188]
28695. ~S(X0) | ~T(X0) [resolution 27249,21188]
42348. 1081 <=> ::(sK161,treg) [avatar definition]
42349. ::(sK161,treg) <- (1081) [avatar component clause 42348]
42484. ~T(sK162) <- (3) [resolution 2834,28695]
42485. ~PD(sK162) <- (3) [resolution 2834,28646]
42487. ~6 | ~3 [avatar split clause 42484,2832,2854]
42490. ~7 | ~3 [avatar split clause 42485,2832,2858]
42491. 6 | 7 | ~4 [avatar split clause 21395,2836,2858,2854]
42498. oP(sK161,sK162) <- (2) [resolution 2829,2312]
53722. oP(sK161,sK163) <- (2, 4) [resolution 21462,42498]
53789. sP9(sK163,sK161) | ~T(sK163) | ~T(sK161) <- (2, 4) [resolution 53722,2314]
54174. sP9(sK163,sK161) | ~PD(sK163) | ~PD(sK161) <- (2, 4) [resolution 53722,2313]
54257. ::(sK163,treg,sK57(sK163)) | ~T(sK162) <- (4) [resolution 10565,21397]
54404. ::(sK161,treg,sK56(sK161)) | ~T(sK162) <- (2) [resolution 10582,42498]
54420. 1348 <=> ::(sK161,treg,sK56(sK161)) [avatar definition]
54422. ::(sK161,treg,sK56(sK161)) <- (1348) [avatar component clause 54420]
54783. ~6 | 1348 | ~2 [avatar split clause 54404,2827,54420,2854]
54795. 1351 <=> ::(sK163,treg,sK57(sK163)) [avatar definition]
54797. ::(sK163,treg,sK57(sK163)) <- (1351) [avatar component clause 54795]
54798. ~6 | 1351 | ~4 [avatar split clause 54257,2836,54795,2854]
54801. 1352 <=> sP9(sK163,sK161) [avatar definition]
54803. sP9(sK163,sK161) <- (1352) [avatar component clause 54801]
54809. 6 | 11 | ~4 [avatar split clause 21394,2836,2891,2854]
56114. ::(sK163,treg) <- (1351) [resolution 54797,2297]
56122. T(sK163) <- (1351) [resolution 56114,2305]
57780. ::(sK161,treg) <- (1348) [resolution 54422,2297]
57785. 1081 | ~1348 [avatar split clause 57780,54420,42348]
57828. T(sK161) <- (1081) [resolution 42349,2305]
57841. 9 | ~1081 [avatar split clause 57828,42348,2868]
57857. P(sK161,sK163) <- (1352) [resolution 54803,2318]
57859. $false <- (1352) [subsumption resolution 57857,2323]
57860. ~1352 [avatar contradiction clause 57859]
57862. ~8 | ~11 | 1352 | ~2 | ~4 [avatar split clause 54174,2836,2827,54801,2891,2863]
57874. sP9(sK163,sK161) | ~T(sK163) <- (2, 4, 9) [subsumption resolution 53789,2870]
57875. ~10 | 1352 | ~2 | ~4 | ~9 [avatar split clause 57874,2868,2836,2827,54801,2887]
57893. 10 | ~1351 [avatar split clause 56122,54795,2887]
57909. $false [avatar sat refutation 2846,2866,4172,21237,21244,21251,21253,21254,21384,42487,42490,42491,54783,
54798,54809,57785,57841,57860,57862,57875,57893]
```

Proof of Theorem (t_{bd}8)

```
255. ! [X6,X7] : (INH(X6,X7) <=> (~::(X7,sreg) & ::(X7,idcnt) & ::(X6,sdcnt) & SDEP(X6,X7))) [input]
264. ! [X6] : (PED(X6) <=> (~::(X6,sreg) & ::(X6,idcnt))) [input]
267. ! [X6] : (PQ(X6) <=> ::(X6,sdcnt)) [input]
271. ! [X6] : (S(X6) <=> ::(X6,sreg)) [input]
279. ! [X6,X7] : (DQT(X6,X7) <=> INH(X6,X7)) [input]
291. ! [X6,X7] : (DQT(X6,X7) => (PED(X7) & PQ(X6))) [input]
292. ~! [X6,X7] : (DQT(X6,X7) => (PED(X7) & PQ(X6))) [negated conjecture 291]
702. ! [X0,X1] : (INH(X0,X1) <=> (~::(X1,sreg) & ::(X1,idcnt) & ::(X0,sdcnt) & SDEP(X0,X1))) [rectify 255]
712. ! [X0] : (PED(X0) <=> (~::(X0,sreg) & ::(X0,idcnt))) [rectify 264]
715. ! [X0] : (PQ(X0) <=> ::(X0,sdcnt)) [rectify 267]
719. ! [X0] : (S(X0) <=> ::(X0,sreg)) [rectify 271]
729. ! [X0,X1] : (DQT(X0,X1) <=> INH(X0,X1)) [rectify 279]
741. ~! [X0,X1] : (DQT(X0,X1) => (PED(X1) & PQ(X0))) [rectify 292]
742. ! [X0] : (::(X0,sdcnt) => PQ(X0)) [unused predicate definition removal 715]
743. ! [X0,X1] : (DQT(X0,X1) => INH(X0,X1)) [unused predicate definition removal 729]
744. ! [X0] : ((~::(X0,sreg) & ::(X0,idcnt)) => PED(X0)) [unused predicate definition removal 712]
1044. ! [X0] : (PED(X0) | (::(X0,sreg) | ~::(X0,idcnt))) [ennf transformation 744]
1045. ! [X0] : (PED(X0) | ::(X0,sreg) | ~::(X0,idcnt)) [flattening 1044]
1046. ! [X0] : (PQ(X0) | ~::(X0,sdcnt)) [ennf transformation 742]
1048. ! [X0,X1] : (INH(X0,X1) | ~DQT(X0,X1)) [ennf transformation 743]
1049. ? [X0,X1] : ((~PED(X1) | ~PQ(X0)) & DQT(X0,X1)) [ennf transformation 741]
1390. ! [X0,X1] : ((INH(X0,X1) | (::(X1,sreg) | ~::(X1,idcnt) | ~::(X0,sdcnt) | ~SDEP(X0,X1))) & ((~::(X1,sreg) &
       ::(X1,idcnt) & ::(X0,sdcnt) & SDEP(X0,X1)) | ~INH(X0,X1))) [nnf transformation 702]
1391. ! [X0,X1] : ((INH(X0,X1) | ::(X1,sreg) | ~::(X1,idcnt) | ~::(X0,sdcnt) | ~SDEP(X0,X1)) & ((~::(X1,sreg) &
       ::(X1,idcnt) & ::(X0,sdcnt) & SDEP(X0,X1)) | ~INH(X0,X1))) [flattening 1390]
1401. ! [X0] : ((S(X0) | ~::(X0,sreg)) & (::(X0,sreg) | ~S(X0))) [nnf transformation 719]
1410. ? [X0,X1] : ((~PED(X1) | ~PQ(X0)) & DQT(X0,X1)) => ((~PED(sK162) | ~PQ(sK161)) & DQT(sK161,sK162)) [choice
       axiom]
1411. (~PED(sK162) | ~PQ(sK161)) & DQT(sK161,sK162) [skolemisation 1049,1410]
2298. ~INH(X0,X1) | ::(X0,sdcnt) [cnf transformation 1391]
2299. ~INH(X0,X1) | ::(X1,idcnt) [cnf transformation 1391]
2300. ~::(X1,sreg) | ~INH(X0,X1) [cnf transformation 1391]
2310. ~::(X0,idcnt) | ::(X0,sreg) | PED(X0) [cnf transformation 1045]
2311. ~::(X0,sdcnt) | PQ(X0) [cnf transformation 1046]
2314. ::(X0,sreg) | ~S(X0) [cnf transformation 1401]
2315. ~::(X0,sreg) | S(X0) [cnf transformation 1401]
2329. ~DQT(X0,X1) | INH(X0,X1) [cnf transformation 1048]
2330. DQT(sK161,sK162) [cnf transformation 1411]
```

```
2331. ~PED(sK162) | ~PQ(sK161) [cnf transformation 1411]
2380. 1 <=> PQ(sK161) [avatar definition]
2384. 2 <=> PED(sK162) [avatar definition]
2386. ~PED(sK162) <- (~2) [avatar component clause 2384]
2387. ~1 | ~2 [avatar split clause 2331,2384,2380]
2421. ~INH(X0,X1) | ~S(X1) [resolution 2300,2314]
2422. INH(sK161,sK162) [resolution 2329,2330]
2424. ::(sK162,idcnt) [resolution 2422,2299]
2425. ::(sK161,sdcnt) [resolution 2422,2298]
2427. PQ(sK161) [resolution 2425,2311]
2430. 1 [avatar split clause 2427,2380]
2431. ~S(sK162) [resolution 2421,2422]
2852. ::(sK162,sreg) | PED(sK162) [resolution 2310,2424]
2881. ::(sK162,sreg) <- (~2) [subsumption resolution 2852,2386]
2883. S(sK162) <- (~2) [resolution 2881,2315]
2884. $false <- (~2) [subsumption resolution 2883,2431]
2885. 2 [avatar contradiction clause 2884]
2886. $false [avatar sat refutation 2387,2430,2885]
```

Proof of Theorem ($t_{bd}10$)

```
1 (all i all u all t (::(i,u,t) -> PAR(i) & UNI(u) & ::(t,treg,t))) # label("instance-of-domain-range_"). [
      assumption].
2 (all a all t ((exists u (UNI(u) & ::(a,u,t) & ::(t,treg,t))) <-> PAR(a) & ::(t,treg,t) & EX(a,t))) # label("
      instantiation-existence"). [assumption].
3 (all a all b (INH(a,b) <-> SDEP(a,b) & (exists t (::(a,sdcnt,t) & ::(b,idcnt,t) & -::(b,sreg,t))))) # label("
      inheres-in-definition"). [assumption].
4 (all s ((exists t ::(s,sdcnt,t)) <-> (exists c exists t (::(s,cnt,t) & ::(c,idcnt,t) & -::(c,sreg,t) & SDEP(s,c)
      )))) # label("definition-of-specifically-dependent-continuant"). [assumption].
5 (all x ((exists t ::(x,sdcnt,t)) -> (all t (EX(x,t) -> ::(x,sdcnt,t))))) # label("specifically-dependent-
      continuant-is-rigid"). [assumption].
6 INH(A,B) <-> SDEP(A,B) & ::(A,sdcnt) & ::(B,idcnt) & -::(B,sreg). [assumption].
7 ::(A,B) <-> (exists C ::(A,B,C)). [assumption].
8 PQ(A) <-> ::(A,sdcnt) & -::(A,rqlt). [assumption].
9 Q(A) <-> PQ(A). [assumption].
10 S(A) <-> ::(A,sreg). [assumption].
11 DQT(A,B) <-> INH(A,B). [assumption].
12 Q(A) -> (exists B DQT(A,B)). [goal].
13 -PQ(A) | ::(A,sdcnt). [clausify(8)].
14 -Q(A) | PQ(A). [clausify(9)].
15 Q(c3). [deny(12)].
16 -Q(A) | ::(A,sdcnt). [resolve(14,b,13,a)].
17 -::(A,B,C) | ::(C,treg,C) # label("instance-of-domain-range_"). [clausify(1)].
18 -UNI(A) | -::(B,A,C) | -::(C,treg,C) | EX(B,C) # label("instantiation-existence"). [clausify(2)].
19 INH(A,B) | -SDEP(A,B) | -::(A,sdcnt,C) | -::(B,idcnt,C) | ::(B,sreg,C) # label("inheres-in-definition"). [
      clausify(3)].
20 -::(A,sdcnt,B) | ::(A,cnt,f127(A)) # label("definition-of-specifically-dependent-continuant"). [clausify(4)].
21 -::(A,sdcnt,B) | ::(f126(A),idcnt,f127(A)) # label("definition-of-specifically-dependent-continuant"). [
      clausify(4)].
22 -::(A,sdcnt,B) | -::(f126(A),sreg,f127(A)) # label("definition-of-specifically-dependent-continuant"). [
      clausify(4)].
23 -::(A,sdcnt,B) | SDEP(A,f126(A)) # label("definition-of-specifically-dependent-continuant"). [clausify(4)].
24 UNI(cnt) # label("continuant-is-a-universal"). [assumption].
25 -::(A,sdcnt,B) | -EX(A,C) | ::(A,sdcnt,C) # label("specifically-dependent-continuant-is-rigid"). [clausify(5)
      ].
26 -INH(A,B) | -::(B,sreg). [clausify(6)].
27 INH(A,B) | -SDEP(A,B) | -::(A,sdcnt) | -::(B,idcnt) | ::(B,sreg). [clausify(6)].
28 -::(A,B) | ::(A,B,f143(A,B)). [clausify(7)].
29 ::(A,B) | -::(A,B,C). [clausify(7)].
30 -S(A) | ::(A,sreg). [clausify(10)].
31 S(A) | -::(A,sreg). [clausify(10)].
32 DQT(A,B) | -INH(A,B). [clausify(11)].
33 -DQT(c3,A). [deny(12)].
34 ::(c3,sdcnt). [resolve(16,a,15,a)].
35 ::(c3,sdcnt,f143(c3,sdcnt)). [resolve(34,a,28,a)].
36 -EX(c3,A) | ::(c3,sdcnt,A). [resolve(35,a,25,a)].
37 SDEP(c3,f126(c3)). [resolve(35,a,23,a)].
38 -::(f126(c3),sreg,f127(c3)). [resolve(35,a,22,a)].
39 ::(f126(c3),idcnt,f127(c3)). [resolve(35,a,21,a)].
40 ::(c3,cnt,f127(c3)). [resolve(35,a,20,a)].
41 INH(c3,f126(c3)) | -::(f126(c3),idcnt) | ::(f126(c3),sreg). [resolve(37,a,27,b),unit_del(b,34)].
42 -::(f127(c3),treg,f127(c3)) | EX(c3,f127(c3)). [resolve(40,a,18,b),unit_del(a,24)].
43 ::(f127(c3),treg,f127(c3)). [resolve(40,a,17,a)].
44 EX(c3,f127(c3)). [back_unit_del(42),unit_del(a,43)].
45 ::(f126(c3),idcnt). [resolve(39,a,29,b)].
46 INH(A,f126(c3)) | -SDEP(A,f126(c3)) | -::(A,sdcnt,f127(c3)). [resolve(39,a,19,d),unit_del(d,38)].
47 INH(c3,f126(c3)) | ::(f126(c3),sreg). [back_unit_del(41),unit_del(b,45)].
48 ::(c3,sdcnt,f127(c3)). [resolve(36,a,44,a)].
49 INH(c3,f126(c3)) | S(f126(c3)). [resolve(47,b,31,b)].
50 S(f126(c3)). [resolve(49,a,32,b),unit_del(b,33)].
51 ::(f126(c3),sreg). [resolve(50,a,30,a)].
52 -INH(A,f126(c3)). [resolve(51,a,26,b)].
53 -SDEP(A,f126(c3)) | -::(A,sdcnt,f127(c3)). [back_unit_del(46),unit_del(a,52)].
54 $F. [resolve(53,b,48,a),unit_del(a,37)].
```

Proof of Theorem (t_{bd}13)

```
26. ! [X2] : (PAR(X2) => ? [X1] : EX(X2,X1)) [input]
30. ! [X17] : ! [X1] : (EX(X17,X1) => (::(X1,treg,X1) & PAR(X1) & PAR(X17))) [input]
31. ! [X17] : ! [X16] : ! [X1] : (::(X17,X16,X1) => (::(X1,treg,X1) & UNI(X16) & PAR(X17))) [input]
148. ! [X0] : (? [X1] : ::(X0,occ,X1) => tmP(X0,X0)) [input]
163. ! [X0] : ! [X8] : (tmP(X0,X8) => (? [X1] : ::(X8,occ,X1) & ? [X1] : ::(X0,occ,X1))) [input]
257. ! [X6,X16] : (::(X6,X16) <=> ? [X1] : ::(X6,X16,X1)) [input]
262. ! [X6] : (PD(X6) <=> (::(X6,pbnd) | ::(X6,proc))) [input]
263. ! [X6] : (ED(X6) <=> (~::(X6,sdcnt) & ~::(X6,sreg) & ::(X6,cnt))) [input]
267. ! [X6] : (PQ(X6) <=> (~::(X6,rqlt) & ::(X6,sdcnt))) [input]
268. ! [X6] : (Q(X6) <=> PQ(X6)) [input]
269. ! [X6] : (T(X6) <=> ::(X6,treg)) [input]
275. ! [X6,X1] : (TLC(X6,X1) <=> (! [X16] : (EX(X6,X16) <=> tmP(X16,X1)) & (Q(X6) | ED(X6) | PD(X6)))) [input]
292. ! [X6,X1] : (TLC(X6,X1) => (T(X1) & (Q(X6) | PD(X6) | ED(X6)))) [input]
293. ~! [X6,X1] : (TLC(X6,X1) => (T(X1) & (Q(X6) | PD(X6) | ED(X6)))) [negated conjecture 292]
343. ! [X0] : (PAR(X0) => ? [X1] : EX(X0,X1)) [rectify 26]
350. ! [X0] : ! [X1] : (EX(X0,X1) => (::(X1,treg,X1) & PAR(X1) & PAR(X0))) [rectify 30]
351. ! [X0,X1] : (EX(X0,X1) => (::(X1,treg,X1) & PAR(X1) & PAR(X0))) [flattening 350]
352. ! [X0] : ! [X1] : ! [X2] : (::(X0,X1,X2) => (::(X2,treg,X2) & UNI(X1) & PAR(X0))) [rectify 31]
353. ! [X0,X1,X2] : (::(X0,X1,X2) => (::(X2,treg,X2) & UNI(X1) & PAR(X0))) [flattening 352]
599. ! [X0] : ! [X1] : (tmP(X0,X1) => (? [X2] : ::(X1,occ,X2) & ? [X3] : ::(X0,occ,X3))) [rectify 163]
600. ! [X0,X1] : (tmP(X0,X1) => (? [X2] : ::(X1,occ,X2) & ? [X3] : ::(X0,occ,X3))) [flattening 599]
705. ! [X0,X1] : (::(X0,X1) <=> ? [X2] : ::(X0,X1,X2)) [rectify 257]
711. ! [X0] : (PD(X0) <=> (::(X0,pbnd) | ::(X0,proc))) [rectify 262]
712. ! [X0] : (ED(X0) <=> (~::(X0,sdcnt) & ~::(X0,sreg) & ::(X0,cnt))) [rectify 263]
716. ! [X0] : (PQ(X0) <=> (~::(X0,rqlt) & ::(X0,sdcnt))) [rectify 267]
717. ! [X0] : (Q(X0) <=> PQ(X0)) [rectify 268]
718. ! [X0] : (T(X0) <=> ::(X0,treg)) [rectify 269]
724. ! [X0,X1] : (TLC(X0,X1) <=> (! [X2] : (EX(X0,X2) <=> tmP(X2,X1)) & (Q(X0) | ED(X0) | PD(X0)))) [rectify 275]
741. ~! [X0,X1] : (TLC(X0,X1) => (T(X1) & (Q(X0) | PD(X0) | ED(X0)))) [rectify 293]
742. ! [X0,X1] : (TLC(X0,X1) => (! [X2] : (EX(X0,X2) <=> tmP(X2,X1)) & (Q(X0) | ED(X0) | PD(X0)))) [unused
     predicate definition removal 724]
743. ! [X0] : (::(X0,treg) => T(X0)) [unused predicate definition removal 718]
791. ! [X0] : (? [X1] : EX(X0,X1) | ~PAR(X0)) [ennf transformation 343]
797. ! [X0,X1] : ((::(X1,treg,X1) & PAR(X1) & PAR(X0)) | ~EX(X0,X1)) [ennf transformation 351]
798. ! [X0,X1,X2] : ((::(X2,treg,X2) & UNI(X1) & PAR(X0)) | ~::(X0,X1,X2)) [ennf transformation 353]
959. ! [X0] : (tmP(X0,X0) | ! [X1] : ~::(X0,occ,X1)) [ennf transformation 148]
979. ! [X0,X1] : ((? [X2] : ::(X1,occ,X2) & ? [X3] : ::(X0,occ,X3)) | ~tmP(X0,X1)) [ennf transformation 600]
1043. ! [X0] : (T(X0) | ~::(X0,treg)) [ennf transformation 743]
1044. ! [X0,X1] : ((! [X2] : (EX(X0,X2) <=> tmP(X2,X1)) & (Q(X0) | ED(X0) | PD(X0))) | ~TLC(X0,X1)) [ennf
     transformation 742]
1045. ? [X0,X1] : ((~T(X1) | (~Q(X0) & ~PD(X0) & ~ED(X0))) & TLC(X0,X1)) [ennf transformation 741]
1078. ! [X0] : (? [X1] : EX(X0,X1) => EX(X0,sK13(X0))) [choice axiom]
1079. ! [X0] : (EX(X0,sK13(X0)) | ~PAR(X0)) [skolemisation 791,1078]
1339. ! [X1] : (? [X2] : ::(X1,occ,X2) => ::(X1,occ,sK146(X1))) [choice axiom]
1340. ! [X0] : (? [X3] : ::(X0,occ,X3) => ::(X0,occ,sK147(X0))) [choice axiom]
1341. ! [X0,X1] : ((::(X1,occ,sK146(X1)) & ::(X0,occ,sK147(X0))) | ~tmP(X0,X1)) [skolemisation 979,1340,1339]
1386. ! [X0,X1] : ((::(X0,X1) | ! [X2] : ~::(X0,X1,X2)) & (? [X2] : ::(X0,X1,X2) | ~::(X0,X1))) [nnf
     transformation 705]
1387. ! [X0,X1] : ((::(X0,X1) | ! [X2] : ~::(X0,X1,X2)) & (? [X3] : ::(X0,X1,X3) | ~::(X0,X1))) [rectify 1386]
1388. ! [X1,X0] : (? [X3] : ::(X0,X1,X3) => ::(X0,X1,sK158(X0,X1))) [choice axiom]
1389. ! [X0,X1] : ((::(X0,X1) | ! [X2] : ~::(X0,X1,X2)) & (::(X0,X1,sK158(X0,X1)) | ~::(X0,X1))) [skolemisation
     1387,1388]
1392. ! [X0] : ((PD(X0) | (~::(X0,pbnd) & ~::(X0,proc))) & ((::(X0,pbnd) | ::(X0,proc)) | ~PD(X0))) [nnf
     transformation 711]
1393. ! [X0] : ((PD(X0) | (~::(X0,pbnd) & ~::(X0,proc))) & (::(X0,pbnd) | ::(X0,proc) | ~PD(X0))) [flattening
     1392]
1394. ! [X0] : ((ED(X0) | (::(X0,sdcnt) | ::(X0,sreg) | ~::(X0,cnt))) & ((~::(X0,sdcnt) & ~::(X0,sreg) & ::(X0,cnt
     )) | ~ED(X0))) [nnf transformation 712]
1395. ! [X0] : ((ED(X0) | ::(X0,sdcnt) | ::(X0,sreg) | ~::(X0,cnt)) & ((~::(X0,sdcnt) & ~::(X0,sreg) & ::(X0,cnt))
     | ~ED(X0))) [flattening 1394]
1396. ! [X0] : ((PQ(X0) | (::(X0,rqlt) | ~::(X0,sdcnt))) & ((~::(X0,rqlt) & ::(X0,sdcnt)) | ~PQ(X0))) [nnf
     transformation 716]
1397. ! [X0] : ((PQ(X0) | ::(X0,rqlt) | ~::(X0,sdcnt)) & ((~::(X0,rqlt) & ::(X0,sdcnt)) | ~PQ(X0))) [flattening
     1396]
1398. ! [X0] : ((Q(X0) | ~PQ(X0)) & (PQ(X0) | ~Q(X0))) [nnf transformation 717]
1399. ! [X0,X1] : ((! [X2] : ((EX(X0,X2) | ~tmP(X2,X1)) & (tmP(X2,X1) | ~EX(X0,X2))) & (Q(X0) | ED(X0) | PD(X0)))
     | ~TLC(X0,X1)) [nnf transformation 1044]
1400. ? [X0,X1] : ((~T(X1) | (~Q(X0) & ~PD(X0) & ~ED(X0))) & TLC(X0,X1)) => ((~T(sK160) | (~Q(sK159) & ~PD(sK159)
     & ~ED(sK159))) & TLC(sK159,sK160)) [choice axiom]
1401. (~T(sK160) | (~Q(sK159) & ~PD(sK159) & ~ED(sK159))) & TLC(sK159,sK160) [skolemisation 1045,1400]
1444. EX(X0,sK13(X0)) | ~PAR(X0) [cnf transformation 1079]
1450. ~EX(X0,X1) | ::(X1,treg,X1) [cnf transformation 797]
1451. ~::(X0,X1,X2) | PAR(X0) [cnf transformation 798]
1772. ~::(X0,occ,X1) | tmP(X0,X0) [cnf transformation 959]
1794. ~tmP(X0,X1) | ::(X1,occ,sK146(X1)) [cnf transformation 1341]
2292. ::(X0,X1,sK158(X0,X1)) | ~::(X0,X1) [cnf transformation 1389]
2293. ~::(X0,X1,X2) | ::(X0,X1) [cnf transformation 1389]
2297. ::(X0,pbnd) | ::(X0,proc) | ~PD(X0) [cnf transformation 1393]
2300. ::(X0,cnt) | ~ED(X0) [cnf transformation 1395]
2304. ::(X0,sdcnt) | ~PQ(X0) [cnf transformation 1397]
2307. ~Q(X0) | PQ(X0) [cnf transformation 1398]
2309. ~::(X0,treg) | T(X0) [cnf transformation 1043]
2310. ~TLC(X0,X1) | ED(X0) | PD(X0) | Q(X0) [cnf transformation 1399]
2311. ~TLC(X0,X1) | ~EX(X0,X2) | tmP(X2,X1) [cnf transformation 1399]
2312. ~TLC(X0,X1) | ~tmP(X2,X1) | EX(X0,X2) [cnf transformation 1399]
2313. TLC(sK159,sK160) [cnf transformation 1401]
2314. ~T(sK160) | ~ED(sK159) [cnf transformation 1401]
2315. ~T(sK160) | ~PD(sK159) [cnf transformation 1401]
2316. ~T(sK160) | ~Q(sK159) [cnf transformation 1401]
```

```
2365. 1 <=> Q(sK159) [avatar definition]
2366. Q(sK159) <- (1) [avatar component clause 2365]
2369. 2 <=> T(sK160) [avatar definition]
2371. ~T(sK160) <- (~2) [avatar component clause 2369]
2372. ~1 | ~2 [avatar split clause 2316,2369,2365]
2374. 3 <=> PD(sK159) [avatar definition]
2375. PD(sK159) <- (3) [avatar component clause 2374]
2377. ~3 | ~2 [avatar split clause 2315,2369,2374]
2379. 4 <=> ED(sK159) [avatar definition]
2380. ED(sK159) <- (4) [avatar component clause 2379]
2382. ~4 | ~2 [avatar split clause 2314,2369,2379]
3166. ~::(X8,X9) | PAR(X8) [resolution 2292,1451]
3241. ED(sK159) | PD(sK159) | Q(sK159) [resolution 2310,2313]
3242. 1 | 3 | 4 [avatar split clause 3241,2379,2374,2365]
3243. ~EX(sK159,X0) | tmP(X0,sK160) [resolution 2311,2313]
3269. ~tmP(X0,sK160) | EX(sK159,X0) [resolution 2312,2313]
3270. ~ED(X0) | PAR(X0) [resolution 3166,2300]
3271. ~PQ(X1) | PAR(X1) [resolution 3166,2304]
3272. PAR(X2) | ::(X2,proc) | ~PD(X2) [resolution 3166,2297]
3295. ~PD(X2) | PAR(X2) [subsumption resolution 3272,3166]
3345. tmP(sK13(sK159),sK160) | ~PAR(sK159) [resolution 3243,1444]
3375. 24 <=> PAR(sK159) [avatar definition]
3377. ~PAR(sK159) <- (~24) [avatar component clause 3375]
3379. 25 <=> tmP(sK13(sK159),sK160) [avatar definition]
3381. tmP(sK13(sK159),sK160) <- (25) [avatar component clause 3379]
3382. ~24 | 25 [avatar split clause 3345,3379,3375]
3383. PAR(sK159) <- (4) [resolution 2380,3270]
3384. $false <- (4, ~24) [subsumption resolution 3383,3377]
3385. ~4 | 24 [avatar contradiction clause 3384]
3386. PQ(sK159) <- (1) [resolution 2366,2307]
3393. PAR(sK159) <- (1) [resolution 3386,3271]
3396. $false <- (1, ~24) [subsumption resolution 3393,3377]
3397. ~1 | 24 [avatar contradiction clause 3396]
3398. PAR(sK159) <- (3) [resolution 2375,3295]
3401. 24 | ~3 [avatar split clause 3398,2374,3375]
3502. ::(sK160,occ,sK146(sK160)) <- (25) [resolution 3381,1794]
3549. tmP(sK160,sK160) <- (25) [resolution 3502,1772]
3734. EX(sK159,sK160) <- (25) [resolution 3269,3549]
3743. ::(sK160,treg,sK160) <- (25) [resolution 3734,1450]
3771. ::(sK160,treg) <- (25) [resolution 3743,2293]
3792. T(sK160) <- (25) [resolution 3771,2309]
3795. $false <- (~2, 25) [subsumption resolution 3792,2371]
3796. 2 | ~25 [avatar contradiction clause 3795]
3797. $false [avatar sat refutation 2372,2377,2382,3242,3382,3385,3397,3401,3796]
```

Proof of Theorem (t_{bd}15)

```
26. ! [X2] : (PAR(X2) => ? [X1] : EX(X2,X1)) [input]
31. ! [X17] : ! [X16] : ! [X1] : (::(X17,X16,X1) => (::(X1,treg,X1) & UNI(X16) & PAR(X17))) [input]
148. ! [X0] : (? [X1] : ::(X0,occ,X1) => tmP(X0,X0)) [input]
149. ! [X0] : ! [X8] : ((tmP(X8,X0) & tmP(X0,X8)) => X0 = X8) [input]
163. ! [X0] : ! [X8] : (tmP(X0,X8) => (? [X1] : ::(X8,occ,X1) & ? [X1] : ::(X0,occ,X1))) [input]
257. ! [X6,X16] : (::(X6,X16) <=> ? [X1] : ::(X6,X16,X1)) [input]
262. ! [X6] : (PD(X6) <=> (::(X6,pbnd) | ::(X6,proc))) [input]
263. ! [X6] : (ED(X6) <=> (~::(X6,sdcnt) & ~::(X6,sreg) & ::(X6,cnt))) [input]
267. ! [X6] : (PQ(X6) <=> (~::(X6,rqlt) & ::(X6,sdcnt))) [input]
268. ! [X6] : (Q(X6) <=> PQ(X6)) [input]
275. ! [X6,X1] : (TLC(X6,X1) <=> (! [X16] : (EX(X6,X16) <=> tmP(X16,X1)) & (Q(X6) | ED(X6) | PD(X6)))) [input]
292. ! [X6,X1,X16] : ((TLC(X6,X16) & TLC(X6,X1)) => X1 = X16) [input]
293. ~! [X6,X1,X16] : ((TLC(X6,X16) & TLC(X6,X1)) => X1 = X16) [negated conjecture 292]
343. ! [X0] : (PAR(X0) => ? [X1] : EX(X0,X1)) [rectify 26]
352. ! [X0] : ! [X1] : ! [X2] : (::(X0,X1,X2) => (::(X2,treg,X2) & UNI(X1) & PAR(X0))) [rectify 31]
353. ! [X0,X1,X2] : (::(X0,X1,X2) => (::(X2,treg,X2) & UNI(X1) & PAR(X0))) [flattening 352]
575. ! [X0] : ! [X1] : ((tmP(X1,X0) & tmP(X0,X1)) => X0 = X1) [rectify 149]
576. ! [X0,X1] : ((tmP(X1,X0) & tmP(X0,X1)) => X0 = X1) [flattening 575]
599. ! [X0] : ! [X1] : (tmP(X0,X1) => (? [X2] : ::(X1,occ,X2) & ? [X3] : ::(X0,occ,X3))) [rectify 163]
600. ! [X0,X1] : (tmP(X0,X1) => (? [X2] : ::(X1,occ,X2) & ? [X3] : ::(X0,occ,X3))) [flattening 599]
705. ! [X0,X1] : (::(X0,X1) <=> ? [X2] : ::(X0,X1,X2)) [rectify 257]
711. ! [X0] : (PD(X0) <=> (::(X0,pbnd) | ::(X0,proc))) [rectify 262]
712. ! [X0] : (ED(X0) <=> (~::(X0,sdcnt) & ~::(X0,sreg) & ::(X0,cnt))) [rectify 263]
716. ! [X0] : (PQ(X0) <=> (~::(X0,rqlt) & ::(X0,sdcnt))) [rectify 267]
717. ! [X0] : (Q(X0) <=> PQ(X0)) [rectify 268]
724. ! [X0,X1] : (TLC(X0,X1) <=> (! [X2] : (EX(X0,X2) <=> tmP(X2,X1)) & (Q(X0) | ED(X0) | PD(X0)))) [rectify 275]
741. ~! [X0,X1,X2] : ((TLC(X0,X2) & TLC(X0,X1)) => X1 = X2) [rectify 293]
742. ! [X0,X1] : (TLC(X0,X1) => (! [X2] : (EX(X0,X2) <=> tmP(X2,X1)) & (Q(X0) | ED(X0) | PD(X0)))) [unused
     predicate definition removal 724]
743. ! [X0] : (Q(X0) => PQ(X0)) [unused predicate definition removal 717]
744. ! [X0] : (PQ(X0) => (~::(X0,rqlt) & ::(X0,sdcnt))) [unused predicate definition removal 716]
745. ! [X0] : (PD(X0) => (::(X0,pbnd) | ::(X0,proc))) [unused predicate definition removal 711]
746. ! [X0] : (ED(X0) => (~::(X0,sdcnt) & ~::(X0,sreg) & ::(X0,cnt))) [unused predicate definition removal 712]
794. ! [X0] : (? [X1] : EX(X0,X1) | ~PAR(X0)) [ennf transformation 343]
801. ! [X0,X1,X2] : ((::(X2,treg,X2) & UNI(X1) & PAR(X0)) | ~::(X0,X1,X2)) [ennf transformation 353]
962. ! [X0] : (tmP(X0,X0) | ! [X1] : ~::(X0,occ,X1)) [ennf transformation 148]
963. ! [X0,X1] : (X0 = X1 | (~tmP(X1,X0) | ~tmP(X0,X1))) [ennf transformation 576]
964. ! [X0,X1] : (X0 = X1 | ~tmP(X1,X0) | ~tmP(X0,X1)) [flattening 963]
982. ! [X0,X1] : ((? [X2] : ::(X1,occ,X2) & ? [X3] : ::(X0,occ,X3)) | ~tmP(X0,X1)) [ennf transformation 600]
1046. ! [X0] : ((::(X0,pbnd) | ::(X0,proc)) | ~PD(X0)) [ennf transformation 745]
1047. ! [X0] : (::(X0,pbnd) | ::(X0,proc) | ~PD(X0)) [flattening 1046]
1048. ! [X0] : ((~::(X0,sdcnt) & ~::(X0,sreg) & ::(X0,cnt)) | ~ED(X0)) [ennf transformation 746]
1049. ! [X0] : ((~::(X0,rqlt) & ::(X0,sdcnt)) | ~PQ(X0)) [ennf transformation 744]
```

```
1050. ! [X0] : (PQ(X0) | ~Q(X0)) [ennf transformation 743]
1051. ! [X0,X1] : ((! [X2] : (EX(X0,X2) <=> tmP(X2,X1)) & (Q(X0) | ED(X0) | PD(X0))) | ~TLC(X0,X1)) [ennf
      transformation 742]
1052. ? [X0,X1,X2] : (X1 != X2 & (TLC(X0,X2) & TLC(X0,X1))) [ennf transformation 741]
1053. ? [X0,X1,X2] : (X1 != X2 & TLC(X0,X2) & TLC(X0,X1)) [flattening 1052]
1086. ! [X0] : (? [X1] : EX(X0,X1) => EX(X0,sK13(X0))) [choice axiom]
1087. ! [X0] : (EX(X0,sK13(X0)) | ~PAR(X0)) [skolemisation 794,1086]
1347. ! [X1] : (? [X2] : ::(X1,occ,X2) => ::(X1,occ,sK146(X1))) [choice axiom]
1348. ! [X0] : (? [X3] : ::(X0,occ,X3) => ::(X0,occ,sK147(X0))) [choice axiom]
1349. ! [X0,X1] : ((::(X1,occ,sK146(X1)) & ::(X0,occ,sK147(X0))) | ~tmP(X0,X1)) [skolemisation 982,1348,1347]
1394. ! [X0,X1] : ((::(X0,X1) | ! [X2] : ~::(X0,X1,X2)) & (? [X2] : ::(X0,X1,X2) | ~::(X0,X1))) [nnf
      transformation 705]
1395. ! [X0,X1] : ((::(X0,X1) | ! [X2] : ~::(X0,X1,X2)) & (? [X3] : ::(X0,X1,X3) | ~::(X0,X1))) [rectify 1394]
1396. ! [X1,X0] : (? [X3] : ::(X0,X1,X3) => ::(X0,X1,sK158(X0,X1))) [choice axiom]
1397. ! [X0,X1] : ((::(X0,X1) | ! [X2] : ~::(X0,X1,X2)) & (::(X0,X1,sK158(X0,X1)) | ~::(X0,X1))) [skolemisation
      1395,1396]
1400. ! [X0,X1] : ((! [X2] : ((EX(X0,X2) | ~tmP(X2,X1)) & (tmP(X2,X1) | ~EX(X0,X2))) & (Q(X0) | ED(X0) | PD(X0)))
      | ~TLC(X0,X1)) [nnf transformation 1051]
1401. ? [X0,X1,X2] : (X1 != X2 & TLC(X0,X2) & TLC(X0,X1)) => (sK160 != sK161 & TLC(sK159,sK161) & TLC(sK159,sK160)
      ) [choice axiom]
1402. sK160 != sK161 & TLC(sK159,sK161) & TLC(sK159,sK160) [skolemisation 1053,1401]
1445. EX(X0,sK13(X0)) | ~PAR(X0) [cnf transformation 1087]
1452. ~::(X0,X1,X2) | PAR(X0) [cnf transformation 801]
1773. ~::(X0,occ,X1) | tmP(X0,X0) [cnf transformation 962]
1774. ~tmP(X1,X0) | X0 = X1 | ~tmP(X0,X1) [cnf transformation 964]
1795. ~tmP(X0,X1) | ::(X1,occ,sK146(X1)) [cnf transformation 1349]
2293. ::(X0,X1,sK158(X0,X1)) | ~::(X0,X1) [cnf transformation 1397]
2298. ::(X0,pbnd) | ::(X0,proc) | ~PD(X0) [cnf transformation 1047]
2299. ::(X0,cnt) | ~ED(X0) [cnf transformation 1048]
2302. ::(X0,sdcnt) | ~PQ(X0) [cnf transformation 1049]
2304. ~Q(X0) | PQ(X0) [cnf transformation 1050]
2305. ~TLC(X0,X1) | ED(X0) | PD(X0) | Q(X0) [cnf transformation 1400]
2306. ~TLC(X0,X1) | ~EX(X0,X2) | tmP(X2,X1) [cnf transformation 1400]
2307. ~TLC(X0,X1) | ~tmP(X2,X1) | EX(X0,X2) [cnf transformation 1400]
2308. TLC(sK159,sK160) [cnf transformation 1402]
2309. TLC(sK159,sK161) [cnf transformation 1402]
2310. sK160 != sK161 [cnf transformation 1402]
3143. ~::(X8,X9) | PAR(X8) [resolution 2293,1452]
3243. ED(sK159) | PD(sK159) | Q(sK159) [resolution 2305,2309]
3245. 15 <=> Q(sK159) [avatar definition]
3247. Q(sK159) <- (15) [avatar component clause 3245]
3249. 16 <=> PD(sK159) [avatar definition]
3251. PD(sK159) <- (16) [avatar component clause 3249]
3253. 17 <=> ED(sK159) [avatar definition]
3255. ED(sK159) <- (17) [avatar component clause 3253]
3257. 15 | 16 | 17 [avatar split clause 3243,3253,3249,3245]
3259. ~EX(sK159,X0) | tmP(X0,sK160) [resolution 2306,2308]
3260. ~EX(sK159,X1) | tmP(X1,sK161) [resolution 2306,2309]
3263. ~tmP(X0,sK160) | EX(sK159,X0) [resolution 2307,2308]
3264. ~tmP(X1,sK161) | EX(sK159,X1) [resolution 2307,2309]
3270. ~ED(X0) | PAR(X0) [resolution 3143,2299]
3271. ~PQ(X1) | PAR(X1) [resolution 3143,2302]
3272. PAR(X2) | ::(X2,proc) | ~PD(X2) [resolution 3143,2298]
3294. ~PD(X2) | PAR(X2) [subsumption resolution 3272,3143]
3325. tmP(sK13(sK159),sK160) | ~PAR(sK159) [resolution 3259,1445]
3357. 21 <=> PAR(sK159) [avatar definition]
3358. PAR(sK159) <- (21) [avatar component clause 3357]
3359. ~PAR(sK159) <- (~21) [avatar component clause 3357]
3361. 22 <=> tmP(sK13(sK159),sK160) [avatar definition]
3363. tmP(sK13(sK159),sK160) <- (22) [avatar component clause 3361]
3364. ~21 | 22 [avatar split clause 3325,3361,3357]
3375. PAR(sK159) <- (16) [resolution 3251,3294]
3376. $false <- (16, ~21) [subsumption resolution 3375,3359]
3377. ~16 | 21 [avatar contradiction clause 3376]
3386. PAR(sK159) <- (17) [resolution 3255,3270]
3387. $false <- (17, ~21) [subsumption resolution 3386,3359]
3388. ~17 | 21 [avatar contradiction clause 3387]
3392. PQ(sK159) <- (15) [resolution 3247,2304]
3403. PAR(sK159) <- (15) [resolution 3392,3271]
3407. 21 | ~15 [avatar split clause 3403,3245,3357]
4016. tmP(sK13(sK159),sK161) | ~PAR(sK159) [resolution 3260,1445]
4019. tmP(sK13(sK159),sK161) <- (21) [subsumption resolution 4016,3358]
4899. ::(sK160,occ,sK146(sK160)) <- (22) [resolution 3363,1795]
4910. ::(sK161,occ,sK146(sK161)) <- (21) [resolution 4019,1795]
6001. tmP(sK160,sK160) <- (22) [resolution 4899,1773]
6030. tmP(sK161,sK161) <- (21) [resolution 4910,1773]
6047. EX(sK159,sK161) <- (21) [resolution 6030,3264]
6083. tmP(sK161,sK160) <- (21) [resolution 6047,3259]
6092. EX(sK159,sK160) <- (22) [resolution 6001,3263]
6108. tmP(sK160,sK161) <- (22) [resolution 6092,3260]
6119. sK160 = sK161 | ~tmP(sK160,sK161) <- (21) [resolution 6083,1774]
6124. ~tmP(sK160,sK161) <- (21) [subsumption resolution 6119,2310]
6125. $false <- (21, 22) [subsumption resolution 6124,6108]
6126. ~21 | ~22 [avatar contradiction clause 6125]
6128. $false [avatar sat refutation 3257,3364,3377,3388,3407,6126]
```

Proof of Theorem (t_{bd}16)

```
26. ! [X2] : (PAR(X2) => ? [X1] : EX(X2,X1)) [input]
30. ! [X17] : ! [X1] : (EX(X17,X1) => (::(X1,treg,X1) & PAR(X1) & PAR(X17))) [input]
31. ! [X17] : ! [X16] : ! [X1] : (::(X17,X16,X1) => (::(X1,treg,X1) & UNI(X16) & PAR(X17))) [input]
34. ! [X0] : ! [X1] : (? [X16] : (::(X1,treg,X1) & ::(X0,X16,X1) & UNI(X16)) <=> (EX(X0,X1) & ::(X1,treg,X1) & PAR
    (X0))) [input]
53. ! [X17] : ! [X1] : (::(X17,imen,X1) <=> (~? [X11] : (cP(X11,X17,X1) & ::(X11,mten,X1)) & ::(X17,idcnt,X1))) [
    input]
54. ! [X28] : (? [X1] : ::(X28,idcnt,X1) <=> (~? [X29] : ? [X1] : (GDEP(X28,X29,X1) | SDEP(X28,X29)) & ? [X1] :
    ::(X28,cnt,X1))) [input]
63. ! [X25] : ! [X30] : (oP(X25,X30) => ! [X1] : (EX(X25,X1) => EX(X30,X1))) [input]
64. ! [X0] : ! [X8] : (oP(X0,X8) => (? [X1] : ::(X8,occ,X1) & ? [X1] : ::(X0,occ,X1))) [input]
142. ! [X5] : (? [X1] : ::(X5,sdcnt,X1) <=> ? [X9] : ? [X1] : (SDEP(X5,X9) & ~::(X9,sreg,X1) & ::(X9,idcnt,X1) &
    ::(X5,cnt,X1))) [input]
146. ! [X0] : ! [X8] : (tmP(X0,X8) => oP(X0,X8)) [input]
148. ! [X0] : (? [X1] : ::(X0,occ,X1) => tmP(X0,X0)) [input]
163. ! [X0] : ! [X8] : (tmP(X0,X8) => (? [X1] : ::(X8,occ,X1) & ? [X1] : ::(X0,occ,X1))) [input]
179. UNI(sreg) [input]
186. UNI(imen) [input]
191. UNI(idcnt) [input]
201. ~? [X6] : (PAR(X6) & UNI(X6)) [input]
203. ! [X1] : ! [X6] : (::(X6,proc,X1) => ::(X6,occ,X1)) [input]
208. ! [X1] : ! [X6] : (::(X6,pbnd,X1) => ::(X6,occ,X1)) [input]
210. ~? [X6] : ? [X1] : (::(X6,occ,X1) & ::(X6,cnt,X1)) [input]
214. ! [X1] : ! [X6] : (::(X6,sreg,X1) => ::(X6,imen,X1)) [input]
234. ! [X6] : (? [X1] : ::(X6,cnt,X1) => ! [X1] : (EX(X6,X1) => ::(X6,cnt,X1))) [input]
257. ! [X6,X16] : (::(X6,X16) <=> ? [X1] : ::(X6,X16,X1)) [input]
262. ! [X6] : (PD(X6) <=> (::(X6,pbnd) | ::(X6,proc))) [input]
263. ! [X6] : (ED(X6) <=> (~::(X6,sdcnt) & ~::(X6,sreg) & ::(X6,cnt))) [input]
267. ! [X6] : (PQ(X6) <=> (~::(X6,rqlt) & ::(X6,sdcnt))) [input]
268. ! [X6] : (Q(X6) <=> PQ(X6)) [input]
269. ! [X6] : (T(X6) <=> ::(X6,treg)) [input]
271. ! [X6] : (S(X6) <=> ::(X6,sreg)) [input]
272. ! [X6] : (PR(X6) <=> (? [X7] : (::(X7,X6) & Q(X7)) | S(X6))) [input]
275. ! [X6,X1] : (TLC(X6,X1) <=> (! [X16] : (EX(X6,X16) <=> tmP(X16,X1)) & (Q(X6) | ED(X6) | PD(X6)))) [input]
276. ! [X6,X7] : (P(X6,X7) <=> ((ISA(X6,X7) & ~S(X7) & ~S(X6) & PR(X7) & PR(X6)) | (? [X1] : cP(X6,X7,X1) & S(X7)
    & S(X6)) | (oP(X6,X7) & ((T(X7) & T(X6)) | (PD(X7) & PD(X6)))))) [input]
292. ! [X6,X7,X1,X16] : ((TLC(X7,X16) & TLC(X6,X1) & P(X6,X7)) => P(X1,X16)) [input]
293. ~! [X6,X7,X1,X16] : ((TLC(X7,X16) & TLC(X6,X1) & P(X6,X7)) => P(X1,X16)) [negated conjecture 292]
343. ! [X0] : (PAR(X0) => ? [X1] : EX(X0,X1)) [rectify 26]
350. ! [X0] : ! [X1] : (EX(X0,X1) => (::(X1,treg,X1) & PAR(X1) & PAR(X0))) [rectify 30]
351. ! [X0,X1] : (EX(X0,X1) => (::(X1,treg,X1) & PAR(X1) & PAR(X0))) [flattening 350]
352. ! [X0] : ! [X1] : ! [X2] : (::(X0,X1,X2) => (::(X2,treg,X2) & UNI(X1) & PAR(X0))) [rectify 31]
353. ! [X0,X1,X2] : (::(X0,X1,X2) => (::(X2,treg,X2) & UNI(X1) & PAR(X0))) [flattening 352]
357. ! [X0] : ! [X1] : (? [X2] : (::(X1,treg,X1) & ::(X0,X2,X1) & UNI(X2)) <=> (EX(X0,X1) & ::(X1,treg,X1) & PAR(
    X0))) [rectify 34]
358. ! [X0,X1] : (? [X2] : (::(X1,treg,X1) & ::(X0,X2,X1) & UNI(X2)) <=> (EX(X0,X1) & ::(X1,treg,X1) & PAR(X0))) [
    flattening 357]
393. ! [X0] : ! [X1] : (::(X0,imen,X1) <=> (~? [X2] : (cP(X2,X0,X1) & ::(X2,mten,X1)) & ::(X0,idcnt,X1))) [rectify
    53]
394. ! [X0,X1] : (::(X0,imen,X1) <=> (~? [X2] : (cP(X2,X0,X1) & ::(X2,mten,X1)) & ::(X0,idcnt,X1))) [flattening
    393]
395. ! [X0] : (? [X1] : ::(X0,idcnt,X1) <=> (~? [X2] : ? [X3] : (GDEP(X0,X2,X3) | SDEP(X0,X2)) & ? [X4] : ::(X0,
    cnt,X4))) [rectify 54]
396. ! [X0] : (? [X1] : ::(X0,idcnt,X1) <=> (~? [X2,X3] : (GDEP(X0,X2,X3) | SDEP(X0,X2)) & ? [X4] : ::(X0,cnt,X4))
    ) [flattening 395]
411. ! [X0] : ! [X1] : (oP(X0,X1) => ! [X2] : (EX(X0,X2) => EX(X1,X2))) [rectify 63]
412. ! [X0,X1] : (oP(X0,X1) => ! [X2] : (EX(X0,X2) => EX(X1,X2))) [flattening 411]
413. ! [X0] : ! [X1] : (oP(X0,X1) => (? [X2] : ::(X1,occ,X2) & ? [X3] : ::(X0,occ,X3))) [rectify 64]
414. ! [X0,X1] : (oP(X0,X1) => (? [X2] : ::(X1,occ,X2) & ? [X3] : ::(X0,occ,X3))) [flattening 413]
563. ! [X0] : (? [X1] : ::(X0,sdcnt,X1) <=> ? [X2] : ? [X3] : (SDEP(X0,X2) & ~::(X2,sreg,X3) & ::(X2,idcnt,X3) &
    ::(X0,cnt,X3))) [rectify 142]
564. ! [X0] : (? [X1] : ::(X0,sdcnt,X1) <=> ? [X2,X3] : (SDEP(X0,X2) & ~::(X2,sreg,X3) & ::(X2,idcnt,X3) & ::(X0,
    cnt,X3))) [flattening 563]
571. ! [X0] : ! [X1] : (tmP(X0,X1) => oP(X0,X1)) [rectify 146]
572. ! [X0,X1] : (tmP(X0,X1) => oP(X0,X1)) [flattening 571]
599. ! [X0] : ! [X1] : (tmP(X0,X1) => (? [X2] : ::(X1,occ,X2) & ? [X3] : ::(X0,occ,X3))) [rectify 163]
600. ! [X0,X1] : (tmP(X0,X1) => (? [X2] : ::(X1,occ,X2) & ? [X3] : ::(X0,occ,X3))) [flattening 599]
617. ~? [X0] : (PAR(X0) & UNI(X0)) [rectify 201]
620. ! [X0] : ! [X1] : (::(X1,proc,X0) => ::(X1,occ,X0)) [rectify 203]
621. ! [X0,X1] : (::(X1,proc,X0) => ::(X1,occ,X0)) [flattening 620]
630. ! [X0] : ! [X1] : (::(X1,pbnd,X0) => ::(X1,occ,X0)) [rectify 208]
631. ! [X0,X1] : (::(X1,pbnd,X0) => ::(X1,occ,X0)) [flattening 630]
634. ~? [X0] : ? [X1] : (::(X0,occ,X1) & ::(X0,cnt,X1)) [rectify 210]
635. ~? [X0,X1] : (::(X0,occ,X1) & ::(X0,cnt,X1)) [flattening 634]
642. ! [X0] : ! [X1] : (::(X1,sreg,X0) => ::(X1,imen,X0)) [rectify 214]
643. ! [X0,X1] : (::(X1,sreg,X0) => ::(X1,imen,X0)) [flattening 642]
679. ! [X0] : (? [X1] : ::(X0,cnt,X1) => ! [X2] : (EX(X0,X2) => ::(X0,cnt,X2))) [rectify 234]
705. ! [X0,X1] : (::(X0,X1) <=> ? [X2] : ::(X0,X1,X2)) [rectify 257]
711. ! [X0] : (PD(X0) <=> (::(X0,pbnd) | ::(X0,proc))) [rectify 262]
712. ! [X0] : (ED(X0) <=> (~::(X0,sdcnt) & ~::(X0,sreg) & ::(X0,cnt))) [rectify 263]
716. ! [X0] : (PQ(X0) <=> (~::(X0,rqlt) & ::(X0,sdcnt))) [rectify 267]
717. ! [X0] : (Q(X0) <=> PQ(X0)) [rectify 268]
718. ! [X0] : (T(X0) <=> ::(X0,treg)) [rectify 269]
720. ! [X0] : (S(X0) <=> ::(X0,sreg)) [rectify 271]
721. ! [X0] : (PR(X0) <=> (? [X1] : (::(X1,X0) & Q(X1)) | S(X0))) [rectify 272]
724. ! [X0,X1] : (TLC(X0,X1) <=> (! [X2] : (EX(X0,X2) <=> tmP(X2,X1)) & (Q(X0) | ED(X0) | PD(X0)))) [rectify 275]
725. ! [X0,X1] : (P(X0,X1) <=> ((ISA(X0,X1) & ~S(X1) & ~S(X0) & PR(X1) & PR(X0)) | (? [X2] : cP(X0,X1,X2) & S(X1)
    & S(X0)) | (oP(X0,X1) & ((T(X1) & T(X0)) | (PD(X1) & PD(X0)))))) [rectify 276]
741. ~! [X0,X1,X2,X3] : ((TLC(X1,X3) & TLC(X0,X2) & P(X0,X1)) => P(X2,X3)) [rectify 293]
```

```
742. ! [X0,X1] : (TLC(X0,X1) => (! [X2] : (EX(X0,X2) <=> tmP(X2,X1)) & (Q(X0) | ED(X0) | PD(X0)))) [unused
     predicate definition removal 724]
743. ! [X0] : (ED(X0) => (~::(X0,sdcnt) & ~::(X0,sreg) & ::(X0,cnt))) [unused predicate definition removal 712]
791. ! [X0] : (? [X1] : EX(X0,X1) | ~PAR(X0)) [ennf transformation 343]
797. ! [X0,X1] : ((::(X1,treg,X1) & PAR(X1) & PAR(X0)) | ~EX(X0,X1)) [ennf transformation 351]
798. ! [X0,X1,X2] : ((::(X2,treg,X2) & UNI(X1) & PAR(X0)) | ~::(X0,X1,X2)) [ennf transformation 353]
824. ! [X0,X1] : (::(X0,imen,X1) <=> (! [X2] : (~cP(X2,X0,X1) | ~::(X2,mten,X1)) & ::(X0,idcnt,X1))) [ennf
     transformation 394]
825. ! [X0] : (? [X1] : ::(X0,idcnt,X1) <=> (! [X2,X3] : (~GDEP(X0,X2,X3) & ~SDEP(X0,X2)) & ? [X4] : ::(X0,cnt,X4)
     )) [ennf transformation 396]
840. ! [X0,X1] : (! [X2] : (EX(X1,X2) | ~EX(X0,X2)) | ~oP(X0,X1)) [ennf transformation 412]
841. ! [X0,X1] : ((? [X2] : ::(X1,occ,X2) & ? [X3] : ::(X0,occ,X3)) | ~oP(X0,X1)) [ennf transformation 414]
957. ! [X0,X1] : (oP(X0,X1) | ~tmP(X0,X1)) [ennf transformation 572]
959. ! [X0] : (tmP(X0,X0) | ! [X1] : ~::(X0,occ,X1)) [ennf transformation 148]
979. ! [X0,X1] : ((? [X2] : ::(X1,occ,X2) & ? [X3] : ::(X0,occ,X3)) | ~tmP(X0,X1)) [ennf transformation 600]
992. ! [X0] : (~PAR(X0) | ~UNI(X0)) [ennf transformation 617]
994. ! [X0,X1] : (::(X1,occ,X0) | ~::(X1,proc,X0)) [ennf transformation 621]
999. ! [X0,X1] : (::(X1,occ,X0) | ~::(X1,pbnd,X0)) [ennf transformation 631]
1001. ! [X0,X1] : (~::(X0,occ,X1) | ~::(X0,cnt,X1)) [ennf transformation 635]
1005. ! [X0,X1] : (::(X1,imen,X0) | ~::(X1,sreg,X0)) [ennf transformation 643]
1024. ! [X0] : (! [X2] : (::(X0,cnt,X2) | ~EX(X0,X2)) | ! [X1] : ~::(X0,cnt,X1)) [ennf transformation 679]
1044. ! [X0] : ((~::(X0,sdcnt) & ~::(X0,sreg) & ::(X0,cnt)) | ~ED(X0)) [ennf transformation 743]
1045. ! [X0,X1] : ((! [X2] : (EX(X0,X2) <=> tmP(X2,X1)) & (Q(X0) | ED(X0) | PD(X0))) | ~TLC(X0,X1)) [ennf
     transformation 742]
1046. ? [X0,X1,X2,X3] : (~P(X2,X3) & (TLC(X1,X3) & TLC(X0,X2) & P(X0,X1))) [ennf transformation 741]
1047. ? [X0,X1,X2,X3] : (~P(X2,X3) & TLC(X1,X3) & TLC(X0,X2) & P(X0,X1)) [flattening 1046]
1063. ! [X1,X0] : (sP9(X1,X0) <=> (oP(X0,X1) & ((T(X1) & T(X0)) | (PD(X1) & PD(X0))))) [predicate definition
     introduction]
1064. ! [X1,X0] : (sP10(X1,X0) <=> (ISA(X0,X1) & ~S(X1) & ~S(X0) & PR(X1) & PR(X0))) [predicate definition
     introduction]
1065. ! [X0,X1] : (P(X0,X1) <=> (sP10(X1,X0) | (? [X2] : cP(X0,X1,X2) & S(X1) & S(X0)) | sP9(X1,X0))) [definition
     folding 725,1064,1063]
1083. ! [X0] : (? [X1] : EX(X0,X1) => EX(X0,sK15(X0))) [choice axiom]
1084. ! [X0] : (EX(X0,sK15(X0)) | ~PAR(X0)) [skolemisation 791,1083]
1092. ! [X0,X1] : ((? [X2] : (::(X1,treg,X1) & ::(X0,X2,X1) & UNI(X2)) | (~EX(X0,X1) | ~::(X1,treg,X1) | ~PAR(X0))
     ) & ((EX(X0,X1) & ::(X1,treg,X1) & PAR(X0)) | ! [X2] : (~::(X1,treg,X1) | ~::(X0,X2,X1) | ~UNI(X2)))) [nnf
     transformation 358]
1093. ! [X0,X1] : ((? [X2] : (::(X1,treg,X1) & ::(X0,X2,X1) & UNI(X2)) | ~EX(X0,X1) | ~::(X1,treg,X1) | ~PAR(X0))
     & ((EX(X0,X1) & ::(X1,treg,X1) & PAR(X0)) | ! [X2] : (~::(X1,treg,X1) | ~::(X0,X2,X1) | ~UNI(X2)))) [
     flattening 1092]
1094. ! [X0,X1] : ((? [X2] : (::(X1,treg,X1) & ::(X0,X2,X1) & UNI(X2)) | ~EX(X0,X1) | ~::(X1,treg,X1) | ~PAR(X0))
     & ((EX(X0,X1) & ::(X1,treg,X1) & PAR(X0)) | ! [X3] : (~::(X1,treg,X1) | ~::(X0,X3,X1) | ~UNI(X3)))) [rectify
     1093]
1095. ! [X1,X0] : (? [X2] : (::(X1,treg,X1) & ::(X0,X2,X1) & UNI(X2)) => (::(X1,treg,X1) & ::(X0,sK21(X0,X1),X1) &
      UNI(sK21(X0,X1)))) [choice axiom]
1096. ! [X0,X1] : (((::(X1,treg,X1) & ::(X0,sK21(X0,X1),X1) & UNI(sK21(X0,X1))) | ~EX(X0,X1) | ~::(X1,treg,X1) | ~
     PAR(X0)) & ((EX(X0,X1) & ::(X1,treg,X1) & PAR(X0)) | ! [X3] : (~::(X1,treg,X1) | ~::(X0,X3,X1) | ~UNI(X3))))
      [skolemisation 1094,1095]
1121. ! [X0,X1] : ((::(X0,imen,X1) | (? [X2] : (cP(X2,X0,X1) & ::(X2,mten,X1)) | ~::(X0,idcnt,X1))) & ((! [X2] :
     (~cP(X2,X0,X1) | ~::(X2,mten,X1)) & ::(X0,idcnt,X1)) | ~::(X0,imen,X1))) [nnf transformation 824]
1122. ! [X0,X1] : ((::(X0,imen,X1) | ? [X2] : (cP(X2,X0,X1) & ::(X2,mten,X1)) | ~::(X0,idcnt,X1)) & ((! [X2] : (~
     cP(X2,X0,X1) | ~::(X2,mten,X1)) & ::(X0,idcnt,X1)) | ~::(X0,imen,X1))) [flattening 1121]
1123. ! [X0,X1] : ((::(X0,imen,X1) | ? [X2] : (cP(X2,X0,X1) & ::(X2,mten,X1)) | ~::(X0,idcnt,X1)) & ((! [X3] : (~
     cP(X3,X0,X1) | ~::(X3,mten,X1)) & ::(X0,idcnt,X1)) | ~::(X0,imen,X1))) [rectify 1122]
1124. ! [X1,X0] : (? [X2] : (cP(X2,X0,X1) & ::(X2,mten,X1)) => (cP(sK35(X0,X1),X0,X1) & ::(sK35(X0,X1),mten,X1)))
     [choice axiom]
1125. ! [X0,X1] : ((::(X0,imen,X1) | (cP(sK35(X0,X1),X0,X1) & ::(sK35(X0,X1),mten,X1)) | ~::(X0,idcnt,X1)) & ((! [
     X3] : (~cP(X3,X0,X1) | ~::(X3,mten,X1)) & ::(X0,idcnt,X1)) | ~::(X0,imen,X1))) [skolemisation 1123,1124]
1126. ! [X0] : ((? [X1] : ::(X0,idcnt,X1) | (? [X2,X3] : (GDEP(X0,X2,X3) | SDEP(X0,X2)) | ! [X4] : ~::(X0,cnt,X4))
     ) & ((! [X2,X3] : (~GDEP(X0,X2,X3) & ~SDEP(X0,X2)) & ? [X4] : ::(X0,cnt,X4)) | ! [X1] : ~::(X0,idcnt,X1))) [
     nnf transformation 825]
1127. ! [X0] : ((? [X1] : ::(X0,idcnt,X1) | ? [X2,X3] : (GDEP(X0,X2,X3) | SDEP(X0,X2)) | ! [X4] : ~::(X0,cnt,X4))
     & ((! [X2,X3] : (~GDEP(X0,X2,X3) & ~SDEP(X0,X2)) & ? [X4] : ::(X0,cnt,X4)) | ! [X1] : ~::(X0,idcnt,X1))) [
     flattening 1126]
1128. ! [X0] : ((? [X1] : ::(X0,idcnt,X1) | ? [X2,X3] : (GDEP(X0,X2,X3) | SDEP(X0,X2)) | ! [X4] : ~::(X0,cnt,X4))
     & ((! [X5,X6] : (~GDEP(X0,X5,X6) & ~SDEP(X0,X5)) & ? [X7] : ::(X0,cnt,X7)) | ! [X8] : ~::(X0,idcnt,X8))) [
     rectify 1127]
1129. ! [X0] : (? [X1] : ::(X0,idcnt,X1) => ::(X0,idcnt,sK36(X0))) [choice axiom]
1130. ! [X0] : (? [X2,X3] : (GDEP(X0,X2,X3) | SDEP(X0,X2)) => (GDEP(X0,sK37(X0),sK38(X0)) | SDEP(X0,sK37(X0)))) [
     choice axiom]
1131. ! [X0] : (? [X7] : ::(X0,cnt,X7) => ::(X0,cnt,sK39(X0))) [choice axiom]
1132. ! [X0] : ((::(X0,idcnt,sK36(X0)) | (GDEP(X0,sK37(X0),sK38(X0)) | SDEP(X0,sK37(X0))) | ! [X4] : ~::(X0,cnt,X4
     )) & ((! [X5,X6] : (~GDEP(X0,X5,X6) & ~SDEP(X0,X5)) & ::(X0,cnt,sK39(X0))) | ! [X8] : ~::(X0,idcnt,X8))) [
     skolemisation 1128,1131,1130,1129]
1148. ! [X1] : (? [X2] : ::(X1,occ,X2) => ::(X1,occ,sK48(X1))) [choice axiom]
1149. ! [X0] : (? [X3] : ::(X0,occ,X3) => ::(X0,occ,sK49(X0))) [choice axiom]
1150. ! [X0,X1] : ((::(X1,occ,sK48(X1)) & ::(X0,occ,sK49(X0))) | ~oP(X0,X1)) [skolemisation 841,1149,1148]
1326. ! [X0] : ((? [X1] : ::(X0,sdcnt,X1) | ! [X2,X3] : (~SDEP(X0,X2) | ::(X2,sreg,X3) | ~::(X2,idcnt,X3) | ~::(X0
     ,cnt,X3))) & (? [X2,X3] : (SDEP(X0,X2) & ~::(X2,sreg,X3) & ::(X2,idcnt,X3) & ::(X0,cnt,X3)) | ! [X1] : ~::(
     X0,sdcnt,X1))) [nnf transformation 564]
1327. ! [X0] : ((? [X1] : ::(X0,sdcnt,X1) | ! [X2,X3] : (~SDEP(X0,X2) | ::(X2,sreg,X3) | ~::(X2,idcnt,X3) | ~::(X0
     ,cnt,X3))) & (? [X4,X5] : (SDEP(X0,X4) & ~::(X4,sreg,X5) & ::(X4,idcnt,X5) & ::(X0,cnt,X5)) | ! [X6] : ~::(
     X0,sdcnt,X6))) [rectify 1326]
1328. ! [X0] : (? [X1] : ::(X0,sdcnt,X1) => ::(X0,sdcnt,sK138(X0))) [choice axiom]
1329. ! [X0] : (? [X4,X5] : (SDEP(X0,X4) & ~::(X4,sreg,X5) & ::(X4,idcnt,X5) & ::(X0,cnt,X5)) => (SDEP(X0,sK139(X0
     )) & ~::(sK139(X0),sreg,sK140(X0)) & ::(sK139(X0),idcnt,sK140(X0)) & ::(X0,cnt,sK140(X0)))) [choice axiom]
1330. ! [X0] : ((::(X0,sdcnt,sK138(X0)) | ! [X2,X3] : (~SDEP(X0,X2) | ::(X2,sreg,X3) | ~::(X2,idcnt,X3) | ~::(X0,
     cnt,X3))) & ((SDEP(X0,sK139(X0)) & ~::(sK139(X0),sreg,sK140(X0)) & ::(sK139(X0),idcnt,sK140(X0)) & ::(X0,cnt
     ,sK140(X0))) | ! [X6] : ~::(X0,sdcnt,X6))) [skolemisation 1327,1329,1328]
1344. ! [X1] : (? [X2] : ::(X1,occ,X2) => ::(X1,occ,sK148(X1))) [choice axiom]
1345. ! [X0] : (? [X3] : ::(X0,occ,X3) => ::(X0,occ,sK149(X0))) [choice axiom]
```

```
1346. ! [X0,X1] : ((::(X1,occ,sK148(X1)) & ::(X0,occ,sK149(X0))) | ~tmP(X0,X1)) [skolemisation 979,1345,1344]
1373. ! [X0] : (! [X1] : (::(X0,cnt,X1) | ~EX(X0,X1)) | ! [X2] : ~::(X0,cnt,X2)) [rectify 1024]
1391. ! [X0,X1] : ((::(X0,X1) | ! [X2] : ~::(X0,X1,X2)) & (? [X2] : ::(X0,X1,X2) | ~::(X0,X1))) [nnf
      transformation 705]
1392. ! [X0,X1] : ((::(X0,X1) | ! [X2] : ~::(X0,X1,X2)) & (? [X3] : ::(X0,X1,X3) | ~::(X0,X1))) [rectify 1391]
1393. ! [X1,X0] : (? [X3] : ::(X0,X1,X3) => ::(X0,X1,sK160(X0,X1))) [choice axiom]
1394. ! [X0,X1] : ((::(X0,X1) | ! [X2] : ~::(X0,X1,X2)) & (::(X0,X1,sK160(X0,X1)) | ~::(X0,X1))) [skolemisation
      1392,1393]
1401. ! [X0] : ((PD(X0) | (~::(X0,pbnd) & ~::(X0,proc))) & ((::(X0,pbnd) | ::(X0,proc)) | ~PD(X0))) [nnf
      transformation 711]
1402. ! [X0] : ((PD(X0) | (~::(X0,pbnd) & ~::(X0,proc))) & (::(X0,pbnd) | ::(X0,proc) | ~PD(X0))) [flattening
      1401]
1403. ! [X0] : ((PQ(X0) | (::(X0,rqlt) | ~::(X0,sdcnt))) & ((~::(X0,rqlt) & ::(X0,sdcnt)) | ~PQ(X0))) [nnf
      transformation 716]
1404. ! [X0] : ((PQ(X0) | ::(X0,rqlt) | ~::(X0,sdcnt)) & ((~::(X0,rqlt) & ::(X0,sdcnt)) | ~PQ(X0))) [flattening
      1403]
1405. ! [X0] : ((Q(X0) | ~PQ(X0)) & (PQ(X0) | ~Q(X0))) [nnf transformation 717]
1406. ! [X0] : ((T(X0) | ~::(X0,treg)) & (::(X0,treg) | ~T(X0))) [nnf transformation 718]
1407. ! [X0] : ((S(X0) | ~::(X0,sreg)) & (::(X0,sreg) | ~S(X0))) [nnf transformation 720]
1408. ! [X0] : ((PR(X0) | (! [X1] : (~::(X1,X0) | ~Q(X1)) & ~S(X0))) & ((? [X1] : (::(X1,X0) & Q(X1)) | S(X0)) | ~
      PR(X0))) [nnf transformation 721]
1409. ! [X0] : ((PR(X0) | (! [X1] : (~::(X1,X0) | ~Q(X1)) & ~S(X0))) & (? [X1] : (::(X1,X0) & Q(X1)) | S(X0) | ~PR
      (X0))) [flattening 1408]
1410. ! [X0] : ((PR(X0) | (! [X1] : (~::(X1,X0) | ~Q(X1)) & ~S(X0))) & (? [X2] : (::(X2,X0) & Q(X2)) | S(X0) | ~PR
      (X0))) [rectify 1409]
1411. ! [X0] : (? [X2] : (::(X2,X0) & Q(X2)) => (::(sK163(X0),X0) & Q(sK163(X0)))) [choice axiom]
1412. ! [X0] : ((PR(X0) | (! [X1] : (~::(X1,X0) | ~Q(X1)) & ~S(X0))) & ((::(sK163(X0),X0) & Q(sK163(X0))) | S(X0)
      | ~PR(X0))) [skolemisation 1410,1411]
1413. ! [X0,X1] : ((! [X2] : ((EX(X0,X2) | ~tmP(X2,X1)) & (tmP(X2,X1) | ~EX(X0,X2))) & (Q(X0) | ED(X0) | PD(X0)))
      | ~TLC(X0,X1)) [nnf transformation 1045]
1414. ! [X1,X0] : ((sP10(X1,X0) | (~ISA(X0,X1) | S(X1) | S(X0) | ~PR(X1) | ~PR(X0))) & ((ISA(X0,X1) & ~S(X1) & ~S(
      X0) & PR(X1) & PR(X0)) | ~sP10(X1,X0))) [nnf transformation 1064]
1415. ! [X1,X0] : ((sP10(X1,X0) | ~ISA(X0,X1) | S(X1) | S(X0) | ~PR(X1) | ~PR(X0)) & ((ISA(X0,X1) & ~S(X1) & ~S(X0
      ) & PR(X1) & PR(X0)) | ~sP10(X1,X0))) [flattening 1414]
1416. ! [X0,X1] : ((sP10(X0,X1) | ~ISA(X1,X0) | S(X0) | S(X1) | ~PR(X0) | ~PR(X1)) & ((ISA(X1,X0) & ~S(X0) & ~S(X1
      ) & PR(X0) & PR(X1)) | ~sP10(X0,X1))) [rectify 1415]
1417. ! [X1,X0] : ((sP9(X1,X0) | (~oP(X0,X1) | ((~T(X1) | ~T(X0)) & (~PD(X1) | ~PD(X0))))) & ((oP(X0,X1) & ((T(X1)
      & T(X0)) | (PD(X1) & PD(X0)))) | ~sP9(X1,X0))) [nnf transformation 1063]
1418. ! [X1,X0] : ((sP9(X1,X0) | ~oP(X0,X1) | ((~T(X1) | ~T(X0)) & (~PD(X1) | ~PD(X0)))) & ((oP(X0,X1) & ((T(X1) &
      T(X0)) | (PD(X1) & PD(X0)))) | ~sP9(X1,X0))) [flattening 1417]
1419. ! [X0,X1] : ((sP9(X0,X1) | ~oP(X1,X0) | ((~T(X0) | ~T(X1)) & (~PD(X0) | ~PD(X1)))) & ((oP(X1,X0) & ((T(X0) &
      T(X1)) | (PD(X0) & PD(X1)))) | ~sP9(X0,X1))) [rectify 1418]
1420. ! [X0,X1] : ((P(X0,X1) | (~sP10(X1,X0) & (! [X2] : ~cP(X0,X1,X2) | ~S(X1) | ~S(X0)) & ~sP9(X1,X0))) & ((sP10
      (X1,X0) | (? [X2] : cP(X0,X1,X2) & S(X1) & S(X0)) | sP9(X1,X0)) | ~P(X0,X1))) [nnf transformation 1065]
1421. ! [X0,X1] : ((P(X0,X1) | (~sP10(X1,X0) & (! [X2] : ~cP(X0,X1,X2) | ~S(X1) | ~S(X0)) & ~sP9(X1,X0))) & (sP10(
      X1,X0) | (? [X2] : cP(X0,X1,X2) & S(X1) & S(X0)) | sP9(X1,X0) | ~P(X0,X1))) [flattening 1420]
1422. ! [X0,X1] : ((P(X0,X1) | (~sP10(X1,X0) & (! [X2] : ~cP(X0,X1,X2) | ~S(X1) | ~S(X0)) & ~sP9(X1,X0))) & (sP10(
      X1,X0) | (? [X3] : cP(X0,X1,X3) & S(X1) & S(X0)) | sP9(X1,X0) | ~P(X0,X1))) [rectify 1421]
1423. ! [X1,X0] : (? [X3] : cP(X0,X1,X3) => cP(X0,X1,sK164(X0,X1))) [choice axiom]
1424. ! [X0,X1] : ((P(X0,X1) | (~sP10(X1,X0) & (! [X2] : ~cP(X0,X1,X2) | ~S(X1) | ~S(X0)) & ~sP9(X1,X0))) & (sP10(
      X1,X0) | (cP(X0,X1,sK164(X0,X1)) & S(X1) & S(X0)) | sP9(X1,X0) | ~P(X0,X1))) [skolemisation 1422,1423]
1425. ? [X0,X1,X2,X3] : (~P(X2,X3) & TLC(X1,X3) & TLC(X0,X2) & P(X0,X1)) => (~P(sK167,sK168) & TLC(sK166,sK168) &
      TLC(sK165,sK167) & P(sK165,sK166)) [choice axiom]
1426. ~P(sK167,sK168) & TLC(sK166,sK168) & TLC(sK165,sK167) & P(sK165,sK166) [skolemisation 1047,1425]
1469. EX(X0,sK15(X0)) | ~PAR(X0) [cnf transformation 1084]
1475. ~EX(X0,X1) | ::(X1,treg,X1) [cnf transformation 797]
1476. ~::(X0,X1,X2) | PAR(X0) [cnf transformation 798]
1477. ~::(X0,X1,X2) | UNI(X1) [cnf transformation 798]
1478. ~::(X0,X1,X2) | ::(X2,treg,X2) [cnf transformation 798]
1487. EX(X0,X1) | ~::(X1,treg,X1) | ~::(X0,X3,X1) | ~UNI(X3) [cnf transformation 1096]
1532. ~::(X0,imen,X1) | ::(X0,idcnt,X1) [cnf transformation 1125]
1536. ~::(X0,idcnt,X8) | ::(X0,cnt,sK39(X0)) [cnf transformation 1132]
1537. ~::(X0,idcnt,X8) | ~SDEP(X0,X5) [cnf transformation 1132]
1563. ~oP(X0,X1) | ~EX(X0,X2) | EX(X1,X2) [cnf transformation 840]
1565. ~oP(X0,X1) | ::(X1,occ,sK48(X1)) [cnf transformation 1150]
1784. ~::(X0,sdcnt,X6) | SDEP(X0,sK139(X0)) [cnf transformation 1330]
1795. ~tmP(X0,X1) | oP(X0,X1) [cnf transformation 957]
1797. ~::(X0,occ,X1) | tmP(X0,X0) [cnf transformation 959]
1819. ~tmP(X0,X1) | ::(X1,occ,sK148(X1)) [cnf transformation 1346]
1846. UNI(sreg) [cnf transformation 179]
1853. UNI(imen) [cnf transformation 186]
1858. UNI(idcnt) [cnf transformation 191]
1868. ~UNI(X0) | ~PAR(X0) [cnf transformation 992]
1870. ~::(X1,proc,X0) | ::(X1,occ,X0) [cnf transformation 994]
1875. ~::(X1,pbnd,X0) | ::(X1,occ,X0) [cnf transformation 999]
1877. ~::(X0,occ,X1) | ~::(X0,cnt,X1) [cnf transformation 1001]
1881. ~::(X1,sreg,X0) | ::(X1,imen,X0) [cnf transformation 1005]
1900. ~::(X0,cnt,X2) | ~EX(X0,X1) | ::(X0,cnt,X1) [cnf transformation 1373]
2317. ::(X0,X1,sK160(X0,X1)) | ~::(X0,X1) [cnf transformation 1394]
2318. ~::(X0,X1,X2) | ::(X0,X1) [cnf transformation 1394]
2325. ::(X0,pbnd) | ::(X0,proc) | ~PD(X0) [cnf transformation 1402]
2328. ::(X0,cnt) | ~ED(X0) [cnf transformation 1044]
2329. ~::(X0,sreg) | ~ED(X0) [cnf transformation 1044]
2331. ::(X0,sdcnt) | ~PQ(X0) [cnf transformation 1404]
2334. ~Q(X0) | PQ(X0) [cnf transformation 1405]
2337. ~::(X0,treg) | T(X0) [cnf transformation 1406]
2338. ::(X0,sreg) | ~S(X0) [cnf transformation 1407]
2340. Q(sK163(X0)) | S(X0) | ~PR(X0) [cnf transformation 1412]
2341. ::(sK163(X0),X0) | S(X0) | ~PR(X0) [cnf transformation 1412]
2343. ~::(X1,X0) | PR(X0) | ~Q(X1) [cnf transformation 1412]
2344. ~TLC(X0,X1) | ED(X0) | PD(X0) | Q(X0) [cnf transformation 1413]
```

```
2345. ~TLC(X0,X1) | ~EX(X0,X2) | tmP(X2,X1) [cnf transformation 1413]
2346. ~TLC(X0,X1) | ~tmP(X2,X1) | EX(X0,X2) [cnf transformation 1413]
2347. ~sP10(X0,X1) | PR(X1) [cnf transformation 1416]
2349. ~sP10(X0,X1) | ~S(X1) [cnf transformation 1416]
2357. ~sP9(X0,X1) | oP(X1,X0) [cnf transformation 1419]
2359. ~oP(X1,X0) | sP9(X0,X1) | ~T(X0) | ~T(X1) [cnf transformation 1419]
2361. ~P(X0,X1) | S(X1) | sP9(X1,X0) | sP10(X1,X0) [cnf transformation 1424]
2363. ~sP9(X1,X0) | P(X0,X1) [cnf transformation 1424]
2366. P(sK165,sK166) [cnf transformation 1426]
2367. TLC(sK165,sK167) [cnf transformation 1426]
2368. TLC(sK166,sK168) [cnf transformation 1426]
2369. ~P(sK167,sK168) [cnf transformation 1426]
2408. EX(X0,X1) | ~::(X1,treg,X1) | ~::(X0,X3,X1) [subsumption resolution 1487,1477]
2409. ~::(X0,X3,X1) | EX(X0,X1) [subsumption resolution 2408,1478]
2418. ~PAR(idcnt) [resolution 1868,1858]
2420. ~PAR(sreg) [resolution 1868,1846]
2429. ~PAR(imen) [resolution 1868,1853]
2448. ~S(X0) | ~ED(X0) [resolution 2338,2329]
2454. PQ(sK163(X0)) | ~PR(X0) | S(X0) [resolution 2340,2334]
2455. PR(sreg) | ~Q(X0) | ~S(X0) [resolution 2343,2338]
2460. 1 <=> ! [X0] : (~Q(X0) | ~S(X0)) [avatar definition]
2461. ~S(X0) | ~Q(X0) <- (1) [avatar component clause 2460]
2463. 2 <=> PR(sreg) [avatar definition]
2465. PR(sreg) <- (2) [avatar component clause 2463]
2466. 1 | 2 [avatar split clause 2455,2463,2460]
2580. 10 <=> PR(idcnt) [avatar definition]
2581. ~PR(idcnt) <- (~10) [avatar component clause 2580]
2894. 22 <=> PR(imen) [avatar definition]
2895. ~PR(imen) <- (~22) [avatar component clause 2894]
2896. PR(imen) <- (22) [avatar component clause 2894]
3334. EX(X2,sK160(X2,X3)) | ~::(X2,X3) [resolution 2317,2409]
3336. ~::(X6,X7) | UNI(X7) [resolution 2317,1477]
3337. ~::(X8,X9) | PAR(X8) [resolution 2317,1476]
3338. ::(X10,cnt,sK39(X10)) | ~::(X10,idcnt) [resolution 2317,1536]
3341. ~::(X13,idcnt) | ~SDEP(X13,X14) [resolution 2317,1537]
3345. ::(X18,imen,sK160(X18,sreg)) | ~::(X18,sreg) [resolution 2317,1881]
3363. ::(X36,idcnt,sK160(X36,imen)) | ~::(X36,imen) [resolution 2317,1532]
3375. SDEP(X48,sK139(X48)) | ~::(X48,sdcnt) [resolution 2317,1784]
3383. ::(X56,occ,sK160(X56,proc)) | ~::(X56,proc) [resolution 2317,1870]
3390. ~::(X63,cnt,sK160(X63,occ)) | ~::(X63,occ) [resolution 2317,1877]
3395. ::(X68,occ,sK160(X68,pbnd)) | ~::(X68,pbnd) [resolution 2317,1875]
3412. ED(sK165) | PD(sK165) | Q(sK165) [resolution 2344,2367]
3415. 44 <=> Q(sK165) [avatar definition]
3417. Q(sK165) <- (44) [avatar component clause 3415]
3419. 45 <=> PD(sK165) [avatar definition]
3421. PD(sK165) <- (45) [avatar component clause 3419]
3423. 46 <=> ED(sK165) [avatar definition]
3425. ED(sK165) <- (46) [avatar component clause 3423]
3426. 44 | 45 | 46 [avatar split clause 3412,3423,3419,3415]
3428. 47 <=> Q(sK166) [avatar definition]
3429. ~Q(sK166) <- (~47) [avatar component clause 3428]
3432. 48 <=> PD(sK166) [avatar definition]
3434. PD(sK166) <- (48) [avatar component clause 3432]
3436. 49 <=> ED(sK166) [avatar definition]
3445. ~EX(sK165,X0) | tmP(X0,sK167) [resolution 2345,2367]
3446. ~EX(sK166,X1) | tmP(X1,sK168) [resolution 2345,2368]
3447. ~tmP(X0,sK167) | EX(sK165,X0) [resolution 2346,2367]
3448. ~tmP(X1,sK168) | EX(sK166,X1) [resolution 2346,2368]
3495. ~PR(X6) | S(X6) | UNI(X6) [resolution 3336,2341]
3503. ~S(X0) | PAR(X0) [resolution 3337,2338]
3504. ~ED(X1) | PAR(X1) [resolution 3337,2328]
3506. ~PQ(X3) | PAR(X3) [resolution 3337,2331]
3507. PAR(X4) | ::(X4,proc) | ~PD(X4) [resolution 3337,2325]
3528. ~PD(X4) | PAR(X4) [subsumption resolution 3507,3337]
3594. tmP(sK15(sK165),sK167) | ~PAR(sK165) [resolution 3445,1469]
3624. 53 <=> PAR(sK165) [avatar definition]
3625. PAR(sK165) <- (53) [avatar component clause 3624]
3626. ~PAR(sK165) <- (~53) [avatar component clause 3624]
3628. 54 <=> tmP(sK15(sK165),sK167) [avatar definition]
3630. tmP(sK15(sK165),sK167) <- (54) [avatar component clause 3628]
3631. ~53 | 54 [avatar split clause 3594,3628,3624]
3640. PAR(sK165) <- (46) [resolution 3425,3504]
3641. $false <- (46, ~53) [subsumption resolution 3640,3626]
3642. ~46 | 53 [avatar contradiction clause 3641]
3696. PQ(sK165) <- (44) [resolution 3417,2334]
3749. PAR(sK165) <- (44) [resolution 3506,3696]
3751. $false <- (44, ~53) [subsumption resolution 3749,3626]
3752. ~44 | 53 [avatar contradiction clause 3751]
3760. PAR(sK165) <- (45) [resolution 3421,3528]
3761. $false <- (45, ~53) [subsumption resolution 3760,3626]
3762. ~45 | 53 [avatar contradiction clause 3761]
3905. 78 <=> T(sK167) [avatar definition]
3906. T(sK167) <- (78) [avatar component clause 3905]
4179. ::(X1,cnt,X2) | ~EX(X1,X2) | ~::(X1,cnt) [resolution 1900,2317]
4448. 85 <=> sP10(sK166,sK165) [avatar definition]
4450. sP10(sK166,sK165) <- (85) [avatar component clause 4448]
4452. 86 <=> sP9(sK166,sK165) [avatar definition]
4454. sP9(sK166,sK165) <- (86) [avatar component clause 4452]
4456. 87 <=> S(sK165) [avatar definition]
4457. ~S(sK165) <- (~87) [avatar component clause 4456]
4458. S(sK165) <- (87) [avatar component clause 4456]
```

```
4476. S(sK166) | sP9(sK166,sK165) | sP10(sK166,sK165) [resolution 2361,2366]
5410. 133 <=> S(idcnt) [avatar definition]
5412. S(idcnt) <- (133) [avatar component clause 5410]
5593. 145 <=> ::(sK167,treg,sK167) [avatar definition]
5594. ::(sK167,treg,sK167) <- (145) [avatar component clause 5593]
5599. 146 <=> S(sK166) [avatar definition]
5601. S(sK166) <- (146) [avatar component clause 5599]
5603. 85 | 86 | 146 [avatar split clause 4476,5599,4452,4448]
5628. ~Q(sK166) <- (1, 146) [resolution 5601,2461]
5630. ~ED(sK166) <- (146) [resolution 5601,2448]
5632. ~49 | ~146 [avatar split clause 5630,5599,3436]
5675. oP(sK165,sK166) <- (86) [resolution 4454,2357]
5704. ::(sK166,occ,sK48(sK166)) <- (86) [resolution 5675,1565]
5758. ::(sK167,occ,sK148(sK167)) <- (54) [resolution 3630,1819]
6593. PAR(sK166) <- (86) [resolution 5704,1476]
6699. tmP(sK167,sK167) <- (54) [resolution 5758,1797]
6839. tmP(sK15(sK166),sK168) | ~PAR(sK166) [resolution 3446,1469]
6853. EX(sK165,sK167) <- (54) [resolution 3447,6699]
6865. ::(sK167,treg,sK167) <- (54) [resolution 6853,1475]
6870. 145 | ~54 [avatar split clause 6865,3628,5593]
6904. ~47 | ~1 | ~146 [avatar split clause 5628,5599,2460,3428]
6908. 179 <=> PAR(sK166) [avatar definition]
6912. 180 <=> tmP(sK15(sK166),sK168) [avatar definition]
6914. tmP(sK15(sK166),sK168) <- (180) [avatar component clause 6912]
6915. ~179 | 180 [avatar split clause 6839,6912,6908]
6921. 179 | ~86 [avatar split clause 6593,4452,6908]
7190. 201 <=> T(sK168) [avatar definition]
7191. T(sK168) <- (201) [avatar component clause 7190]
7192. ~T(sK168) <- (~201) [avatar component clause 7190]
7297. ~SDEP(sK163(idcnt),X4) | S(idcnt) | ~PR(idcnt) [resolution 3341,2341]
7887. ~::(X8,idcnt) | ::(X8,cnt) [resolution 3338,2318]
7909. 228 <=> ::(sK167,treg) [avatar definition]
7910. ::(sK167,treg) <- (228) [avatar component clause 7909]
7918. 230 <=> ::(sK168,treg) [avatar definition]
7919. ::(sK168,treg) <- (230) [avatar component clause 7918]
8298. ::(sK168,occ,sK148(sK168)) <- (180) [resolution 6914,1819]
8695. ~::(X9,sreg) | ::(X9,imen) [resolution 3345,2318]
8817. ::(sK167,treg) <- (145) [resolution 5594,2318]
8822. 228 | ~145 [avatar split clause 8817,5593,7909]
8824. T(sK167) <- (228) [resolution 7910,2337]
8830. 78 | ~228 [avatar split clause 8824,7909,3905]
8884. ~::(X16,imen) | ::(X16,idcnt) [resolution 3363,2318]
8982. ~::(X20,proc) | ::(X20,occ) [resolution 3383,2318]
9092. ~::(X20,pbnd) | ::(X20,occ) [resolution 3395,2318]
10821. ~EX(X5,sK160(X5,occ)) | ~::(X5,cnt) | ~::(X5,occ) [resolution 4179,3390]
10832. ~::(X5,occ) | ~::(X5,cnt) [subsumption resolution 10821,3334]
12374. 334 <=> S(imen) [avatar definition]
12375. ~S(imen) <- (~334) [avatar component clause 12374]
12376. S(imen) <- (334) [avatar component clause 12374]
13118. tmP(sK168,sK168) <- (180) [resolution 8298,1797]
13167. EX(sK166,sK168) <- (180) [resolution 13118,3448]
13240. ::(sK168,treg,sK168) <- (180) [resolution 13167,1475]
13312. ::(sK168,treg) <- (180) [resolution 13240,2318]
13317. 230 | ~180 [avatar split clause 13312,6912,7918]
13342. T(sK168) <- (230) [resolution 7919,2337]
13346. $false <- (~201, 230) [subsumption resolution 13342,7192]
13347. 201 | ~230 [avatar contradiction clause 13346]
13358. PR(sK165) <- (85) [resolution 4450,2347]
41074. S(sK165) | UNI(sK165) <- (85) [resolution 3495,13358]
41076. UNI(sK165) <- (85, ~87) [subsumption resolution 41074,4457]
41095. ~PAR(sK165) <- (85, ~87) [resolution 41076,1868]
41096. $false <- (53, 85, ~87) [subsumption resolution 41095,3625]
41097. ~53 | ~85 | 87 [avatar contradiction clause 41096]
41262. ED(sK166) | PD(sK166) | Q(sK166) [resolution 2368,2344]
41568. ED(sK166) | PD(sK166) <- (~47) [subsumption resolution 41262,3429]
41569. 48 | 49 | 47 [avatar split clause 41568,3428,3436,3432]
41593. 2378 <=> S(sreg) [avatar definition]
41595. S(sreg) <- (2378) [avatar component clause 41593]
46037. ::(sK163(sreg),imen) | S(sreg) | ~PR(sreg) [resolution 8695,2341]
46038. ::(sK163(sreg),imen) | S(sreg) <- (2) [subsumption resolution 46037,2465]
46040. 2485 <=> ::(sK163(sreg),imen) [avatar definition]
46042. ::(sK163(sreg),imen) <- (2485) [avatar component clause 46040]
46043. 2378 | 2485 | ~2 [avatar split clause 46038,2463,46040,41593]
46193. ::(sK163(imen),idcnt) | S(imen) | ~PR(imen) [resolution 8884,2341]
46344. PR(imen) | ~Q(sK163(sreg)) <- (2485) [resolution 46042,2343]
46345. ~Q(sK163(sreg)) <- (~22, 2485) [subsumption resolution 46344,2895]
46351. S(sreg) | ~PR(sreg) <- (~22, 2485) [resolution 46345,2340]
46352. S(sreg) <- (2, ~22, 2485) [subsumption resolution 46351,2465]
46353. 2378 | ~2 | 22 | ~2485 [avatar split clause 46352,46040,2894,2463,41593]
46360. 2502 <=> ::(sK163(imen),idcnt) [avatar definition]
46362. ::(sK163(imen),idcnt) <- (2502) [avatar component clause 46360]
46363. ~22 | 334 | 2502 [avatar split clause 46193,46360,12374,2894]
46385. PAR(imen) <- (334) [resolution 12376,3503]
46388. $false <- (334) [subsumption resolution 46385,2429]
46389. ~334 [avatar contradiction clause 46388]
46613. PR(idcnt) | ~Q(sK163(imen)) <- (2502) [resolution 46362,2343]
46622. ~Q(sK163(imen)) <- (~10, 2502) [subsumption resolution 46613,2581]
46682. S(imen) | ~PR(imen) <- (~10, 2502) [resolution 46622,2340]
46683. ~PR(imen) <- (~10, ~334, 2502) [subsumption resolution 46682,12375]
46684. $false <- (~10, 22, ~334, 2502) [subsumption resolution 46683,2896]
46685. 10 | ~22 | 334 | ~2502 [avatar contradiction clause 46684]
```

```
46735. 2517 <=> ! [X4] : ~SDEP(sK163(idcnt),X4) [avatar definition]
46736. ~SDEP(sK163(idcnt),X4) <- (2517) [avatar component clause 46735]
46737. ~10 | 133 | 2517 [avatar split clause 7297,46735,5410,2580]
46766. ~::(sK163(idcnt),sdcnt) <- (2517) [resolution 46736,3375]
47052. ~PQ(sK163(idcnt)) <- (2517) [resolution 46766,2331]
47053. ~PR(idcnt) | S(idcnt) <- (2517) [resolution 47052,2454]
47064. PAR(idcnt) <- (133) [resolution 5412,3503]
47067. $false <- (133) [subsumption resolution 47064,2418]
47068. ~133 [avatar contradiction clause 47067]
47070. 133 | ~10 | ~2517 [avatar split clause 47053,46735,2580,5410]
47180. PAR(sreg) <- (2378) [resolution 41595,3503]
47183. $false <- (2378) [subsumption resolution 47180,2420]
47184. ~2378 [avatar contradiction clause 47183]
48809. 2598 <=> ::(sK166,proc) [avatar definition]
48810. ::(sK166,proc) <- (2598) [avatar component clause 48809]
48811. ~::(sK166,proc) <- (~2598) [avatar component clause 48809]
48854. 2602 <=> ::(sK166,pbnd) [avatar definition]
48855. ::(sK166,pbnd) <- (2602) [avatar component clause 48854]
48856. ~::(sK166,pbnd) <- (~2602) [avatar component clause 48854]
48930. ::(sK166,proc) | ~PD(sK166) <- (~2602) [resolution 48856,2325]
48931. ~PD(sK166) <- (~2598, ~2602) [subsumption resolution 48930,48811]
48932. $false <- (48, ~2598, ~2602) [subsumption resolution 48931,3434]
48933. ~48 | 2598 | 2602 [avatar contradiction clause 48932]
49649. 2652 <=> ::(sK166,idcnt) [avatar definition]
49650. ::(sK166,idcnt) <- (2652) [avatar component clause 49649]
90503. 4322 <=> ::(sK166,sreg) [avatar definition]
90504. ::(sK166,sreg) <- (4322) [avatar component clause 90503]
90505. ~::(sK166,sreg) <- (~4322) [avatar component clause 90503]
90556. 4327 <=> ::(sK166,cnt) [avatar definition]
90557. ::(sK166,cnt) <- (4327) [avatar component clause 90556]
90985. ::(sK166,occ) <- (2598) [resolution 48810,8982]
91055. ~::(sK166,cnt) <- (2598) [resolution 90985,10832]
91061. ~4327 | ~2598 [avatar split clause 91055,48809,90556]
91352. ~S(sK166) <- (~4322) [resolution 90505,2338]
91353. $false <- (146, ~4322) [subsumption resolution 91352,5601]
91354. ~146 | 4322 [avatar contradiction clause 91353]
91361. ::(sK166,imen) <- (4322) [resolution 90504,8695]
91507. ::(sK166,idcnt) <- (4322) [resolution 91361,8884]
91514. 2652 | ~4322 [avatar split clause 91507,90503,49649]
91524. ::(sK166,cnt) <- (2652) [resolution 49650,7887]
91532. 4327 | ~2652 [avatar split clause 91524,49649,90556]
91551. ::(sK166,occ) <- (2602) [resolution 48855,9092]
91601. ~::(sK166,cnt) <- (2602) [resolution 91551,10832]
91607. $false <- (2602, 4327) [subsumption resolution 91601,90557]
91608. ~2602 | ~4327 [avatar contradiction clause 91607]
91614. ~S(sK165) <- (85) [resolution 4450,2349]
91617. $false <- (85, 87) [subsumption resolution 91614,4458]
91618. ~85 | ~87 [avatar contradiction clause 91617]
91957. oP(sK165,sK166) <- (86) [resolution 4454,2357]
92017. ~EX(sK165,X1) | EX(sK166,X1) <- (86) [resolution 91957,1563]
315750. EX(sK166,sK167) <- (54, 86) [resolution 92017,6853]
315870. tmP(sK167,sK168) <- (54, 86) [resolution 315750,3446]
316268. oP(sK167,sK168) <- (54, 86) [resolution 315870,1795]
320210. sP9(sK168,sK167) | ~T(sK168) | ~T(sK167) <- (54, 86) [resolution 316268,2359]
320219. sP9(sK168,sK167) | ~T(sK167) <- (54, 86, 201) [subsumption resolution 320210,7191]
320220. sP9(sK168,sK167) <- (54, 78, 86, 201) [subsumption resolution 320219,3906]
321906. P(sK167,sK168) <- (54, 78, 86, 201) [resolution 320220,2363]
321908. $false <- (54, 78, 86, 201) [subsumption resolution 321906,2369]
321909. ~54 | ~78 | ~86 | ~201 [avatar contradiction clause 321908]
321910. $false [avatar sat refutation
        2466,3426,3631,3642,3752,3762,5603,5632,6870,6904,6915,6921,8822,8830,13317,13347,41097,
41569,46043,46353,46363,46389,46685,46737,47068,47070,47184,48933,91061,91354,91514,
91532,91608,91618,321909]
```

Proof of Theorem (t_{bd}17)

```
26. ! [X2] : (PAR(X2) => ? [X1] : EX(X2,X1)) [input]
30. ! [X17] : ! [X1] : (EX(X17,X1) => (::(X1,treg,X1) & PAR(X1) & PAR(X17))) [input]
138. ! [X5] : ! [X9] : (SDEP(X5,X9) => (! [X1] : (EX(X5,X1) => EX(X9,X1)) & ? [X1] : (EX(X9,X1) & EX(X5,X1)))) [
     input]
141. ! [X0] : ! [X8] : (INH(X0,X8) <=> (? [X1] : (~::(X8,sreg,X1) & ::(X8,idcnt,X1) & ::(X0,sdcnt,X1)) & SDEP(X0,
     X8))) [input]
146. ! [X0] : ! [X8] : (tmP(X0,X8) => oP(X0,X8)) [input]
148. ! [X0] : (? [X1] : ::(X0,occ,X1) => tmP(X0,X0)) [input]
163. ! [X0] : ! [X8] : (tmP(X0,X8) => (? [X1] : ::(X8,occ,X1) & ? [X1] : ::(X0,occ,X1))) [input]
257. ! [X6,X16] : (::(X6,X16) <=> ? [X1] : ::(X6,X16,X1)) [input]
269. ! [X6] : (T(X6) <=> ::(X6,treg)) [input]
275. ! [X6,X1] : (TLC(X6,X1) <=> (! [X16] : (EX(X6,X16) <=> tmP(X16,X1)) & (Q(X6) | ED(X6) | PD(X6)))) [input]
276. ! [X6,X7] : (P(X6,X7) <=> ((ISA(X6,X7) & ~S(X7) & ~S(X6) & PR(X7) & PR(X6)) | (? [X1] : cP(X6,X7,X1) & S(X7)
     & S(X6)) | (oP(X6,X7) & ((T(X7) & T(X6)) | (PD(X7) & PD(X6)))))) [input]
279. ! [X6,X7] : (DQT(X6,X7) <=> INH(X6,X7)) [input]
292. ! [X6,X7,X1,X16] : ((TLC(X7,X16) & TLC(X6,X1) & DQT(X6,X7)) => P(X1,X16)) [input]
293. ~! [X6,X7,X1,X16] : ((TLC(X7,X16) & TLC(X6,X1) & DQT(X6,X7)) => P(X1,X16)) [negated conjecture 292]
343. ! [X0] : (PAR(X0) => ? [X1] : EX(X0,X1)) [rectify 26]
350. ! [X0] : ! [X1] : (EX(X0,X1) => (::(X1,treg,X1) & PAR(X1) & PAR(X0))) [rectify 30]
351. ! [X0,X1] : (EX(X0,X1) => (::(X1,treg,X1) & PAR(X1) & PAR(X0))) [flattening 350]
555. ! [X0] : ! [X1] : (SDEP(X0,X1) => (! [X2] : (EX(X0,X2) => EX(X1,X2)) & ? [X3] : (EX(X1,X3) & EX(X0,X3)))) [
     rectify 138]
556. ! [X0,X1] : (SDEP(X0,X1) => (! [X2] : (EX(X0,X2) => EX(X1,X2)) & ? [X3] : (EX(X1,X3) & EX(X0,X3)))) [
     flattening 555]
```

```
561. ! [X0] : ! [X1] : (INH(X0,X1) <=> (? [X2] : (~::(X1,sreg,X2) & ::(X1,idcnt,X2) & ::(X0,sdcnt,X2)) & SDEP(X0,
     X1))) [rectify 141]
562. ! [X0,X1] : (INH(X0,X1) <=> (? [X2] : (~::(X1,sreg,X2) & ::(X1,idcnt,X2) & ::(X0,sdcnt,X2)) & SDEP(X0,X1))) [
     flattening 561]
571. ! [X0] : ! [X1] : (tmP(X0,X1) => oP(X0,X1)) [rectify 146]
572. ! [X0,X1] : (tmP(X0,X1) => oP(X0,X1)) [flattening 571]
599. ! [X0] : ! [X1] : (tmP(X0,X1) => (? [X2] : ::(X1,occ,X2) & ? [X3] : ::(X0,occ,X3))) [rectify 163]
600. ! [X0,X1] : (tmP(X0,X1) => (? [X2] : ::(X1,occ,X2) & ? [X3] : ::(X0,occ,X3))) [flattening 599]
705. ! [X0,X1] : (::(X0,X1) <=> ? [X2] : ::(X0,X1,X2)) [rectify 257]
718. ! [X0] : (T(X0) <=> ::(X0,treg)) [rectify 269]
724. ! [X0,X1] : (TLC(X0,X1) <=> (! [X2] : (EX(X0,X2) <=> tmP(X2,X1)) & (Q(X0) | ED(X0) | PD(X0)))) [rectify 275]
725. ! [X0,X1] : (P(X0,X1) <=> ((ISA(X0,X1) & ~S(X1) & ~S(X0) & PR(X1) & PR(X0)) | (? [X2] : cP(X0,X1,X2) & S(X1)
     & S(X0)) | (oP(X0,X1) & ((T(X1) & T(X0)) | (PD(X1) & PD(X0)))))) [rectify 276]
728. ! [X0,X1] : (DQT(X0,X1) <=> INH(X0,X1)) [rectify 279]
741. ~! [X0,X1,X2,X3] : ((TLC(X1,X3) & TLC(X0,X2) & DQT(X0,X1)) => P(X2,X3)) [rectify 293]
742. ! [X0,X1] : (TLC(X0,X1) => (! [X2] : (EX(X0,X2) <=> tmP(X2,X1)) & (Q(X0) | ED(X0) | PD(X0)))) [unused
     predicate definition removal 724]
743. ! [X0,X1] : (((ISA(X0,X1) & ~S(X1) & ~S(X0) & PR(X1) & PR(X0)) | (? [X2] : cP(X0,X1,X2) & S(X1) & S(X0)) | (
     oP(X0,X1) & ((T(X1) & T(X0)) | (PD(X1) & PD(X0))))) => P(X0,X1)) [unused predicate definition removal 725]
745. ! [X0] : (::(X0,treg) => T(X0)) [unused predicate definition removal 718]
747. ! [X0,X1] : (DQT(X0,X1) => INH(X0,X1)) [unused predicate definition removal 728]
796. ! [X0] : (? [X1] : EX(X0,X1) | ~PAR(X0)) [ennf transformation 343]
802. ! [X0,X1] : ((::(X1,treg,X1) & PAR(X1) & PAR(X0)) | ~EX(X0,X1)) [ennf transformation 351]
957. ! [X0,X1] : ((! [X2] : (EX(X1,X2) | ~EX(X0,X2)) & ? [X3] : (EX(X1,X3) & EX(X0,X3))) | ~SDEP(X0,X1)) [ennf
     transformation 556]
962. ! [X0,X1] : (oP(X0,X1) | ~tmP(X0,X1)) [ennf transformation 572]
964. ! [X0] : (tmP(X0,X0) | ! [X1] : ~::(X0,occ,X1)) [ennf transformation 148]
984. ! [X0,X1] : ((? [X2] : ::(X1,occ,X2) & ? [X3] : ::(X0,occ,X3)) | ~tmP(X0,X1)) [ennf transformation 600]
1050. ! [X0] : (T(X0) | ~::(X0,treg)) [ennf transformation 745]
1052. ! [X0,X1] : ((! [X2] : (EX(X0,X2) <=> tmP(X2,X1)) & (Q(X0) | ED(X0) | PD(X0))) | ~TLC(X0,X1)) [ennf
      transformation 742]
1053. ! [X0,X1] : (P(X0,X1) | ((~ISA(X0,X1) | S(X1) | S(X0) | ~PR(X1) | ~PR(X0)) & (! [X2] : ~cP(X0,X1,X2) | ~S(X1
      ) | ~S(X0)) & (~oP(X0,X1) | ((~T(X1) | ~T(X0)) & (~PD(X1) | ~PD(X0)))))) [ennf transformation 743]
1054. ! [X0,X1] : (INH(X0,X1) | ~DQT(X0,X1)) [ennf transformation 747]
1055. ? [X0,X1,X2,X3] : (~P(X2,X3) & (TLC(X1,X3) & TLC(X0,X2) & DQT(X0,X1))) [ennf transformation 741]
1056. ? [X0,X1,X2,X3] : (~P(X2,X3) & TLC(X1,X3) & TLC(X0,X2) & DQT(X0,X1)) [flattening 1055]
1089. ! [X0] : (? [X1] : EX(X0,X1) => EX(X0,sK13(X0))) [choice axiom]
1090. ! [X0] : (EX(X0,sK13(X0)) | ~PAR(X0)) [skolemisation 796,1089]
1315. ! [X1,X0] : (? [X3] : (EX(X1,X3) & EX(X0,X3)) => (EX(X1,sK128(X0,X1)) & EX(X0,sK128(X0,X1)))) [choice axiom]
1316. ! [X0,X1] : ((! [X2] : (EX(X1,X2) | ~EX(X0,X2)) & (EX(X1,sK128(X0,X1)) & EX(X0,sK128(X0,X1)))) | ~SDEP(X0,X1
      )) [skolemisation 957,1315]
1327. ! [X0,X1] : ((INH(X0,X1) | (! [X2] : (::(X1,sreg,X2) | ~::(X1,idcnt,X2) | ~::(X0,sdcnt,X2)) | ~SDEP(X0,X1)))
      & ((? [X2] : (~::(X1,sreg,X2) & ::(X1,idcnt,X2) & ::(X0,sdcnt,X2)) & SDEP(X0,X1)) | ~INH(X0,X1))) [nnf
      transformation 562]
1328. ! [X0,X1] : ((INH(X0,X1) | ! [X2] : (::(X1,sreg,X2) | ~::(X1,idcnt,X2) | ~::(X0,sdcnt,X2)) | ~SDEP(X0,X1)) &
       ((? [X2] : (~::(X1,sreg,X2) & ::(X1,idcnt,X2) & ::(X0,sdcnt,X2)) & SDEP(X0,X1)) | ~INH(X0,X1))) [flattening
      1327]
1329. ! [X0,X1] : ((INH(X0,X1) | ! [X2] : (::(X1,sreg,X2) | ~::(X1,idcnt,X2) | ~::(X0,sdcnt,X2)) | ~SDEP(X0,X1)) &
       ((? [X3] : (~::(X1,sreg,X3) & ::(X1,idcnt,X3) & ::(X0,sdcnt,X3)) & SDEP(X0,X1)) | ~INH(X0,X1))) [rectify
      1328]
1330. ! [X1,X0] : (? [X3] : (~::(X1,sreg,X3) & ::(X1,idcnt,X3) & ::(X0,sdcnt,X3)) => (~::(X1,sreg,sK135(X0,X1)) &
      ::(X1,idcnt,sK135(X0,X1)) & ::(X0,sdcnt,sK135(X0,X1)))) [choice axiom]
1331. ! [X0,X1] : ((INH(X0,X1) | ! [X2] : (::(X1,sreg,X2) | ~::(X1,idcnt,X2) | ~::(X0,sdcnt,X2)) | ~SDEP(X0,X1)) &
       (((~::(X1,sreg,sK135(X0,X1)) & ::(X1,idcnt,sK135(X0,X1)) & ::(X0,sdcnt,sK135(X0,X1))) & SDEP(X0,X1)) | ~INH
      (X0,X1))) [skolemisation 1329,1330]
1350. ! [X1] : (? [X2] : ::(X1,occ,X2) => ::(X1,occ,sK146(X1))) [choice axiom]
1351. ! [X0] : (? [X3] : ::(X0,occ,X3) => ::(X0,occ,sK147(X0))) [choice axiom]
1352. ! [X0,X1] : ((::(X1,occ,sK146(X1)) & ::(X0,occ,sK147(X0))) | ~tmP(X0,X1)) [skolemisation 984,1351,1350]
1397. ! [X0,X1] : ((::(X0,X1) | ! [X2] : ~::(X0,X1,X2)) & (? [X2] : ::(X0,X1,X2) | ~::(X0,X1))) [nnf
      transformation 705]
1398. ! [X0,X1] : ((::(X0,X1) | ! [X2] : ~::(X0,X1,X2)) & (? [X3] : ::(X0,X1,X3) | ~::(X0,X1))) [rectify 1397]
1399. ! [X1,X0] : (? [X3] : ::(X0,X1,X3) => ::(X0,X1,sK158(X0,X1))) [choice axiom]
1400. ! [X0,X1] : ((::(X0,X1) | ! [X2] : ~::(X0,X1,X2)) & (::(X0,X1,sK158(X0,X1)) | ~::(X0,X1))) [skolemisation
      1398,1399]
1411. ! [X0,X1] : ((! [X2] : ((EX(X0,X2) | ~tmP(X2,X1)) & (tmP(X2,X1) | ~EX(X0,X2))) & (Q(X0) | ED(X0) | PD(X0)))
      | ~TLC(X0,X1)) [nnf transformation 1052]
1412. ? [X0,X1,X2,X3] : (~P(X2,X3) & TLC(X1,X3) & TLC(X0,X2) & DQT(X0,X1)) => (~P(sK163,sK164) & TLC(sK162,sK164)
      & TLC(sK161,sK163) & DQT(sK161,sK162)) [choice axiom]
1413. ~P(sK163,sK164) & TLC(sK162,sK164) & TLC(sK161,sK163) & DQT(sK161,sK162) [skolemisation 1056,1412]
1456. EX(X0,sK13(X0)) | ~PAR(X0) [cnf transformation 1090]
1460. ~EX(X0,X1) | PAR(X0) [cnf transformation 802]
1462. ~EX(X0,X1) | ::(X1,treg,X1) [cnf transformation 802]
1752. EX(X0,sK128(X0,X1)) | ~SDEP(X0,X1) [cnf transformation 1316]
1753. EX(X1,sK128(X0,X1)) | ~SDEP(X0,X1) [cnf transformation 1316]
1754. ~SDEP(X0,X1) | ~EX(X0,X2) | EX(X1,X2) [cnf transformation 1316]
1763. ~INH(X0,X1) | SDEP(X0,X1) [cnf transformation 1331]
1782. ~tmP(X0,X1) | oP(X0,X1) [cnf transformation 962]
1784. ~::(X0,occ,X1) | tmP(X0,X0) [cnf transformation 964]
1806. ~tmP(X0,X1) | ::(X1,occ,sK146(X1)) [cnf transformation 1352]
2305. ~::(X0,X1,X2) | ::(X0,X1) [cnf transformation 1400]
2322. ~::(X0,treg) | T(X0) [cnf transformation 1050]
2328. ~TLC(X0,X1) | ~EX(X0,X2) | tmP(X2,X1) [cnf transformation 1411]
2329. ~TLC(X0,X1) | ~tmP(X2,X1) | EX(X0,X2) [cnf transformation 1411]
2331. ~oP(X0,X1) | P(X0,X1) | ~T(X1) | ~T(X0) [cnf transformation 1053]
2334. ~DQT(X0,X1) | INH(X0,X1) [cnf transformation 1054]
2335. DQT(sK161,sK162) [cnf transformation 1413]
2336. TLC(sK161,sK163) [cnf transformation 1413]
2337. TLC(sK162,sK164) [cnf transformation 1413]
2338. ~P(sK163,sK164) [cnf transformation 1413]
2421. INH(sK161,sK162) [resolution 2334,2335]
2425. SDEP(sK161,sK162) [resolution 2421,1763]
```

```
2651. ~SDEP(X4,X5) | PAR(X4) [resolution 1752,1460]
2654. ~SDEP(X4,X5) | PAR(X5) [resolution 1753,1460]
2671. PAR(sK161) [resolution 2651,2425]
2675. PAR(sK162) [resolution 2654,2425]
3207. ~EX(sK161,X0) | EX(sK162,X0) [resolution 1754,2425]
3403. ~EX(sK161,X0) | tmP(X0,sK163) [resolution 2328,2336]
3404. ~EX(sK162,X1) | tmP(X1,sK164) [resolution 2328,2337]
3438. ~tmP(X0,sK163) | EX(sK161,X0) [resolution 2329,2336]
3439. ~tmP(X1,sK164) | EX(sK162,X1) [resolution 2329,2337]
3571. tmP(sK13(sK161),sK163) | ~PAR(sK161) [resolution 3403,1456]
3574. tmP(sK13(sK161),sK163) [subsumption resolution 3571,2671]
3584. ::(sK163,occ,sK146(sK163)) [resolution 3574,1806]
3999. 74 <=> T(sK163) [avatar definition]
4000. T(sK163) <- (74) [avatar component clause 3999]
4647. tmP(sK163,sK163) [resolution 3584,1784]
5125. tmP(sK13(sK162),sK164) | ~PAR(sK162) [resolution 3404,1456]
5128. tmP(sK13(sK162),sK164) [subsumption resolution 5125,2675]
5140. ::(sK164,occ,sK146(sK164)) [resolution 5128,1806]
5205. 102 <=> T(sK164) [avatar definition]
5206. T(sK164) <- (102) [avatar component clause 5205]
6532. tmP(sK164,sK164) [resolution 5140,1784]
6972. EX(sK161,sK163) [resolution 3438,4647]
6984. ::(sK163,treg,sK163) [resolution 6972,1462]
6991. EX(sK162,sK164) [resolution 3439,6532]
7003. ::(sK164,treg,sK164) [resolution 6991,1462]
7980. 230 <=> ::(sK163,treg) [avatar definition]
7981. ::(sK163,treg) <- (230) [avatar component clause 7980]
7989. 232 <=> ::(sK164,treg) [avatar definition]
7990. ::(sK164,treg) <- (232) [avatar component clause 7989]
8034. ::(sK163,treg) [resolution 6984,2305]
8039. 230 [avatar split clause 8034,7980]
8045. T(sK163) <- (230) [resolution 7981,2322]
8051. 74 | ~230 [avatar split clause 8045,7980,3999]
8341. ::(sK164,treg) [resolution 7003,2305]
8346. 232 [avatar split clause 8341,7989]
8368. T(sK164) <- (232) [resolution 7990,2322]
16627. 102 | ~232 [avatar split clause 8368,7989,5205]
139248. EX(sK162,sK163) [resolution 3207,6972]
139380. tmP(sK163,sK164) [resolution 139248,3404]
139404. oP(sK163,sK164) [resolution 139380,1782]
141343. P(sK163,sK164) | ~T(sK164) | ~T(sK163) [resolution 139404,2331]
141350. ~T(sK164) | ~T(sK163) [subsumption resolution 141343,2338]
141351. ~T(sK163) <- (102) [subsumption resolution 141350,5206]
141352. $false <- (74, 102) [subsumption resolution 141351,4000]
141353. ~74 | ~102 [avatar contradiction clause 141352]
141354. $false [avatar sat refutation 8039,8051,8346,16627,141353]
```

Proof of Theorem (t_{bd}18)

```
13. ! [X0] : ! [X8] : ! [X1] : (cP(X0,X8,X1) => (::(X1,treg,X1) & ::(X8,cnt,X1) & ::(X0,cnt,X1))) [input]
257. ! [X6,X16] : (::(X6,X16) <=> ? [X1] : ::(X6,X16,X1)) [input]
269. ! [X6] : (T(X6) <=> ::(X6,treg)) [input]
277. ! [X6,X7,X1] : (tP(X6,X7,X1) <=> (cP(X6,X7,X1) & ED(X7) & ED(X6))) [input]
292. ! [X6,X7,X1] : (tP(X6,X7,X1) => (T(X1) & ED(X7) & ED(X6))) [input]
293. ~! [X6,X7,X1] : (tP(X6,X7,X1) => (T(X1) & ED(X7) & ED(X6))) [negated conjecture 292]
317. ! [X0] : ! [X1] : ! [X2] : (cP(X0,X1,X2) => (::(X2,treg,X2) & ::(X1,cnt,X2) & ::(X0,cnt,X2))) [rectify 13]
318. ! [X0,X1,X2] : (cP(X0,X1,X2) => (::(X2,treg,X2) & ::(X1,cnt,X2) & ::(X0,cnt,X2))) [flattening 317]
705. ! [X0,X1] : (::(X0,X1) <=> ? [X2] : ::(X0,X1,X2)) [rectify 257]
718. ! [X0] : (T(X0) <=> ::(X0,treg)) [rectify 269]
726. ! [X0,X1,X2] : (tP(X0,X1,X2) <=> (cP(X0,X1,X2) & ED(X1) & ED(X0))) [rectify 277]
741. ~! [X0,X1,X2] : (tP(X0,X1,X2) => (T(X2) & ED(X1) & ED(X0))) [rectify 293]
742. ! [X0,X1,X2] : (tP(X0,X1,X2) => (cP(X0,X1,X2) & ED(X1) & ED(X0))) [unused predicate definition removal 726]
743. ! [X0] : (::(X0,treg) => T(X0)) [unused predicate definition removal 718]
768. ! [X0,X1,X2] : ((::(X2,treg,X2) & ::(X1,cnt,X2) & ::(X0,cnt,X2)) | ~cP(X0,X1,X2)) [ennf transformation 318]
1043. ! [X0] : (T(X0) | ~::(X0,treg)) [ennf transformation 743]
1044. ! [X0,X1,X2] : ((cP(X0,X1,X2) & ED(X1) & ED(X0)) | ~tP(X0,X1,X2)) [ennf transformation 742]
1045. ? [X0,X1,X2] : ((~T(X2) | ~ED(X1) | ~ED(X0)) & tP(X0,X1,X2)) [ennf transformation 741]
1386. ! [X0,X1] : ((::(X0,X1) | ! [X2] : ~::(X0,X1,X2)) & (? [X2] : ::(X0,X1,X2) | ~::(X0,X1))) [nnf
      transformation 705]
1387. ! [X0,X1] : ((::(X0,X1) | ! [X2] : ~::(X0,X1,X2)) & (? [X3] : ::(X0,X1,X3) | ~::(X0,X1))) [rectify 1386]
1388. ! [X1,X0] : (? [X3] : ::(X0,X1,X3) => ::(X0,X1,sK158(X0,X1))) [choice axiom]
1389. ! [X0,X1] : ((::(X0,X1) | ! [X2] : ~::(X0,X1,X2)) & (::(X0,X1,sK158(X0,X1)) | ~::(X0,X1))) [skolemisation
      1387,1388]
1394. ? [X0,X1,X2] : ((~T(X2) | ~ED(X1) | ~ED(X0)) & tP(X0,X1,X2)) => ((~T(sK161) | ~ED(sK160) | ~ED(sK159)) & tP(
      sK159,sK160,sK161)) [choice axiom]
1395. (~T(sK161) | ~ED(sK160) | ~ED(sK159)) & tP(sK159,sK160,sK161) [skolemisation 1045,1394]
1414. ~cP(X0,X1,X2) | ::(X2,treg,X2) [cnf transformation 768]
2287. ~::(X0,X1,X2) | ::(X0,X1) [cnf transformation 1389]
2295. ~::(X0,treg) | T(X0) [cnf transformation 1043]
2296. ~tP(X0,X1,X2) | ED(X0) [cnf transformation 1044]
2297. ~tP(X0,X1,X2) | ED(X1) [cnf transformation 1044]
2298. ~tP(X0,X1,X2) | cP(X0,X1,X2) [cnf transformation 1044]
2299. tP(sK159,sK160,sK161) [cnf transformation 1395]
2300. ~T(sK161) | ~ED(sK160) | ~ED(sK159) [cnf transformation 1395]
2349. 1 <=> ED(sK159) [avatar definition]
2353. 2 <=> ED(sK160) [avatar definition]
2357. 3 <=> T(sK161) [avatar definition]
2359. ~T(sK161) <- (~3) [avatar component clause 2357]
2360. ~1 | ~2 | ~3 [avatar split clause 2300,2357,2353,2349]
2392. ED(sK159) [resolution 2296,2299]
```

```
2395. 1 [avatar split clause 2392,2349]
2396. ED(sK160) [resolution 2297,2299]
2399. 2 [avatar split clause 2396,2353]
2806. cP(sK159,sK160,sK161) [resolution 2298,2299]
2807. ::(sK161,treg,sK161) [resolution 2806,1414]
2824. ::(sK161,treg) [resolution 2807,2287]
2942. T(sK161) [resolution 2824,2295]
2943. $false <- (~3) [subsumption resolution 2942,2359]
2944. 3 [avatar contradiction clause 2943]
2945. $false [avatar sat refutation 2360,2395,2399,2944]
```

Proof of Theorem (t_{bd}20)

```
This went KO with both provers
```

Proof of Theorem (t_{bd}23)

```
12. ! [X0] : ! [X8] : ! [X9] : ! [X1] : ! [X10] : (((tmP(X1,X10) & cP(X8,X9,X10) & cP(X0,X8,X1)) => cP(X0,X9,X1)) [
      input]
13. ! [X0] : ! [X8] : ! [X1] : (cP(X0,X8,X1) => (::(X1,treg,X1) & ::(X8,cnt,X1) & ::(X0,cnt,X1))) [input]
152. ! [X0] : (? [X1] : ::(X0,treg,X1) => tmP(X0,X0)) [input]
277. ! [X6,X7,X1] : (tP(X6,X7,X1) <=> (cP(X6,X7,X1) & ED(X7) & ED(X6))) [input]
292. ! [X6,X7,X1,X13] : (((tP(X7,X13,X1) & tP(X6,X7,X1)) => tP(X6,X13,X1)) [input]
293. ~! [X6,X7,X1,X13] : (((tP(X7,X13,X1) & tP(X6,X7,X1)) => tP(X6,X13,X1)) [negated conjecture 292]
315. ! [X0] : ! [X1] : ! [X2] : ! [X3] : ! [X4] : (((tmP(X3,X4) & cP(X1,X2,X4) & cP(X0,X1,X3)) => cP(X0,X2,X3)) [
      rectify 12]
316. ! [X0,X1,X2,X3,X4] : ((tmP(X3,X4) & cP(X1,X2,X4) & cP(X0,X1,X3)) => cP(X0,X2,X3)) [flattening 315]
317. ! [X0] : ! [X1] : ! [X2] : (cP(X0,X1,X2) => (::(X2,treg,X2) & ::(X1,cnt,X2) & ::(X0,cnt,X2))) [rectify 13]
318. ! [X0,X1,X2] : (cP(X0,X1,X2) => (::(X2,treg,X2) & ::(X1,cnt,X2) & ::(X0,cnt,X2))) [flattening 317]
726. ! [X0,X1,X2] : (tP(X0,X1,X2) <=> (cP(X0,X1,X2) & ED(X1) & ED(X0))) [rectify 277]
741. ~! [X0,X1,X2,X3] : ((tP(X1,X3,X2) & tP(X0,X1,X2)) => tP(X0,X3,X2)) [rectify 293]
764. ! [X0,X1,X2,X3,X4] : (cP(X0,X2,X3) | (~tmP(X3,X4) | ~cP(X1,X2,X4) | ~cP(X0,X1,X3))) [ennf transformation 316]
765. ! [X0,X1,X2,X3,X4] : (cP(X0,X2,X3) | ~tmP(X3,X4) | ~cP(X1,X2,X4) | ~cP(X0,X1,X3)) [flattening 764]
766. ! [X0,X1,X2] : ((::(X2,treg,X2) & ::(X1,cnt,X2) & ::(X0,cnt,X2)) | ~cP(X0,X1,X2)) [ennf transformation 318]
963. ! [X0] : (tmP(X0,X0) | ! [X1] : ~::(X0,treg,X1)) [ennf transformation 152]
1041. ? [X0,X1,X2,X3] : (~tP(X0,X3,X2) & (tP(X1,X3,X2) & tP(X0,X1,X2))) [ennf transformation 741]
1042. ? [X0,X1,X2,X3] : (~tP(X0,X3,X2) & tP(X1,X3,X2) & tP(X0,X1,X2)) [flattening 1041]
1391. ! [X0,X1,X2] : (((tP(X0,X1,X2) | ~cP(X0,X1,X2) | ~ED(X1) | ~ED(X0))) & ((cP(X0,X1,X2) & ED(X1) & ED(X0)) | ~
      tP(X0,X1,X2))) [nnf transformation 726]
1392. ! [X0,X1,X2] : (((tP(X0,X1,X2) | ~cP(X0,X1,X2) | ~ED(X1) | ~ED(X0)) & ((cP(X0,X1,X2) & ED(X1) & ED(X0)) | ~tP
      (X0,X1,X2))) [flattening 1391]
1393. ? [X0,X1,X2,X3] : (~tP(X0,X3,X2) & tP(X1,X3,X2) & tP(X0,X1,X2)) => (~tP(sK159,sK162,sK161) & tP(sK160,sK162,
      sK161) & tP(sK159,sK160,sK161)) [choice axiom]
1394. ~tP(sK159,sK162,sK161) & tP(sK160,sK162,sK161) & tP(sK159,sK160,sK161) [skolemisation 1042,1393]
1410. ~cP(X1,X2,X4) | ~tmP(X3,X4) | cP(X0,X2,X3) | ~cP(X0,X1,X3) [cnf transformation 765]
1413. ~cP(X0,X1,X2) | ::(X2,treg,X2) [cnf transformation 766]
1769. ~::(X0,treg,X1) | tmP(X0,X0) [cnf transformation 963]
2294. ~tP(X0,X1,X2) | ED(X0) [cnf transformation 1392]
2295. ~tP(X0,X1,X2) | ED(X1) [cnf transformation 1392]
2296. ~tP(X0,X1,X2) | cP(X0,X1,X2) [cnf transformation 1392]
2297. ~cP(X0,X1,X2) | tP(X0,X1,X2) | ~ED(X1) | ~ED(X0) [cnf transformation 1392]
2298. tP(sK159,sK160,sK161) [cnf transformation 1394]
2299. tP(sK160,sK162,sK161) [cnf transformation 1394]
2300. ~tP(sK159,sK162,sK161) [cnf transformation 1394]
2379. ED(sK159) [resolution 2294,2298]
2382. ED(sK162) [resolution 2295,2299]
2799. cP(sK159,sK160,sK161) [resolution 2296,2298]
2800. cP(sK160,sK162,sK161) [resolution 2296,2299]
2804. ::(sK161,treg,sK161) [resolution 2799,1413]
2821. tmP(sK161,sK161) [resolution 2804,1769]
4763. ~cP(X23,sK160,X22) | cP(X23,sK162,X22) | ~tmP(X22,sK161) [resolution 1410,2800]
53818. cP(sK159,sK162,sK161) | ~tmP(sK161,sK161) [resolution 4763,2799]
53824. cP(sK159,sK162,sK161) [subsumption resolution 53818,2821]
53852. tP(sK159,sK162,sK161) | ~ED(sK162) | ~ED(sK159) [resolution 53824,2297]
53890. ~ED(sK162) | ~ED(sK159) [subsumption resolution 53852,2300]
53891. ~ED(sK159) [subsumption resolution 53890,2382]
53892. $false [subsumption resolution 53891,2379]
```

Proof of Theorem (t_{bd}25)

```
1. ! [X0] : ! [X1] : (::(X0,idcnt,X1) => cP(X0,X0,X1)) [input]
11. ! [X2] : ! [X3] : ! [X1] : (cP(X2,X3,X1) => (::(X2,idcnt,X1) <=> ::(X3,idcnt,X1))) [input]
13. ! [X0] : ! [X8] : ! [X1] : (cP(X0,X8,X1) => (::(X1,treg,X1) & ::(X8,cnt,X1) & ::(X0,cnt,X1))) [input]
31. ! [X17] : ! [X16] : ! [X1] : (::(X17,X16,X1) => (::(X1,treg,X1) & UNI(X16) & PAR(X17))) [input]
34. ! [X0] : ! [X1] : (? [X16] : (::(X1,treg,X1) & ::(X0,X16,X1) & UNI(X16)) <=> (EX(X0,X1) & ::(X1,treg,X1) & PAR
      (X0))) [input]
54. ! [X28] : (? [X1] : ::(X28,idcnt,X1) <=> (~? [X29] : ? [X1] : (GDEP(X28,X29,X1) | SDEP(X28,X29)) & ? [X1] :
      ::(X28,cnt,X1))) [input]
142. ! [X5] : (? [X1] : ::(X5,sdcnt,X1) <=> ? [X9] : ? [X1] : (SDEP(X5,X9) & ~::(X9,sreg,X1) & ::(X9,idcnt,X1) &
      ::(X5,cnt,X1))) [input]
240. ! [X6] : (? [X1] : ::(X6,idcnt,X1) => ! [X1] : (EX(X6,X1) => ::(X6,idcnt,X1))) [input]
257. ! [X6,X16] : (::(X6,X16) <=> ? [X1] : ::(X6,X16,X1)) [input]
263. ! [X6] : (ED(X6) <=> (~::(X6,sdcnt) & ~::(X6,sreg) & ::(X6,cnt))) [input]
264. ! [X6] : (PED(X6) <=> (~::(X6,sreg) & ::(X6,idcnt))) [input]
277. ! [X6,X7,X1] : (tP(X6,X7,X1) <=> (cP(X6,X7,X1) & ED(X7) & ED(X6))) [input]
292. ! [X6] : (PED(X6) <=> (! [X7] : ! [X1] : (tP(X7,X6,X1) => PED(X7)) & ? [X7] : ? [X1] : tP(X7,X6,X1))) [input]
293. ~! [X6] : (PED(X6) <=> (! [X7] : ! [X1] : (tP(X7,X6,X1) => PED(X7)) & ? [X7] : ? [X1] : tP(X7,X6,X1))) [
      negated conjecture 292]
```

```
294. ! [X0,X1] : (::(X0,idcnt,X1) => cP(X0,X0,X1)) [flattening 1]
313. ! [X0] : ! [X1] : ! [X2] : (cP(X0,X1,X2) => (::(X0,idcnt,X2) <=> ::(X1,idcnt,X2))) [rectify 11]
314. ! [X0,X1,X2] : (cP(X0,X1,X2) => (::(X0,idcnt,X2) <=> ::(X1,idcnt,X2))) [flattening 313]
317. ! [X0] : ! [X1] : ! [X2] : (cP(X0,X1,X2) => (::(X2,treg,X2) & ::(X1,cnt,X2) & ::(X0,cnt,X2))) [rectify 13]
318. ! [X0,X1,X2] : (cP(X0,X1,X2) => (::(X2,treg,X2) & ::(X1,cnt,X2) & ::(X0,cnt,X2))) [flattening 317]
352. ! [X0] : ! [X1] : ! [X2] : (::(X0,X1,X2) => (::(X2,treg,X2) & UNI(X1) & PAR(X0))) [rectify 31]
353. ! [X0,X1,X2] : (::(X0,X1,X2) => (::(X2,treg,X2) & UNI(X1) & PAR(X0))) [flattening 352]
357. ! [X0] : ! [X1] : (? [X2] : (::(X1,treg,X1) & ::(X0,X2,X1) & UNI(X2)) <=> (EX(X0,X1) & ::(X1,treg,X1) & PAR(
     X0))) [rectify 34]
358. ! [X0,X1] : (? [X2] : (::(X1,treg,X1) & ::(X0,X2,X1) & UNI(X2)) <=> (EX(X0,X1) & ::(X1,treg,X1) & PAR(X0))) [
     flattening 357]
395. ! [X0] : (? [X1] : ::(X0,idcnt,X1) <=> (~? [X2] : ? [X3] : (GDEP(X0,X2,X3) | SDEP(X0,X2)) & ? [X4] : ::(X0,
     cnt,X4))) [rectify 54]
396. ! [X0] : (? [X1] : ::(X0,idcnt,X1) <=> (~? [X2,X3] : (GDEP(X0,X2,X3) | SDEP(X0,X2)) & ? [X4] : ::(X0,cnt,X4))
     ) [flattening 395]
563. ! [X0] : (? [X1] : ::(X0,sdcnt,X1) <=> ? [X2] : ? [X3] : (SDEP(X0,X2) & ~::(X2,sreg,X3) & ::(X2,idcnt,X3) &
     ::(X0,cnt,X3))) [rectify 142]
564. ! [X0] : (? [X1] : ::(X0,sdcnt,X1) <=> ? [X2,X3] : (SDEP(X0,X2) & ~::(X2,sreg,X3) & ::(X2,idcnt,X3) & ::(X0,
     cnt,X3))) [flattening 563]
685. ! [X0] : (? [X1] : ::(X0,idcnt,X1) => ! [X2] : (EX(X0,X2) => ::(X0,idcnt,X2))) [rectify 240]
705. ! [X0,X1] : (::(X0,X1) <=> ? [X2] : ::(X0,X1,X2)) [rectify 257]
712. ! [X0] : (ED(X0) <=> (~::(X0,sdcnt) & ~::(X0,sreg) & ::(X0,cnt))) [rectify 263]
713. ! [X0] : (PED(X0) <=> (~::(X0,sreg) & ::(X0,idcnt))) [rectify 264]
726. ! [X0,X1,X2] : (tP(X0,X1,X2) <=> (cP(X0,X1,X2) & ED(X1) & ED(X0))) [rectify 277]
741. ~! [X0] : (PED(X0) <=> (! [X1] : ! [X2] : (tP(X1,X0,X2) => PED(X1)) & ? [X3] : ? [X4] : tP(X3,X0,X4))) [
     rectify 293]
742. ~! [X0] : (PED(X0) <=> (! [X1,X2] : (tP(X1,X0,X2) => PED(X1)) & ? [X3,X4] : tP(X3,X0,X4))) [flattening 741]
748. ! [X0,X1] : (cP(X0,X0,X1) | ~::(X0,idcnt,X1)) [ennf transformation 294]
764. ! [X0,X1,X2] : (((::(X0,idcnt,X2) <=> ::(X1,idcnt,X2)) | ~cP(X0,X1,X2)) [ennf transformation 314]
767. ! [X0,X1,X2] : (((::(X2,treg,X2) & ::(X1,cnt,X2) & ::(X0,cnt,X2)) | ~cP(X0,X1,X2)) [ennf transformation 318]
797. ! [X0,X1,X2] : (((::(X2,treg,X2) & UNI(X1) & PAR(X0)) | ~::(X0,X1,X2)) [ennf transformation 353]
824. ! [X0] : (? [X1] : ::(X0,idcnt,X1) <=> (! [X2,X3] : (~GDEP(X0,X2,X3) & ~SDEP(X0,X2)) & ? [X4] : ::(X0,cnt,X4)
     )) [ennf transformation 396]
1029. ! [X0] : (! [X2] : (::(X0,idcnt,X2) | ~EX(X0,X2)) | ! [X1] : ~::(X0,idcnt,X1)) [ennf transformation 685]
1042. ? [X0] : (PED(X0) <~> (! [X1,X2] : (PED(X1) | ~tP(X1,X0,X2)) & ? [X3,X4] : tP(X3,X0,X4))) [ennf
     transformation 742]
1061. ! [X0,X1,X2] : (((::(X0,idcnt,X2) | ~::(X1,idcnt,X2)) & (::(X1,idcnt,X2) | ~::(X0,idcnt,X2))) | ~cP(X0,X1,X2
     )) [nnf transformation 764]
1084. ! [X0,X1] : ((? [X2] : (::(X1,treg,X1) & ::(X0,X2,X1) & UNI(X2)) | (~EX(X0,X1) | ~::(X1,treg,X1) | ~PAR(X0))
     ) & ((EX(X0,X1) & ::(X1,treg,X1) & PAR(X0)) | ! [X2] : (~::(X1,treg,X1) | ~::(X0,X2,X1) | ~UNI(X2)))) [nnf
     transformation 358]
1085. ! [X0,X1] : ((? [X2] : (::(X1,treg,X1) & ::(X0,X2,X1) & UNI(X2)) | ~EX(X0,X1) | ~::(X1,treg,X1) | ~PAR(X0))
     & ((EX(X0,X1) & ::(X1,treg,X1) & PAR(X0)) | ! [X2] : (~::(X1,treg,X1) | ~::(X0,X2,X1) | ~UNI(X2)))) [
     flattening 1084]
1086. ! [X0,X1] : ((? [X2] : (::(X1,treg,X1) & ::(X0,X2,X1) & UNI(X2)) | ~EX(X0,X1) | ~::(X1,treg,X1) | ~PAR(X0))
     & ((EX(X0,X1) & ::(X1,treg,X1) & PAR(X0)) | ! [X3] : (~::(X1,treg,X1) | ~::(X0,X3,X1) | ~UNI(X3)))) [rectify
     1085]
1087. ! [X1,X0] : (? [X2] : (::(X1,treg,X1) & ::(X0,X2,X1) & UNI(X2)) => (::(X1,treg,X1) & ::(X0,sK19(X0,X1),X1) &
     UNI(sK19(X0,X1)))) [choice axiom]
1088. ! [X0,X1] : (((::(X1,treg,X1) & ::(X0,sK19(X0,X1),X1) & UNI(sK19(X0,X1))) | ~EX(X0,X1) | ~::(X1,treg,X1) | ~
     PAR(X0)) & ((EX(X0,X1) & ::(X1,treg,X1) & PAR(X0)) | ! [X3] : (~::(X1,treg,X1) | ~::(X0,X3,X1) | ~UNI(X3)))
     ) [skolemisation 1086,1087]
1118. ! [X0] : ((? [X1] : ::(X0,idcnt,X1) | (? [X2,X3] : (GDEP(X0,X2,X3) | SDEP(X0,X2)) | ! [X4] : ~::(X0,cnt,X4))
     ) & (! [X2,X3] : (~GDEP(X0,X2,X3) & ~SDEP(X0,X2)) & ? [X4] : ::(X0,cnt,X4)) | ! [X1] : ~::(X0,idcnt,X1))) [
     nnf transformation 824]
1119. ! [X0] : ((? [X1] : ::(X0,idcnt,X1) | ? [X2,X3] : (GDEP(X0,X2,X3) | SDEP(X0,X2)) | ! [X4] : ~::(X0,cnt,X4))
     & ((! [X2,X3] : (~GDEP(X0,X2,X3) & ~SDEP(X0,X2)) & ? [X4] : ::(X0,cnt,X4)) | ! [X1] : ~::(X0,idcnt,X1))) [
     flattening 1118]
1120. ! [X0] : ((? [X1] : ::(X0,idcnt,X1) | ? [X2,X3] : (GDEP(X0,X2,X3) | SDEP(X0,X2)) | ! [X4] : ~::(X0,cnt,X4))
     & ((! [X5,X6] : (~GDEP(X0,X5,X6) & ~SDEP(X0,X5)) & ? [X7] : ::(X0,cnt,X7)) | ! [X8] : ~::(X0,idcnt,X8))) [
     rectify 1119]
1121. ! [X0] : (? [X1] : ::(X0,idcnt,X1) => ::(X0,idcnt,sK34(X0))) [choice axiom]
1122. ! [X0] : (? [X2,X3] : (GDEP(X0,X2,X3) | SDEP(X0,X2)) => (GDEP(X0,sK35(X0),sK36(X0)) | SDEP(X0,sK35(X0)))) [
     choice axiom]
1123. ! [X0] : (? [X7] : ::(X0,cnt,X7) => ::(X0,cnt,sK37(X0))) [choice axiom]
1124. ! [X0] : ((::(X0,idcnt,sK34(X0)) | (GDEP(X0,sK35(X0),sK36(X0)) | SDEP(X0,sK35(X0))) | ! [X4] : ~::(X0,cnt,X4
     )) & ((! [X5,X6] : (~GDEP(X0,X5,X6) & ~SDEP(X0,X5)) & ::(X0,cnt,sK37(X0))) | ! [X8] : ~::(X0,idcnt,X8))) [
     skolemisation 1120,1123,1122,1121]
1318. ! [X0] : ((? [X1] : ::(X0,sdcnt,X1) | ! [X2,X3] : (~SDEP(X0,X2) | ::(X2,sreg,X3) | ~::(X2,idcnt,X3) | ~::(X0
     ,cnt,X3))) & (? [X2,X3] : (SDEP(X0,X2) & ~::(X2,sreg,X3) & ::(X2,idcnt,X3) & ::(X0,cnt,X3)) | ! [X1] : ~::(
     X0,sdcnt,X1))) [nnf transformation 564]
1319. ! [X0] : ((? [X1] : ::(X0,sdcnt,X1) | ! [X2,X3] : (~SDEP(X0,X2) | ::(X2,sreg,X3) | ~::(X2,idcnt,X3) | ~::(X0
     ,cnt,X3))) & (? [X4,X5] : (SDEP(X0,X4) & ~::(X4,sreg,X5) & ::(X4,idcnt,X5) & ::(X0,cnt,X5)) | ! [X6] : ~::(
     X0,sdcnt,X6))) [rectify 1318]
1320. ! [X0] : (? [X1] : ::(X0,sdcnt,X1) => ::(X0,sdcnt,sK136(X0))) [choice axiom]
1321. ! [X0] : (? [X4,X5] : (SDEP(X0,X4) & ~::(X4,sreg,X5) & ::(X4,idcnt,X5) & ::(X0,cnt,X5)) => (SDEP(X0,sK137(X0
     )) & ~::(sK137(X0),sreg,sK138(X0)) & ::(sK137(X0),idcnt,sK138(X0)) & ::(X0,cnt,sK138(X0)))) [choice axiom]
1322. ! [X0] : (((::(X0,sdcnt,sK136(X0)) | ! [X2,X3] : (~SDEP(X0,X2) | ::(X2,sreg,X3) | ~::(X2,idcnt,X3) | ~::(X0,
     cnt,X3))) & ((SDEP(X0,sK137(X0)) & ~::(sK137(X0),sreg,sK138(X0)) & ::(sK137(X0),idcnt,sK138(X0)) & ::(X0,cnt
     ,sK138(X0))) | ! [X6] : ~::(X0,sdcnt,X6))) [skolemisation 1319,1321,1320]
1371. ! [X0] : (! [X1] : (::(X0,idcnt,X1) | ~EX(X0,X1)) | ! [X2] : ~::(X0,idcnt,X2)) [rectify 1029]
1383. ! [X0,X1] : (((::(X0,X1) | ! [X2] : ~::(X0,X1,X2)) & (? [X2] : ::(X0,X1,X2) | ~::(X0,X1))) [nnf
     transformation 705]
1384. ! [X0,X1] : (((::(X0,X1) | ! [X2] : ~::(X0,X1,X2)) & (? [X3] : ::(X0,X1,X3) | ~::(X0,X1))) [rectify 1383]
1385. ! [X1,X0] : (? [X3] : ::(X0,X1,X3) => ::(X0,X1,sK158(X0,X1))) [choice axiom]
1386. ! [X0,X1] : (((::(X0,X1) | ! [X2] : ~::(X0,X1,X2)) & (::(X0,X1,sK158(X0,X1)) | ~::(X0,X1))) [skolemisation
     1384,1385]
1389. ! [X0] : ((ED(X0) | (::(X0,sdcnt) | ::(X0,sreg) | ~::(X0,cnt))) & ((~::(X0,sdcnt) & ~::(X0,sreg) & ::(X0,cnt
     )) | ~ED(X0))) [nnf transformation 712]
1390. ! [X0] : ((ED(X0) | ::(X0,sdcnt) | ::(X0,sreg) | ~::(X0,cnt)) & ((~::(X0,sdcnt) & ~::(X0,sreg) & ::(X0,cnt))
     | ~ED(X0))) [flattening 1389]
```

```
1391. ! [X0] : ((PED(X0) | (::(X0,sreg) | ~::(X0,idcnt))) & ((~::(X0,sreg) & ::(X0,idcnt)) | ~PED(X0))) [nnf
      transformation 713]
1392. ! [X0] : ((PED(X0) | ::(X0,sreg) | ~::(X0,idcnt)) & ((~::(X0,sreg) & ::(X0,idcnt)) | ~PED(X0))) [flattening
      1391]
1393. ! [X0,X1,X2] : (((tP(X0,X1,X2) | (~cP(X0,X1,X2) | ~ED(X1) | ~ED(X0))) & ((cP(X0,X1,X2) & ED(X1) & ED(X0)) | ~
      tP(X0,X1,X2))) [nnf transformation 726]
1394. ! [X0,X1,X2] : (((tP(X0,X1,X2) | ~cP(X0,X1,X2) | ~ED(X1) | ~ED(X0)) & ((cP(X0,X1,X2) & ED(X1) & ED(X0)) | ~tP
      (X0,X1,X2))) [flattening 1393]
1395. ? [X0] : (((? [X1,X2] : (~PED(X1) & tP(X1,X0,X2)) | ! [X3,X4] : ~tP(X3,X0,X4)) | ~PED(X0)) & ((! [X1,X2] : (
      PED(X1) | ~tP(X1,X0,X2)) & ? [X3,X4] : tP(X3,X0,X4)) | PED(X0))) [nnf transformation 1042]
1396. ? [X0] : ((? [X1,X2] : (~PED(X1) & tP(X1,X0,X2)) | ! [X3,X4] : ~tP(X3,X0,X4) | ~PED(X0)) & ((! [X1,X2] : (
      PED(X1) | ~tP(X1,X0,X2)) & ? [X3,X4] : tP(X3,X0,X4)) | PED(X0))) [flattening 1395]
1397. ? [X0] : ((? [X1,X2] : (~PED(X1) & tP(X1,X0,X2)) | ! [X3,X4] : ~tP(X3,X0,X4) | ~PED(X0)) & ((! [X5,X6] : (
      PED(X5) | ~tP(X5,X0,X6)) & ? [X7,X8] : tP(X7,X0,X8)) | PED(X0))) [rectify 1396]
1398. ? [X0] : ((? [X1,X2] : (~PED(X1) & tP(X1,X0,X2)) | ! [X3,X4] : ~tP(X3,X0,X4) | ~PED(X0)) & ((! [X5,X6] : (
      PED(X5) | ~tP(X5,X0,X6)) & ? [X7,X8] : tP(X7,X0,X8)) | PED(X0))) => ((? [X2,X1] : (~PED(X1) & tP(X1,sK159,X2
      )) | ! [X4,X3] : ~tP(X3,sK159,X4) | ~PED(sK159)) & ((! [X6,X5] : (PED(X5) | ~tP(X5,sK159,X6)) & ? [X8,X7] :
      tP(X7,sK159,X8)) | PED(sK159))) [choice axiom]
1399. ? [X2,X1] : (~PED(X1) & tP(X1,sK159,X2)) => (~PED(sK160) & tP(sK160,sK159,sK161)) [choice axiom]
1400. ? [X8,X7] : tP(X7,sK159,X8) => tP(sK162,sK159,sK163) [choice axiom]
1401. ((~PED(sK160) & tP(sK160,sK159,sK161)) | ! [X3,X4] : ~tP(X3,sK159,X4) | ~PED(sK159)) & ((! [X5,X6] : (PED(X5
      ) | ~tP(X5,sK159,X6)) & tP(sK162,sK159,sK163)) | PED(sK159)) [skolemisation 1397,1400,1399,1398]
1402. ~::(X0,idcnt,X1) | cP(X0,X0,X1) [cnf transformation 748]
1415. ~cP(X0,X1,X2) | ~::(X0,idcnt,X2) | ::(X1,idcnt,X2) [cnf transformation 1061]
1416. ~cP(X0,X1,X2) | ~::(X1,idcnt,X2) | ::(X0,idcnt,X2) [cnf transformation 1061]
1418. ~cP(X0,X1,X2) | ::(X0,cnt,X2) [cnf transformation 767]
1419. ~cP(X0,X1,X2) | ::(X1,cnt,X2) [cnf transformation 767]
1452. ~::(X0,X1,X2) | UNI(X1) [cnf transformation 797]
1453. ~::(X0,X1,X2) | ::(X2,treg,X2) [cnf transformation 797]
1462. EX(X0,X1) | ~::(X1,treg,X1) | ~::(X0,X3,X1) | ~UNI(X3) [cnf transformation 1088]
1511. ~::(X0,idcnt,X8) | ::(X0,cnt,sK37(X0)) [cnf transformation 1124]
1512. ~::(X0,idcnt,X8) | ~SDEP(X0,X5) [cnf transformation 1124]
1759. ~::(X0,sdcnt,X6) | SDEP(X0,sK137(X0)) [cnf transformation 1322]
1881. ~::(X0,idcnt,X2) | ~EX(X0,X1) | ::(X0,idcnt,X1) [cnf transformation 1371]
2292. ::(X0,X1,sK158(X0,X1)) | ~::(X0,X1) [cnf transformation 1386]
2293. ~::(X0,X1,X2) | ::(X0,X1) [cnf transformation 1386]
2298. ~::(X0,sreg) | ~ED(X0) [cnf transformation 1390]
2300. ~::(X0,cnt) | ::(X0,sdcnt) | ::(X0,sreg) | ED(X0) [cnf transformation 1390]
2301. ::(X0,idcnt) | ~PED(X0) [cnf transformation 1392]
2302. ~::(X0,sreg) | ~PED(X0) [cnf transformation 1392]
2303. ~::(X0,idcnt) | ::(X0,sreg) | PED(X0) [cnf transformation 1392]
2304. ~tP(X0,X1,X2) | ED(X0) [cnf transformation 1394]
2305. ~tP(X0,X1,X2) | ED(X1) [cnf transformation 1394]
2306. ~tP(X0,X1,X2) | cP(X0,X1,X2) [cnf transformation 1394]
2307. ~cP(X0,X1,X2) | tP(X0,X1,X2) | ~ED(X1) | ~ED(X0) [cnf transformation 1394]
2308. tP(sK162,sK159,sK163) | PED(sK159) [cnf transformation 1401]
2309. PED(X5) | ~tP(X5,sK159,X6) | PED(sK159) [cnf transformation 1401]
2310. tP(sK160,sK159,sK161) | ~tP(X3,sK159,X4) | ~PED(sK159) [cnf transformation 1401]
2311. ~PED(sK160) | ~tP(X3,sK159,X4) | ~PED(sK159) [cnf transformation 1401]
2350. EX(X0,X1) | ~::(X1,treg,X1) | ~::(X0,X3,X1) [subsumption resolution 1462,1452]
2351. ~::(X0,X3,X1) | EX(X0,X1) [subsumption resolution 2350,1453]
2360. 1 <=> PED(sK159) [avatar definition]
2361. PED(sK159) <- (1) [avatar component clause 2360]
2362. ~PED(sK159) <- (~1) [avatar component clause 2360]
2364. 2 <=> ! [X3,X4] : ~tP(X3,sK159,X4) [avatar definition]
2365. ~tP(X3,sK159,X4) <- (2) [avatar component clause 2364]
2367. 3 <=> PED(sK160) [avatar definition]
2369. ~PED(sK160) <- (~3) [avatar component clause 2367]
2370. ~1 | 2 | ~3 [avatar split clause 2311,2367,2364,2360]
2372. 4 <=> tP(sK160,sK159,sK161) [avatar definition]
2374. tP(sK160,sK159,sK161) <- (4) [avatar component clause 2372]
2375. ~1 | 2 | 4 [avatar split clause 2310,2372,2364,2360]
2377. 5 <=> ! [X5,X6] : (PED(X5) | ~tP(X5,sK159,X6)) [avatar definition]
2378. ~tP(X5,sK159,X6) | PED(X5) <- (5) [avatar component clause 2377]
2379. 1 | 5 [avatar split clause 2309,2377,2360]
2381. 6 <=> tP(sK162,sK159,sK163) [avatar definition]
2383. tP(sK162,sK159,sK163) <- (6) [avatar component clause 2381]
2384. 1 | 6 [avatar split clause 2308,2381,2360]
3219. ::(X10,cnt,sK37(X10)) | ~::(X10,idcnt) [resolution 2292,1511]
3221. cP(X12,X12,sK158(X12,idcnt)) | ~::(X12,idcnt) [resolution 2292,1402]
3222. ~::(X13,idcnt) | ~SDEP(X13,X14) [resolution 2292,1512]
3256. SDEP(X48,sK137(X48)) | ~::(X48,sdcnt) [resolution 2292,1759]
3865. ::(X6,idcnt,X7) | ~EX(X6,X7) | ~::(X6,idcnt) [resolution 1881,2292]
4181. ~SDEP(X0,X1) | ~PED(X0) [resolution 3222,2301]
6233. ~::(X10,sdcnt) | ~PED(X10) [resolution 3256,4181]
6601. ~::(X7,idcnt) | ::(X7,cnt) [resolution 3219,2293]
7329. ~::(X32,idcnt) | tP(X32,X32,sK158(X32,idcnt)) | ~ED(X32) | ~ED(X32) [resolution 3221,2307]
7335. tP(X32,X32,sK158(X32,idcnt)) | ~::(X32,idcnt) | ~ED(X32) [duplicate literal removal 7329]
16317. 516 <=> ED(sK159) [avatar definition]
16318. ED(sK159) <- (516) [avatar component clause 16317]
16319. ~ED(sK159) <- (~516) [avatar component clause 16317]
16830. ::(X0,cnt) | ~PED(X0) [resolution 6601,2301]
16962. ~PED(X0) | ::(X0,sdcnt) | ::(X0,sreg) | ED(X0) [resolution 16830,2300]
16970. ~PED(X0) | ::(X0,sreg) | ED(X0) [subsumption resolution 16962,6233]
16971. ~PED(X0) | ED(X0) [subsumption resolution 16970,2302]
17006. ED(sK159) <- (1) [resolution 16971,2361]
17011. $false <- (1, ~516) [subsumption resolution 17006,16319]
17012. ~1 | 516 [avatar contradiction clause 17011]
17015. PED(sK162) <- (5, 6) [resolution 2383,2378]
17016. cP(sK162,sK159,sK163) <- (6) [resolution 2383,2306]
17017. ED(sK159) <- (6) [resolution 2383,2305]
```

```
17021. 516 | ~6 [avatar split clause 17017,2381,16317]
17034. ~::(sK162,idcnt,sK163) | ::(sK159,idcnt,sK163) <- (6) [resolution 17016,1415]
17037. ::(sK162,cnt,sK163) <- (6) [resolution 17016,1418]
17062. 519 <=> EX(sK162,sK163) [avatar definition]
17064. EX(sK162,sK163) <- (519) [avatar component clause 17062]
17118. 531 <=> ::(sK159,idcnt,sK163) [avatar definition]
17120. ::(sK159,idcnt,sK163) <- (531) [avatar component clause 17118]
17122. 532 <=> ::(sK162,idcnt,sK163) [avatar definition]
17124. ~::(sK162,idcnt,sK163) <- (~532) [avatar component clause 17122]
17125. 531 | ~532 | ~6 [avatar split clause 17034,2381,17122,17118]
17252. EX(sK162,sK163) <- (6) [resolution 17037,2351]
17267. 519 | ~6 [avatar split clause 17252,2381,17062]
17760. ~EX(sK162,sK163) | ~::(sK162,idcnt) <- (~532) [resolution 17124,3865]
17761. ~::(sK162,idcnt) <- (519, ~532) [subsumption resolution 17760,17064]
17778. ~PED(sK162) <- (519, ~532) [resolution 17761,2301]
17779. $false <- (5, 6, 519, ~532) [subsumption resolution 17778,17015]
17780. ~5 | ~6 | ~519 | 532 [avatar contradiction clause 17779]
17804. ::(sK159,idcnt) <- (531) [resolution 17120,2293]
18078. ::(sK159,sreg) | PED(sK159) <- (531) [resolution 17804,2303]
18085. ::(sK159,sreg) <- (~1, 531) [subsumption resolution 18078,2362]
18157. ~ED(sK159) <- (~1, 531) [resolution 18085,2298]
18160. $false <- (~1, 516, 531) [subsumption resolution 18157,16318]
18161. 1 | ~516 | ~531 [avatar contradiction clause 18160]
18174. cP(sK160,sK159,sK161) <- (4) [resolution 2374,2306]
18176. ED(sK160) <- (4) [resolution 2374,2304]
18193. ::(sK159,cnt,sK161) <- (4) [resolution 18174,1419]
18213. 567 <=> EX(sK159,sK161) [avatar definition]
18214. EX(sK159,sK161) <- (567) [avatar component clause 18213]
18273. 580 <=> ::(sK159,idcnt,sK161) [avatar definition]
18274. ~::(sK159,idcnt,sK161) <- (~580) [avatar component clause 18273]
18277. 581 <=> ::(sK160,idcnt,sK161) [avatar definition]
18278. ::(sK160,idcnt,sK161) <- (581) [avatar component clause 18277]
19139. EX(sK159,sK161) <- (4) [resolution 18193,2351]
19143. 567 | ~4 [avatar split clause 19139,2372,18213]
19898. ~::(sK159,idcnt,sK161) | ::(sK160,idcnt,sK161) <- (4) [resolution 18174,1416]
20086. ::(sK160,idcnt) <- (581) [resolution 18278,2293]
20142. ::(sK160,sreg) | PED(sK160) <- (581) [resolution 20086,2303]
20164. ::(sK160,sreg) <- (~3, 581) [subsumption resolution 20142,2369]
20221. ~ED(sK160) <- (~3, 581) [resolution 20164,2298]
20224. $false <- (~3, 4, 581) [subsumption resolution 20221,18176]
20225. 3 | ~4 | ~581 [avatar contradiction clause 20224]
20230. 581 | ~580 | ~4 [avatar split clause 19898,2372,18273,18277]
22263. ~EX(sK159,sK161) | ~::(sK159,idcnt) <- (~580) [resolution 18274,3865]
22264. ~::(sK159,idcnt) <- (567, ~580) [subsumption resolution 22263,18214]
22282. ~PED(sK159) <- (567, ~580) [resolution 22264,2301]
22283. $false <- (1, 567, ~580) [subsumption resolution 22282,2361]
22284. ~1 | ~567 | 580 [avatar contradiction clause 22283]
24134. 692 <=> ::(sK159,idcnt) [avatar definition]
24135. ::(sK159,idcnt) <- (692) [avatar component clause 24134]
24136. ~::(sK159,idcnt) <- (~692) [avatar component clause 24134]
24147. ~PED(sK159) <- (~692) [resolution 24136,2301]
24148. $false <- (1, ~692) [subsumption resolution 24147,2361]
24149. ~1 | 692 [avatar contradiction clause 24148]
54369. ~::(sK159,idcnt) | ~ED(sK159) <- (2) [resolution 7335,2365]
54370. ~ED(sK159) <- (2, 692) [subsumption resolution 54369,24135]
54371. $false <- (2, 516, 692) [subsumption resolution 54370,16318]
54372. ~2 | ~516 | ~692 [avatar contradiction clause 54371]
54373. $false [avatar sat refutation
       2370,2375,2379,2384,17012,17021,17125,17267,17780,18161,19143,20225,20230,22284,24149,54372]
```

Proof of Theorem (t_{bd}27)

```
83. ! [X0] : ! [X8] : ! [X1] : (PTC(X0,X8,X1) => (::(X1,treg,X1) & ::(X8,proc,X1) & (::(X0,gdcnt,X1) | ::(X0,sdcnt
      ,X1) | (~::(X0,sreg,X1) & ::(X0,idcnt,X1))))) [input]
257. ! [X6,X16] : (::(X6,X16) <=> ? [X1] : ::(X6,X16,X1)) [input]
269. ! [X6] : (T(X6) <=> ::(X6,treg)) [input]
278. ! [X6,X7,X1] : (PC(X6,X7,X1) <=> (? [X13] : (PTC(X6,X13,X1) & tmP(X7,X13)) & PD(X7) & ED(X6))) [input]
292. ! [X6,X7,X1] : (PC(X6,X7,X1) => (T(X1) & PD(X7) & ED(X6))) [input]
293. ~! [X6,X7,X1] : (PC(X6,X7,X1) => (T(X1) & PD(X7) & ED(X6))) [negated conjecture 292]
450. ! [X0] : ! [X1] : ! [X2] : (PTC(X0,X1,X2) => (::(X2,treg,X2) & ::(X1,proc,X2) & (::(X0,gdcnt,X2) | ::(X0,
      sdcnt,X2) | (~::(X0,sreg,X2) & ::(X0,idcnt,X2))))) [rectify 83]
451. ! [X0,X1,X2] : (PTC(X0,X1,X2) => (::(X2,treg,X2) & ::(X1,proc,X2) & (::(X0,gdcnt,X2) | ::(X0,sdcnt,X2) |
      (~::(X0,sreg,X2) & ::(X0,idcnt,X2))))) [flattening 450]
705. ! [X0,X1] : (::(X0,X1) <=> ? [X2] : ::(X0,X1,X2)) [rectify 257]
718. ! [X0] : (T(X0) <=> ::(X0,treg)) [rectify 269]
727. ! [X0,X1,X2] : (PC(X0,X1,X2) <=> (? [X3] : (PTC(X0,X3,X2) & tmP(X1,X3)) & PD(X1) & ED(X0))) [rectify 278]
741. ~! [X0,X1,X2] : (PC(X0,X1,X2) => (T(X2) & PD(X1) & ED(X0))) [rectify 293]
742. ! [X0] : (::(X0,treg) => T(X0)) [unused predicate definition removal 718]
743. ! [X0,X1,X2] : (PC(X0,X1,X2) => (? [X3] : (PTC(X0,X3,X2) & tmP(X1,X3)) & PD(X1) & ED(X0))) [unused predicate
      definition removal 727]
869. ! [X0,X1,X2] : ((::(X2,treg,X2) & ::(X1,proc,X2) & (::(X0,gdcnt,X2) | ::(X0,sdcnt,X2) | (~::(X0,sreg,X2) &
      ::(X0,idcnt,X2)))) | ~PTC(X0,X1,X2)) [ennf transformation 451]
1043. ! [X0] : (T(X0) | ~::(X0,treg)) [ennf transformation 742]
1044. ! [X0,X1,X2] : ((? [X3] : (PTC(X0,X3,X2) & tmP(X1,X3)) & PD(X1) & ED(X0)) | ~PC(X0,X1,X2)) [ennf
      transformation 743]
1045. ? [X0,X1,X2] : ((~T(X2) | ~PD(X1) | ~ED(X0)) & PC(X0,X1,X2)) [ennf transformation 741]
1386. ! [X0,X1] : ((::(X0,X1) | ! [X2] : ~::(X0,X1,X2)) & (? [X2] : ::(X0,X1,X2) | ~::(X0,X1))) [nnf
      transformation 705]
1387. ! [X0,X1] : ((::(X0,X1) | ! [X2] : ~::(X0,X1,X2)) & (? [X3] : ::(X0,X1,X3) | ~::(X0,X1))) [rectify 1386]
1388. ! [X1,X0] : (? [X3] : ::(X0,X1,X3) => ::(X0,X1,sK158(X0,X1))) [choice axiom]
```

```
1389. ! [X0,X1] : ((::(X0,X1) | ! [X2] : ~::(X0,X1,X2)) & (::(X0,X1,sK158(X0,X1)) | ~::(X0,X1))) [skolemisation
      1387,1388]
1396. ! [X2,X1,X0] : (? [X3] : (PTC(X0,X3,X2) & tmP(X1,X3)) => (PTC(X0,sK159(X0,X1,X2),X2) & tmP(X1,sK159(X0,X1,X2
      )))) [choice axiom]
1397. ! [X0,X1,X2] : (((PTC(X0,sK159(X0,X1,X2),X2) & tmP(X1,sK159(X0,X1,X2))) & PD(X1) & ED(X0)) | ~PC(X0,X1,X2))
      [skolemisation 1044,1396]
1398. ? [X0,X1,X2] : ((~T(X2) | ~PD(X1) | ~ED(X0)) & PC(X0,X1,X2)) => ((~T(sK162) | ~PD(sK161) | ~ED(sK160)) & PC(
      sK160,sK161,sK162)) [choice axiom]
1399. (~T(sK162) | ~PD(sK161) | ~ED(sK160)) & PC(sK160,sK161,sK162) [skolemisation 1045,1398]
1604. ~PTC(X0,X1,X2) | ::(X2,treg,X2) [cnf transformation 869]
2291. ~::(X0,X1,X2) | ::(X0,X1) [cnf transformation 1389]
2302. ~::(X0,treg) | T(X0) [cnf transformation 1043]
2303. ~PC(X0,X1,X2) | ED(X0) [cnf transformation 1397]
2304. ~PC(X0,X1,X2) | PD(X1) [cnf transformation 1397]
2306. PTC(X0,sK159(X0,X1,X2),X2) | ~PC(X0,X1,X2) [cnf transformation 1397]
2307. PC(sK160,sK161,sK162) [cnf transformation 1399]
2308. ~T(sK162) | ~PD(sK161) | ~ED(sK160) [cnf transformation 1399]
2357. 1 <=> ED(sK160) [avatar definition]
2361. 2 <=> PD(sK161) [avatar definition]
2365. 3 <=> T(sK162) [avatar definition]
2367. ~T(sK162) <- (~3) [avatar component clause 2365]
2368. ~1 | ~2 | ~3 [avatar split clause 2308,2365,2361,2357]
2400. ED(sK160) [resolution 2303,2307]
2403. 1 [avatar split clause 2400,2357]
2404. PD(sK161) [resolution 2304,2307]
2407. 2 [avatar split clause 2404,2361]
3870. ~PC(X4,X5,X6) | ::(X6,treg,X6) [resolution 2306,1604]
7220. ::(sK162,treg,sK162) [resolution 3870,2307]
7248. ::(sK162,treg) [resolution 7220,2291]
7305. T(sK162) [resolution 7248,2302]
7308. $false <- (~3) [subsumption resolution 7305,2367]
7309. 3 [avatar contradiction clause 7308]
7310. $false [avatar sat refutation 2368,2403,2407,7309]
```

Proof of Theorem (t_{bd}29)

This went KO with both provers

Proof of Theorem (t_{bd}38)

This went KO with both provers

Proof of Theorem (t_{bd}41)

```
31. ! [X17] : ! [X16] : ! [X1] : (::(X17,X16,X1) => (::(X1,treg,X1) & UNI(X16) & PAR(X17))) [input]
34. ! [X0] : ! [X1] : (? [X16] : (::(X1,treg,X1) & ::(X0,X16,X1) & UNI(X16)) <=> (EX(X0,X1) & ::(X1,treg,X1) & PAR
    (X0))) [input]
53. ! [X17] : ! [X1] : (::(X17,imen,X1) <=> (~? [X11] : (cP(X11,X17,X1) & ::(X11,mten,X1)) & ::(X17,idcnt,X1))) [
    input]
54. ! [X28] : (? [X1] : ::(X28,idcnt,X1) <=> (~? [X29] : ? [X1] : (GDEP(X28,X29,X1) | SDEP(X28,X29)) & ? [X1] :
    ::(X28,cnt,X1))) [input]
101. ! [X0] : ! [X8] : ! [X1] : (SREG(X0,X8,X1) => (::(X1,treg,X1) & ::(X8,sreg,X1) & ~::(X0,sreg,X1) & ::(X0,
     idcnt,X1))) [input]
120. ! [X0] : ! [X8] : ! [X1] : (SPROJ(X0,X8,X1) => (::(X1,treg,X1) & ::(X8,sreg,X1) & ::(X0,streg,X1))) [input]
139. ! [X8] : (? [X1] : ::(X8,rqlt,X1) <=> (? [X1] : ::(X8,qlt,X1) & ? [X9] : ? [X31] : (SDEP(X8,X31) & INH(X8,X9)
     & X9 != X31))) [input]
141. ! [X0] : ! [X8] : (INH(X0,X8) <=> (? [X1] : (~::(X8,sreg,X1) & ::(X8,idcnt,X1) & ::(X0,sdcnt,X1)) & SDEP(X0,
     X8))) [input]
142. ! [X5] : (? [X1] : ::(X5,sdcnt,X1) <=> ? [X9] : ? [X1] : (SDEP(X5,X9) & ~::(X9,sreg,X1) & ::(X9,idcnt,X1) &
     ::(X5,cnt,X1))) [input]
214. ! [X1] : ! [X6] : (::(X6,sreg,X1) => ::(X6,imen,X1)) [input]
215. ! [X1] : ! [X6] : (::(X6,idcnt,X1) => ::(X6,cnt,X1)) [input]
222. ! [X1] : ! [X6] : (::(X6,gdcnt,X1) => ::(X6,cnt,X1)) [input]
235. ! [X6] : (? [X1] : ::(X6,sreg,X1) => ! [X1] : (EX(X6,X1) => ::(X6,sreg,X1))) [input]
245. ! [X6] : (? [X1] : ::(X6,gdcnt,X1) => ! [X1] : (EX(X6,X1) => ::(X6,gdcnt,X1))) [input]
250. ~? [X6] : ? [X1] : (::(X6,gdcnt,X1) & ::(X6,idcnt,X1)) & ~? [X6] : ? [X1] : (::(X6,gdcnt,X1) & ::(X6,sdcnt,X1
     )) & ~? [X6] : ? [X1] : (::(X6,idcnt,X1) & ::(X6,sdcnt,X1)) [input]
257. ! [X6,X16] : (::(X6,X16) <=> ? [X1] : ::(X6,X16,X1)) [input]
263. ! [X6] : (ED(X6) <=> (~::(X6,sdcnt) & ~::(X6,sreg) & ::(X6,cnt))) [input]
267. ! [X6] : (PQ(X6) <=> (~::(X6,rqlt) & ::(X6,sdcnt))) [input]
268. ! [X6] : (Q(X6) <=> PQ(X6)) [input]
271. ! [X6] : (S(X6) <=> ::(X6,sreg)) [input]
282. ! [X6,X5,X1] : (SLC(X6,X5,X1) <=> ((? [X7] : (SPROJ(X7,X5,X1) & STREG(X6,X7)) & PD(X6)) | (? [X7] : (SREG(X7,
     X5,X1) & ! [X13] : (INH(X6,X13) => cP(X13,X7,X1)) & INH(X6,X7)) & Q(X6)) | (? [X7] : (SREG(X7,X5,X1) & ! [
     X13] : (GDEP(X6,X13,X1) => cP(X13,X7,X1)) & GDEP(X6,X7,X1)) & ::(X6,gdcnt,X1)) | (SREG(X6,X5,X1) & ~::(X6,
     sreg,X1) & ::(X6,idcnt,X1)))) [input]
292. ! [X6,X5,X1] : (SLC(X6,X5,X1) => (S(X5) & (PD(X6) | PQ(X6) | ED(X6)))) [input]
293. ~! [X6,X5,X1] : (SLC(X6,X5,X1) => (S(X5) & (PD(X6) | PQ(X6) | ED(X6)))) [negated conjecture 292]
352. ! [X0] : ! [X1] : ! [X2] : (::(X0,X1,X2) => (::(X2,treg,X2) & UNI(X1) & PAR(X0))) [rectify 31]
353. ! [X0,X1,X2] : (::(X0,X1,X2) => (::(X2,treg,X2) & UNI(X1) & PAR(X0))) [flattening 352]
357. ! [X0] : ! [X1] : (? [X2] : (::(X1,treg,X1) & ::(X0,X2,X1) & UNI(X2)) <=> (EX(X0,X1) & ::(X1,treg,X1) & PAR(
     X0))) [rectify 34]
358. ! [X0,X1] : (? [X2] : (::(X1,treg,X1) & ::(X0,X2,X1) & UNI(X2)) <=> (EX(X0,X1) & ::(X1,treg,X1) & PAR(X0))) [
     flattening 357]
393. ! [X0] : ! [X1] : (::(X0,imen,X1) <=> (~? [X2] : (cP(X2,X0,X1) & ::(X2,mten,X1)) & ::(X0,idcnt,X1))) [rectify
     53]
394. ! [X0,X1] : (::(X0,imen,X1) <=> (~? [X2] : (cP(X2,X0,X1) & ::(X2,mten,X1)) & ::(X0,idcnt,X1))) [flattening
     393]
```

```
395. ! [X0] : (? [X1] : ::(X0,idcnt,X1) <=> (~? [X2] : ? [X3] : (GDEP(X0,X2,X3) | SDEP(X0,X2)) & ? [X4] : ::(X0,
     cnt,X4))) [rectify 54]
396. ! [X0] : (? [X1] : ::(X0,idcnt,X1) <=> (~? [X2,X3] : (GDEP(X0,X2,X3) | SDEP(X0,X2)) & ? [X4] : ::(X0,cnt,X4))
     ) [flattening 395]
486. ! [X0] : ! [X1] : ! [X2] : (SREG(X0,X1,X2) => (::(X2,treg,X2) & ::(X1,sreg,X2) & ~::(X0,sreg,X2) & ::(X0,
     idcnt,X2))) [rectify 101]
487. ! [X0,X1,X2] : (SREG(X0,X1,X2) => (::(X2,treg,X2) & ::(X1,sreg,X2) & ~::(X0,sreg,X2) & ::(X0,idcnt,X2))) [
     flattening 486]
521. ! [X0] : ! [X1] : ! [X2] : (SPROJ(X0,X1,X2) => (::(X2,treg,X2) & ::(X1,sreg,X2) & ::(X0,streg,X2))) [rectify
     120]
522. ! [X0,X1,X2] : (SPROJ(X0,X1,X2) => (::(X2,treg,X2) & ::(X1,sreg,X2) & ::(X0,streg,X2))) [flattening 521]
557. ! [X0] : (? [X1] : ::(X0,rqlt,X1) <=> (? [X2] : ::(X0,qlt,X2) & ? [X3] : ? [X4] : (SDEP(X0,X4) & INH(X0,X3) &
     X3 != X4))) [rectify 139]
558. ! [X0] : (? [X1] : ::(X0,rqlt,X1) <=> (? [X2] : ::(X0,qlt,X2) & ? [X3,X4] : (SDEP(X0,X4) & INH(X0,X3) & X3 !=
     X4))) [flattening 557]
561. ! [X0] : ! [X1] : (INH(X0,X1) <=> (? [X2] : (~::(X1,sreg,X2) & ::(X1,idcnt,X2) & ::(X0,sdcnt,X2)) & SDEP(X0,
     X1))) [rectify 141]
562. ! [X0,X1] : (INH(X0,X1) <=> (? [X2] : (~::(X1,sreg,X2) & ::(X1,idcnt,X2) & ::(X0,sdcnt,X2)) & SDEP(X0,X1))) [
     flattening 561]
563. ! [X0] : (? [X1] : ::(X0,sdcnt,X1) <=> ? [X2] : ? [X3] : (SDEP(X0,X2) & ~::(X2,sreg,X3) & ::(X2,idcnt,X3) &
     ::(X0,cnt,X3))) [rectify 142]
564. ! [X0] : (? [X1] : ::(X0,sdcnt,X1) <=> ? [X2,X3] : (SDEP(X0,X2) & ~::(X2,sreg,X3) & ::(X2,idcnt,X3) & ::(X0,
     cnt,X3))) [flattening 563]
642. ! [X0] : ! [X1] : (::(X1,sreg,X0) => ::(X1,imen,X0)) [rectify 214]
643. ! [X0,X1] : (::(X1,sreg,X0) => ::(X1,imen,X0)) [flattening 642]
644. ! [X0] : ! [X1] : (::(X1,idcnt,X0) => ::(X1,cnt,X0)) [rectify 215]
645. ! [X0,X1] : (::(X1,idcnt,X0) => ::(X1,cnt,X0)) [flattening 644]
658. ! [X0] : ! [X1] : (::(X1,gdcnt,X0) => ::(X1,cnt,X0)) [rectify 222]
659. ! [X0,X1] : (::(X1,gdcnt,X0) => ::(X1,cnt,X0)) [flattening 658]
680. ! [X0] : (? [X1] : ::(X0,sreg,X1) => ! [X2] : (EX(X0,X2) => ::(X0,sreg,X2))) [rectify 235]
690. ! [X0] : (? [X1] : ::(X0,gdcnt,X1) => ! [X2] : (EX(X0,X2) => ::(X0,gdcnt,X2))) [rectify 245]
696. ~? [X0] : ? [X1] : (::(X0,gdcnt,X1) & ::(X0,idcnt,X1)) & ~? [X2] : ? [X3] : (::(X2,gdcnt,X3) & ::(X2,sdcnt,X3
     )) & ~? [X4] : ? [X5] : (::(X4,idcnt,X5) & ::(X4,sdcnt,X5))) [rectify 250]
697. ~? [X0,X1] : (::(X0,gdcnt,X1) & ::(X0,idcnt,X1)) & ~? [X2,X3] : (::(X2,gdcnt,X3) & ::(X2,sdcnt,X3)) & ~? [X4,
     X5] : (::(X4,idcnt,X5) & ::(X4,sdcnt,X5)) [flattening 696]
705. ! [X0,X1] : (::(X0,X1) <=> ? [X2] : ::(X0,X1,X2)) [rectify 257]
712. ! [X0] : (ED(X0) <=> (~::(X0,sdcnt) & ~::(X0,sreg) & ::(X0,cnt))) [rectify 263]
716. ! [X0] : (PQ(X0) <=> (~::(X0,rqlt) & ::(X0,sdcnt))) [rectify 267]
717. ! [X0] : (Q(X0) <=> PQ(X0)) [rectify 268]
720. ! [X0] : (S(X0) <=> ::(X0,sreg)) [rectify 271]
731. ! [X0,X1,X2] : (SLC(X0,X1,X2) <=> ((? [X3] : (SPROJ(X3,X1,X2) & STREG(X0,X3)) & PD(X0)) | (? [X4] : (SREG(X4,
     X1,X2) & ! [X5] : (INH(X0,X5) => cP(X5,X4,X2)) & INH(X0,X4)) & Q(X0)) | (? [X6] : (SREG(X6,X1,X2) & ! [X7] :
     (GDEP(X0,X7,X2) => cP(X7,X6,X2)) & GDEP(X0,X6,X2)) & ::(X0,gdcnt,X2)) | (SREG(X0,X1,X2) & ~::(X0,sreg,X2) &
     ::(X0,idcnt,X2)))) [rectify 282]
741. ~! [X0,X1,X2] : (SLC(X0,X1,X2) => (S(X1) & (PD(X0) | PQ(X0) | ED(X0)))) [rectify 293]
742. ! [X0,X1,X2] : (SLC(X0,X1,X2) => ((? [X3] : (SPROJ(X3,X1,X2) & STREG(X0,X3)) & PD(X0)) | (? [X4] : (SREG(X4,
     X1,X2) & ! [X5] : (INH(X0,X5) => cP(X5,X4,X2)) & INH(X0,X4)) & Q(X0)) | (? [X6] : (SREG(X6,X1,X2) & ! [X7] :
     (GDEP(X0,X7,X2) => cP(X7,X6,X2)) & GDEP(X0,X6,X2)) & ::(X0,gdcnt,X2)) | (SREG(X0,X1,X2) & ~::(X0,sreg,X2) &
     ::(X0,idcnt,X2)))) [unused predicate definition removal 731]
743. ! [X0] : (::(X0,sreg) => S(X0)) [unused predicate definition removal 720]
744. ! [X0] : (Q(X0) => PQ(X0)) [unused predicate definition removal 717]
745. ! [X0] : ((~::(X0,sdcnt) & ~::(X0,sreg) & ::(X0,cnt)) => ED(X0)) [unused predicate definition removal 712]
800. ! [X0,X1,X2] : ((::(X2,treg,X2) & UNI(X1) & PAR(X0)) | ~::(X0,X1,X2)) [ennf transformation 353]
826. ! [X0,X1] : (::(X0,imen,X1) <=> (! [X2] : (~cP(X2,X0,X1) | ~::(X2,mten,X1)) & ::(X0,idcnt,X1))) [ennf
     transformation 394]
827. ! [X0] : (? [X1] : ::(X0,idcnt,X1) <=> (! [X2,X3] : (~GDEP(X0,X2,X3) & ~SDEP(X0,X2)) & ? [X4] : ::(X0,cnt,X4)
     )) [ennf transformation 396]
901. ! [X0,X1,X2] : (((::(X2,treg,X2) & ::(X1,sreg,X2) & ~::(X0,sreg,X2) & ::(X0,idcnt,X2)) | ~SREG(X0,X1,X2)) [
     ennf transformation 487]
931. ! [X0,X1,X2] : (((::(X2,treg,X2) & ::(X1,sreg,X2) & ::(X0,streg,X2)) | ~SPROJ(X0,X1,X2)) [ennf transformation
     522]
1007. ! [X0,X1] : (::(X1,imen,X0) | ~::(X1,sreg,X0)) [ennf transformation 643]
1008. ! [X0,X1] : (::(X1,cnt,X0) | ~::(X1,idcnt,X0)) [ennf transformation 645]
1015. ! [X0,X1] : (::(X1,cnt,X0) | ~::(X1,gdcnt,X0)) [ennf transformation 659]
1027. ! [X0] : (! [X2] : (::(X0,sreg,X2) | ~EX(X0,X2)) | ! [X1] : ~::(X0,sreg,X1)) [ennf transformation 680]
1037. ! [X0] : (! [X2] : (::(X0,gdcnt,X2) | ~EX(X0,X2)) | ! [X1] : ~::(X0,gdcnt,X1)) [ennf transformation 690]
1042. ! [X0,X1] : (~::(X0,gdcnt,X1) | ~::(X0,idcnt,X1)) & ! [X2,X3] : (~::(X2,gdcnt,X3) | ~::(X2,sdcnt,X3)) & ! [
     X4,X5] : (~::(X4,idcnt,X5) | ~::(X4,sdcnt,X5)) [ennf transformation 697]
1045. ! [X0] : (ED(X0) | (::(X0,sdcnt) | ::(X0,sreg) | ~::(X0,cnt))) [ennf transformation 745]
1046. ! [X0] : (ED(X0) | ::(X0,sdcnt) | ::(X0,sreg) | ~::(X0,cnt)) [flattening 1045]
1047. ! [X0] : (PQ(X0) | ~Q(X0)) [ennf transformation 744]
1048. ! [X0] : (S(X0) | ~::(X0,sreg)) [ennf transformation 743]
1049. ! [X0,X1,X2] : (((? [X3] : (SPROJ(X3,X1,X2) & STREG(X0,X3)) & PD(X0)) | (? [X4] : (SREG(X4,X1,X2) & ! [X5] :
     (cP(X5,X4,X2) | ~INH(X0,X5)) & INH(X0,X4)) & Q(X0)) | (? [X6] : (SREG(X6,X1,X2) & ! [X7] : (cP(X7,X6,X2) |
     ~GDEP(X0,X7,X2)) & GDEP(X0,X6,X2)) & ::(X0,gdcnt,X2)) | (SREG(X0,X1,X2) & ~::(X0,sreg,X2) & ::(X0,idcnt,X2))
     ) | ~SLC(X0,X1,X2)) [ennf transformation 742]
1050. ! [X0,X1,X2] : ((? [X3] : (SPROJ(X3,X1,X2) & STREG(X0,X3)) & PD(X0)) | (? [X4] : (SREG(X4,X1,X2) & ! [X5] :
     (cP(X5,X4,X2) | ~INH(X0,X5)) & INH(X0,X4)) & Q(X0)) | (? [X6] : (SREG(X6,X1,X2) & ! [X7] : (cP(X7,X6,X2) | ~
     GDEP(X0,X7,X2)) & GDEP(X0,X6,X2)) & ::(X0,gdcnt,X2)) | (SREG(X0,X1,X2) & ~::(X0,sreg,X2) & ::(X0,idcnt,X2))
     | ~SLC(X0,X1,X2)) [flattening 1049]
1051. ? [X0,X1,X2] : ((~S(X1) | (~PD(X0) & ~PQ(X0) & ~ED(X0))) & SLC(X0,X1,X2)) [ennf transformation 741]
1067. ! [X2,X1,X0] : ((? [X6] : (SREG(X6,X1,X2) & ! [X7] : (cP(X7,X6,X2) | ~GDEP(X0,X7,X2)) & GDEP(X0,X6,X2)) &
     ::(X0,gdcnt,X2)) | ~sP9(X2,X1,X0)) [predicate definition introduction]
1068. ! [X2,X1,X0] : ((? [X4] : (SREG(X4,X1,X2) & ! [X5] : (cP(X5,X4,X2) | ~INH(X0,X5)) & INH(X0,X4)) & Q(X0)) | ~
     sP10(X2,X1,X0)) [predicate definition introduction]
1069. ! [X2,X1,X0] : ((SREG(X0,X1,X2) & ~::(X0,sreg,X2) & ::(X0,idcnt,X2)) | ~sP11(X2,X1,X0)) [predicate
     definition introduction]
1070. ! [X0,X1,X2] : ((? [X3] : (SPROJ(X3,X1,X2) & STREG(X0,X3)) & PD(X0)) | sP10(X2,X1,X0) | sP9(X2,X1,X0) | sP11
     (X2,X1,X0) | ~SLC(X0,X1,X2)) [definition folding 1050,1069,1068,1067]
1097. ! [X0,X1] : ((? [X2] : (::(X1,treg,X1) & ::(X0,X2,X1) & UNI(X2)) | (~EX(X0,X1) | ~::(X1,treg,X1) | ~PAR(X0))
     ) & ((EX(X0,X1) & ::(X1,treg,X1) & PAR(X0)) | ! [X2] : (~::(X1,treg,X1) | ~::(X0,X2,X1) | ~UNI(X2)))) [nnf
```

```
transformation 358]
1098. ! [X0,X1] : ((? [X2] : (::(X1,treg,X1) & ::(X0,X2,X1) & UNI(X2)) | ~EX(X0,X1) | ~::(X1,treg,X1) | ~PAR(X0)) & ((EX(X0,X1) & ::(X1,treg,X1) & PAR(X0)) | ! [X2] : (~::(X1,treg,X1) | ~::(X0,X2,X1) | ~UNI(X2)))) [
flattening 1097]
1099. ! [X0,X1] : ((? [X2] : (::(X1,treg,X1) & ::(X0,X2,X1) & UNI(X2)) | ~EX(X0,X1) | ~::(X1,treg,X1) | ~PAR(X0)) & ((EX(X0,X1) & ::(X1,treg,X1) & PAR(X0)) | ! [X3] : (~::(X1,treg,X1) | ~::(X0,X3,X1) | ~UNI(X3)))) [rectify
1098]
1100. ! [X1,X0] : (? [X2] : (::(X1,treg,X1) & ::(X0,X2,X1) & UNI(X2)) => (::(X1,treg,X1) & ::(X0,sK22(X0,X1),X1) & UNI(sK22(X0,X1)))) [choice axiom]
1101. ! [X0,X1] : (((::(X1,treg,X1) & ::(X0,sK22(X0,X1),X1) & UNI(sK22(X0,X1))) | ~EX(X0,X1) | ~::(X1,treg,X1) | ~PAR(X0)) & ((EX(X0,X1) & ::(X1,treg,X1) & PAR(X0)) | ! [X3] : (~::(X1,treg,X1) | ~::(X0,X3,X1) | ~UNI(X3)))) [skolemisation 1099,1100]
1126. ! [X0,X1] : ((::(X0,imen,X1) | (? [X2] : (cP(X2,X0,X1) & ::(X2,mten,X1)) | ~::(X0,idcnt,X1))) & ((! [X2] : (~cP(X2,X0,X1) | ~::(X2,mten,X1)) & ::(X0,idcnt,X1)) | ~::(X0,imen,X1))) [nnf transformation 826]
1127. ! [X0,X1] : ((::(X0,imen,X1) | ? [X2] : (cP(X2,X0,X1) & ::(X2,mten,X1)) | ~::(X0,idcnt,X1)) & ((! [X2] : (~cP(X2,X0,X1) | ~::(X2,mten,X1)) & ::(X0,idcnt,X1)) | ~::(X0,imen,X1))) [flattening 1126]
1128. ! [X0,X1] : ((::(X0,imen,X1) | ? [X2] : (cP(X2,X0,X1) & ::(X2,mten,X1)) | ~::(X0,idcnt,X1)) & ((! [X3] : (~cP(X3,X0,X1) | ~::(X3,mten,X1)) & ::(X0,idcnt,X1)) | ~::(X0,imen,X1))) [rectify 1127]
1129. ! [X1,X0] : (? [X2] : (cP(X2,X0,X1) & ::(X2,mten,X1)) => (cP(sK36(X0,X1),X0,X1) & ::(sK36(X0,X1),mten,X1))) [choice axiom]
1130. ! [X0,X1] : ((::(X0,imen,X1) | (cP(sK36(X0,X1),X0,X1) & ::(sK36(X0,X1),mten,X1)) | ~::(X0,idcnt,X1)) & ((! [X3] : (~cP(X3,X0,X1) | ~::(X3,mten,X1)) & ::(X0,idcnt,X1)) | ~::(X0,imen,X1))) [skolemisation 1128,1129]
1131. ! [X0] : ((? [X1] : ::(X0,idcnt,X1) | (? [X2,X3] : (GDEP(X0,X2,X3) | SDEP(X0,X2)) | ! [X4] : ~::(X0,cnt,X4)) & ((! [X2,X3] : (~GDEP(X0,X2,X3) & ~SDEP(X0,X2)) & ? [X4] : ::(X0,cnt,X4)) | ! [X1] : ~::(X0,idcnt,X1))) [
nnf transformation 827]
1132. ! [X0] : ((? [X1] : ::(X0,idcnt,X1) | ? [X2,X3] : (GDEP(X0,X2,X3) | SDEP(X0,X2)) | ! [X4] : ~::(X0,cnt,X4)) & ((! [X2,X3] : (~GDEP(X0,X2,X3) & ~SDEP(X0,X2)) & ? [X4] : ::(X0,cnt,X4)) | ! [X1] : ~::(X0,idcnt,X1))) [
flattening 1131]
1133. ! [X0] : ((? [X1] : ::(X0,idcnt,X1) | ? [X2,X3] : (GDEP(X0,X2,X3) | SDEP(X0,X2)) | ! [X4] : ~::(X0,cnt,X4)) & ((! [X5,X6] : (~GDEP(X0,X5,X6) & ~SDEP(X0,X5)) & ? [X7] : ::(X0,cnt,X7)) | ! [X8] : ~::(X0,idcnt,X8))) [
rectify 1132]
1134. ! [X0] : (? [X1] : ::(X0,idcnt,X1) => ::(X0,idcnt,sK37(X0))) [choice axiom]
1135. ! [X0] : (? [X2,X3] : (GDEP(X0,X2,X3) | SDEP(X0,X2)) => (GDEP(X0,sK38(X0),sK39(X0)) | SDEP(X0,sK38(X0)))) [
choice axiom]
1136. ! [X0] : (? [X7] : ::(X0,cnt,X7) => ::(X0,cnt,sK40(X0))) [choice axiom]
1137. ! [X0] : ((::(X0,idcnt,sK37(X0)) | (GDEP(X0,sK38(X0),sK39(X0)) | SDEP(X0,sK38(X0))) | ! [X4] : ~::(X0,cnt,X4)) & ((! [X5,X6] : (~GDEP(X0,X5,X6) & ~SDEP(X0,X5)) & ::(X0,cnt,sK40(X0))) | ! [X8] : ~::(X0,idcnt,X8))) [
skolemisation 1133,1136,1135,1134]
1316. ! [X0] : ((? [X1] : ::(X0,rqlt,X1) | (! [X2] : ~::(X0,qlt,X2) | ! [X3,X4] : (~SDEP(X0,X4) | ~INH(X0,X3) | X3 = X4))) & ((? [X2] : ::(X0,qlt,X2) & ? [X3,X4] : (SDEP(X0,X4) & INH(X0,X3) & X3 != X4)) | ! [X1] : ~::(X0,
rqlt,X1))) [nnf transformation 558]
1317. ! [X0] : ((? [X1] : ::(X0,rqlt,X1) | ! [X2] : ~::(X0,qlt,X2) | ! [X3,X4] : (~SDEP(X0,X4) | ~INH(X0,X3) | X3 = X4)) & ((? [X2] : ::(X0,qlt,X2) & ? [X3,X4] : (SDEP(X0,X4) & INH(X0,X3) & X3 != X4)) | ! [X1] : ~::(X0,
rqlt,X1))) [flattening 1316]
1318. ! [X0] : ((? [X1] : ::(X0,rqlt,X1) | ! [X2] : ~::(X0,qlt,X2) | ! [X3,X4] : (~SDEP(X0,X4) | ~INH(X0,X3) | X3 = X4)) & ((? [X5] : ::(X0,qlt,X5) & ? [X6,X7] : (SDEP(X0,X7) & INH(X0,X6) & X6 != X7)) | ! [X8] : ~::(X0,
rqlt,X8))) [rectify 1317]
1319. ! [X0] : (? [X1] : ::(X0,rqlt,X1) => ::(X0,rqlt,sK132(X0))) [choice axiom]
1320. ! [X0] : (? [X5] : ::(X0,qlt,X5) => ::(X0,qlt,sK133(X0))) [choice axiom]
1321. ! [X0] : (? [X6,X7] : (SDEP(X0,X7) & INH(X0,X6) & X6 != X7) => (SDEP(X0,sK135(X0)) & INH(X0,sK134(X0)) & sK134(X0) != sK135(X0))) [choice axiom]
1322. ! [X0] : ((::(X0,rqlt,sK132(X0)) | ! [X2] : ~::(X0,qlt,X2) | ! [X3,X4] : (~SDEP(X0,X4) | ~INH(X0,X3) | X3 = X4)) & ((::(X0,qlt,sK133(X0)) & (SDEP(X0,sK135(X0)) & INH(X0,sK134(X0)) & sK134(X0) != sK135(X0))) | ! [X8] : ~::(X0,rqlt,X8))) [skolemisation 1318,1321,1320,1319]
1326. ! [X0,X1] : ((INH(X0,X1) | (! [X2] : (::(X1,sreg,X2) | ~::(X1,idcnt,X2) | ~::(X0,sdcnt,X2)) | ~SDEP(X0,X1))) & ((? [X2] : (~::(X1,sreg,X2) & ::(X1,idcnt,X2) & ::(X0,sdcnt,X2)) & SDEP(X0,X1)) | ~INH(X0,X1))) [nnf
transformation 562]
1327. ! [X0,X1] : ((INH(X0,X1) | ! [X2] : (::(X1,sreg,X2) | ~::(X1,idcnt,X2) | ~::(X0,sdcnt,X2)) | ~SDEP(X0,X1)) & ((? [X2] : (~::(X1,sreg,X2) & ::(X1,idcnt,X2) & ::(X0,sdcnt,X2)) & SDEP(X0,X1)) | ~INH(X0,X1))) [flattening
1326]
1328. ! [X0,X1] : ((INH(X0,X1) | ! [X2] : (::(X1,sreg,X2) | ~::(X1,idcnt,X2) | ~::(X0,sdcnt,X2)) | ~SDEP(X0,X1)) & ((? [X3] : (~::(X1,sreg,X3) & ::(X1,idcnt,X3) & ::(X0,sdcnt,X3)) & SDEP(X0,X1)) | ~INH(X0,X1))) [rectify
1327]
1329. ! [X1,X0] : (? [X3] : (~::(X1,sreg,X3) & ::(X1,idcnt,X3) & ::(X0,sdcnt,X3)) => (~::(X1,sreg,sK138(X0,X1)) & ::(X1,idcnt,sK138(X0,X1)) & ::(X0,sdcnt,sK138(X0,X1)))) [choice axiom]
1330. ! [X0,X1] : ((INH(X0,X1) | ! [X2] : (::(X1,sreg,X2) | ~::(X1,idcnt,X2) | ~::(X0,sdcnt,X2)) | ~SDEP(X0,X1)) & (((~::(X1,sreg,sK138(X0,X1)) & ::(X1,idcnt,sK138(X0,X1)) & ::(X0,sdcnt,sK138(X0,X1))) & SDEP(X0,X1)) | ~INH
(X0,X1))) [skolemisation 1328,1329]
1331. ! [X0] : ((? [X1] : ::(X0,sdcnt,X1) | ! [X2,X3] : (~SDEP(X0,X2) | ::(X2,sreg,X3) | ~::(X2,idcnt,X3) | ~::(X0,cnt,X3))) & (? [X2,X3] : (SDEP(X0,X2) & ~::(X2,sreg,X3) & ::(X2,idcnt,X3) & ::(X0,cnt,X3)) | ! [X1] : ~::(
X0,sdcnt,X1))) [nnf transformation 564]
1332. ! [X0] : ((? [X1] : ::(X0,sdcnt,X1) | ! [X2,X3] : (~SDEP(X0,X2) | ::(X2,sreg,X3) | ~::(X2,idcnt,X3) | ~::(X0,cnt,X3))) & (? [X4,X5] : (SDEP(X0,X4) & ~::(X4,sreg,X5) & ::(X4,idcnt,X5) & ::(X0,cnt,X5)) | ! [X6] : ~::(
X0,sdcnt,X6))) [rectify 1331]
1333. ! [X0] : (? [X1] : ::(X0,sdcnt,X1) => ::(X0,sdcnt,sK139(X0))) [choice axiom]
1334. ! [X0] : (? [X4,X5] : (SDEP(X0,X4) & ~::(X4,sreg,X5) & ::(X4,idcnt,X5) & ::(X0,cnt,X5)) => (SDEP(X0,sK140(X0)) & ~::(sK140(X0),sreg,sK141(X0)) & ::(sK140(X0),idcnt,sK141(X0)) & ::(X0,cnt,sK141(X0)))) [choice axiom]
1335. ! [X0] : ((::(X0,sdcnt,sK139(X0)) | ! [X2,X3] : (~SDEP(X0,X2) | ::(X2,sreg,X3) | ~::(X2,idcnt,X3) | ~::(X0,cnt,X3))) & ((SDEP(X0,sK140(X0)) & ~::(sK140(X0),sreg,sK141(X0)) & ::(sK140(X0),idcnt,sK141(X0)) & ::(X0,cnt
,sK141(X0))) | ! [X6] : ~::(X0,sdcnt,X6))) [skolemisation 1332,1334,1333]
1379. ! [X0] : (! [X1] : (::(X0,sreg,X1) | ~EX(X0,X1)) | ! [X2] : ~::(X0,sreg,X2)) [rectify 1027]
1389. ! [X0] : (! [X1] : (::(X0,gdcnt,X1) | ~EX(X0,X1)) | ! [X2] : ~::(X0,gdcnt,X2)) [rectify 1037]
1396. ! [X0,X1] : ((::(X0,X1) | ! [X2] : ~::(X0,X1,X2)) & (? [X2] : ::(X0,X1,X2) | ~::(X0,X1))) [nnf
transformation 705]
1397. ! [X0,X1] : ((::(X0,X1) | ! [X2] : ~::(X0,X1,X2)) & (? [X3] : ::(X0,X1,X3) | ~::(X0,X1))) [rectify 1396]
1398. ! [X1,X0] : (? [X3] : ::(X0,X1,X3) => ::(X0,X1,sK161(X0,X1))) [choice axiom]
1399. ! [X0,X1] : ((::(X0,X1) | ! [X2] : ~::(X0,X1,X2)) & (::(X0,X1,sK161(X0,X1)) | ~::(X0,X1))) [skolemisation
1397,1398]
1404. ! [X0] : ((PQ(X0) | (::(X0,rqlt) | ~::(X0,sdcnt))) & ((~::(X0,rqlt) & ::(X0,sdcnt)) | ~PQ(X0))) [nnf
transformation 716]
1405. ! [X0] : ((PQ(X0) | ::(X0,rqlt) | ~::(X0,sdcnt)) & ((~::(X0,rqlt) & ::(X0,sdcnt)) | ~PQ(X0))) [flattening
```

```
1404]
1406. ! [X2,X1,X0] : ((SREG(X0,X1,X2) & ~::(X0,sreg,X2) & ::(X0,idcnt,X2)) | ~sP11(X2,X1,X0)) [nnf transformation
1069]
1407. ! [X0,X1,X2] : ((SREG(X2,X1,X0) & ~::(X2,sreg,X0) & ::(X2,idcnt,X0)) | ~sP11(X0,X1,X2)) [rectify 1406]
1408. ! [X2,X1,X0] : ((? [X4] : (SREG(X4,X1,X2) & ! [X5] : (cP(X5,X4,X2) | ~INH(X0,X5)) & INH(X0,X4)) & Q(X0)) | ~
sP10(X2,X1,X0)) [nnf transformation 1068]
1409. ! [X0,X1,X2] : ((? [X3] : (SREG(X3,X1,X0) & ! [X4] : (cP(X4,X3,X0) | ~INH(X2,X4)) & INH(X2,X3)) & Q(X2)) | ~
sP10(X0,X1,X2)) [rectify 1408]
1410. ! [X2,X1,X0] : (? [X3] : (SREG(X3,X1,X0) & ! [X4] : (cP(X4,X3,X0) | ~INH(X2,X4)) & INH(X2,X3)) => (SREG(
sK162(X0,X1,X2),X1,X0) & ! [X4] : (cP(X4,sK162(X0,X1,X2),X0) | ~INH(X2,X4)) & INH(X2,sK162(X0,X1,X2)))) [
choice axiom]
1411. ! [X0,X1,X2] : (((SREG(sK162(X0,X1,X2),X1,X0) & ! [X4] : (cP(X4,sK162(X0,X1,X2),X0) | ~INH(X2,X4)) & INH(X2,
sK162(X0,X1,X2))) & Q(X2)) | ~sP10(X0,X1,X2)) [skolemisation 1409,1410]
1412. ! [X2,X1,X0] : ((? [X6] : (SREG(X6,X1,X2) & ! [X7] : (cP(X7,X6,X2) | ~GDEP(X0,X7,X2)) & GDEP(X0,X6,X2)) &
::(X0,gdcnt,X2)) | ~sP9(X2,X1,X0)) [nnf transformation 1067]
1413. ! [X0,X1,X2] : ((? [X3] : (SREG(X3,X1,X0) & ! [X4] : (cP(X4,X3,X0) | ~GDEP(X2,X4,X0)) & GDEP(X2,X3,X0)) &
::(X2,gdcnt,X0)) | ~sP9(X0,X1,X2)) [rectify 1412]
1414. ! [X2,X1,X0] : (? [X3] : (SREG(X3,X1,X0) & ! [X4] : (cP(X4,X3,X0) | ~GDEP(X2,X4,X0)) & GDEP(X2,X3,X0)) => (
SREG(sK163(X0,X1,X2),X1,X0) & ! [X4] : (cP(X4,sK163(X0,X1,X2),X0) | ~GDEP(X2,X4,X0)) & GDEP(X2,sK163(X0,X1,
X2),X0))) [choice axiom]
1415. ! [X0,X1,X2] : (((SREG(sK163(X0,X1,X2),X1,X0) & ! [X4] : (cP(X4,sK163(X0,X1,X2),X0) | ~GDEP(X2,X4,X0)) &
GDEP(X2,sK163(X0,X1,X2),X0)) & ::(X2,gdcnt,X0)) | ~sP9(X0,X1,X2)) [skolemisation 1413,1414]
1416. ! [X2,X1,X0] : (? [X3] : (SPROJ(X3,X1,X2) & STREG(X0,X3)) => (SPROJ(sK164(X0,X1,X2),X1,X2) & STREG(X0,sK164(
X0,X1,X2)))) [choice axiom]
1417. ! [X0,X1,X2] : (((SPROJ(sK164(X0,X1,X2),X1,X2) & STREG(X0,sK164(X0,X1,X2))) & PD(X0)) | sP10(X2,X1,X0) | sP9
(X2,X1,X0) | sP11(X2,X1,X0) | ~SLC(X0,X1,X2)) [skolemisation 1070,1416]
1418. ? [X0,X1,X2] : ((~S(X1) | (~PD(X0) & ~PQ(X0) & ~ED(X0))) & SLC(X0,X1,X2)) => ((~S(sK166) | (~PD(sK165) & ~PQ
(sK165) & ~ED(sK165))) & SLC(sK165,sK166,sK167)) [choice axiom]
1419. (~S(sK166) | (~PD(sK165) & ~PQ(sK165) & ~ED(sK165))) & SLC(sK165,sK166,sK167) [skolemisation 1051,1418]
1470. ~::(X0,X1,X2) | UNI(X1) [cnf transformation 800]
1471. ~::(X0,X1,X2) | ::(X2,treg,X2) [cnf transformation 800]
1480. EX(X0,X1) | ~::(X1,treg,X1) | ~::(X0,X3,X1) | ~UNI(X3) [cnf transformation 1101]
1525. ~::(X0,imen,X1) | ::(X0,idcnt,X1) [cnf transformation 1130]
1530. ~::(X0,idcnt,X8) | ~SDEP(X0,X5) [cnf transformation 1137]
1531. ~GDEP(X0,X5,X6) | ~::(X0,idcnt,X8) [cnf transformation 1137]
1663. ~SREG(X0,X1,X2) | ::(X1,sreg,X2) [cnf transformation 901]
1704. ~SPROJ(X0,X1,X2) | ::(X1,sreg,X2) [cnf transformation 931]
1762. ~::(X0,rqlt,X8) | INH(X0,sK134(X0)) [cnf transformation 1322]
1770. ::(X0,sdcnt,sK138(X0,X1)) | ~INH(X0,X1) [cnf transformation 1330]
1777. ~::(X0,sdcnt,X6) | SDEP(X0,sK140(X0)) [cnf transformation 1335]
1874. ~::(X1,sreg,X0) | ::(X1,imen,X0) [cnf transformation 1007]
1875. ~::(X1,idcnt,X0) | ::(X1,cnt,X0) [cnf transformation 1008]
1882. ~::(X1,gdcnt,X0) | ::(X1,cnt,X0) [cnf transformation 1015]
1894. ~::(X0,sreg,X2) | ~EX(X0,X1) | ::(X0,sreg,X1) [cnf transformation 1379]
1904. ~::(X0,gdcnt,X2) | ~EX(X0,X1) | ::(X0,gdcnt,X1) [cnf transformation 1389]
1913. ~::(X2,sdcnt,X3) | ~::(X2,gdcnt,X3) [cnf transformation 1042]
2310. ::(X0,X1,sK161(X0,X1)) | ~::(X0,X1) [cnf transformation 1399]
2311. ~::(X0,X1,X2) | ::(X0,X1) [cnf transformation 1399]
2318. ~::(X0,cnt) | ::(X0,sdcnt) | ::(X0,sreg) | ED(X0) [cnf transformation 1046]
2321. ~::(X0,sdcnt) | ::(X0,rqlt) | PQ(X0) [cnf transformation 1405]
2322. ~Q(X0) | PQ(X0) [cnf transformation 1047]
2323. ~::(X0,sreg) | S(X0) [cnf transformation 1048]
2324. ~sP11(X0,X1,X2) | ::(X2,idcnt,X0) [cnf transformation 1407]
2325. ~sP11(X0,X1,X2) | ~::(X2,sreg,X0) [cnf transformation 1407]
2326. ~sP11(X0,X1,X2) | SREG(X2,X1,X0) [cnf transformation 1407]
2327. ~sP10(X0,X1,X2) | Q(X2) [cnf transformation 1411]
2330. SREG(sK162(X0,X1,X2),X1,X0) | ~sP10(X0,X1,X2) [cnf transformation 1411]
2331. ~sP9(X0,X1,X2) | ::(X2,gdcnt,X0) [cnf transformation 1415]
2332. GDEP(X2,sK163(X0,X1,X2),X0) | ~sP9(X0,X1,X2) [cnf transformation 1415]
2334. SREG(sK163(X0,X1,X2),X1,X0) | ~sP9(X0,X1,X2) [cnf transformation 1415]
2335. ~SLC(X0,X1,X2) | sP10(X2,X1,X0) | sP9(X2,X1,X0) | sP11(X2,X1,X0) | PD(X0) [cnf transformation 1417]
2337. SPROJ(sK164(X0,X1,X2),X1,X2) | sP10(X2,X1,X0) | sP9(X2,X1,X0) | sP11(X2,X1,X0) | ~SLC(X0,X1,X2) [cnf
transformation 1417]
2338. SLC(sK165,sK166,sK167) [cnf transformation 1419]
2339. ~S(sK166) | ~ED(sK165) [cnf transformation 1419]
2340. ~S(sK166) | ~PQ(sK165) [cnf transformation 1419]
2341. ~S(sK166) | ~PD(sK165) [cnf transformation 1419]
2380. EX(X0,X1) | ~::(X1,treg,X1) | ~::(X0,X3,X1) [subsumption resolution 1480,1470]
2381. ~::(X0,X3,X1) | EX(X0,X1) [subsumption resolution 2380,1471]
2390. 1 <=> PD(sK165) [avatar definition]
2394. 2 <=> S(sK166) [avatar definition]
2397. ~1 | ~2 [avatar split clause 2341,2394,2390]
2399. 3 <=> PQ(sK165) [avatar definition]
2401. ~PQ(sK165) <= (~3) [avatar component clause 2399]
2402. ~3 | ~2 [avatar split clause 2340,2394,2399]
2404. 4 <=> ED(sK165) [avatar definition]
2406. ~ED(sK165) <= (~4) [avatar component clause 2404]
2407. ~4 | ~2 [avatar split clause 2339,2394,2404]
3133. ~::(X0,gdcnt,sK138(X0,X1)) | ~INH(X0,X1) [resolution 1770,1913]
3138. EX(X10,sK138(X10,X11)) | ~INH(X10,X11) [resolution 1770,2381]
3207. ::(X18,imen,sK161(X18,sreg)) | ~::(X18,sreg) [resolution 2310,1874]
3225. ::(X36,idcnt,sK161(X36,imen)) | ~::(X36,imen) [resolution 2310,1525]
3237. SDEP(X48,sK140(X48)) | ~::(X48,sdcnt) [resolution 2310,1777]
3271. INH(X82,sK134(X82)) | ~::(X82,rqlt) [resolution 2310,1762]
3802. ::(X5,sreg,X6) | ~EX(X5,X6) | ~::(X5,sreg) [resolution 1894,2310]
3862. ::(X1,gdcnt,X2) | ~EX(X1,X2) | ~::(X1,gdcnt) [resolution 1904,2310]
3948. ~sP10(X11,X12,X13) | ::(X12,sreg,X11) [resolution 2330,1663]
3951. ~sP9(X0,X1,X2) | ~::(X2,idcnt,X3) [resolution 2332,1531]
3959. ~sP9(X11,X12,X13) | ::(X12,sreg,X11) [resolution 2334,1663]
5532. sP10(sK167,sK166,sK165) | sP9(sK167,sK166,sK165) | sP11(sK167,sK166,sK165) | PD(sK165) [resolution
2335,2338]
```

```
5534. 103 <=> sP11(sK167,sK166,sK165) [avatar definition]
5535. ~sP11(sK167,sK166,sK165) <- (~103) [avatar component clause 5534]
5536. sP11(sK167,sK166,sK165) <- (103) [avatar component clause 5534]
5538. 104 <=> sP9(sK167,sK166,sK165) [avatar definition]
5540. sP9(sK167,sK166,sK165) <- (104) [avatar component clause 5538]
5542. 105 <=> sP10(sK167,sK166,sK165) [avatar definition]
5544. sP10(sK167,sK166,sK165) <- (105) [avatar component clause 5542]
5545. 1 | 103 | 104 | 105 [avatar split clause 5532,5542,5538,5534,2390]
6133. sP10(X11,X12,X13) | sP9(X11,X12,X13) | sP11(X11,X12,X13) | ~SLC(X13,X12,X11) | ::(X12,sreg,X11) [resolution
      2337,1704]
6137. sP9(X11,X12,X13) | sP11(X11,X12,X13) | ~SLC(X13,X12,X11) | ::(X12,sreg,X11) [subsumption resolution
      6133,3948]
6138. ~SLC(X13,X12,X11) | sP11(X11,X12,X13) | ::(X12,sreg,X11) [subsumption resolution 6137,3959]
7761. ~::(X8,sreg) | ::(X8,imen) [resolution 3207,2311]
9901. ~EX(X9,sK138(X9,X10)) | ~::(X9,gdcnt) | ~INH(X9,X10) [resolution 3862,3133]
9912. ~::(X9,gdcnt) | ~INH(X9,X10) [subsumption resolution 9901,3138]
11728. SREG(sK165,sK166,sK167) <- (103) [resolution 5536,2326]
11729. ~::(sK165,sreg,sK167) <- (103) [resolution 5536,2325]
11730. ::(sK165,idcnt,sK167) <- (103) [resolution 5536,2324]
11731. sP11(sK167,sK166,sK165) | ::(sK166,sreg,sK167) [resolution 6138,2338]
11736. ::(sK166,sreg,sK167) <- (103) [resolution 11728,1663]
11902. ~EX(sK165,sK167) | ~::(sK165,sreg) <- (103) [resolution 11729,3802]
11904. 276 <=> ::(sK165,sreg) [avatar definition]
11905. ::(sK165,sreg) <- (276) [avatar component clause 11904]
11906. ~::(sK165,sreg) <- (~276) [avatar component clause 11904]
11908. 277 <=> EX(sK165,sK167) [avatar definition]
11911. ~276 | ~277 | ~103 [avatar split clause 11902,5534,11908,11904]
11932. ~SDEP(sK165,X0) <- (103) [resolution 11730,1530]
11935. ::(sK165,cnt,sK167) <- (103) [resolution 11730,1875]
11941. EX(sK165,sK167) <- (103) [resolution 11730,2381]
11958. 277 | ~103 [avatar split clause 11941,5534,11908]
12008. ~::(sK165,sdcnt) <- (103) [resolution 11932,3237]
12042. ::(sK166,sreg) <- (103) [resolution 11736,2311]
12088. S(sK166) <- (103) [resolution 12042,2323]
12093. 2 | ~103 [avatar split clause 12088,5534,2394]
12741. ::(sK165,cnt) <- (103) [resolution 11935,2311]
12762. ::(sK165,sdcnt) | ::(sK165,sreg) | ED(sK165) <- (103) [resolution 12741,2318]
12765. ::(sK165,sreg) | ED(sK165) <- (103) [subsumption resolution 12762,12008]
12766. ED(sK165) <- (103, ~276) [subsumption resolution 12765,11906]
12767. $false <- (~4, 103, ~276) [subsumption resolution 12766,2406]
12768. 4 | ~103 | 276 [avatar contradiction clause 12767]
12821. Q(sK165) <- (105) [resolution 5544,2327]
12839. PQ(sK165) <- (105) [resolution 12821,2322]
12840. $false <- (~3, 105) [subsumption resolution 12839,2401]
12841. 3 | ~105 [avatar contradiction clause 12840]
12845. ~::(sK165,idcnt,X1) <- (104) [resolution 5540,3951]
12846. ::(sK165,gdcnt,sK167) <- (104) [resolution 5540,2331]
12856. ~::(sK165,imen) <- (104) [resolution 12845,3225]
12902. ::(sK165,cnt,sK167) <- (104) [resolution 12846,1882]
12906. ::(sK165,gdcnt) <- (104) [resolution 12846,2311]
12997. ::(sK165,cnt) <- (104) [resolution 12902,2311]
13017. ::(sK165,sdcnt) | ::(sK165,sreg) | ED(sK165) <- (104) [resolution 12997,2318]
13020. ::(sK165,sdcnt) | ED(sK165) <- (104, ~276) [subsumption resolution 13017,11906]
13021. ::(sK165,sdcnt) <- (~4, 104, ~276) [subsumption resolution 13020,2406]
13031. ::(sK165,rqlt) | PQ(sK165) <- (~4, 104, ~276) [resolution 13021,2321]
13034. ::(sK165,rqlt) <- (~3, ~4, 104, ~276) [subsumption resolution 13031,2401]
14489. 374 <=> ::(sK166,sreg,sK167) [avatar definition]
14491. ::(sK166,sreg,sK167) <- (374) [avatar component clause 14489]
14506. 378 <=> ::(sK166,sreg) [avatar definition]
14508. ::(sK166,sreg) <- (378) [avatar component clause 14506]
14521. ::(sK166,sreg,sK167) <- (~103) [subsumption resolution 11731,5535]
14522. 374 | 103 [avatar split clause 14521,5534,14489]
16001. ::(sK166,sreg) <- (374) [resolution 14491,2311]
16020. 378 | ~374 [avatar split clause 16001,14489,14506]
16082. S(sK166) <- (378) [resolution 14508,2323]
27911. ~INH(sK165,X2) <- (104) [resolution 9912,12906]
27913. ~::(sK165,rqlt) <- (104) [resolution 27911,3271]
27914. $false <- (~3, ~4, 104, ~276) [subsumption resolution 27913,13034]
27915. 3 | 4 | ~104 | 276 [avatar contradiction clause 27914]
27932. ::(sK165,imen) <- (276) [resolution 11905,7761]
27937. $false <- (104, 276) [subsumption resolution 27932,12856]
27938. ~104 | ~276 [avatar contradiction clause 27937]
27939. 2 | ~378 [avatar split clause 16082,14506,2394]
27965. $false [avatar sat refutation
      2397,2402,2407,5545,11911,11958,12093,12768,12841,14522,16020,27915,27938,27939]
```

Proof of Theorem (t_{bd}43)

This went KO with both provers

Proof of Theorem (t_{bd}44)

This went KO with both provers

Proof of Theorem (t_{bd}46)

This went KO with both provers

Proof of Theorem ($t_{bd}48$)

```
30. ! [X17] : ! [X1] : (EX(X17,X1) => (::(X1,treg,X1) & PAR(X1) & PAR(X17))) [input]
101. ! [X0] : ! [X8] : ! [X1] : (SREG(X0,X8,X1) => (::(X1,treg,X1) & ::(X8,sreg,X1) & ~::(X0,sreg,X1) & ::(X0,
     idcnt,X1))) [input]
120. ! [X0] : ! [X8] : ! [X1] : (SPROJ(X0,X8,X1) => (::(X1,treg,X1) & ::(X8,sreg,X1) & ::(X0,streg,X1))) [input]
191. UNI(idcnt) [input]
201. ~? [X6] : (PAR(X6) & UNI(X6)) [input]
257. ! [X6,X16] : (::(X6,X16) <=> ? [X1] : ::(X6,X16,X1)) [input]
271. ! [X6] : (S(X6) <=> ::(X6,sreg)) [input]
276. ! [X6,X7] : (P(X6,X7) <=> ((! [X6] : (EX(X6,X1) => cP(X6,X7,X1)) & S(X7) & S(X6)) | (oP(X6,X7) & ((T(X7) & T(
     X6)) | (PD(X7) & PD(X6)))))) [input]
281. ! [X6,X5,X1] : (SLC(X6,X5,X1) <=> ((? [X7] : (SPROJ(X7,X5,X1) & STREG(X6,X7)) & PD(X6)) | (? [X7] : (SREG(X7,
     X5,X1) & ! [X13] : (INH(X6,X13) => cP(X13,X7,X1)) & INH(X6,X7)) & Q(X6)) | (? [X7] : (SREG(X7,X5,X1) & ! [
     X13] : (GDEP(X6,X13,X1) => cP(X13,X7,X1)) & GDEP(X6,X7,X1)) & ::(X6,gdcnt,X1)) | (SREG(X6,X5,X1) & ~::(X6,
     sreg,X1) & ::(X6,idcnt,X1))))) [input]
291. ! [X6,X7,X1,X4,X5] : ((SLC(X7,X5,X1) & SLC(X6,X4,X1) & PC(X6,X7,X1)) => P(X4,X5)) [input]
292. ~! [X6,X7,X1,X4,X5] : ((SLC(X7,X5,X1) & SLC(X6,X4,X1) & PC(X6,X7,X1)) => P(X4,X5)) [negated conjecture 291]
349. ! [X0] : ! [X1] : (EX(X0,X1) => (::(X1,treg,X1) & PAR(X1) & PAR(X0))) [rectify 30]
350. ! [X0,X1] : (EX(X0,X1) => (::(X1,treg,X1) & PAR(X1) & PAR(X0))) [flattening 349]
485. ! [X0] : ! [X1] : ! [X2] : (SREG(X0,X1,X2) => (::(X2,treg,X2) & ::(X1,sreg,X2) & ~::(X0,sreg,X2) & ::(X0,
     idcnt,X2))) [rectify 101]
486. ! [X0,X1,X2] : (SREG(X0,X1,X2) => (::(X2,treg,X2) & ::(X1,sreg,X2) & ~::(X0,sreg,X2) & ::(X0,idcnt,X2))) [
     flattening 485]
520. ! [X0] : ! [X1] : ! [X2] : (SPROJ(X0,X1,X2) => (::(X2,treg,X2) & ::(X1,sreg,X2) & ::(X0,streg,X2))) [rectify
     120]
521. ! [X0,X1,X2] : (SPROJ(X0,X1,X2) => (::(X2,treg,X2) & ::(X1,sreg,X2) & ::(X0,streg,X2))) [flattening 520]
616. ~? [X0] : (PAR(X0) & UNI(X0)) [rectify 201]
704. ! [X0,X1] : (::(X0,X1) <=> ? [X2] : ::(X0,X1,X2)) [rectify 257]
719. ! [X0] : (S(X0) <=> ::(X0,sreg)) [rectify 271]
724. ! [X0,X1] : (P(X0,X1) <=> ((! [X2] : (EX(X2,X3) => cP(X2,X1,X3)) & S(X1) & S(X0)) | (oP(X0,X1) & ((T(X1) & T(
     X0)) | (PD(X1) & PD(X0)))))) [rectify 276]
725. ! [X3] : ! [X0,X1] : (P(X0,X1) <=> ((! [X2] : (EX(X2,X3) => cP(X2,X1,X3)) & S(X1) & S(X0)) | (oP(X0,X1) & ((T
     (X1) & T(X0)) | (PD(X1) & PD(X0)))))) [closure 724]
726. ! [X3,X0,X1] : (P(X0,X1) <=> ((! [X2] : (EX(X2,X3) => cP(X2,X1,X3)) & S(X1) & S(X0)) | (oP(X0,X1) & ((T(X1) &
     T(X0)) | (PD(X1) & PD(X0)))))) [flattening 725]
731. ! [X0,X1,X2] : (SLC(X0,X1,X2) <=> ((? [X3] : (SPROJ(X3,X1,X2) & STREG(X0,X3)) & PD(X0)) | (? [X4] : (SREG(X4,
     X1,X2) & ! [X5] : (INH(X0,X5) => cP(X5,X4,X2)) & INH(X0,X4)) & Q(X0)) | (? [X6] : (SREG(X6,X1,X2) & ! [X7] :
      (GDEP(X0,X7,X2) => cP(X7,X6,X2)) & GDEP(X0,X6,X2)) & ::(X0,gdcnt,X2)) | (SREG(X0,X1,X2) & ~::(X0,sreg,X2) &
      ::(X0,idcnt,X2)))) [rectify 281]
741. ~! [X0,X1,X2,X3,X4] : ((SLC(X1,X4,X2) & SLC(X0,X3,X2) & PC(X0,X1,X2)) => P(X3,X4)) [rectify 292]
742. ! [X0,X1,X2] : (SLC(X0,X1,X2) => ((? [X3] : (SPROJ(X3,X1,X2) & STREG(X0,X3)) & PD(X0)) | (? [X4] : (SREG(X4,
     X1,X2) & ! [X5] : (INH(X0,X5) => cP(X5,X4,X2)) & INH(X0,X4)) & Q(X0)) | (? [X6] : (SREG(X6,X1,X2) & ! [X7] :
      (GDEP(X0,X7,X2) => cP(X7,X6,X2)) & GDEP(X0,X6,X2)) & ::(X0,gdcnt,X2)) | (SREG(X0,X1,X2) & ~::(X0,sreg,X2) &
      ::(X0,idcnt,X2)))) [unused predicate definition removal 731]
800. ! [X0,X1] : ((::(X1,treg,X1) & PAR(X1) & PAR(X0)) | ~EX(X0,X1)) [ennf transformation 350]
902. ! [X0,X1,X2] : ((::(X2,treg,X2) & ::(X1,sreg,X2) & ~::(X0,sreg,X2) & ::(X0,idcnt,X2)) | ~SREG(X0,X1,X2)) [
     ennf transformation 486]
932. ! [X0,X1,X2] : ((::(X2,treg,X2) & ::(X1,sreg,X2) & ::(X0,streg,X2)) | ~SPROJ(X0,X1,X2)) [ennf transformation
     521]
995. ! [X0] : (~PAR(X0) | ~UNI(X0)) [ennf transformation 616]
1049. ! [X3,X0,X1] : (P(X0,X1) <=> ((! [X2] : (cP(X2,X1,X3) | ~EX(X2,X3)) & S(X1) & S(X0)) | (oP(X0,X1) & ((T(X1)
      & T(X0)) | (PD(X1) & PD(X0)))))) [ennf transformation 726]
1051. ! [X0,X1,X2] : (((? [X3] : (SPROJ(X3,X1,X2) & STREG(X0,X3)) & PD(X0)) | (? [X4] : (SREG(X4,X1,X2) & ! [X5] :
      (cP(X5,X4,X2) | ~INH(X0,X5)) & INH(X0,X4)) & Q(X0)) | (? [X6] : (SREG(X6,X1,X2) & ! [X7] : (cP(X7,X6,X2) |
      ~GDEP(X0,X7,X2)) & GDEP(X0,X6,X2)) & ::(X0,gdcnt,X2)) | (SREG(X0,X1,X2) & ~::(X0,sreg,X2) & ::(X0,idcnt,X2))
     ) | ~SLC(X0,X1,X2)) [ennf transformation 742]
1052. ! [X0,X1,X2] : ((? [X3] : (SPROJ(X3,X1,X2) & STREG(X0,X3)) & PD(X0)) | (? [X4] : (SREG(X4,X1,X2) & ! [X5] :
      (cP(X5,X4,X2) | ~INH(X0,X5)) & INH(X0,X4)) & Q(X0)) | (? [X6] : (SREG(X6,X1,X2) & ! [X7] : (cP(X7,X6,X2) | ~
     GDEP(X0,X7,X2)) & GDEP(X0,X6,X2)) & ::(X0,gdcnt,X2)) | (SREG(X0,X1,X2) & ~::(X0,sreg,X2) & ::(X0,idcnt,X2))
     | ~SLC(X0,X1,X2)) [flattening 1051]
1053. ? [X0,X1,X2,X3,X4] : (~P(X3,X4) & (SLC(X1,X4,X2) & SLC(X0,X3,X2) & PC(X0,X1,X2))) [ennf transformation 741]
1054. ? [X0,X1,X2,X3,X4] : (~P(X3,X4) & SLC(X1,X4,X2) & SLC(X0,X3,X2) & PC(X0,X1,X2)) [flattening 1053]
1070. ! [X1,X0] : (sP9(X1,X0) <=> (oP(X0,X1) & ((T(X1) & T(X0)) | (PD(X1) & PD(X0))))) [predicate definition
      introduction]
1071. ! [X3,X0,X1] : (P(X0,X1) <=> ((! [X2] : (cP(X2,X1,X3) | ~EX(X2,X3)) & S(X1) & S(X0)) | sP9(X1,X0))) [
      definition folding 1049,1070]
1072. ! [X2,X1,X0] : ((? [X6] : (SREG(X6,X1,X2) & ! [X7] : (cP(X7,X6,X2) | ~GDEP(X0,X7,X2)) & GDEP(X0,X6,X2)) &
      ::(X0,gdcnt,X2)) | ~sP10(X2,X1,X0)) [predicate definition introduction]
1073. ! [X2,X1,X0] : ((? [X4] : (SREG(X4,X1,X2) & ! [X5] : (cP(X5,X4,X2) | ~INH(X0,X5)) & INH(X0,X4)) & Q(X0)) | ~
     sP11(X2,X1,X0)) [predicate definition introduction]
1074. ! [X2,X1,X0] : ((SREG(X0,X1,X2) & ~::(X0,sreg,X2) & ::(X0,idcnt,X2)) | ~sP12(X2,X1,X0)) [predicate
      definition introduction]
1075. ! [X0,X1,X2] : ((? [X3] : (SPROJ(X3,X1,X2) & STREG(X0,X3)) & PD(X0)) | sP11(X2,X1,X0) | sP10(X2,X1,X0) |
     sP12(X2,X1,X0) | ~SLC(X0,X1,X2)) [definition folding 1052,1074,1073,1072]
1401. ! [X0,X1] : ((::(X0,X1) | ! [X2] : ~::(X0,X1,X2)) & (? [X2] : ::(X0,X1,X2) | ~::(X0,X1))) [nnf
      transformation 704]
1402. ! [X0,X1] : ((::(X0,X1) | ! [X2] : ~::(X0,X1,X2)) & (? [X3] : ::(X0,X1,X3) | ~::(X0,X1))) [rectify 1401]
1403. ! [X1,X0] : (? [X3] : ::(X0,X1,X3) => ::(X0,X1,sK162(X0,X1))) [choice axiom]
1404. ! [X0,X1] : ((::(X0,X1) | ! [X2] : ~::(X0,X1,X2)) & (::(X0,X1,sK162(X0,X1)) | ~::(X0,X1))) [skolemisation
      1402,1403]
1410. ! [X0] : ((S(X0) | ~::(X0,sreg)) & (::(X0,sreg) | ~S(X0))) [nnf transformation 719]
1414. ! [X3,X0,X1] : ((P(X0,X1) | ((? [X2] : (~cP(X2,X1,X3) & EX(X2,X3)) | ~S(X1) | ~S(X0)) & ~sP9(X1,X0))) & (((!
      [X2] : (cP(X2,X1,X3) | ~EX(X2,X3)) & S(X1) & S(X0)) | sP9(X1,X0)) | ~P(X0,X1))) [nnf transformation 1071]
1415. ! [X3,X0,X1] : ((P(X0,X1) | ((? [X2] : (~cP(X2,X1,X3) & EX(X2,X3)) | ~S(X1) | ~S(X0)) & ~sP9(X1,X0))) & ((!
      [X2] : (cP(X2,X1,X3) | ~EX(X2,X3)) & S(X1) & S(X0)) | sP9(X1,X0) | ~P(X0,X1))) [flattening 1414]
1416. ! [X0,X1,X2] : ((P(X1,X2) | ((? [X3] : (~cP(X3,X2,X0) & EX(X3,X0)) | ~S(X2) | ~S(X1)) & ~sP9(X2,X1))) & ((!
      [X4] : (cP(X4,X2,X0) | ~EX(X4,X0)) & S(X2) & S(X1)) | sP9(X2,X1) | ~P(X1,X2))) [rectify 1415]
1417. ! [X2,X0] : (? [X3] : (~cP(X3,X2,X0) & EX(X3,X0)) => (~cP(sK163(X0,X2),X2,X0) & EX(sK163(X0,X2),X0))) [
      choice axiom]
1418. ! [X0,X1,X2] : ((P(X1,X2) | (((~cP(sK163(X0,X2),X2,X0) & EX(sK163(X0,X2),X0)) | ~S(X2) | ~S(X1)) & ~sP9(X2,
```

```
X1))) & ((! [X4] : (cP(X4,X2,X0) | ~EX(X4,X0)) & S(X2) & S(X1)) | sP9(X2,X1) | ~P(X1,X2))) [skolemisation
1416,1417]
1421. ! [X2,X1,X0] : ((SREG(X0,X1,X2) & ~::(X0,sreg,X2) & ::(X0,idcnt,X2)) | ~sP12(X2,X1,X0)) [nnf transformation
1074]
1422. ! [X0,X1,X2] : ((SREG(X2,X1,X0) & ~::(X2,sreg,X0) & ::(X2,idcnt,X0)) | ~sP12(X0,X1,X2)) [rectify 1421]
1423. ! [X2,X1,X0] : ((? [X4] : (SREG(X4,X1,X2) & ! [X5] : (cP(X5,X4,X2) | ~INH(X0,X5)) & INH(X0,X4)) & Q(X0)) | ~
sP11(X2,X1,X0)) [nnf transformation 1073]
1424. ! [X0,X1,X2] : ((? [X3] : (SREG(X3,X1,X0) & ! [X4] : (cP(X4,X3,X0) | ~INH(X2,X4)) & INH(X2,X3)) & Q(X2)) | ~
sP11(X0,X1,X2)) [rectify 1423]
1425. ! [X2,X1,X0] : (? [X3] : (SREG(X3,X1,X0) & ! [X4] : (cP(X4,X3,X0) | ~INH(X2,X4)) & INH(X2,X3)) => (SREG(
sK165(X0,X1,X2),X1,X0) & ! [X4] : (cP(X4,sK165(X0,X1,X2),X0) | ~INH(X2,X4)) & INH(X2,sK165(X0,X1,X2)))) [
choice axiom]
1426. ! [X0,X1,X2] : (((SREG(sK165(X0,X1,X2),X1,X0) & ! [X4] : (cP(X4,sK165(X0,X1,X2),X0) | ~INH(X2,X4)) & INH(X2,
sK165(X0,X1,X2))) & Q(X2)) | ~sP11(X0,X1,X2)) [skolemisation 1424,1425]
1427. ! [X2,X1,X0] : ((? [X6] : (SREG(X6,X1,X2) & ! [X7] : (cP(X7,X6,X2) | ~GDEP(X0,X7,X2)) & GDEP(X0,X6,X2)) &
::(X0,gdcnt,X2)) | ~sP10(X2,X1,X0)) [nnf transformation 1072]
1428. ! [X0,X1,X2] : ((? [X3] : (SREG(X3,X1,X0) & ! [X4] : (cP(X4,X3,X0) | ~GDEP(X2,X4,X0)) & GDEP(X2,X3,X0)) &
::(X2,gdcnt,X0)) | ~sP10(X0,X1,X2)) [rectify 1427]
1429. ! [X2,X1,X0] : (? [X3] : (SREG(X3,X1,X0) & ! [X4] : (cP(X4,X3,X0) | ~GDEP(X2,X4,X0)) & GDEP(X2,X3,X0)) => (
SREG(sK166(X0,X1,X2),X1,X0) & ! [X4] : (cP(X4,sK166(X0,X1,X2),X0) | ~GDEP(X2,X4,X0)) & GDEP(X2,sK166(X0,X1,
X2),X0))) [choice axiom]
1430. ! [X0,X1,X2] : (((SREG(sK166(X0,X1,X2),X1,X0) & ! [X4] : (cP(X4,sK166(X0,X1,X2),X0) | ~GDEP(X2,X4,X0)) &
GDEP(X2,sK166(X0,X1,X2),X0)) & ::(X2,gdcnt,X0)) | ~sP10(X0,X1,X2)) [skolemisation 1428,1429]
1431. ! [X2,X1,X0] : (? [X3] : (SPROJ(X3,X1,X2) & STREG(X0,X3)) => (SPROJ(sK167(X0,X1,X2),X1,X2) & STREG(X0,sK167(
X0,X1,X2)))) [choice axiom]
1432. ! [X0,X1,X2] : (((SPROJ(sK167(X0,X1,X2),X1,X2) & STREG(X0,sK167(X0,X1,X2))) & PD(X0)) | sP11(X2,X1,X0) |
sP10(X2,X1,X0) | sP12(X2,X1,X0) | ~SLC(X0,X1,X2)) [skolemisation 1075,1431]
1433. ? [X0,X1,X2,X3,X4] : (~P(X3,X4) & SLC(X1,X4,X2) & SLC(X0,X3,X2) & PC(X0,X1,X2)) => (~P(sK171,sK172) & SLC(
sK169,sK172,sK170) & SLC(sK168,sK171,sK170) & PC(sK168,sK169,sK170)) [choice axiom]
1434. ~P(sK171,sK172) & SLC(sK169,sK172,sK170) & SLC(sK168,sK171,sK170) & PC(sK168,sK169,sK170) [skolemisation
1054,1433]
1482. ~EX(X0,X1) | PAR(X1) [cnf transformation 800]
1678. ~SREG(X0,X1,X2) | ::(X1,sreg,X2) [cnf transformation 902]
1719. ~SPROJ(X0,X1,X2) | ::(X1,sreg,X2) [cnf transformation 932]
1866. UNI(idcnt) [cnf transformation 191]
1876. ~UNI(X0) | ~PAR(X0) [cnf transformation 995]
2326. ~::(X0,X1,X2) | ::(X0,X1) [cnf transformation 1404]
2341. ~::(X0,sreg) | S(X0) [cnf transformation 1410]
2353. EX(sK163(X0,X2),X0) | P(X1,X2) | ~S(X2) | ~S(X1) [cnf transformation 1418]
2361. ~sP12(X0,X1,X2) | SREG(X2,X1,X0) [cnf transformation 1422]
2365. SREG(sK165(X0,X1,X2),X1,X0) | ~sP11(X0,X1,X2) [cnf transformation 1426]
2369. SREG(sK166(X0,X1,X2),X1,X0) | ~sP10(X0,X1,X2) [cnf transformation 1430]
2372. SPROJ(sK167(X0,X1,X2),X1,X2) | sP11(X2,X1,X0) | sP10(X2,X1,X0) | sP12(X2,X1,X0) | ~SLC(X0,X1,X2) [cnf
transformation 1432]
2374. SLC(sK168,sK171,sK170) [cnf transformation 1434]
2375. SLC(sK169,sK172,sK170) [cnf transformation 1434]
2376. ~P(sK171,sK172) [cnf transformation 1434]
2425. ~PAR(idcnt) [resolution 1876,1866]
4085. ~sP11(X11,X12,X13) | ::(X12,sreg,X11) [resolution 2365,1678]
4099. ~sP10(X11,X12,X13) | ::(X12,sreg,X11) [resolution 2369,1678]
4263. P(X3,X4) | ~S(X4) | ~S(X3) | PAR(X5) [resolution 2353,1482]
4272. 37 <=> ! [X1,X0] : (P(X0,X1) | ~S(X0) | ~S(X1)) [avatar definition]
4273. P(X0,X1) | ~S(X0) | ~S(X1) <- (37) [avatar component clause 4272]
4276. 38 <=> ! [X5] : PAR(X5) [avatar definition]
4277. PAR(X5) <- (38) [avatar component clause 4276]
4278. 38 | 37 [avatar split clause 4263,4272,4276]
4280. 39 <=> S(sK171) [avatar definition]
4281. S(sK171) <- (39) [avatar component clause 4280]
4284. 40 <=> S(sK172) [avatar definition]
5695. 113 <=> sP12(sK170,sK171,sK168) [avatar definition]
5697. sP12(sK170,sK171,sK168) <- (113) [avatar component clause 5695]
6304. sP11(X11,X12,X13) | sP10(X11,X12,X13) | sP12(X11,X12,X13) | ~SLC(X13,X12,X11) | ::(X12,sreg,X11) [resolution
2372,1719]
6308. sP10(X11,X12,X13) | sP12(X11,X12,X13) | ~SLC(X13,X12,X11) | ::(X12,sreg,X11) [subsumption resolution
6304,4085]
6309. ~SLC(X13,X12,X11) | sP12(X11,X12,X13) | ::(X12,sreg,X11) [subsumption resolution 6308,4099]
7058. ~S(sK171) | ~S(sK172) <- (37) [resolution 4273,2376]
12070. sP12(sK170,sK171,sK168) | ::(sK171,sreg,sK170) [resolution 6309,2374]
12071. sP12(sK170,sK172,sK169) | ::(sK172,sreg,sK170) [resolution 6309,2375]
12073. 307 <=> ::(sK171,sreg,sK170) [avatar definition]
12075. ::(sK171,sreg,sK170) <- (307) [avatar component clause 12073]
12076. 307 | 113 [avatar split clause 12070,5695,12073]
12078. 308 <=> ::(sK172,sreg,sK170) [avatar definition]
12080. ::(sK172,sreg,sK170) <- (308) [avatar component clause 12078]
12082. 309 <=> sP12(sK170,sK172,sK169) [avatar definition]
12084. sP12(sK170,sK172,sK169) <- (309) [avatar component clause 12082]
12085. 308 | 309 [avatar split clause 12071,12082,12078]
12086. SREG(sK168,sK171,sK170) <- (113) [resolution 5697,2361]
12134. 314 <=> ::(sK172,sreg) [avatar definition]
12135. ::(sK172,sreg) <- (314) [avatar component clause 12134]
12145. SREG(sK169,sK172,sK170) <- (309) [resolution 12084,2361]
12154. ::(sK171,sreg,sK170) <- (113) [resolution 12086,1678]
12158. 307 | ~113 [avatar split clause 12154,5695,12073]
12199. ::(sK171,sreg) <- (307) [resolution 12075,2326]
12256. S(sK171) <- (307) [resolution 12199,2341]
12260. 39 | ~307 [avatar split clause 12256,12073,4280]
12441. ::(sK172,sreg,sK170) <- (309) [resolution 12145,1678]
12460. 308 | ~309 [avatar split clause 12441,12082,12078]
12478. ::(sK172,sreg) <- (308) [resolution 12080,2326]
12484. 314 | ~308 [avatar split clause 12478,12078,12134]
12513. S(sK172) <- (314) [resolution 12135,2341]
```

```
12519. 40 | ~314 [avatar split clause 12513,12134,4284]
12525. ~S(sK172) <- (37, 39) [subsumption resolution 7058,4281]
12526. ~40 | ~37 | ~39 [avatar split clause 12525,4280,4272,4284]
12529. $false <- (38) [resolution 4277,2425]
12619. ~38 [avatar contradiction clause 12529]
12620. $false [avatar sat refutation 4278,12076,12085,12158,12260,12460,12484,12519,12526,12619]
```

Proof of Theorem (t_{bd}50)

```
30. ! [X17] : ! [X1] : (EX(X17,X1) => (::(X1,treg,X1) & PAR(X1) & PAR(X17))) [input]
101. ! [X0] : ! [X8] : ! [X1] : (SREG(X0,X8,X1) => (::(X1,treg,X1) & ::(X8,sreg,X1) & ~::(X0,sreg,X1) & ::(X0,
     idcnt,X1))) [input]
120. ! [X0] : ! [X8] : ! [X1] : (SPROJ(X0,X8,X1) => (::(X1,treg,X1) & ::(X8,sreg,X1) & ::(X0,streg,X1))) [input]
191. UNI(idcnt) [input]
201. ~? [X6] : (PAR(X6) & UNI(X6)) [input]
257. ! [X6,X16] : (::(X6,X16) <=> ? [X1] : ::(X6,X16,X1)) [input]
271. ! [X6] : (S(X6) <=> ::(X6,sreg)) [input]
276. ! [X6,X7] : (P(X6,X7) <=> ((! [X6] : (EX(X6,X1) => cP(X6,X7,X1)) & S(X7) & S(X6)) | (oP(X6,X7) & ((T(X7) & T(
     X6)) | (PD(X7) & PD(X6)))))) [input]
281. ! [X6,X5,X1] : (SLC(X6,X5,X1) <=> ((? [X7] : (SPROJ(X7,X5,X1) & STREG(X6,X7)) & PD(X6)) | (? [X7] : (SREG(X7,
     X5,X1) & ! [X13] : (INH(X6,X13) => cP(X13,X7,X1)) & INH(X6,X7)) & Q(X6)) | (? [X7] : (SREG(X7,X5,X1) & ! [
     X13] : (GDEP(X6,X13,X1) => cP(X13,X7,X1)) & GDEP(X6,X7,X1)) & ::(X6,gdcnt,X1)) | (SREG(X6,X5,X1) & ~::(X6,
     sreg,X1) & ::(X6,idcnt,X1)))) [input]
291. ! [X6,X7,X1,X4,X5] : ((SLC(X7,X5,X1) & SLC(X6,X4,X1) & tP(X6,X7,X1)) => P(X4,X5)) [input]
292. ~! [X6,X7,X1,X4,X5] : ((SLC(X7,X5,X1) & SLC(X6,X4,X1) & tP(X6,X7,X1)) => P(X4,X5)) [negated conjecture 291]
349. ! [X0] : ! [X1] : (EX(X0,X1) => (::(X1,treg,X1) & PAR(X1) & PAR(X0))) [rectify 30]
350. ! [X0,X1] : (EX(X0,X1) => (::(X1,treg,X1) & PAR(X1) & PAR(X0))) [flattening 349]
485. ! [X0] : ! [X1] : ! [X2] : (SREG(X0,X1,X2) => (::(X2,treg,X2) & ::(X1,sreg,X2) & ~::(X0,sreg,X2) & ::(X0,
     idcnt,X2))) [rectify 101]
486. ! [X0,X1,X2] : (SREG(X0,X1,X2) => (::(X2,treg,X2) & ::(X1,sreg,X2) & ~::(X0,sreg,X2) & ::(X0,idcnt,X2))) [
     flattening 485]
520. ! [X0] : ! [X1] : ! [X2] : (SPROJ(X0,X1,X2) => (::(X2,treg,X2) & ::(X1,sreg,X2) & ::(X0,streg,X2))) [rectify
     120]
521. ! [X0,X1,X2] : (SPROJ(X0,X1,X2) => (::(X2,treg,X2) & ::(X1,sreg,X2) & ::(X0,streg,X2))) [flattening 520]
616. ~? [X0] : (PAR(X0) & UNI(X0)) [rectify 201]
704. ! [X0,X1] : (::(X0,X1) <=> ? [X2] : ::(X0,X1,X2)) [rectify 257]
719. ! [X0] : (S(X0) <=> ::(X0,sreg)) [rectify 271]
724. ! [X0,X1] : (P(X0,X1) <=> ((! [X2] : (EX(X2,X3) => cP(X2,X1,X3)) & S(X1) & S(X0)) | (oP(X0,X1) & ((T(X1) & T(
     X0)) | (PD(X1) & PD(X0)))))) [rectify 276]
725. ! [X3] : ! [X0,X1] : (P(X0,X1) <=> ((! [X2] : (EX(X2,X3) => cP(X2,X1,X3)) & S(X1) & S(X0)) | (oP(X0,X1) & ((T
     (X1) & T(X0)) | (PD(X1) & PD(X0)))))) [closure 724]
726. ! [X3,X0,X1] : (P(X0,X1) <=> ((! [X2] : (EX(X2,X3) => cP(X2,X1,X3)) & S(X1) & S(X0)) | (oP(X0,X1) & ((T(X1) &
     T(X0)) | (PD(X1) & PD(X0)))))) [flattening 725]
731. ! [X0,X1,X2] : (SLC(X0,X1,X2) <=> ((? [X3] : (SPROJ(X3,X1,X2) & STREG(X0,X3)) & PD(X0)) | (? [X4] : (SREG(X4,
     X1,X2) & ! [X5] : (INH(X0,X5) => cP(X5,X4,X2)) & INH(X0,X4)) & Q(X0)) | (? [X6] : (SREG(X6,X1,X2) & ! [X7] :
     (GDEP(X0,X7,X2) => cP(X7,X6,X2)) & GDEP(X0,X6,X2)) & ::(X0,gdcnt,X2)) | (SREG(X0,X1,X2) & ~::(X0,sreg,X2) &
     ::(X0,idcnt,X2)))) [rectify 281]
741. ~! [X0,X1,X2,X3,X4] : ((SLC(X1,X4,X2) & SLC(X0,X3,X2) & tP(X0,X1,X2)) => P(X3,X4)) [rectify 292]
742. ! [X0,X1,X2] : (SLC(X0,X1,X2) => ((? [X3] : (SPROJ(X3,X1,X2) & STREG(X0,X3)) & PD(X0)) | (? [X4] : (SREG(X4,
     X1,X2) & ! [X5] : (INH(X0,X5) => cP(X5,X4,X2)) & INH(X0,X4)) & Q(X0)) | (? [X6] : (SREG(X6,X1,X2) & ! [X7] :
     (GDEP(X0,X7,X2) => cP(X7,X6,X2)) & GDEP(X0,X6,X2)) & ::(X0,gdcnt,X2)) | (SREG(X0,X1,X2) & ~::(X0,sreg,X2) &
     ::(X0,idcnt,X2)))) [unused predicate definition removal 731]
800. ! [X0,X1] : ((::(X1,treg,X1) & PAR(X1) & PAR(X0)) | ~EX(X0,X1)) [ennf transformation 350]
902. ! [X0,X1,X2] : ((::(X2,treg,X2) & ::(X1,sreg,X2) & ~::(X0,sreg,X2) & ::(X0,idcnt,X2)) | ~SREG(X0,X1,X2)) [
     ennf transformation 486]
932. ! [X0,X1,X2] : ((::(X2,treg,X2) & ::(X1,sreg,X2) & ::(X0,streg,X2)) | ~SPROJ(X0,X1,X2)) [ennf transformation
     521]
995. ! [X0] : (~PAR(X0) | ~UNI(X0)) [ennf transformation 616]
1049. ! [X3,X0,X1] : (P(X0,X1) <=> ((! [X2] : (cP(X2,X1,X3) | ~EX(X2,X3)) & S(X1) & S(X0)) | (oP(X0,X1) & ((T(X1)
      & T(X0)) | (PD(X1) & PD(X0)))))) [ennf transformation 726]
1051. ! [X0,X1,X2] : (((? [X3] : (SPROJ(X3,X1,X2) & STREG(X0,X3)) & PD(X0)) | (? [X4] : (SREG(X4,X1,X2) & ! [X5] :
      (cP(X5,X4,X2) | ~INH(X0,X5)) & INH(X0,X4)) & Q(X0)) | (? [X6] : (SREG(X6,X1,X2) & ! [X7] : (cP(X7,X6,X2) |
      ~GDEP(X0,X7,X2)) & GDEP(X0,X6,X2)) & ::(X0,gdcnt,X2)) | (SREG(X0,X1,X2) & ~::(X0,sreg,X2) & ::(X0,idcnt,X2))
     ) | ~SLC(X0,X1,X2)) [ennf transformation 742]
1052. ! [X0,X1,X2] : ((? [X3] : (SPROJ(X3,X1,X2) & STREG(X0,X3)) & PD(X0)) | (? [X4] : (SREG(X4,X1,X2) & ! [X5] :
      (cP(X5,X4,X2) | ~INH(X0,X5)) & INH(X0,X4)) & Q(X0)) | (? [X6] : (SREG(X6,X1,X2) & ! [X7] : (cP(X7,X6,X2) | ~
     GDEP(X0,X7,X2)) & GDEP(X0,X6,X2)) & ::(X0,gdcnt,X2)) | (SREG(X0,X1,X2) & ~::(X0,sreg,X2) & ::(X0,idcnt,X2))
     | ~SLC(X0,X1,X2)) [flattening 1051]
1053. ? [X0,X1,X2,X3,X4] : (~P(X3,X4) & (SLC(X1,X4,X2) & SLC(X0,X3,X2) & tP(X0,X1,X2))) [ennf transformation 741]
1054. ? [X0,X1,X2,X3,X4] : (~P(X3,X4) & SLC(X1,X4,X2) & SLC(X0,X3,X2) & tP(X0,X1,X2)) [flattening 1053]
1070. ! [X1,X0] : (sP9(X1,X0) <=> (oP(X0,X1) & ((T(X1) & T(X0)) | (PD(X1) & PD(X0))))) [predicate definition
      introduction]
1071. ! [X3,X0,X1] : (P(X0,X1) <=> ((! [X2] : (cP(X2,X1,X3) | ~EX(X2,X3)) & S(X1) & S(X0)) | sP9(X1,X0))) [
      definition folding 1049,1070]
1072. ! [X2,X1,X0] : ((? [X6] : (SREG(X6,X1,X2) & ! [X7] : (cP(X7,X6,X2) | ~GDEP(X0,X7,X2)) & GDEP(X0,X6,X2)) &
      ::(X0,gdcnt,X2)) | ~sP10(X2,X1,X0)) [predicate definition introduction]
1073. ! [X2,X1,X0] : ((? [X4] : (SREG(X4,X1,X2) & ! [X5] : (cP(X5,X4,X2) | ~INH(X0,X5)) & INH(X0,X4)) & Q(X0)) | ~
      sP11(X2,X1,X0)) [predicate definition introduction]
1074. ! [X2,X1,X0] : ((SREG(X0,X1,X2) & ~::(X0,sreg,X2) & ::(X0,idcnt,X2)) | ~sP12(X2,X1,X0)) [predicate
      definition introduction]
1075. ! [X0,X1,X2] : ((? [X3] : (SPROJ(X3,X1,X2) & STREG(X0,X3)) & PD(X0)) | sP11(X2,X1,X0) | sP10(X2,X1,X0) |
      sP12(X2,X1,X0) | ~SLC(X0,X1,X2)) [definition folding 1052,1074,1073,1072]
1401. ! [X0,X1] : ((::(X0,X1) | ! [X2] : ~::(X0,X1,X2)) & (? [X2] : ::(X0,X1,X2) | ~::(X0,X1))) [nnf
      transformation 704]
1402. ! [X0,X1] : ((::(X0,X1) | ! [X2] : ~::(X0,X1,X2)) & (? [X3] : ::(X0,X1,X3) | ~::(X0,X1))) [rectify 1401]
1403. ! [X1,X0] : (? [X3] : ::(X0,X1,X3) => ::(X0,X1,sK162(X0,X1))) [choice axiom]
1404. ! [X0,X1] : ((::(X0,X1) | ! [X2] : ~::(X0,X1,X2)) & (::(X0,X1,sK162(X0,X1)) | ~::(X0,X1))) [skolemisation
      1402,1403]
1410. ! [X0] : ((S(X0) | ~::(X0,sreg)) & (::(X0,sreg) | ~S(X0))) [nnf transformation 719]
```

```
1414. ! [X3,X0,X1] : ((P(X0,X1) | ((? [X2] : (~cP(X2,X1,X3) & EX(X2,X3)) | ~S(X1) | ~S(X0)) & ~sP9(X1,X0))) & (((!
      [X2] : (cP(X2,X1,X3) | ~EX(X2,X3)) & S(X1) & S(X0)) | sP9(X1,X0)) | ~P(X0,X1))) [nnf transformation 1071]
1415. ! [X3,X0,X1] : ((P(X0,X1) | ((? [X2] : (~cP(X2,X1,X3) & EX(X2,X3)) | ~S(X1) | ~S(X0)) & ~sP9(X1,X0))) & ((!
      [X2] : (cP(X2,X1,X3) | ~EX(X2,X3)) & S(X1) & S(X0)) | sP9(X1,X0) | ~P(X0,X1))) [flattening 1414]
1416. ! [X0,X1,X2] : ((P(X1,X2) | ((? [X3] : (~cP(X3,X2,X0) & EX(X3,X0)) | ~S(X2) | ~S(X1)) & ~sP9(X2,X1))) & ((!
      [X4] : (cP(X4,X2,X0) | ~EX(X4,X0)) & S(X2) & S(X1)) | sP9(X2,X1) | ~P(X1,X2))) [rectify 1415]
1417. ! [X2,X0] : (? [X3] : (~cP(X3,X2,X0) & EX(X3,X0)) => (~cP(sK163(X0,X2),X2,X0) & EX(sK163(X0,X2),X0))) [
      choice axiom]
1418. ! [X0,X1,X2] : ((P(X1,X2) | (((~cP(sK163(X0,X2),X2,X0) & EX(sK163(X0,X2),X0)) | ~S(X2) | ~S(X1)) & ~sP9(X2,
      X1))) & ((! [X4] : (cP(X4,X2,X0) | ~EX(X4,X0)) & S(X2) & S(X1)) | sP9(X2,X1) | ~P(X1,X2))) [skolemisation
      1416,1417]
1419. ! [X2,X1,X0] : ((SREG(X0,X1,X2) & ~::(X0,sreg,X2) & ::(X0,idcnt,X2)) | ~sP12(X2,X1,X0)) [nnf transformation
      1074]
1420. ! [X0,X1,X2] : ((SREG(X2,X1,X0) & ~::(X2,sreg,X0) & ::(X2,idcnt,X0)) | ~sP12(X0,X1,X2)) [rectify 1419]
1421. ! [X2,X1,X0] : ((? [X4] : (SREG(X4,X1,X2) & ! [X5] : (cP(X5,X4,X2) | ~INH(X0,X5)) & INH(X0,X4)) & Q(X0)) | ~
      sP11(X2,X1,X0)) [nnf transformation 1073]
1422. ! [X0,X1,X2] : ((? [X3] : (SREG(X3,X1,X0) & ! [X4] : (cP(X4,X3,X0) | ~INH(X2,X4)) & INH(X2,X3)) & Q(X2)) | ~
      sP11(X0,X1,X2)) [rectify 1421]
1423. ! [X2,X1,X0] : (? [X3] : (SREG(X3,X1,X0) & ! [X4] : (cP(X4,X3,X0) | ~INH(X2,X4)) & INH(X2,X3)) => (SREG(
      sK164(X0,X1,X2),X1,X0) & ! [X4] : (cP(X4,sK164(X0,X1,X2),X0) | ~INH(X2,X4)) & INH(X2,sK164(X0,X1,X2)))) [
      choice axiom]
1424. ! [X0,X1,X2] : (((SREG(sK164(X0,X1,X2),X1,X0) & ! [X4] : (cP(X4,sK164(X0,X1,X2),X0) | ~INH(X2,X4)) & INH(X2,
      sK164(X0,X1,X2))) & Q(X2)) | ~sP11(X0,X1,X2)) [skolemisation 1422,1423]
1425. ! [X2,X1,X0] : ((? [X6] : (SREG(X6,X1,X2) & ! [X7] : (cP(X7,X6,X2) | ~GDEP(X0,X7,X2)) & GDEP(X0,X6,X2)) &
      ::(X0,gdcnt,X2)) | ~sP10(X2,X1,X0)) [nnf transformation 1072]
1426. ! [X0,X1,X2] : ((? [X3] : (SREG(X3,X1,X0) & ! [X4] : (cP(X4,X3,X0) | ~GDEP(X2,X4,X0)) & GDEP(X2,X3,X0)) &
      ::(X2,gdcnt,X0)) | ~sP10(X0,X1,X2)) [rectify 1425]
1427. ! [X2,X1,X0] : (? [X3] : (SREG(X3,X1,X0) & ! [X4] : (cP(X4,X3,X0) | ~GDEP(X2,X4,X0)) & GDEP(X2,X3,X0)) => (
      SREG(sK165(X0,X1,X2),X1,X0) & ! [X4] : (cP(X4,sK165(X0,X1,X2),X0) | ~GDEP(X2,X4,X0)) & GDEP(X2,sK165(X0,X1,
      X2),X0))) [choice axiom]
1428. ! [X0,X1,X2] : (((SREG(sK165(X0,X1,X2),X1,X0) & ! [X4] : (cP(X4,sK165(X0,X1,X2),X0) | ~GDEP(X2,X4,X0)) &
      GDEP(X2,sK165(X0,X1,X2),X0)) & ::(X2,gdcnt,X0)) | ~sP10(X0,X1,X2)) [skolemisation 1426,1427]
1429. ! [X2,X1,X0] : (? [X3] : (SPROJ(X3,X1,X2) & STREG(X0,X3)) => (SPROJ(sK166(X0,X1,X2),X1,X2) & STREG(X0,sK166(
      X0,X1,X2)))) [choice axiom]
1430. ! [X0,X1,X2] : (((SPROJ(sK166(X0,X1,X2),X1,X2) & STREG(X0,sK166(X0,X1,X2))) & PD(X0)) | sP11(X2,X1,X0) |
      sP10(X2,X1,X0) | sP12(X2,X1,X0) | ~SLC(X0,X1,X2)) [skolemisation 1075,1429]
1431. ? [X0,X1,X2,X3,X4] : (~P(X3,X4) & SLC(X1,X4,X2) & SLC(X0,X3,X2) & tP(X0,X1,X2)) => (~P(sK170,sK171) & SLC(
      sK168,sK171,sK169) & SLC(sK167,sK170,sK169) & tP(sK167,sK168,sK169)) [choice axiom]
1432. ~P(sK170,sK171) & SLC(sK168,sK171,sK169) & SLC(sK167,sK170,sK169) & tP(sK167,sK168,sK169) [skolemisation
      1054,1431]
1480. ~EX(X0,X1) | PAR(X1) [cnf transformation 800]
1676. ~SREG(X0,X1,X2) | ::(X1,sreg,X2) [cnf transformation 902]
1717. ~SPROJ(X0,X1,X2) | ::(X1,sreg,X2) [cnf transformation 932]
1864. UNI(idcnt) [cnf transformation 191]
1874. ~UNI(X0) | ~PAR(X0) [cnf transformation 995]
2324. ~::(X0,X1,X2) | ::(X0,X1) [cnf transformation 1404]
2339. ~::(X0,sreg) | S(X0) [cnf transformation 1410]
2351. EX(sK163(X0,X2),X0) | P(X1,X2) | ~S(X2) | ~S(X1) [cnf transformation 1418]
2358. ~sP12(X0,X1,X2) | SREG(X2,X1,X0) [cnf transformation 1420]
2362. SREG(sK164(X0,X1,X2),X1,X0) | ~sP11(X0,X1,X2) [cnf transformation 1424]
2366. SREG(sK165(X0,X1,X2),X1,X0) | ~sP10(X0,X1,X2) [cnf transformation 1428]
2369. SPROJ(sK166(X0,X1,X2),X1,X2) | sP11(X2,X1,X0) | sP10(X2,X1,X0) | sP12(X2,X1,X0) | ~SLC(X0,X1,X2) [cnf
      transformation 1430]
2371. SLC(sK167,sK170,sK169) [cnf transformation 1432]
2372. SLC(sK168,sK171,sK169) [cnf transformation 1432]
2373. ~P(sK170,sK171) [cnf transformation 1432]
2422. ~PAR(idcnt) [resolution 1874,1864]
4242. ~sP11(X11,X12,X13) | ::(X12,sreg,X11) [resolution 2362,1676]
4255. ~sP10(X11,X12,X13) | ::(X12,sreg,X11) [resolution 2366,1676]
4506. EX(sK163(X0,sK171),X0) | ~S(sK171) | ~S(sK170) [resolution 2351,2373]
4517. 66 <=> ! [X5] : PAR(X5) [avatar definition]
4518. PAR(X5) <- (66) [avatar component clause 4517]
4521. 67 <=> S(sK170) [avatar definition]
4525. 68 <=> S(sK171) [avatar definition]
4529. 69 <=> ! [X0] : EX(sK163(X0,sK171),X0) [avatar definition]
4530. EX(sK163(X0,sK171),X0) <- (69) [avatar component clause 4529]
4531. ~67 | ~68 | 69 [avatar split clause 4506,4529,4525,4521]
5639. 124 <=> sP12(sK169,sK170,sK167) [avatar definition]
5641. sP12(sK169,sK170,sK167) <- (124) [avatar component clause 5639]
5656. 128 <=> sP12(sK169,sK171,sK168) [avatar definition]
5658. sP12(sK169,sK171,sK168) <- (128) [avatar component clause 5656]
6245. sP11(X11,X12,X13) | sP10(X11,X12,X13) | sP12(X11,X12,X13) | ~SLC(X13,X12,X11) | ::(X12,sreg,X11) [resolution
      2369,1717]
6249. sP10(X11,X12,X13) | sP12(X11,X12,X13) | ~SLC(X13,X12,X11) | ::(X12,sreg,X11) [subsumption resolution
      6245,4242]
6250. ~SLC(X13,X12,X11) | sP12(X11,X12,X13) | ::(X12,sreg,X11) [subsumption resolution 6249,4255]
12046. sP12(sK169,sK170,sK167) | ::(sK170,sreg,sK169) [resolution 6250,2371]
12047. sP12(sK169,sK171,sK168) | ::(sK171,sreg,sK169) [resolution 6250,2372]
12049. 346 <=> ::(sK170,sreg,sK169) [avatar definition]
12050. ~::(sK170,sreg,sK169) <- (~346) [avatar component clause 12049]
12051. ::(sK170,sreg,sK169) <- (346) [avatar component clause 12049]
12052. 346 | 124 [avatar split clause 12046,5639,12049]
12054. 347 <=> ::(sK171,sreg,sK169) [avatar definition]
12056. ::(sK171,sreg,sK169) <- (347) [avatar component clause 12054]
12057. 347 | 128 [avatar split clause 12047,5656,12054]
12254. ::(sK170,sreg) <- (346) [resolution 12051,2324]
12328. S(sK170) <- (346) [resolution 12254,2339]
12332. 67 | ~346 [avatar split clause 12328,12049,4521]
12360. ::(sK171,sreg) <- (347) [resolution 12056,2324]
12398. S(sK171) <- (347) [resolution 12360,2339]
12404. 68 | ~347 [avatar split clause 12398,12054,4525]
```

```
12414. $false <- (66) [resolution 4518,2422]
12496. ~66 [avatar contradiction clause 12414]
12497. SREG(sK168,sK171,sK169) <- (128) [resolution 5658,2358]
12579. PAR(X1) <- (69) [resolution 4530,1480]
12582. 66 | ~69 [avatar split clause 12579,4529,4517]
13037. ::(sK171,sreg,sK169) <- (128) [resolution 12497,1676]
13045. 347 | ~128 [avatar split clause 13037,5656,12054]
13048. SREG(sK167,sK170,sK169) <- (124) [resolution 5641,2358]
13156. ::(sK170,sreg,sK169) <- (124) [resolution 13048,1676]
13160. $false <- (124, ~346) [subsumption resolution 13156,12050]
13161. ~124 | 346 [avatar contradiction clause 13160]
13162. $false [avatar sat refutation 4531,12052,12057,12332,12404,12496,12582,13045,13161]
```

Proof of Theorem ($t_{bd}52$)

```
30. ! [X17] : ! [X1] : (EX(X17,X1) => (::(X1,treg,X1) & PAR(X1) & PAR(X17))) [input]
101. ! [X0] : ! [X8] : ! [X1] : (SREG(X0,X8,X1) => (::(X1,treg,X1) & ::(X8,sreg,X1) & ~::(X0,sreg,X1) & ::(X0,
       idcnt,X1))) [input]
120. ! [X0] : ! [X8] : ! [X1] : (SPROJ(X0,X8,X1) => (::(X1,treg,X1) & ::(X8,sreg,X1) & ::(X0,streg,X1))) [input]
191. UNI(idcnt) [input]
201. ~? [X6] : (PAR(X6) & UNI(X6)) [input]
257. ! [X6,X16] : (::(X6,X16) <=> ? [X1] : ::(X6,X16,X1)) [input]
271. ! [X6] : (S(X6) <=> ::(X6,sreg)) [input]
276. ! [X6,X7] : (P(X6,X7) <=> ((! [X6] : (EX(X6,X1) => cP(X6,X7,X1)) & S(X7) & S(X6)) | (oP(X6,X7) & ((T(X7) & T(
       X6)) | (PD(X7) & PD(X6)))))) [input]
281. ! [X6,X5,X1] : (SLC(X6,X5,X1) <=> ((? [X7] : (SPROJ(X7,X5,X1) & STREG(X6,X7)) & PD(X6)) | (? [X7] : (SREG(X7,
       X5,X1) & ! [X13] : (INH(X6,X13) => cP(X13,X7,X1)) & INH(X6,X7)) & Q(X6)) | (? [X7] : (SREG(X7,X5,X1) & ! [
       X13] : (GDEP(X6,X13,X1) => cP(X13,X7,X1)) & GDEP(X6,X7,X1)) & ::(X6,gdcnt,X1)) | (SREG(X6,X5,X1) & ~::(X6,
       sreg,X1) & ::(X6,idcnt,X1)))) [input]
291. ! [X6,X7,X5,X1,X4] : ((SLC(X7,X4,X1) & SLC(X6,X5,X1) & P(X6,X7)) => P(X5,X4)) [input]
292. ~! [X6,X7,X5,X1,X4] : ((SLC(X7,X4,X1) & SLC(X6,X5,X1) & P(X6,X7)) => P(X5,X4)) [negated conjecture 291]
349. ! [X0] : ! [X1] : (EX(X0,X1) => (::(X1,treg,X1) & PAR(X1) & PAR(X0))) [rectify 30]
350. ! [X0,X1] : (EX(X0,X1) => (::(X1,treg,X1) & PAR(X1) & PAR(X0))) [flattening 349]
485. ! [X0] : ! [X1] : ! [X2] : (SREG(X0,X1,X2) => (::(X2,treg,X2) & ::(X1,sreg,X2) & ~::(X0,sreg,X2) & ::(X0,
       idcnt,X2))) [rectify 101]
486. ! [X0,X1,X2] : (SREG(X0,X1,X2) => (::(X2,treg,X2) & ::(X1,sreg,X2) & ~::(X0,sreg,X2) & ::(X0,idcnt,X2))) [
       flattening 485]
520. ! [X0] : ! [X1] : ! [X2] : (SPROJ(X0,X1,X2) => (::(X2,treg,X2) & ::(X1,sreg,X2) & ::(X0,streg,X2))) [rectify
       120]
521. ! [X0,X1,X2] : (SPROJ(X0,X1,X2) => (::(X2,treg,X2) & ::(X1,sreg,X2) & ::(X0,streg,X2))) [flattening 520]
616. ~? [X0] : (PAR(X0) & UNI(X0)) [rectify 201]
704. ! [X0,X1] : (::(X0,X1) <=> ? [X2] : ::(X0,X1,X2)) [rectify 257]
719. ! [X0] : (S(X0) <=> ::(X0,sreg)) [rectify 271]
724. ! [X0,X1] : (P(X0,X1) <=> ((! [X2] : (EX(X2,X3) => cP(X2,X1,X3)) & S(X1) & S(X0)) | (oP(X0,X1) & ((T(X1) & T(
       X0)) | (PD(X1) & PD(X0)))))) [rectify 276]
725. ! [X3] : ! [X0,X1] : (P(X0,X1) <=> ((! [X2] : (EX(X2,X3) => cP(X2,X1,X3)) & S(X1) & S(X0)) | (oP(X0,X1) & ((T
       (X1) & T(X0)) | (PD(X1) & PD(X0)))))) [closure 724]
726. ! [X3,X0,X1] : (P(X0,X1) <=> ((! [X2] : (EX(X2,X3) => cP(X2,X1,X3)) & S(X1) & S(X0)) | (oP(X0,X1) & ((T(X1) &
       T(X0)) | (PD(X1) & PD(X0)))))) [flattening 725]
731. ! [X0,X1,X2] : (SLC(X0,X1,X2) <=> ((? [X3] : (SPROJ(X3,X1,X2) & STREG(X0,X3)) & PD(X0)) | (? [X4] : (SREG(X4,
       X1,X2) & ! [X5] : (INH(X0,X5) => cP(X5,X4,X2)) & INH(X0,X4)) & Q(X0)) | (? [X6] : (SREG(X6,X1,X2) & ! [X7] :
       (GDEP(X0,X7,X2) => cP(X7,X6,X2)) & GDEP(X0,X6,X2)) & ::(X0,gdcnt,X2)) | (SREG(X0,X1,X2) & ~::(X0,sreg,X2) &
       ::(X0,idcnt,X2)))) [rectify 281]
741. ~! [X0,X1,X2,X3,X4] : ((SLC(X1,X4,X3) & SLC(X0,X2,X3) & P(X0,X1)) => P(X2,X4)) [rectify 292]
742. ! [X0,X1,X2] : (SLC(X0,X1,X2) => ((? [X3] : (SPROJ(X3,X1,X2) & STREG(X0,X3)) & PD(X0)) | (? [X4] : (SREG(X4,
       X1,X2) & ! [X5] : (INH(X0,X5) => cP(X5,X4,X2)) & INH(X0,X4)) & Q(X0)) | (? [X6] : (SREG(X6,X1,X2) & ! [X7] :
       (GDEP(X0,X7,X2) => cP(X7,X6,X2)) & GDEP(X0,X6,X2)) & ::(X0,gdcnt,X2)) | (SREG(X0,X1,X2) & ~::(X0,sreg,X2) &
       ::(X0,idcnt,X2)))) [unused predicate definition removal 731]
798. ! [X0,X1] : ((::(X1,treg,X1) & PAR(X1) & PAR(X0)) | ~EX(X0,X1)) [ennf transformation 350]
900. ! [X0,X1,X2] : ((::(X2,treg,X2) & ::(X1,sreg,X2) & ~::(X0,sreg,X2) & ::(X0,idcnt,X2)) | ~SREG(X0,X1,X2)) [
       ennf transformation 486]
930. ! [X0,X1,X2] : ((::(X2,treg,X2) & ::(X1,sreg,X2) & ::(X0,streg,X2)) | ~SPROJ(X0,X1,X2)) [ennf transformation
       521]
993. ! [X0] : (~PAR(X0) | ~UNI(X0)) [ennf transformation 616]
1046. ! [X3,X0,X1] : (P(X0,X1) <=> ((! [X2] : (cP(X2,X1,X3) | ~EX(X2,X3)) & S(X1) & S(X0)) | (oP(X0,X1) & ((T(X1)
       & T(X0)) | (PD(X1) & PD(X0)))))) [ennf transformation 726]
1047. ! [X0,X1,X2] : (((? [X3] : (SPROJ(X3,X1,X2) & STREG(X0,X3)) & PD(X0)) | (? [X4] : (SREG(X4,X1,X2) & ! [X5] :
       (cP(X5,X4,X2) | ~INH(X0,X5)) & INH(X0,X4)) & Q(X0)) | (? [X6] : (SREG(X6,X1,X2) & ! [X7] : (cP(X7,X6,X2) |
       ~GDEP(X0,X7,X2)) & GDEP(X0,X6,X2)) & ::(X0,gdcnt,X2)) | (SREG(X0,X1,X2) & ~::(X0,sreg,X2) & ::(X0,idcnt,X2))
       ) | ~SLC(X0,X1,X2)) [ennf transformation 742]
1048. ! [X0,X1,X2] : ((? [X3] : (SPROJ(X3,X1,X2) & STREG(X0,X3)) & PD(X0)) | (? [X4] : (SREG(X4,X1,X2) & ! [X5] :
       (cP(X5,X4,X2) | ~INH(X0,X5)) & INH(X0,X4)) & Q(X0)) | (? [X6] : (SREG(X6,X1,X2) & ! [X7] : (cP(X7,X6,X2) | ~
       GDEP(X0,X7,X2)) & GDEP(X0,X6,X2)) & ::(X0,gdcnt,X2)) | (SREG(X0,X1,X2) & ~::(X0,sreg,X2) & ::(X0,idcnt,X2))
       | ~SLC(X0,X1,X2)) [flattening 1047]
1049. ? [X0,X1,X2,X3,X4] : (~P(X2,X4) & (SLC(X1,X4,X3) & SLC(X0,X2,X3) & P(X0,X1))) [ennf transformation 741]
1050. ? [X0,X1,X2,X3,X4] : (~P(X2,X4) & SLC(X1,X4,X3) & SLC(X0,X2,X3) & P(X0,X1)) [flattening 1049]
1066. ! [X1,X0] : (sP9(X1,X0) <=> (oP(X0,X1) & ((T(X1) & T(X0)) | (PD(X1) & PD(X0))))) [predicate definition
       introduction]
1067. ! [X3,X0,X1] : (P(X0,X1) <=> ((! [X2] : (cP(X2,X1,X3) | ~EX(X2,X3)) & S(X1) & S(X0)) | sP9(X1,X0))) [
       definition folding 1046,1066]
1068. ! [X2,X1,X0] : ((? [X6] : (SREG(X6,X1,X2) & ! [X7] : (cP(X7,X6,X2) | ~GDEP(X0,X7,X2)) & GDEP(X0,X6,X2)) &
       ::(X0,gdcnt,X2)) | ~sP10(X2,X1,X0)) [predicate definition introduction]
1069. ! [X2,X1,X0] : ((? [X4] : (SREG(X4,X1,X2) & ! [X5] : (cP(X5,X4,X2) | ~INH(X0,X5)) & INH(X0,X4)) & Q(X0)) | ~
       sP11(X2,X1,X0)) [predicate definition introduction]
1070. ! [X2,X1,X0] : ((SREG(X0,X1,X2) & ~::(X0,sreg,X2) & ::(X0,idcnt,X2)) | ~sP12(X2,X1,X0)) [predicate
       definition introduction]
1071. ! [X0,X1,X2] : ((? [X3] : (SPROJ(X3,X1,X2) & STREG(X0,X3)) & PD(X0)) | sP11(X2,X1,X0) | sP10(X2,X1,X0) |
       sP12(X2,X1,X0) | ~SLC(X0,X1,X2)) [definition folding 1048,1070,1069,1068]
1397. ! [X0,X1] : ((::(X0,X1) | ! [X2] : ~::(X0,X1,X2)) & (? [X2] : ::(X0,X1,X2) | ~::(X0,X1))) [nnf
       transformation 704]
```

```
1398. ! [X0,X1] : ((::(X0,X1) | ! [X2] : ~::(X0,X1,X2)) & (? [X3] : ::(X0,X1,X3) | ~::(X0,X1))) [rectify 1397]
1399. ! [X1,X0] : (? [X3] : ::(X0,X1,X3) => ::(X0,X1,sK162(X0,X1))) [choice axiom]
1400. ! [X0,X1] : ((::(X0,X1) | ! [X2] : ~::(X0,X1,X2)) & (::(X0,X1,sK162(X0,X1)) | ~::(X0,X1))) [skolemisation
      1398,1399]
1406. ! [X0] : ((S(X0) | ~::(X0,sreg)) & (::(X0,sreg) | ~S(X0))) [nnf transformation 719]
1410. ! [X3,X0,X1] : ((P(X0,X1) | ((? [X2] : (~cP(X2,X1,X3) & EX(X2,X3)) | ~S(X1) | ~S(X0)) & ~sP9(X1,X0))) & (((!
      [X2] : (cP(X2,X1,X3) | ~EX(X2,X3)) & S(X1) & S(X0)) | sP9(X1,X0)) | ~P(X0,X1))) [nnf transformation 1067]
1411. ! [X3,X0,X1] : ((P(X0,X1) | ((? [X2] : (~cP(X2,X1,X3) & EX(X2,X3)) | ~S(X1) | ~S(X0)) & ~sP9(X1,X0))) & ((!
      [X2] : (cP(X2,X1,X3) | ~EX(X2,X3)) & S(X1) & S(X0)) | sP9(X1,X0) | ~P(X0,X1))) [flattening 1410]
1412. ! [X0,X1,X2] : ((P(X1,X2) | ((? [X3] : (~cP(X3,X2,X0) & EX(X3,X0)) | ~S(X2) | ~S(X1)) & ~sP9(X2,X1))) & ((!
      [X4] : (cP(X4,X2,X0) | ~EX(X4,X0)) & S(X2) & S(X1)) | sP9(X2,X1) | ~P(X1,X2))) [rectify 1411]
1413. ! [X2,X0] : (? [X3] : (~cP(X3,X2,X0) & EX(X3,X0)) => (~cP(sK163(X0,X2),X2,X0) & EX(sK163(X0,X2),X0))) [
      choice axiom]
1414. ! [X0,X1,X2] : ((P(X1,X2) | (((~cP(sK163(X0,X2),X2,X0) & EX(sK163(X0,X2),X0)) | ~S(X2) | ~S(X1)) & ~sP9(X2,
      X1))) & ((! [X4] : (cP(X4,X2,X0) | ~EX(X4,X0)) & S(X2) & S(X1)) | sP9(X2,X1) | ~P(X1,X2))) [skolemisation
      1412,1413]
1415. ! [X2,X1,X0] : ((SREG(X0,X1,X2) & ~::(X0,sreg,X2) & ::(X0,idcnt,X2)) | ~sP12(X2,X1,X0)) [nnf transformation
      1070]
1416. ! [X0,X1,X2] : ((SREG(X2,X1,X0) & ~::(X2,sreg,X0) & ::(X2,idcnt,X0)) | ~sP12(X0,X1,X2)) [rectify 1415]
1417. ! [X2,X1,X0] : ((? [X4] : (SREG(X4,X1,X2) & ! [X5] : (cP(X5,X4,X2) | ~INH(X0,X5)) & INH(X0,X4)) & Q(X0)) | ~
      sP11(X2,X1,X0)) [nnf transformation 1069]
1418. ! [X0,X1,X2] : ((? [X3] : (SREG(X3,X1,X0) & ! [X4] : (cP(X4,X3,X0) | ~INH(X2,X4)) & INH(X2,X3)) & Q(X2)) | ~
      sP11(X0,X1,X2)) [rectify 1417]
1419. ! [X2,X1,X0] : (? [X3] : (SREG(X3,X1,X0) & ! [X4] : (cP(X4,X3,X0) | ~INH(X2,X4)) & INH(X2,X3)) => (SREG(
      sK164(X0,X1,X2),X1,X0) & ! [X4] : (cP(X4,sK164(X0,X1,X2),X0) | ~INH(X2,X4)) & INH(X2,sK164(X0,X1,X2)))) [
      choice axiom]
1420. ! [X0,X1,X2] : (((SREG(sK164(X0,X1,X2),X1,X0) & ! [X4] : (cP(X4,sK164(X0,X1,X2),X0) | ~INH(X2,X4)) & INH(X2,
      sK164(X0,X1,X2))) & Q(X2)) | ~sP11(X0,X1,X2)) [skolemisation 1418,1419]
1421. ! [X2,X1,X0] : ((? [X6] : (SREG(X6,X1,X2) & ! [X7] : (cP(X7,X6,X2) | ~GDEP(X0,X7,X2)) & GDEP(X0,X6,X2)) &
      ::(X0,gdcnt,X2)) | ~sP10(X2,X1,X0)) [nnf transformation 1068]
1422. ! [X0,X1,X2] : ((? [X3] : (SREG(X3,X1,X0) & ! [X4] : (cP(X4,X3,X0) | ~GDEP(X2,X4,X0)) & GDEP(X2,X3,X0)) &
      ::(X2,gdcnt,X0)) | ~sP10(X0,X1,X2)) [rectify 1421]
1423. ! [X2,X1,X0] : (? [X3] : (SREG(X3,X1,X0) & ! [X4] : (cP(X4,X3,X0) | ~GDEP(X2,X4,X0)) & GDEP(X2,X3,X0)) => (
      SREG(sK165(X0,X1,X2),X1,X0) & ! [X4] : (cP(X4,sK165(X0,X1,X2),X0) | ~GDEP(X2,X4,X0)) & GDEP(X2,sK165(X0,X1,
      X2),X0))) [choice axiom]
1424. ! [X0,X1,X2] : (((SREG(sK165(X0,X1,X2),X1,X0) & ! [X4] : (cP(X4,sK165(X0,X1,X2),X0) | ~GDEP(X2,X4,X0)) &
      GDEP(X2,sK165(X0,X1,X2),X0)) & ::(X2,gdcnt,X0)) | ~sP10(X0,X1,X2)) [skolemisation 1422,1423]
1425. ! [X2,X1,X0] : (? [X3] : (SPROJ(X3,X1,X2) & STREG(X0,X3)) => (SPROJ(sK166(X0,X1,X2),X1,X2) & STREG(X0,sK166(
      X0,X1,X2)))) [choice axiom]
1426. ! [X0,X1,X2] : (((SPROJ(sK166(X0,X1,X2),X1,X2) & STREG(X0,sK166(X0,X1,X2))) & PD(X0)) | sP11(X2,X1,X0) |
      sP10(X2,X1,X0) | sP12(X2,X1,X0) | ~SLC(X0,X1,X2)) [skolemisation 1071,1425]
1427. ? [X0,X1,X2,X3,X4] : (~P(X2,X4) & SLC(X1,X4,X3) & SLC(X0,X2,X3) & P(X0,X1)) => (~P(sK169,sK171) & SLC(sK168,
      sK171,sK170) & SLC(sK167,sK169,sK170) & P(sK167,sK168)) [choice axiom]
1428. ~P(sK169,sK171) & SLC(sK168,sK171,sK170) & SLC(sK167,sK169,sK170) & P(sK167,sK168) [skolemisation 1050,1427]
1476. ~EX(X0,X1) | PAR(X1) [cnf transformation 798]
1672. ~SREG(X0,X1,X2) | ::(X1,sreg,X2) [cnf transformation 900]
1713. ~SPROJ(X0,X1,X2) | ::(X1,sreg,X2) [cnf transformation 930]
1860. UNI(idcnt) [cnf transformation 191]
1870. ~UNI(X0) | ~PAR(X0) [cnf transformation 993]
2320. ~::(X0,X1,X2) | ::(X0,X1) [cnf transformation 1400]
2332. ~::(X0,sreg) | S(X0) [cnf transformation 1406]
2344. EX(sK163(X0,X2),X0) | P(X1,X2) | ~S(X2) | ~S(X1) [cnf transformation 1414]
2348. ~sP12(X0,X1,X2) | SREG(X2,X1,X0) [cnf transformation 1416]
2352. SREG(sK164(X0,X1,X2),X1,X0) | ~sP11(X0,X1,X2) [cnf transformation 1420]
2356. SREG(sK165(X0,X1,X2),X1,X0) | ~sP10(X0,X1,X2) [cnf transformation 1424]
2359. SPROJ(sK166(X0,X1,X2),X1,X2) | sP11(X2,X1,X0) | sP10(X2,X1,X0) | sP12(X2,X1,X0) | ~SLC(X0,X1,X2) [cnf
      transformation 1426]
2361. SLC(sK167,sK169,sK170) [cnf transformation 1428]
2362. SLC(sK168,sK171,sK170) [cnf transformation 1428]
2363. ~P(sK169,sK171) [cnf transformation 1428]
2412. ~PAR(idcnt) [resolution 1870,1860]
4116. ~sP11(X11,X12,X13) | ::(X12,sreg,X11) [resolution 2352,1672]
4131. ~sP10(X11,X12,X13) | ::(X12,sreg,X11) [resolution 2356,1672]
4284. P(X3,X4) | ~S(X4) | ~S(X3) | PAR(X5) [resolution 2344,1476]
4293. 40 <=> ! [X1,X0] : (P(X0,X1) | ~S(X0) | ~S(X1)) [avatar definition]
4294. P(X0,X1) | ~S(X0) | ~S(X1) <- (40) [avatar component clause 4293]
4297. 41 <=> ! [X5] : PAR(X5) [avatar definition]
4298. PAR(X5) <- (41) [avatar component clause 4297]
4299. 41 | 40 [avatar split clause 4284,4293,4297]
4301. 42 <=> S(sK169) [avatar definition]
4302. S(sK169) <- (42) [avatar component clause 4301]
4305. 43 <=> S(sK171) [avatar definition]
6278. sP11(X11,X12,X13) | sP10(X11,X12,X13) | sP12(X11,X12,X13) | ~SLC(X13,X12,X11) | ::(X12,sreg,X11) [resolution
      2359,1713]
6282. sP10(X11,X12,X13) | sP12(X11,X12,X13) | ~SLC(X13,X12,X11) | ::(X12,sreg,X11) [subsumption resolution
      6278,4116]
6283. ~SLC(X13,X12,X11) | sP12(X11,X12,X13) | ::(X12,sreg,X11) [subsumption resolution 6282,4131]
7055. ~S(sK169) | ~S(sK171) <- (40) [resolution 4294,2363]
11963. 307 <=> sP12(sK170,sK169,sK167) [avatar definition]
11965. sP12(sK170,sK169,sK167) <- (307) [avatar component clause 11963]
12002. sP12(sK170,sK169,sK167) | ::(sK169,sreg,sK170) [resolution 6283,2361]
12005. 309 <=> ::(sK169,sreg,sK170) [avatar definition]
12007. ::(sK169,sreg,sK170) <- (309) [avatar component clause 12005]
12008. 309 | 307 [avatar split clause 12002,11963,12005]
12010. 310 <=> ::(sK171,sreg,sK170) [avatar definition]
12012. ::(sK171,sreg,sK170) <- (310) [avatar component clause 12010]
12019. sP12(sK170,sK171,sK168) | ::(sK171,sreg,sK170) [resolution 2362,6283]
12023. 311 <=> sP12(sK170,sK171,sK168) [avatar definition]
12025. sP12(sK170,sK171,sK168) <- (311) [avatar component clause 12023]
12026. 310 | 311 [avatar split clause 12019,12023,12010]
12382. SREG(sK167,sK169,sK170) <- (307) [resolution 11965,2348]
```

```
12421. 318 <=> ::(sK171,sreg) [avatar definition]
12422. ::(sK171,sreg) <- (318) [avatar component clause 12421]
12658. SREG(sK168,sK171,sK170) <- (311) [resolution 12025,2348]
13066. ::(sK169,sreg,sK170) <- (307) [resolution 12382,1672]
13070. 309 | ~307 [avatar split clause 13066,11963,12005]
13345. ::(sK169,sreg) <- (309) [resolution 12007,2320]
13378. S(sK169) <- (309) [resolution 13345,2332]
13381. 42 | ~309 [avatar split clause 13378,12005,4301]
13527. ::(sK171,sreg,sK170) <- (311) [resolution 12658,1672]
13546. 310 | ~311 [avatar split clause 13527,12023,12010]
13571. ::(sK171,sreg) <- (310) [resolution 12012,2320]
13577. 318 | ~310 [avatar split clause 13571,12010,12421]
13689. S(sK171) <- (318) [resolution 12422,2332]
13694. 43 | ~318 [avatar split clause 13689,12421,4305]
13700. ~S(sK171) <- (40, 42) [subsumption resolution 7055,4302]
13701. ~43 | ~40 | ~42 [avatar split clause 13700,4301,4293,4305]
13703. $false <- (41) [resolution 4298,2412]
13801. ~41 [avatar contradiction clause 13703]
13802. $false [avatar sat refutation 4299,12008,12026,13070,13381,13546,13577,13694,13701,13801]
```