Supplementary Material for ICSE Submission: Analyzing and Debugging Normative Requirements via Satisfiability Checking

Anonymous Author(s)

ACM Reference Format:
Anonymous Author(s). 2023. Supplementary Material for ICSE Submission: Analyzing and Debugging Normative Requirements via Satisfiability Checking. In *Proceedings of The 46th ACM International Conference on Software Engineering (ICSE 2024).* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

Sec. A provides detailed explanation and correctness proof of the translation function from SLEEC DSL to FOL*. Sec. B provides details on the derivation tree and the reduction process for causal (un)satisfiability proofs. Sec. C provides additional evaluation results.

A TRANSLATION OF SLEEC DSL TO FOL*

In this section, we provide the translation function T in Tbl. 1, and prove the correctness claim of the translation prove the correctness of the translation in Thm. 1.

We now state and prove the correctness of the FOL* encoding:

Theorem 1 (Correctness of the FOL* translation). *Let a set of rules Rules and facts Facts in* SLEEC DSL *be given. There exists a trace $\sigma = (\mathcal{E}_1, \mathbb{M}_1, \delta_1), (\mathcal{E}_2, \mathbb{M}_2, \delta_2), \ldots (\mathcal{E}_n, \mathbb{M}_n, \delta_n)$ such that $\sigma \in \mathcal{L}(Rules) \cap \mathcal{L}(Facts)$ if and only if $T(Rules) \wedge T(Facts) \wedge axiom_{mc}$ has a satisfying solution (D, v).*

Sketch of Proof 1. *We prove the forward direction by constructing a satisfying solution (D, v) to $T(Rules) \wedge T(Facts)$ from the trace $\sigma \in \mathcal{L}(Rules) \cap \mathcal{L}(Facts)$. For every state $(\mathcal{E}_i, \mathbb{M}_i, \delta_i) \in \sigma$, we follow the construction rules: (1) for every event $e \in \mathcal{E}_i$, add a relational object o^e of class C^e such that $v(o^e.ext) = \top$ and $v(o^e.time) = \delta_i$; and (2) add a relational object o^M such that $o^M.time = \delta_i$ and $v(o^M.m) = \mathbb{M}_i(m)$ for every measure $m \in M$. We then prove that the constructed (D, v) is a solution to $T(Rules) \wedge T(Facts) \wedge axiom_{mc}$.*

We prove the backward direction by constructing σ from a satisfying solution (D, v) to $T(Rules) \wedge T(Facts) \wedge axiom_{mc}$. The construction maps every relational object o^e and o^M to some state $(\mathcal{E}_i, \mathbb{M}_i, \delta_i) \in \sigma$, where (1) $e \in \mathcal{E}_i$ if $v(o^e).ext = \top \wedge v(o^e).time = \delta_i$; and (2) $\mathbb{M}_m = v(o^M.m)$ for every $m \in M$ if $v(o^M).ext = \top \wedge v(o^M).time = \delta_i$. We then prove $\sigma \in \mathcal{L}(Rules) \cap \mathcal{L}(Facts)$ and conclude the proof.

A.1 FOL* Encoding for Situational Conflicts

In this section, we first define the states of obligation in Def. 1, and then use it to prove sufficient condition for situational conflict in Thm. 1. Next, we present the FOL* encoding for the sufficient condition of situational conflict in Tab. 2, and finally provide a sketch of the proof the correctness of the encoding (Thm. 2).

Definition 1 (State of Obligations)**.** Let a rule set *Rules*, a rule $r \in Rules$ and an r-triggering situation $(\sigma_0^k, \vec{M_k})$ be given. The time point k is the last time point of σ_0^k, and it is also when r is triggered. The status of rules, obligation chains, conditional obligations and obligations are defined as follows:

Triggered: A rule $r = \textbf{when}\, e \wedge p\, \textbf{then} \bigvee_{cob}$ is *triggered* at time point i if $i \le k$ and $\sigma_0^k \models_i e$ and $\sigma_o^k \models p$. If a rule $r = \textbf{when}\, e \wedge p\, \textbf{then} \bigvee_{cob}$ is triggered at i, then the obligation chain $\bigvee_{cob}$ is *triggered* at i. If an obligation chain $\bigvee_{cob} = cob$ **otherwise** $\bigvee'_{cob}$ is triggered at i, then (1) the *conditional obligation cob* is triggered at i and (2) $\bigvee'_{cob}$ is *triggered* at $j > i$ if cob is *violated* at j or cob is *blocked* at i. If a conditional obligation $p \Rightarrow ob$ is triggered at i and p is evaluated to $\top$ at i, then ob is triggered at i.
Fulfilled: An obligation ob is *fulfilled* at time point $j \le k$ if it is triggered at some time point $i \le j$ and $\sigma_0^j \models_i ob$. A conditional obligation $p \Rightarrow ob$ (triggered at i) is *fulfilled* at j if its obligation ob is fulfilled at j or p evaluated to $\bot$ at i. An obligation chain $\bigvee_{cob} = cob$ **otherwise** $\bigvee'_{cob}$ is fulfilled at j if cob is fulfilled at j or $\bigvee'_{cob}$ is fulfilled at j.
Violated: An obligation ob (*triggered* at i) is *violated* at a time point $j \le k$ if $\sigma_0^k \not\models_i^j ob$. A conditional obligation $p \Rightarrow ob$ is *violated* at time point j if ob is *violated* at j. An obligation chain $\bigvee_{cob} = \bigvee'_{cob}$ **otherwise** cob is *violated* at time point j if $\bigvee'_{cob}$ is *violated* at some point $j' \le j$, cob is *triggered* at j and cob is *violated* at j'.
Active: An obligation, conditional obligation and obligation chain are *active* at time point j if they are *triggered* at some time point $i \le j$ and are not *fulfilled* and *violated* at any time point $j' \in [i, j]$.
Forced: An obligation chain $\bigvee_{cob}$ is *forced* at time point $j \ge k$ if $\bigvee_{cob}$ is *active* at j. If an obligation chain $\bigvee_{cob} = cob^+$ **otherwise** $\bigvee'_{cob}$ is *forced*, and $\bigvee'_{cob}$ is *blocked* at the time point j' when cob^+ expires (s.t., $\delta_{j'} = \delta_j + \mathbb{M}_j$), then cob^+ is *forced* at time point j. If a conditional obligation $p \Rightarrow ob$ is *forced* at j, and ob is triggered at j, then ob is *forced* at j.
Blocked: An obligation ob (triggered at i) is *blocked* at time point j if it is *active* and there is an obligation ob' such that (1) ob' is *forced* at time point j; (2) if $ob = e\, \textbf{within}\, t$ and $ob' = \neg e\, \textbf{within}\, t'$ then $\mathbb{M}_i(t) + \delta_i \le \mathbb{M}_{i'}(t') + \delta_{i'}$; and (3) if $ob = \neg e\, \textbf{within}\, t$ and $ob' = e\, \textbf{within}\, t'$ then $\mathbb{M}_i(t) + \delta_i \ge \mathbb{M}_{i'}(t') + \delta_{i'}$. A conditional obligation $p \Rightarrow ob$ is *blocked* at j if ob is *blocked* j and p evaluates to $\top$ at j. An obligation chain cob^+ **otherwise** $\bigvee_{cob}$ is *blocked* at time point j if cob^+ is *blocked* at some time point j and $\bigvee_{cob}$ is *blocked* at time point j' when cob expires (s.t., $\delta_{j'} = \delta_j + \mathbb{M}_j$).

ICSE 2024, 14 - 20 April, 2024, Lisbon, Portugal

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

$$
\begin{array}{ll}
T(\textbf{when}\, e \,\textbf{and}\, p \,\textbf{then}\, \bigvee_{cob}) & \rightarrow \forall o^e : C^e \exists o^M : C^M(o^m.time = o^e.time \wedge (T^*(p, o^M) \Rightarrow T^*(\bigvee_{cob}, o^M))) \\
T(\textbf{exists}\, e \,\textbf{and}\, p \,\textbf{while}\, \bigvee_{cob}) & \rightarrow \exists o^e : C^e \exists o^M : C^M(o^m.time = o^e.time \wedge (T^*(p, o^M) \wedge T^*(\bigvee_{cob}, o^M))) \\
T^*(cob_1 \,\textbf{otherwise}\, cob_2 \ldots cob_n,\ o^M) & \rightarrow T^*(cob_1, o^M) \vee \exists o^M_j : C^M(\textbf{Violation}(cob_1, o^M, o^M_j) \wedge T^*(cob_2 \,\textbf{otherwise}\, \ldots cob_n, o^M_j)) \\
T^*(\textbf{not}\ \bigvee_{cob},\ o^M) & \rightarrow \neg T^*(\bigvee_{cob}) \\
T^*(p \Rightarrow ob, o^M) & \rightarrow \neg T^*(p, o^M) \wedge T^*(ob, o^M) \\
\textbf{Violation}(p \Rightarrow ob, o^M, o^M_j) & \rightarrow T^*(p, o^M) \wedge \textbf{Violation}(ob, o^M, o^M_j) \\
T^*(e \,\textbf{within}\, t, o^M) & \rightarrow \exists o^e : C^e(o^M.time \le o^e.time \le o^M.time + T^*(t, o^M)) \\
\textbf{violation}(e \,\textbf{within}\, t, o^M, o^M_j) & \rightarrow o^M_j.time = o^M.time + T^*(t, o^M) \wedge \neg T^*(\textbf{not}\ e \,\textbf{within}\, t, o^M) \\
T^*(\textbf{not}\ e \,\textbf{within}\, t, o^M) & \rightarrow \neg T^*(\textbf{not}\ e \,\textbf{within}\, t, o^M) \\
\textbf{Violation}(\neg e \,\textbf{within}\, t, o^M, o^M_j) & \rightarrow \begin{array}{c} (o^M.time \le o^M_j.time \le o^M.time + T^*(t, o^M)) \wedge \exists o^e : C^e(o^M_j.time = o^e.time) \\ \neg(\exists o^e_1 : C^e(o^M.time \le o^e_1.time < o^M_j.time)) \end{array}
\end{array}
$$

$$
\begin{array}{lll}
T^*(c, o^M) \rightarrow c & T^*(m, o^M) \rightarrow o^M.m & \\
T^*(-t, o^M) \rightarrow -T^*(t, o^M) & T^*(t_1 + t_2, o^M) \rightarrow T^*(t_1, o^M) + T^*(t_2, o^M) & T^*(c \times t, o^M) \rightarrow T^*(c, o^M) \times T^*(t, o^M) \\
T^*(\top, o^M) \rightarrow \top & T^*(t_1 > t_2, o^M) \rightarrow T^*(t_1, o^M) > T^*(t_2, o^M) & T^*(t_1 = t_2, o^M) \rightarrow T^*(t_1, o^M) = T^*(t_2, o^M) \\
T^*(\neg p, o^M) \rightarrow \neg T^*(p, o^M) & T^*(p_1 \wedge p_2, o^M) \rightarrow T^*(p_1, o^M) \wedge T^*(p_2, o^M) & T^*(p_1 \vee p_2, o^M) \rightarrow T^*(p_1, o^M) \vee T^*(p_2, o^M)
\end{array}
$$

Figure 1: Translation rules from `SLEEC DSL` to FOL*. Given a `SLEEC DSL` rule r, T translates r to an FOL* formula using the translation function T^*. The function T^* recursively visits the elements (i.e., term, proposition and obligations) of r and translates them into FOL* constraints under a relational object o^M representing the measures when r is triggered.

Noticed that there are circular dependencies between forced and blocked obligation chains. Fortunately, by encoding the status definitions into FOL*, we can leverage FOL* solver's ability to incrementally and lazily unroll the necessary definitions to resolve the dependencies.

The status of obligations at time point k is determined by the historical states leading up to k (i.e. the prefix of a trace σ_0^k) and the future measures after k (i.e., $\vec{M}_k$). When a rule is triggered at time point k while its response (the entire obligation chain) is blocked given a situation $(\sigma_0^k, \vec{M}_k)$, then a conflict is inevitable.

Lemma 1. *For every rule r = **when** $e \wedge p$ **then** $\bigvee_{cob}$ in a rule set Rules, if there exists a r-triggering situation $(\sigma_0^k, \vec{M}_k)$ (see Def. ??) where the obligation chain $\bigvee_{cob}$ is blocked at time point k, then r is situationally conflicting with respect to the situation $(\sigma_0^k, \vec{M}_k)$.*

The Proof of Lemma 1.

Sketch of Proof 2. *Proof by contradiction, we assume there exists a trace σ such that σ is an extension to σ_0^k and is also consistent with $\vec{M}_k$. Since $\sigma \in \mathcal{L}(Rules)$, then σ fulfill the obligation chain $\bigvee_{cob} = (cob_1, \ldots cob_n)$ triggered at k ($\sigma \models_k \bigvee_{cob}$). Therefore, by the semantics of obligation chain fulfillment, either $\sigma \models_k cob_1$ or cob_1 is positive and there exists a time point $k' \ge k$ such that $\sigma \not\models_k^{k'} cob_1$ and $\sigma \models_{k'} (cob_2, \ldots cob_n)$*

Since $\bigvee_{cob}$ is blocked at k, by Def. 1, the obligation ob_1 in cob_1 is blocked at k, and for every conditional obligation cob_m, the obligation ob_m is cob_m is blocked at the unique time when ob_{m-1} is violated (the violation time is unique since $cob_1 \ldots cob_{n-1}$ are all positive). Therefore, it is sufficient to show that if an obligation ob is blocked, then σ does not fulfill ob_m. There are two cases:

*First, we consider the case $ob = e$ **within** t. There is an event occurred at some time point $k \ge j$ where $\delta_j \le \delta_k \le \delta_j + \mathbb{M}_j$. Since ob is blocked, then there exists an obligation forced by some rule r' (triggered at j') such that $ob' = \neg e$ **within** t' where $\mathbb{M}_{j'}(t') \ge \mathbb{M}_j$. Therefore, the occurrence of e at time point k violates r'. Contradiction.*

*The case where $ob = \neg e$ **within** t can be proved analogously.* □

Remark 1. The sufficient condition for situational conflict defined in Lemma 1 is not a necessary condition, as some situational conflicts do not require a rule's response to be blocked at the last state of the situation. Let's consider the set of rules $\{r_1, r_2, r_3, r_4\}$, where r_1 = **when** e_1 **then** e_2 **within** 5, r_2 = **when** e_3 **then** $\neg e_2$ **within** 4, r_3 = **when** e_4 **then** e_3 **within** 3, and r_4 = **when** e_1 **then** $\neg e_3$ **within** 1. The rule r_1 is situational conflicting in the situation $(\sigma^1, ^*)$ where $\sigma^1 = (r_1, r_2, r_3, r_4, \mathbb{M}_1, 1)$ because, according to r_1 and r_2, e_2 must occur within the interval $(4, 5]$. Additionally, based on r_3 and r_4, the event e_3 must occur at a time $t \in (1, 3]$. For all possible values of t, according to r_2, e_2 cannot occur within the interval $(t, t+4]$, which covers the interval $(4, 5]$ and thus conflicts with r_1. However, in the situation σ^1, the obligation of r_1 is not blocked at time point 1. We refer to the situational conflicts caused by forced obligations "after" the situation as"transitive situation conflicts". Identifying situations that lead to transitive situation conflicts is not easily expressed as satisfiability (e.g., we need to cover the entire range of t in the example) and is left as future work.

We present the FOL* encoding in Fig.2 to describe the situation for a rule to be situational conflicting. Given a set of rules *Rules* and a rule $r \in Rules$, every satisfying solution to the FOL* formula TSC(*Rules*, r) represents a situation where r is situational conflicting. The top-level encoding TSC(r, *Rules*) is presented in part (1) of Tab.2, which describes the existence of a situation $(\sigma_0^k, \vec{M}_k)$ where r is triggered at the last state of σ_0^k (o^M). The situation should be non-violating ($\mathrm{T}_{\downarrow}(Rules, o^M.time)$) and should block the response of r ($\textsc{Blocked}(\bigvee_{cob}, o^M, o^M)$). The FOL* encoding for non-violating and obligation blocking is presented in parts (2) and (3) of Tab. 2, respectively.

Note that eagerly expanding the definition of obligation blocking blows up the size of encoding exponentially (with respect to the number of obligations in *Rules*) due to the transitive dependencies between blocked obligations and forced obligations. To avoid the blow-up, we lazily expand the definition of obligation blocking by introducing an *internal* class of relational object $C^{blockob}$ for every obligation *ob* in *Rules* to indicate if and when *ob* is blocked. The axiom *axiomBlockob* is added to describe the definition of a

Table 1: FOL* causal proof of UNSAT for ϕ_1 and ϕ_2 in Ex. 1.

ID	Lemma	Derivation Rule	Deps
1	$\forall a : A \exists b : B \cdot (a.time \leq b.time \leq a.time + 10)$	**INPUT**: ϕ_1	{}
2	$\exists a : A \forall b : B \cdot (b.time \geq a.time + 20 \land p(a, b))$	**INPUT**: ϕ_2	{}
3	$\forall b : B \cdot (b.time \geq a_1.time + 20 \land p(a_1, b))$	**EI**: $[a \leftarrow a_1]$	{2}
4	$\exists b : B \cdot (a_1.time \leq b.time \leq a_1.time + 10)$	**UI**: $[a \leftarrow a_1]$	{1, 3}
5	$a_1.time \leq b_1.time \leq a_1.time + 10$	**EI**: $[b \leftarrow b_1]$	{4}
6	$b_1.time \geq a_1.time + 20 \land p(a_1, b_1)$	**UI**:$[b \leftarrow b_1]$	{3, 5}
7	$b_1.time \geq a_1.time + 20$	**And**	{6}
8	$\bot$	**Impl**	{5, 7}

blocked obligation (Def. 1) and is lazily applied to relational objects of $C^{block_{ob}}$ in a given domain.

Theorem 2 (Encoding of Situational Conflict). *Let a rule set Rules be given. For every rule $r \in$ Rules, if the FOL* formula TSC$(r, Rules)$ is satisfiable, then r is situationally conflicting.*

Sketch of Proof 3. *If (D, v) is a satisfying solution to TSC$(r, Rules)$, then we use the same method in the proof of Thm. 1 to construct a situation σ from (D, v). We then show that the encoding in Tab. 2 conforms with the semantics of non-violation and the status of obligations (in Def. 1). Finally, we show that the constructed σ satisfies the sufficient condition for situational conflict (Lemma. 1), and thus r is situational conflicting w.r.t σ.* □

B FOL* * PROOF OF UNSAT

A causal FOL* proof is a sequence of derivation steps $L_1, L_2 \ldots, L_n$. Each step L_i is a tuple (i, ψ, o, *Deps*) where (1) i is the ID of the derivation step; (2) ψ is the derived FOL* lemma; (3) o is the name of the *derivation rule* used to derive ψ; and (4) *Deps* are IDs of dependent lemmas for deriving ψ. A *derivation step is sound* if the lemma ψ can be obtained via the derivation rule o using lemmas from *Deps*. A *proof is sound* if every derivation step is sound. The proof is *refutational* if the final derivation contains the lemma $\bot$ (UNSAT).

Example 1. Let FOL* formulas, $\phi_1 : \forall a : A \exists b : B \cdot (a.time \leq b.time \leq a.time + 10)$ and $\phi_2 : \exists a : A \forall b : B \cdot (b.time \geq a.time + 20 \land p(a, b))$ be given, where A and B are classes of relational objects, *time* is an attribute of type $\mathbb{N}$ and p is a complex predicate. $\phi_1 \land \phi_2$ is UNSAT, and the proof of UNSAT is shown in Tbl. 1. The proof starts by introducing the **Input** ϕ_1 and ϕ_2 (steps 1, 2), and then uses existential instantiation (**EI**) in steps 3, 5 and universal instantiation (**UI**) in steps 4, 6 to eliminate quantifiers in ϕ_1 and ϕ_2. In step 7, the **And** rule decomposes the conjunction and derives part of it as a new lemma. Finally, in step 8, $\bot$ is derived with the **Impl** rule from the quantifier-free (QF) lemmas derived in steps 5 and 7. The proof is refutational because step 8 derives $\bot$.

B.1 Derivation Tree and proof reduction

Given a refutation proof, one can construct a *derivation graph* where every lemma is a node and its dependencies are the incoming edges to the node. The roots of the derivation graph are the input formulas and axioms (e.g., $axiom_{mc}$) and the (only) leaf of the graph is the derived $\bot$.

Using the derivation graph, one can check the soundness of the proof and reduce it by traversing the graph backwards from the leaf (step 8 in Fig. 1). While visiting a node, we first check if the lemma represented by the node can be soundly derived using the derivation rule with the lemmas in its dependency, and then reduce the dependency if not every lemma is necessary. Only the nodes representing the lemmas in the reduced dependencies are scheduled to be visited in the future. For instance, after checking the derivation step 7, its dependencies, 5 and 8, are scheduled to be visited next. If every scheduled node is visited without any failure, the proof is successfully verified, and the visited portion of the graph constitutes *the reduced proof.* In a reduced proof, every derived lemma is used to derive $\bot$.

We use two special derivation rules, **Input** and **Implication**. The rule **Input** adds an input FOL* formula as a fact to the proof. The rule **Implication** derives new QF lemmas via logical implication, and it can be verified using an SMT solver by solving the formula $deps \land \neg C$ where C is the derived lemmas and *deps* are lemmas in the implication step's dependencies. The derivation is *valid* if and only if the formula is UNSAT, and the UNSAT core returned by the SMT solver becomes the reduced dependencies.

Example 2 (Reduced Dependencies)**.** Let $L1 : A > B$, $L2 : B > C$ and $L3 : C > 5$ be three (derived) lemmas. Suppose a lemma $L4 : A > C$ is derived using the **implication** rule given the dependencies $L1$, $L2$, $L3$. The rule can be verified by checking the satisfiability of $L1 \land L2 \land L3 \land \neg L4$. The result is UNSAT with an UNSAT core $L1$, $L2$ and $\neg L4$. Therefore, the reduced dependencies are $L1$ and $L2$.

B.2 Condition for involved atomic elements

In addition to reporting a binary "yes" or "no" answer to the satisfiability, the FOL* satisfiability checker `LEGOS` also provides a causal proof of UNSAT if the encoded formula is unsatisfiable. We project the proof into the input `SLEEC DSL` rules to highlight the causes of WFI problems at the level of *atomic elements*. More specifically, we want to highlight every *involved atomic elements* in the proof.

Definition 2 (Atomic element)**.** An *atomic element* in `SLEEC DSL` is one the followings: (1) an *atomic proposition* ap: $\top \mid \bot \mid t = t \mid t \geq t \mid \neg ap$, (2) a triggering event e (where e in **when** $e \land \ldots$ **then** $\ldots$ or **exists** $e \land \ldots$ **while** $\ldots$), (3) a response event e' in an obligation (in e' **within** $\ldots$), or (4) a deadline t of an obligation (in $\ldots$ **within** t).

Definition 3 (Involved atomic element)**.** Let a proof L be given. We denote $Imp(L)$ as the set of QF lemmas derived via or listed as dependencies for the derivation rule **Impl**. An atomic proposition ap is *involved* if $Imp(L)$ contains the quantifier-free (QF) formula $T^*(ap, o^M)$ for some relational object of class C^M where T^* is the translation function for `SLEEC DSL` element defined in Tab. 1. A triggering event e for "**when** $e \land p$ **then** $\bigvee_{cob}$" is involved if $Imp(L)$ contains a QF formula $\neg(o^e.ext) \lor \neg T^*(p, o^M) \lor F$ where F is a QF formula, and both o^e and o^M are relational objects of class C^e and C^M, respectively. Similarly, a triggering event e for "**exists** $e \land p$ **while** $\bigvee_{cob}$" is involved if $Imp(L)$ contains a formula $o^e.ext \land T^*(p, o^M) \land F$. An obligation head e for e **within** t is involved if $Imp(L)$ contains a QF lemma $o^e.ext \land o^e.time \geq o^M.time \land F$ for some object o^e and o^M. An obligation deadline t for e **within** t is involved if $Imp(L)$ contains a quantifier-free lemma $o^e.time \leq o^M.time + T^*(t, o^M) \land F$ for some object o^e and o^M.

$\mathrm{TSC}(\textbf{when}\ e \land p\ \textbf{then}\ \bigvee_{cob}, Rules)$	$\rightarrow \exists o^e : C^e \exists o^M : C^M(o^M.time = o^e.time \land T^*(p, o^M) \land \textsc{Blocked}(\bigvee_{cob}, o^M, o^M)$ $\land \mathrm{T}_{\downarrow}(Rules, o^M.time)) \land axiom_{mc} \land axiomBlock_{ob}$ for every obligations ob in $Rules$
$\mathrm{T}_{\downarrow}(\textbf{when}\ e \land p\ \textbf{then}\ \bigvee_{cob}, end\_time)$	$\rightarrow \forall o^e : C^e(o^e.time \le end\_time \Rightarrow \exists o^M : C^M$ $(o^m.time = o^e.time \land (T^*(p, o^M) \Rightarrow \mathrm{T}^*_{\downarrow}(\bigvee_{cob}, o^M, end\_time))))$
$\mathrm{T}^*_{\downarrow}(ob_1\ \textbf{otherwise}\ \ldots ob_n, o^M, end\_time)$	$\rightarrow \mathrm{T}^*_{\downarrow}(ob_1, o^M, end\_time) \lor \exists o^M_v : C^M(\textbf{violation}(ob_1, o^M, o^M_v) \lor \mathrm{T}^*_{\downarrow}(ob_n, o^M_v, end\_time))$
$\mathrm{T}^*_{\downarrow}(e\ \textbf{within}\ t, o^M, end\_time)$	$\rightarrow \exists o^e : C^e(o^M.time \le o^e.time \le o^M.time + T^*(t, o^M)\ \lor\ o^M.time + T^*(t, o^M) > end\_time$
$\mathrm{T}^*_{\downarrow}(\neg e\ \textbf{within}\ t, o^M, end\_time)$	$\rightarrow \neg(\exists o^e : C^e(o^M.time \le o^e.time \le \textsc{Min}(o^M.time + T^*(t, o^M),, end\_time))$
$\textsc{Blocked}((p \Rightarrow e\ \textbf{within}\ t)\ \textbf{otherwise}\ \bigvee_{cob}, o^M_i, o^M_c)$	$\rightarrow \textsc{Blocked}((p \Rightarrow e\ \textbf{within}\ t), o^M_i, o^M_c) \land \exists o^M_v : C^M$ $(o^M_v.time = o^M_i.time + T * (t, o^M_i)) \land \textsc{Blocked}(\bigvee_{cob}, o^M_v, o^M_c)$
$\textsc{Blocked}(p \Rightarrow ob, o^M_i, o^M_c)$	$\rightarrow T^*(p, o^M_i) \land \textsc{Blocked}(ob, o^M_i, o^M_c)$
$\textsc{Blocked}(ob, o^M_i, o^M_c)$	$\rightarrow \exists o^{block_{ob}} : C^{block_{ob}}(o^{block_{ob}}.i = o^M_i.time \land o^{block_{ob}}.c = o^M_c.time)$
$axiomBlock_{ob}$	$\rightarrow \forall o^{block_{ob}} : C^{block_{ob}} \exists o^M_i, o^M_c : C^M$ $(o^M_i.time = o^{block_{ob}}.i \land o^M_c.time = o^{block_{ob}}.c \land \_\textsc{Blocked}(ob, o^M_i, o^{M_c}))$
$\_\textsc{Blocked}(e\ \textbf{within}\ t, o^M_i, o^{M_c})$	$\rightarrow \textsc{Active}(e\ \textbf{within}\ t, o^M_i, o^{M_c}) \land \bigvee_{ob \in OBG(\neg e)}(\exists o^M_1 : C^M(o^M_1.time \le o^M.time)$ $\land \textsc{Forced}(ob, o^M_1, M_c) \land (o^M.time + T^*(t, o^M) \ge o^M_1.time + T^*(t_1, o^M))$ where t_1 is ob's time limit
$\_\textsc{Blocked}(\neg e\ \textbf{within}\ t, o^M_i, o^{M_c})$	$\rightarrow \textsc{Active}(\neg e\ \textbf{within}\ t, o^M_i, o^{M_c}) \land \bigvee_{ob \in OBG(e)}(\exists o^M_1 : C^M(o^M_1.time \le o^M.time)$ $\land \textsc{Forced}(ob, o^M_1, M_c) \land (o^M.time + T^*(t, o^M) \le o^M_1.time + T^*(t_1, o^M))$ where t_1 is ob's time limit
$\textsc{Active}(ob, o^M_i, o^M_c)$	$\rightarrow \textsc{Triggered}(ob, o^M_i) \land \neg\textsc{Violated}(ob, o^M_i, o^M_c) \land \neg\textsc{Fulfilled}(ob, o^M_i, o^M_c)$
$\textsc{Fulfilled}(e\ \textbf{within}\ t, o^M_i, o^M_c,,)$	$\rightarrow T^*(e\ \textbf{within}\ \textsc{Min}(t, o^M_c.time)), o^M$
$\textsc{Fulfilled}(\neg e\ \textbf{within}\ t, o^M_i, o^M_c)$	$\rightarrow \mathrm{T}^*_{\downarrow}(\neg e\ \textbf{within}\ t, o^M, o^M_c.time)$
$\textsc{Violated}(ob, o^M_i, o^M_c)$	$\rightarrow \textsc{Fulfilled}(\textsc{Noncomp}(ob), o^M_i, o^M_c.time,,)$
$\textsc{Triggered}(ob, o^M)$	$\rightarrow$ let $\textsc{trigger\_rule}(ob) = \textbf{when}\ e \land p\ \textbf{then}\ \bigvee_{cob}$ where $(p_m \Rightarrow ob) = \bigvee_{cob}[m]$ if $m = 1$ then $\exists o^e : C^e(o^e.time = o^M.time \land T^*(p \land p_m, o^M))$ else $\exists o^M_i : C^M(\textsc{Violated}(\bigvee_{cob}[m-1], o^M_i, o^M) \lor \textsc{Blocked}(\bigvee_{cob}[m-1], o^M_i, o^M)))$
$\textsc{Forced}(p \Rightarrow ob, o^M_i, o^M_c)$ where $cob = \bigvee_{cob}[m]$	$\rightarrow T^*(p, o^M_i) \land \textsc{Active}(ob, o^M_i, o^M_c) \land \neg(\textsc{Blocked}(ob, o^M_i, o^M_c)) \land$ $\exists o^M_v : C^M(\textbf{Violation}(ob, o^M_i, o^M_v) \land \textsc{Blocked}(\bigvee_{cob}[m+1:], o^M_v, o^M_c))$
$OBG(h)$	$= \{ob \mid (h\ \textbf{within}\ t)$ in the rule set$\}$
$\textsc{trigger\_rule}(ob) = \textbf{when}\ e\ \textbf{then}\ \bigvee_{cob}$	if and only if $ob \in \bigvee_{cob}$

Figure 2: The FOL* encoding TSC$(r, \textit{Rules})$ that describes a situation where the rule $r \in \textit{Rules}$ to be situational conflicting. The table contains three parts: (1) the top-level encoding that describes the existence of a situation where r is triggered at the last state (o^M); (2) the FOL* constraint for describing the situation is non-violating (i.e., $\mathrm{T}_{\downarrow}(\textit{Rules}, o^M.\textit{time})$)); and (3) the FOL* encoding for describing blocked obligations (i.e., $\textsc{Blocked}(\bigvee_{cob}, o^M_i, o^M_c)$) as well as another status of obligations where $o^M{}_i$ is the state when the obligation is triggered and $o^M{}_c$ is the last state of the situation.

C ADDITIONAL EVALUATION RESULTS

To ensure correctness, we augmented the proof of absence of vacuous conflicts produced by N-Tool with feedback produced by the existing tool for analyzing vacuous conflicts, AutoCheck, see results in Tbl. 2.

case studies	T-Tool	AutoCheck
ALMI	0	0
ASPEN	0	0
AutoCar	0	0
BSN	0	0
DressAssist	0	0
CSI-Cobot	0	0
DAISY	0	0
DPA	0	0
SafeSCAD	0	0

Table 2: Vacuous conflicts identified by N-Tool compared to AutoCheck