Probabilistic Analysis of Binary Sessions

Omar Inverso
Gran Sasso Science Institute, Italy

Hernán Melgratti
ICC – Universidad de Buenos Aires – Conicet, Argentina

Luca Padovani
Università di Torino, Italy

Catia Trubiani
Gran Sasso Science Institute, Italy

Emilio Tuosto
Gran Sasso Science Institute, Italy

Abstract

We study a probabilistic variant of binary session types that relate to a class of Finite-State Markov Chains. The probability annotations in session types enable the reasoning on the probability that a session terminates successfully, for some user-definable notion of successful termination. We develop a type system for a simple session calculus featuring probabilistic choices and show that the success probability of well-typed processes agrees with that of the sessions they use. To this aim, the type system needs to track the propagation of probabilistic choices across different sessions.

2012 ACM Subject Classification Theory of computation → Type structures

Keywords and phrases Probabilistic choices; session types; static analysis; deadlock freedom.

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2020.36

Funding Omar Inverso has been partially supported by MIUR project PRIN 2017FTXR7S *IT MATTERS* (Methods and Tools for Trustworthy Smart Systems). Catia Trubiani has been partially supported by MIUR project PRIN 2017TWRCNB *SEDUCE* (Designing Spatially Distributed Cyber-Physical Systems under Uncertainty). Hernán Melgratti, Luca Padovani and Emilio Tuosto have been partially supported by EU H2020 RISE programme under the Marie Skłodowska-Curie grant agreement No 778233. Hernán Melgratti has been partially supported by UBACyT projects 20020170100544BA and 20020170100086BA and PIP project 11220130100148CO.

Acknowledgements

1 Introduction

Session types [29, 30] have consolidated as a formalism for the modular analysis of complex systems of communicating processes. A *session* is a private channel connecting two (sometimes more) processes, each owning one *endpoint* of the session and using the endpoint according to a specification – the *session type* – that constrains the sequence of messages that can be sent and received through that endpoint. As an example, the session type

$$!\mathsf{int}.(\circ \mathbin{\&} ?\mathsf{int}.(\circ \oplus T)) \tag{1.1}$$

could describe (part of) an auction protocol as seen from the viewpoint of a buyer process, which sends a bid ($!\mathsf{int}$) and waits for a decision from the auctioneer. The protocol proceeds in two different ways, as specified by the two sides of the branching operator &. The auctioneer may declare that the item is sold, in which case the session terminates immediately ($\circ$), or it may inform the buyer of a different (higher) bid ($?\mathsf{int}$). At that point the buyer may choose ($\oplus$) to quit the auction or to restart the same protocol, here denoted by T, with another bid.

31st International Conference on Concurrency Theory (CONCUR 2020).
Editors: Igor Konnov and Laura Kovács; Article No. 36; pp. 36:1–36:31
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Most session type theories are aimed at enforcing qualitative properties of a system, such as type safety, protocol compliance, deadlock and livelock freedom, and so on [30]. In these theories, branches (&) and choices ($\oplus$) are given a non-deterministic interpretation since all that matters is understanding whether the system "behaves well" . In this work, we propose a session type system for *quantitative analysis* of session-based networks of communicating processes. More specifically, we shift from a non-deterministic to a *probabilistic* interpretation of branches and choices in session types and study a type system aimed at determining the probability with which a particular session terminates *successfully*. Since there is no universal interpretation of "successful termination", we differentiate successful from unsuccessful termination of a session by means of a dedicated . For example, in our type system we can refine (1.1) as

$$\mathsf{!int}.(\bullet\ {}_{p}\&\ \mathsf{?int}.(\circ\ {}_{q}\oplus T)) \tag{1.2}$$

where the session type $\bullet$ indicates successful termination and branches and choices are annotated with probabilities p and q. In particular, the auctioneer declares the item sold with probability p and answers with a counteroffer with probability $1-p$, whereas the buyer decides to quit the auction with probability q and to bid again with probability $1-q$.

From an abstract description such as (1.2), we can easily compute the probability that the interaction ends up in a particular state (*e.g.*, the probability with which the buyer wins the auction). However, (1.2) is "just" the type of one endpoint of a single session in a system, while the system itself could be much more complex: there could be many different processes involved, each making probabilistic choices affecting the behavior of faraway processes that directly or indirectly receive information about such choices through messages exchanged in sessions. Also, new processes and sessions could be created and the network topology could evolve dynamically as the system runs. How do we know that (1.2) is a faithful abstraction of our system? How do we know that the probability annotations we see in (1.2) correspond to the actual probabilities that the system evolves in a certain way? Here is where our type system comes into play: by certifying that a system of processes is well typed with respect to a given set of session types with probability annotations, we support the computation of the probability that the system evolves in certain way statically – *i.e.*, before the system runs – and solely looking at the session types we are interested in as opposed to the system itself.

Summary of contributions and structure of the paper. We define a session calculus in which processes may perform probabilistic choices (Section 2). We study a variant of session types based on a probabilistic interpretation of branches and choices so that session types correspond to a particular class of Discrete-Time Markov Chains (Section 3). We provide syntax-directed typing rules for relating processes and session types (Section 4). Well-typed processes are shown to behave probabilistically as specified by the corresponding session types. We are able to trace this correspondence not just for finite processes (Theorem 4.8) but also for processes engaged in potentially infinite interactions (Corollary 4.9). We discuss related work in Section 5 and ideas for further developments in Section 6. Example details and proofs of all the results are relegated to the appendices.

2 A Probabilistic Session Calculus

We let p, q and r range over *probabilities*, namely real numbers in the range $[0,1]$. We let x, y and z range over an infinite set $\mathcal{N}$ of *channel names*. We write $\overline{x}$ for finite sequences of names and other entities. Processes, ranged over by P, Q and R, are defined by the grammar

Domains	p, q, r	$\in$	$[0, 1]$	probability	$\mid \mathtt{case}\, x\, [P, Q]$	branch
	x, y, z	$\in$	$\mathcal{N}$	name	$\mid \mathtt{inl}\, x.P$	left selection
					$\mid \mathtt{inr}\, x.P$	right selection
Processes	P, Q	$::=$	$\mathtt{idle}$	inaction	$\mid P \mid Q$	parallel composition
		$\mid$	$\mathtt{done}\, x$	success	$\mid (x)P$	session restriction
		$\mid$	$x?(y).P$	message input	$\mid P\ {}_{p}\boxplus\ Q$	probabilistic choice
		$\mid$	$x!y.P$	message output	$\mid A\langle \overline{x} \rangle$	process invocation

Table 1 Syntax of processes.

in Table 1. We have two distinct terms, $\mathtt{idle}$ and $\mathtt{done}\, x$, for modeling inactive processes. We use $\mathtt{idle}$ to denote plain termination and $\mathtt{done}\, x$ to denote successful termination of session x. This way, we are able to relate the success rate resulting from processes to that inferrable from session types (Theorem 4.8). The terms $x?(y).P$ and $x!y.P$ denote a process that respectively performs an input and an output of a message y on session x and then continues as P. For simplicity, in the model we only consider messages that are themselves (session) channels, while in some examples we will also use more elaborate message types. The term $\mathtt{case}\, x\, [P, Q]$ represents a process that waits for a selection (either "left" or "right") on session x and continues as either P or Q accordingly. The terms $\mathtt{inl}\, x.P$ and $\mathtt{inr}\, x.P$ represent processes that perform a selection (respectively "left" and "right") on session x and continue as P. Parallel composition $P \mid Q$, channel restriction $(x)P$ and process invocation $A\langle \overline{x} \rangle$ are standard. We assume that for every process variable A there is an equation $A(\overline{x}) := P$ defining it. Finally, the term $P\ {}_{p}\boxplus\ Q$ represents a process that has performed a probabilistic choice and that behaves as P with probability p and as Q with probability $1 - p$.

The notions of free and bound names are standard. In the following, we write $\mathsf{fn}(P)$ and $\mathsf{bn}(P)$ for the set of free and bound names of P, respectively. For the sake of readability, we occasionally omit $\mathtt{idle}$ terms and we assume that input/output prefixes and selections bind more tightly than choices and parallel compositions. So for example, $\mathtt{inl}\, x.\mathtt{done}\, y\ {}_{p}\boxplus\ \mathtt{inr}\, x$ is to be read $(\mathtt{inl}\, x.\mathtt{done}\, y)\ {}_{p}\boxplus\ (\mathtt{inr}\, x.\mathtt{idle})$.

The operational semantics of processes is given by a structural precongruence relation $\preccurlyeq$ and a reduction relation $\rightarrow$, which are defined by the axioms and rules in Table 2 where we abbreviate with $P \equiv Q$ the two relations $P \preccurlyeq Q$ and $Q \preccurlyeq P$. We use a pre-congruence instead of a symmetric relation because careless rewriting of processes may compromise their well typing. Nonetheless, the use of a pre-congruence does not affect the ability of processes to reduce (*cf.* Theorem 4.5) and most relations are symmetric anyway. We now describe the structural pre-congruence and reduction, focusing on the former relation since it is the only one that deals with probabilistic choices.

The relations described by s-par-comm, s-new-comm and s-par-new are standard and need no commentary. Axiom s-choice-comm allows us to commute a probabilistic choice. The probability needs to be suitably adjusted so as to preserve the semantics of the process. Axiom s-no-choice turns a probabilistic choice into a deterministic one when the probability is trivial. On the contrary, knowing that $P\ {}_{1}\boxplus\ Q$ is well typed allows us to easily derive that P alone is also well typed. Axiom s-choice-idem states that the probabilistic choice is idempotent, namely that a probabilistic choice between equal behaviors is not really a choice. Rule s-choice-assoc expresses the standard associativity property for probabilistic choices, which requires a normalization of the involved probabilities. Note that this rule is applicable only when $pq < 1$, or else the rightmost probability in the conclusion would be undefined. When $pq = 1$, the process can be simplified using s-no-choice. Rule s-par-assoc expresses the

Structural pre-congruence $\boxed{P \preccurlyeq Q}$

$$\textsc{s-no-choice}\quad P \;{}_{1}\boxplus\; Q \preccurlyeq P \qquad \textsc{s-choice-idem}\quad P \;{}_{p}\boxplus\; P \equiv P \qquad \textsc{s-choice-comm}\quad P \;{}_{p}\boxplus\; Q \equiv Q \;{}_{1-p}\boxplus\; P \qquad \textsc{s-new-comm}\quad (x)(y)P \equiv (y)(x)P$$

$$\textsc{s-par-comm}\quad P \mid Q \equiv Q \mid P \qquad \textsc{s-par-choice}\quad (P \;{}_{p}\boxplus\; Q) \mid R \preccurlyeq (P \mid R) \;{}_{p}\boxplus\; (Q \mid R) \qquad \textsc{s-par-new}\quad \frac{x \notin \mathsf{fn}(Q)}{(x)P \mid Q \equiv (x)(P \mid Q)}$$

$$\textsc{s-choice-assoc}\quad \frac{pq < 1}{(P \;{}_{q}\boxplus\; Q) \;{}_{p}\boxplus\; R \equiv P \;{}_{pq}\boxplus\; (Q \;{}_{\frac{p-pq}{1-pq}}\boxplus\; R)} \qquad \textsc{s-par-assoc}\quad \frac{\mathsf{fn}(Q) \cap \mathsf{fn}(R) \neq \emptyset}{(P \mid Q) \mid R \preccurlyeq P \mid (Q \mid R)}$$

Reduction $\boxed{P \to Q}$

$$\textsc{r-com}\quad x!y.P \mid x?(y).Q \to P \mid Q \qquad \textsc{r-left}\quad \mathtt{inl}\,x.P \mid \mathtt{case}\,x\,[Q,R] \to P \mid Q \qquad \textsc{r-var}\quad \frac{A(\overline{x}) := P}{A\langle\overline{x}\rangle \to P}$$

$$\textsc{r-par}\quad \frac{P \to Q}{P \mid R \to Q \mid R} \qquad \textsc{r-new}\quad \frac{P \to Q}{(x)P \to (x)Q} \qquad \textsc{r-choice}\quad \frac{P \to Q}{P \;{}_{p}\boxplus\; R \to Q \;{}_{p}\boxplus\; R} \qquad \textsc{r-struct}\quad \frac{P \preccurlyeq R \to R' \preccurlyeq Q}{P \to Q}$$

Table 2 Structural pre-congruence and reduction of processes.

associativity property for the parallel composition. The side condition, requiring the middle and rightmost processes to be connected by one shared name, is needed by the type system (*cf.* Section 4). The reader might side conditions imposed on the associativity rule, since are limiting the ability to rewrite processes to an extent which could prevent processes to be placed next to each other and reduce according to the reduction relation. *proximity property* (Lemma D.5) ensuring that this is not the case, namely that it is always possible to rearrange (well-typed) processes in such a way that processes connected by a session can communicate. The symmetric relation $P \mid (Q \mid R) \preccurlyeq (P \mid Q) \mid R$ when $\mathsf{fn}(P) \cap \mathsf{fn}(Q) \neq \emptyset$ is derivable using s-par-assoc and s-par-comm. Rule s-par-choice distributes parallel compositions over probabilistic choices. This rule is pivotal in our model, for two different reasons. First, being able to distribute a process over a probabilistic choice is essential to make sure that processes connected by a session can be placed next to each other so that they can reduce according to $\to$. Second, the relation is quite challenging to handle at the typing level: when R is composed in parallel with P and Q, it might be necessary to type R differently depending on whether or not the session that connects R with P and Q is affected by the probabilistic choice. This is doable provided that R uses the session *safely*, namely if it does not delegate the session before it becomes aware of the probabilistic choice (*cf.* Section 4).

The reduction relation is standard. The base cases consist of the usual rules for communication eM, branch selection (r-left and r-right, the latter omitted) along with the expansion of process variables (r-var). Reduction is closed under parallel compositions (r-par), restrictions (r-new), probabilistic choices (r-choice) and structural precongruence (r-struct). Note that a probabilistic choice $P \;{}_{p}\boxplus\; Q$ is *persistent*, in the sense that neither P nor Q is discarded by reduction even though they morally represent two mutually-exclusive evolutions of the same process. This is one of the standard approaches for describing the semantics of probabilistic processes [28, 53, 37]. As a consequence, a process like $A := \mathtt{idle}\ {}_{0.001}\boxplus\ A$ diverges but terminates with probability 1. We will be able to state interesting properties of such processes through a soundness result that is relativized to the probability of termination.

We write $\Rightarrow$ for the reflexive, transitive closure of $\rightarrow$, we write $P \rightarrow$ if there exists Q such that $P \rightarrow Q$ and $P \nrightarrow$ if not $P \rightarrow$. In the above example, $A \Rightarrow P$ implies $P \rightarrow$.

▶ **Example 2.1** (Auction). We end this section showing how to represent in our calculus the auction example informally described in Section 1. We define two processes, a *Buyer* and a *Seller* connected by a session x:

$$
\begin{aligned}
\mathit{Buyer}(x) &:= x!\mathit{bid}.\mathtt{case}\, x\, [\mathtt{done}\, x, x?(y).(\mathtt{inl}\, x\ {}_q\boxplus\ \mathtt{inr}\, x.\mathit{Buyer}\langle x\rangle)] \\
\mathit{Seller}(x) &:= x?(z).(\mathtt{inl}\, x.\mathtt{done}\, x\ {}_p\boxplus\ \mathtt{inr}\, x.x!\mathit{counteroffer}.\mathtt{case}\, x\, [\mathtt{idle}, \mathit{Seller}\langle x\rangle])
\end{aligned}
$$

The buyer sends the current *bid* on x and waits for a reaction from the seller. The seller accepts the bid with probability p and rejects it with probability $1-p$. If the seller accepts (by selecting the left branch of the session), the buyer terminates successfully. Otherwise, the seller proposes a counteroffer, which the buyer rejects with probability q and accepts with probability $1-q$. In the first case, the session terminates without satisfaction of the buyer. In the second case, the buyer starts a new negotiation. ■

3 Probabilistic Session Types

Session types. Probabilistic session types describe communication protocols taking place through session endpoints and their (finite) syntax is given by the following grammar:

$$
\textbf{Session type} \qquad T, S \ ::=\ \circ \ \mid\ \bullet \ \mid\ ?t.T \ \mid\ !t.T \ \mid\ T\ {}_p\&\ S \ \mid\ T\ {}_p\oplus\ S \tag{3.1}
$$

The session types $\circ$ and $\bullet$ describe a session endpoint on which no further input/output operations are possible. We use $\bullet$ to mark those termination points of a protocol that represent success and that we target in our probabilistic analysis. The precise meaning of "successful termination" is domain specific but also irrelevant in the technical development that follows. The session types $?t.T$ and $!t.T$ describe session endpoints used for receiving (respectively, sending) a message of type t and then according to T. Types will be discussed shortly. The session types $T\ {}_p\&\ S$ and $T\ {}_p\oplus\ S$ describe a session endpoint used for receiving (respectively, sending) a binary choice which is "left" with probability p and "right" with probability $1-p$. The endpoint is then used according to T or S, respectively.

We do not use any special syntax for specifying infinite session types. Rather, we interpret the productions for T coinductively and we call session types the possibly infinite trees generated by the productions in (3.1) that satisfy the following conditions:

Regularity We require every tree to consist of finitely many *distinct* subtrees. This condition ensures that session types are finitely representable either using the so-called "μ notation" [47] or as solutions of finite sets of equations [16].

Reachability We require every subtree T of a session type to contain a *reachable leaf* labelled by $\circ$ or $\bullet$. This condition ensures that it is always possible to terminate a session regardless of how long it has been running.

To formalize these conditions, we define a relation $T \rightsquigarrow_p S$ modeling the fact that (the behavior described by) T may evolve into S with probability p in a single step:

$$
\begin{array}{llll}
\circ \rightsquigarrow_1 \circ & \quad ?t.T \rightsquigarrow_1 T & \quad T\ {}_p\&\ S \rightsquigarrow_p T & \quad T\ {}_p\&\ S \rightsquigarrow_{1-p} S \\
\bullet \rightsquigarrow_1 \bullet & \quad !t.T \rightsquigarrow_1 T & \quad T\ {}_p\oplus\ S \rightsquigarrow_p T & \quad T\ {}_p\oplus\ S \rightsquigarrow_{1-p} S
\end{array}
$$

We also consider the relation $\rightsquigarrow_p^*$, which accounts for multiple steps in the expected way:

$$T \rightsquigarrow_1^* T \qquad \frac{T \rightsquigarrow_p S}{T \rightsquigarrow_p^* S} \qquad \frac{T \rightsquigarrow_p^* T' \qquad T' \rightsquigarrow_q^* S}{T \rightsquigarrow_{pq}^* S}$$

Roughly speaking, $\rightsquigarrow_p^*$ is the reflexive, transitive closure of $\rightsquigarrow_p$ except that the probability annotation p accounts for the cumulative transition probability between two session types.

▶ **Definition 3.1** (well-formed session type)**.** *Let $\mathcal{T}(T) \overset{\text{def}}{=} \{S \mid \exists p, S : T \rightsquigarrow_p^* S\}$. A (possibly infinite) tree T generated by the productions in* (3.1) *is a* well-formed session type *if $\mathcal{T}(T)$ is finite and, for every $S \in \mathcal{T}(T)$, there exists $p > 0$ such that either $S \rightsquigarrow_p^* \circ$ or $S \rightsquigarrow_p^* \bullet$.*

▶ **Example 3.2** (auction protocol, buyer side)**.** Even though we have not presented the typing rules for the calculus of Section 2, we can speculate on the session type of the endpoint used *e.g.*, by the buyer process in Example 2.1, which satisfies the equation

$$T = !\mathsf{int}.(\bullet \;{}_p\& \; ?\mathsf{int}.(\circ \;{}_q\oplus \; T))$$

In this case we have $\mathcal{T}(T) = \{\circ, \bullet, \bullet \;{}_p\& \; (?\mathsf{int}.(\circ \;{}_q\oplus \; T)), ?\mathsf{int}.(\circ \;{}_q\oplus \; T), \circ \;{}_q\oplus \; T, T\}$ and it is easy to see that T is well formed provided that at least one among p and q is positive. ■

From now on we assume that all the session types we work with are well formed.

Success probability. We now define the probability that a protocol described by a session type T terminates successfully. Intuitively, this probability is computed by accounting for all paths in the structure of T that lead to a leaf labelled by $\bullet$. Formally:

▶ **Definition 3.3** (success probability)**.** *The* success probability *of a session type T, denoted by $[\![T]\!]$, is determined by the following equations:*

$$\begin{array}{lll} [\![\circ]\!] = 0 & [\![?t.T]\!] = [\![T]\!] & [\![T \;{}_p\& \; S]\!] = p[\![T]\!] + (1-p)[\![S]\!] \\ [\![\bullet]\!] = 1 & [\![!t.T]\!] = [\![T]\!] & [\![T \;{}_p\oplus \; S]\!] = p[\![T]\!] + (1-p)[\![S]\!] \end{array}$$

For a *finite* session type T, Definition 3.3 gives a straightforward recursive algorithm for computing $[\![T]\!]$. When T is infinite, however, it is less obvious that Definition 3.3 provides a way for determining $[\![T]\!]$. To address the problem in the general case we observe that, by interpreting $[\![T]\!]$ as a *probability variable*, Definition 3.3 allows us to derive a *finite system of equations* relating such variables. Indeed, the right hand side of each equation for $[\![T]\!]$ in Definition 3.3 is expressed in terms of probability variables corresponding to the children nodes in the tree of T. Since T has finitely many subtrees, we end up with finitely many equations. Then, we observe that every session type T corresponds to a Discrete-Time Markov Chain (DTMC) [33, 48] whose state space is $\mathcal{T}(T) = \{S_1, \dots, S_n\}$ and such that the probability p_{ij} of performing a transition from state S_i to state S_j is given by

$$p_{ij} \overset{\text{def}}{=} \begin{cases} p & \text{if } S_i \rightsquigarrow_p S_j \\ 0 & \text{otherwise} \end{cases}$$

Regularity and reachability imply that the DTMC we obtain from any session type T is finite state and absorbing. That is, it is always possible to reach an *absorbing state* (either $\circ$ or $\bullet$) from any *transient state* (any other session type). In any finite-state, absorbing DTMC, the probability of reaching a specific absorbing state from any transient state can be computed by solving a particular system of equations which is guaranteed to have a unique solution [33]. Moreover, the system that we obtain for $[\![T]\!]$ using Definition 3.3 is precisely the one whose solution is the probability of reaching $\bullet$ from T (see Appendix A).

▶ **Example 3.4.** We compute the success probability of T from Example 3.2 where, for the sake of illustration, we take $p = \frac{1}{4}$ and $q = \frac{2}{3}$. Let $T_1 = \bullet\ {}_{\frac{1}{4}}\&\ T_2$ and $T_2 = ?\mathsf{int}.T_3$ and $T_3 = \circ\ {}_{\frac{2}{3}}\oplus\ T$ be convenient names for some of its subtrees. Using Definition 3.3 we obtain the system of equations

$$\begin{array}{lll}
[\![T]\!] = [\![T_1]\!] & [\![\bullet]\!] = 1 & [\![T_3]\!] = \frac{2}{3}[\![\circ]\!] + \frac{1}{3}[\![T]\!] \\
[\![T_1]\!] = \frac{1}{4}[\![\bullet]\!] + \frac{3}{4}[\![T_2]\!] & [\![T_2]\!] = [\![T_3]\!] & [\![\circ]\!] = 0
\end{array}$$

from which we compute $[\![T]\!] = \frac{1}{3}$ (Appendix A details the corresponding DTMC). ■

Duality. We write $\overline{T}$ for the *dual* of T, that is the session type obtained from T by swapping input actions with output actions and leaving the remaining forms unchanged. Formally, $\overline{T}$ is the session type obtained from T that satisfies the following equations:

$$\begin{array}{lll}
\overline{\circ} = \circ & \overline{?t.T} = !t.\overline{T} & \overline{T\ {}_p\&\ S} = \overline{T}\ {}_p\oplus\ \overline{S} \\
\overline{\bullet} = \bullet & \overline{!t.T} = ?t.\overline{T} & \overline{T\ {}_p\oplus\ S} = \overline{T}\ {}_p\&\ \overline{S}
\end{array}$$

It is easy to see that duality is an involution (that is, $\overline{\overline{T}} = T$) and that the success probability is unaffected by duality, that is $[\![T]\!] = [\![\overline{T}]\!]$. This means that we can compute the success probability of a session from either of its two endpoints.

Types. Types describe resources used by processes and exchanged as messages. We distinguish between session endpoints, whose type is a session type T, from sessions with success probability p, whose type has the form $\langle p\rangle$:

$$\textbf{Type} \qquad t, s \ ::= \ T \ \mid \ \langle p\rangle \tag{3.2}$$

We will see in Section 4 that a type $\langle p\rangle$ results from "joining" the two peer endpoints of a session having dual sessions types T and $\overline{T}$ such that $p = [\![T]\!] = [\![\overline{T}]\!]$. For brevity we omit message types such as unit and int from the formal development as their handling is folklore and does not affect the presented results. We occasionally use them in the examples though.

A key aspect of the type system is that processes may use session endpoints differently, depending on the outcome of probabilistic choices. Nonetheless, we need to capture the overall effect of such different uses in a single type. For this reason, we introduce a *probabilistic type combinator* that allows us to combine types by weighing the different ways in which a resource is used according to a given probability.

▶ **Definition 3.5** (probabilistic type combination). *We write $t\ {}_p\boxplus\ s$ for the* combination of t and s weighed by p, *which is defined by cases on the form of t and s as follows:*

$$t\ {}_p\boxplus\ s \overset{\text{def}}{=} \begin{cases} t & \textit{if } t = s \\ T\ {}_{pq+(1-p)r}\oplus\ S & \textit{if } t = T\ {}_q\oplus\ S \textit{ and } s = T\ {}_r\oplus\ S \\ \langle pq + (1-p)r\rangle & \textit{if } t = \langle q\rangle \textit{ and } s = \langle r\rangle \\ \textit{undefined} & \textit{otherwise} \end{cases}$$

Intuitively, $t\ {}_p\boxplus\ s$ describes a resource that is used according to t with probability p and according to s with probability $1 - p$. The combination of t and s is only defined when t and s have "compatible shapes", the trivial case being when they are the same type. The interesting cases are when t and s describe a choice (a point of the protocol where one process performs a selection) and when t and s describe a session as a whole. In both cases, the success probability of the choice (respectively, of the session) is weighed by p. As an example, consider a session endpoint that is used according to $T\ {}_1\!\oplus\ S$ with probability p and according to $T\ {}_0\!\oplus\ S$ with probability $1-p$. In the first case, we are certain that the session endpoint is used for selecting "left" and then according to T. In the second case, we are certain that the session endpoint is used for selecting "right" and then according to S. Overall, the session endpoint is used according to the type $T\ {}_p\!\oplus\ S$.

The combination $\langle q\rangle\ {}_p\!\boxplus\ \langle r\rangle = \langle pq+(1-p)r\rangle$ captures the fact that the success probability of a whole session that is carried out in two different ways having success probabilities respectively q and r is the convex sum of q and r weighed by p. The success probability with which we annotate this type allows us to state the soundness properties of the type system, by relating the success probabilities in session types with those of a process that behaves according to those session types. Speaking of success probability, a fundamental property that is used extensively in the soundness proofs is the following one. Any conceivable generalization of Definition 3.5 must guarantee this property for the type system to be sound.

▶ **Proposition 3.6.** $[\![T_1\ {}_p\!\boxplus\ T_2]\!] = p[\![T_1]\!] + (1-p)[\![T_2]\!]$.

Definition 3.5 is quite conservative in that, except for top-level choices, any other session type can only be combined with itself. It is conceivable to generalize ${}_p\!\boxplus$ to permit the combination of "deep choices" found after a common prefix. For example, we could have $!\mathsf{int}.(T\ {}_1\!\oplus\ S)\ {}_p\!\boxplus\ !\mathsf{int}.(T\ {}_0\!\oplus\ S) = !\mathsf{int}.(T\ {}_p\!\oplus\ S)$. This generalization is not for free, though. As we will see in Section 4, session endpoints that are affected by a probabilistic choice must be "handled with care" and Definition 3.5 as it stands helps ensuring that this is actually the case. We leave the combination of "deep choices" to future work.

4 Typing Rules

We use contexts for tracking the type of free variables occurring in processes. A *context* is a finite map from variables to types written $x_1 : t_1, \dots, x_n : t_n$. We let Γ and Δ range over contexts, we write $\emptyset$ for the empty context, $\mathsf{dom}(\Gamma)$ for the domain of Γ and Γ, Δ for the union of Γ and Δ when $\mathsf{dom}(\Gamma) \cap \mathsf{dom}(\Delta) = \emptyset$. We also extend ${}_p\!\boxplus$ pointwise to contexts in the obvious way.

Before we discuss the typing rules, we have to introduce two predicates to single out types that have particular properties. The class of *unrestricted types*, defined next, is aimed at describing resources that can be discarded and duplicated at will.

▶ **Definition 4.1** (unrestricted type and context). *We say that t is* unrestricted *and we write* $\mathsf{un}(t)$ *if* $t = \circ$. *We write* $\mathsf{un}(\Gamma)$ *if* $\mathsf{un}(\Gamma(x))$ *for all* $x \in \mathsf{dom}(\Gamma)$.

In our case, the only unrestricted type is $\circ$, but if the type system is extended with basic types such as unit and int, these would be unrestricted as well. Next we introduce the class of *safe types*, those describing resources that can be safely sent in messages and used in process invocations because they cannot be passively affected by a probabilistic choice.

▶ **Definition 4.2** (safe type). *We write* $\mathsf{safe}(t)$ *if t is* not *of the form* $T\ {}_p\&\ S$.

The ultimate motivation for the safety predicate has its roots in the soundness proof of the type system. In a nutshell, an unsafe session type is one whose dual admits a non-trivial probabilistic combination (Definition 3.5) and therefore that may change unpredictably, from the standpoint of a process using a resource with that (unsafe) type. In this case, the process must wait to be notified of the (probabilistic) choice that has occurred before using the

$$\textsc{t-idle}\ \dfrac{\mathsf{un}(\Gamma)}{\Gamma \vdash \texttt{idle}} \qquad \textsc{t-done}\ \dfrac{\mathsf{un}(\Gamma)}{\Gamma, x : \bullet \vdash \texttt{done}\, x} \qquad \textsc{t-var}\ \dfrac{\mathsf{un}(\Gamma) \qquad A : \overline{t} \qquad \mathsf{safe}(\overline{t})}{\Gamma, \overline{x : t} \vdash A\langle \overline{x} \rangle}$$

$$\textsc{t-in}\ \dfrac{\Gamma, x : T, y : t \vdash P}{\Gamma, x : ?t.T \vdash x?(y).P} \qquad \textsc{t-branch}\ \dfrac{\Gamma, x : T \vdash P \qquad \Delta, x : S \vdash Q}{\Gamma \ {}_p\boxplus\ \Delta, x : T \ {}_p\&\ S \vdash \texttt{case}\, x\, [P, Q]}$$

$$\textsc{t-out}\ \dfrac{\Gamma, x : T \vdash P \qquad \mathsf{safe}(t)}{\Gamma, x : !t.T, y : t \vdash x!y.P} \qquad \textsc{t-left}\ \dfrac{\Gamma, x : T \vdash P}{\Gamma, x : T \ {}_1\oplus\ S \vdash \texttt{inl}\, x.P} \qquad \textsc{t-right}\ \dfrac{\Gamma, x : S \vdash P}{\Gamma, x : T \ {}_0\oplus\ S \vdash \texttt{inr}\, x.P}$$

$$\textsc{t-par}\ \dfrac{\Gamma, x : T \vdash P \qquad \Delta, x : \overline{T} \vdash Q}{\Gamma, \Delta, x : \langle [\![T]\!] \rangle \vdash P \mid Q} \qquad \textsc{t-choice}\ \dfrac{\Gamma \vdash P \qquad \Delta \vdash Q}{\Gamma \ {}_p\boxplus\ \Delta \vdash P \ {}_p\boxplus\ Q} \qquad \textsc{t-new}\ \dfrac{\Gamma, x : \langle p \rangle \vdash P}{\Gamma \vdash (x)P}$$

Table 3 Typing rules.

resource in a message. Should the need arise to send an unsafe endpoint in a message, it is possible to patch the endpoint's session type so as to make it safe, for example by prefixing the session type with a dummy input/output action. We will see an instance where this patch is necessary in Example 4.12.

Judgments have the form $\Gamma \vdash P$, meaning that P is well typed in Γ, and are derived by the rules in Table 3. The typing rules are syntax directed, so that each process form corresponds to a typing rule. We now discuss each rule in detail. Rules T-IDLE and T-DONE deal with terminated processes. In T-IDLE the whole context must be unrestricted, since the `idle` process does not use any resource. Rule T-DONE is similar, except that the session x flagged by the process must have type $\bullet$. This way, we enforce the correspondence between successful termination in processes and successful termination in session types. Rule T-VAR establishes that a process invocation is well typed provided that the type of the parameters passed to the process match the expected ones and that any unused resource has an unrestricted type. Observe that the type of such parameters must be safe. This way, we prevent to use as parameters resources whose type can be (passively) affected by a probabilistic choice. Rules T-IN and T-OUT deal with the exchange of a message y on session x. The rules update the type of x from the conclusion to the premise of the rule to account for the communication. As usual, a linear resource y being sent in a message is no longer available in the continuation of the process. As anticipated earlier, T-OUT requires the type of y to be safe, again to ensure that the type of y does not suddenly change under the effect of a probabilistic choice.

The typing rules described so far are fairly standard for any session calculus. We now move on to the part of the type system that handles probabilities. Rules T-LEFT and T-RIGHT deal with selections. In these cases, the type of x must be of the form $T \ {}_p\oplus\ S$ and the process continuation uses x according to either T or S respectively. The key aspect is the probability p with which the process selects "left", which is 1 in the case of $\texttt{inl}\, x$ and 0 in the case of $\texttt{inr}\, x$. These processes behave deterministically, hence the probability annotation in the session type is trivial. Rule T-BRANCH illustrates the typing of a branch, whereby a process receives a choice from a session x and continues accordingly. The type of x must be of the form $T \ {}_p\&\ S$, where p is the probability with which the process will receive a "left" choice. The key part of the rule concerns *all the other resources* used by the process, which will be used according to Γ if the process receives a "left" choice and according to Δ otherwise. That is, the process is becoming aware of a probabilistic choice that has been performed elsewhere and whose outcome is communicated on x. Depending on this information, the process uses its resources (not just x) accordingly. The behavior of the process as a whole is described by the combination $\Gamma \;{}_{p}\!\boxplus\; \Delta$ of the contexts in the two branches. Recall that the ${}_{p}\!\boxplus$ operator, when used on session types, is idempotent in all cases but for selections (Definition 3.5). Hence, $\Gamma \;{}_{p}\!\boxplus\; \Delta$ is *nearly the same* as Γ and Δ, except that the probabilities with which some future selections will be performed on endpoints in Γ and Δ may have been adjusted as a side effect of the information received from x. This mechanism enables the propagation of probabilistic choices through the system as messages are exchanged on sessions.

Rule T-PAR deals with parallel compositions $P \mid Q$, where P and Q must use x according to dual session types. Writing Γ, Δ in the conclusion of the rule ensures that P and Q do not share any name other than x, thus preventing the creation of network topologies that may lead to deadlocks [12]. In the conclusion of the rule the type of x becomes of the form $\langle p \rangle$ to record the fact that both endpoints of x have been used. The success probability p coincides with that of one of the endpoints and is well defined since $[\![T]\!] = [\![\overline{T}]\!]$. Rule T-CHOICE deals with probabilistic choices performed by a process and partially overlaps with T-BRANCH in that the contexts of the two alternative evolutions of the process after the choice are combined by ${}_{p}\!\boxplus$. Finally, rule T-NEW removes a session x from the context when x is restricted.

Let us now discuss the main properties enjoyed by well-typed processes. First and foremost, typing is preserved by reductions.

▶ **Theorem 4.3** (subject reduction). *If $\Gamma \vdash P$ and $P \to Q$, then $\Gamma \vdash Q$.*

Although this result is considered standard, one detail makes it special in our setting. Specifically, we observe that the reduct Q is well typed in the *very same environment* used for typing P, despite the fact that a communication may have taken place on a session x in P, determining a change in the session types associated with the endpoints of x. A communication can occur only if P contains *both* endpoints for x, and more precisely if there are two subprocesses of P that use x according to dual session types and that are composed in parallel using T-PAR. Then, x in Γ must be associated with a type of the form $\langle p \rangle$, where p is the success probability of P. Then, Theorem 4.3 guarantees that not only the typing, but also the *success probability of sessions is preserved by reductions*. This is counterintuitive at first, given that a session may evolve through different branches each having different success probabilities. However, recall that probabilistic choices are *persistent* in our calculus, meaning that the reduct Q accounts for *all possible evolutions* of P. This is what entails such strong formulation of Theorem 4.3.

Next we turn our attention to termination. To this aim, we provide two characterizations of process termination respectively concerning the present and the future states of a process.

▶ **Definition 4.4** (immediate and eventual termination). *We say that P is* terminated *if $P\!\downarrow$ is derivable using the following axioms and rules:*

$$\texttt{idle}\!\downarrow \qquad \texttt{done}\, x\!\downarrow \qquad \frac{P\!\downarrow \qquad Q\!\downarrow}{P \mid Q\!\downarrow} \qquad \frac{P\!\downarrow \qquad Q\!\downarrow}{P \;{}_{p}\!\boxplus\; Q\!\downarrow} \qquad \frac{P\!\downarrow}{(x)P\!\downarrow}$$

We say that P terminates with probability *p, notation $P\!\Downarrow_p$, if there exist (P_n), (Q_n) and (p_n) for $n \in \mathbb{N}$ such that $P \Rightarrow P_n \;{}_{p_n}\!\boxplus\; Q_n$ and $P_n\!\downarrow$ for every $n \in \mathbb{N}$ and $\lim_{n\to\infty} p_n = p$.*

In words, $P\!\downarrow$ means that P does not contain any pending communications, whereas $P\!\Downarrow_p$ means that P evolves with probability p to states in which there are no pending communications. Our type system is not strong enough to guarantee (probable) termination.

.

▶ **Theorem 4.5** (deadlock freedom). *If $\emptyset \vdash P$ and $P \Rightarrow Q$, then either $Q \rightarrow$ or $Q\downarrow$.*

Note that deadlock freedom is not simply a bonus feature of our type system. It is actually a requirement for proving the properties of the type system that specifically pertain probabilities, which we will discuss shortly. Before doing so, we need an operational characterization of successful termination relative to a particular session.

▶ **Definition 4.6** (successful termination of a session). *We say that P* successfully terminates session x with probability p *if $P \uparrow^x_p$ is derivable using the following axioms and rules:*

$$\textsc{p-done}\ \frac{}{\mathtt{done}\,x \uparrow^x_1} \qquad \textsc{p-par-1}\ \frac{P \uparrow^x_p}{P \mid Q \uparrow^x_p} \qquad \textsc{p-par-2}\ \frac{Q \uparrow^x_p}{P \mid Q \uparrow^x_p} \qquad \textsc{p-res}\ \frac{P \uparrow^x_p \qquad x \neq y}{(y)P \uparrow^x_p} \qquad \textsc{p-choice}\ \frac{P \uparrow^x_q \qquad Q \uparrow^x_r}{P \;{}_p\!\boxplus\; Q \uparrow^x_{pq+(1-p)r}} \qquad \textsc{p-any}\ \frac{}{P \uparrow^x_0}$$

Axiom p-done states that a process of the form $\mathtt{done}\,x$ has successfully terminated session x with probability 1. The rules p-par-*i* state that the successful termination of a parallel composition $P \mid Q$ with respect to a session x can be reduced to the successful termination of either P or Q. In particular, we do not require that *both* P and Q have successfully terminated x, for two reasons: first, it could be the case that P and Q are connected by a session different from x, hence only one among P and Q could own x; second, if a process has successfully terminated a session through one of its endpoints, then duality ensures that the peer owning the other endpoint cannot have pending operations on it, so the session as a whole can be considered successfully terminated even if only one peer has become $\mathtt{done}\,x$.

Rule p-res accounts for session restrictions in the expected way and p-choice states that the successful termination of x in a process distribution is obtained by weighing the probabilities of successful termination of the processes in the distribution. Note that p-choice can be applied only if it is possible to derive the successful termination of x for *all* of the processes in the distribution, whereas in general only *some* of such processes will have successfully terminated x. To account for this possibility, we can use p-any to *approximate* the probability of successful termination of x for any process to 0.

The type system gives us an upper bound to the success probability of any session:

▶ **Proposition 4.7.** *If $x : \langle p\rangle \vdash P$ and $P \uparrow^x_q$, then $q \leq p$.*

In particular, a session with type $\langle 0\rangle$ cannot be successfully completed, which could indicate a flaw in the system. The upper bound is matched exactly by terminated processes:

▶ **Theorem 4.8.** *If $x : \langle p\rangle \vdash P$ and $P \nrightarrow$, then $P \uparrow^x_p$.*

Note that Theorem 4.8 does not hold unless processes are deadlock free, whence the key role of Theorem 4.5. As stated, Theorem 4.8 appears of limited use since it only concerns processes that cannot reduce any further, whereas in general we are interested in computing the probability of successful termination also for processes engaged in arbitrarily long interactions, for which the predicate $P \nrightarrow$ might never hold. It turns out that Theorem 4.8 can be relativized to the probability that a process terminates, thus:

▶ **Corollary 4.9** (relative success). *Let $P \Uparrow^x_p$ if there exist (P_n) and (p_n) such that $P \Rightarrow P_n$ and $P_n \uparrow^x_{p_n}$ for all $n \in \mathbb{N}$ and $\lim_{n\to\infty} p_n = p$. Then (1) $x : \langle 1\rangle \vdash P$ and $P \Downarrow_p$ imply $P \Uparrow^x_p$ and (2) $x : \langle p\rangle \vdash P$ and $P \Downarrow_1$ imply $P \Uparrow^x_p$.*

Property (1) states that a well-typed process using a session with type $x : \langle 1\rangle$ successfully completes the session with the same probability with which it terminates. Property (2) extends Theorem 4.8 to processes that are known to terminate with probability 1.

▶ **Example 4.10.** Below is the type derivation for the process *Buyer* from Example 2.1 using T from Example 3.2 and assuming the type assignment $\mathit{Buyer} : T$.

$$\dfrac{\dfrac{}{x : \bullet \vdash \mathtt{done}\,x}\ \text{T-DONE} \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{x : \circ \vdash \mathtt{idle}}\ \text{T-IDLE}}{x : \circ\ {}_{1}\!\oplus T \vdash \mathtt{inl}\,x}\ \text{T-LEFT} \qquad \dfrac{\dfrac{}{x : T, y : \mathsf{int} \vdash \mathit{Buyer}\langle x\rangle}\ \text{T-VAR}}{x : \circ\ {}_{0}\!\oplus T, y : \mathsf{int} \vdash \mathtt{inr}\,x.\mathit{Buyer}\langle x\rangle}\ \text{T-RIGHT}}{x : \circ\ {}_{q}\!\oplus T, y : \mathsf{int} \vdash \mathtt{inl}\,x\ {}_{q}\!\boxplus \mathtt{inr}\,x.\mathit{Buyer}\langle x\rangle}\ \text{T-CHOICE}}{x : ?\mathsf{int}.(\circ\ {}_{q}\!\oplus T) \vdash x?(y).(\mathtt{inl}\,x\ {}_{q}\!\boxplus \mathtt{inr}\,x.\mathit{Buyer}\langle x\rangle)}\ \text{T-IN}}{\dfrac{x : \bullet\ {}_{p}\!\&\ ?\mathsf{int}.(\circ\ {}_{q}\!\oplus T) \vdash \mathtt{case}\,x\,[\mathtt{done}\,x, x?(y).(\mathtt{inl}\,x\ {}_{q}\!\boxplus \mathtt{inr}\,x.\mathit{Buyer}\langle x\rangle)]}{x : T \vdash x!\mathit{bid}.\mathtt{case}\,x\,[\mathtt{done}\,x, x?(y).(\mathtt{inl}\,x\ {}_{p}\!\boxplus \mathtt{inr}\,x.\mathit{Buyer}\langle x\rangle)]}\ \text{T-OUT}}\ \text{T-BRANCH}$$

Observe the application of T-CHOICE, which turns the probabilistic choice $\circ\ {}_{q}\!\oplus T$ in the conclusion of the rule into a deterministic one in the two premises. There exists an analogous derivation for $x : \overline{T} \vdash Q$ where Q is the body of $\mathit{Seller}\langle x\rangle$ in Example 2.1. By taking p and q as in Example 3.4, we derive $x : \langle\frac{1}{3}\rangle \vdash \mathit{Buyer}\langle x\rangle \mid \mathit{Seller}\langle x\rangle$ with one application of T-PAR. It is easy to establish that this process terminates with probability 1, hence by Corollary 4.9(2) the buyer wins the auction with probability $\frac{1}{3}$. ■

▶ **Example 4.11.** The separation of probabilistic choices from the communication of information ("left" and "right" selections) that depends on such choices implies that there is no 1-to-1 correspondence between choices as seen in session types and choices performed by processes. Below are a few instances in which the type system performs a non-trivial reconciliation between the probability annotations in types and those in processes. The type derivations are detailed in Appendix B.1.

1. The process $\mathtt{case}\,x\,[\mathtt{inr}\,y.\mathtt{done}\,x, \mathtt{inl}\,y.\mathtt{done}\,y]$ inverts a choice from session x to y, so that it successfully completes x if and only if it does not successfully complete y. It is well typed in the context $x : \bullet\ {}_{p}\!\&\ \circ, y : \bullet\ {}_{1-p}\!\oplus \circ$, which reflects the effect of the inversion.
2. The process $\mathtt{case}\,x\,[\mathtt{case}\,y\,[\mathtt{inl}\,z.\mathtt{done}\,z, \mathtt{inr}\,z], \mathtt{case}\,y\,[\mathtt{inr}\,z, \mathtt{inr}\,z]]$ coalesces two choices received from x and y into a choice sent on z. The process is well typed in the context $x : \circ\ {}_{p}\!\&\ \circ, y : \circ\ {}_{q}\!\&\ \circ, z : \bullet\ {}_{pq}\!\oplus \circ$, indicating that the success probability for z is the product of the probabilities of receiving "left" from both x and y.
3. The process $\mathtt{inl}\,x.\mathtt{inl}\,x.\mathtt{done}\,x\ {}_{\frac{1}{2}}\!\boxplus \mathtt{inr}\,x.\mathtt{inr}\,x$ sends the same probabilistic choice twice on session x. It is well typed in the context $x : (\bullet\ {}_{1}\!\oplus \circ)\ {}_{\frac{1}{2}}\!\oplus (\circ\ {}_{0}\!\oplus \circ)$ but *not* in the context $x : (\bullet\ {}_{\frac{1}{2}}\!\oplus \circ)\ {}_{\frac{1}{2}}\!\oplus (\circ\ {}_{\frac{1}{2}}\!\oplus \circ)$. Once the choice is communicated, subsequent "left" or "right" selections that depend on that choice become deterministic. ■

▶ **Example 4.12** (Work sharing)**.** Consider a system $C\langle x\rangle \mid x?(z).B\langle x, y, z\rangle \mid A\langle y\rangle$ modeling (from left to right) a master process C connected with two slave processes which can be "busy" handling jobs or "idle" waiting for jobs. The processes are defined as follows:

$$\begin{aligned}
C(x) &:= x!\mathit{job}.\mathtt{case}\,x\,[\mathtt{done}\,x, \mathtt{idle}]\\
B(x, y, \mathit{job}) &:= y!\langle\rangle.\big(\mathtt{inl}\,x.\mathtt{inl}\,y.\mathtt{done}\,x\ {}_{p}\!\boxplus \big(\mathtt{inr}\,x.\mathtt{inl}\,y\ {}_{q}\!\boxplus \mathtt{inr}\,y.y!x.y!\mathit{job}.A\langle y\rangle\big)\big)\\
A(y) &:= y?().\mathtt{case}\,y\,[\mathtt{idle}, y?(x).y?(z).B\langle x, y, z\rangle]
\end{aligned}$$

The master sends a job to the first slave and waits for a notification indicating whether the job has been handled or not. Obviously, the master succeeds only in the first case. A busy slave decides whether to handle the job (with probability p) or not (with probability $1-p$). In the first case, it notifies the master and the idle slave that the job has been handled and terminates. In the second case, it decides whether to discard the job (with probability q) or to hand it over to the other slave (with probability $1-q$). Note that the busy slave sends on y a dummy value to the idle one before taking any decision so that the type of y is *safe* when y is used in $A\langle y\rangle$. This way, by the time the busy slave makes a probabilistic choice that may affect (and will be communicated to) the idle slave, the idle slave is blocked on a `case` waiting for such choice, and therefore its typing can be suitably adjusted when it is moved (by s-par-choice) into the scope of the choice.

Now, take $T = !\mathsf{unit}.(\circ\ {}_{p-pq+q}\!\oplus\ !S.!\mathsf{int}.\overline{T})$ and $S = \bullet\ {}_{r}\!\oplus\ \circ$ where $\max\{p,q\} > 0$ and $r = \frac{p}{p-pq+q}$. It is possible to show that the above composition is well typed under the global type assignments $C : !\mathsf{int}.\overline{S}$, $B : S, T, \mathsf{int}$ and $A : \overline{T}$, where we assume that *job* has type int. From the fact that the system terminates with probability 1, we conclude that the master succeeds with probability r. Details can be found in Appendix B.2. ■

5 Related Work

Type systems for probabilistic, concurrent programs. Despite their close relationship with process algebras, many of which have been extensively studied in a probabilistic setting, there are few results concerning probabilistic variants of session types. A notable exception is [2], which considers a probabilistic variant of multiparty session types (MST) eM

Some type systems for probabilistic programs have been developed to characterize precisely the space of the possible execution traces [38] or to ensure that well-typed programs do not leak secret information [17]. The work [53] considers a sub-structural type system for a probabilistic variant of the linear π-calculus. Although the type system is not concerned with probabilities directly, there are interesting analogies with our typing discipline: it is only by relying on the properties of well-typed processes – most notably, race and deadlock freedom – that we are able to relate the probabilities in processes with those in types.

Probabilistic models of concurrent processes. The design of computational models that combine concurrency and probabilities has a long tradition [54, 49] and gave birth to a variety of operational approaches [50] and concrete probabilistic extensions of well-known concurrency models, such as CCS [27], CSP [40, 25], Petri nets [9], Klaim [19], and name-passing process calculi [28, 53, 42, 26]. Our language for processes can be seen as the session-based counterpart of (a synchronous version of) the *simple probabilistic π-calculus* [42], which features both probabilistic and non-deterministic choices. While non-deterministic choices in [42] correspond to the standard choice operator ($+$) of the π-calculus, we adopted a session discipline, and hence a choice is realised by communicating a label over a session.

The development of a denotational semantics for languages that combine non-determinism, concurrency and probabilities has revealed challenging. On the one hand, probabilistic choices do not distribute over non-deterministic ones, *i.e.*, it matters whether the environment chooses before or after a probabilistic choice is made, as highlighted in [52]. This observation appears to be reflected in our type system by the typing rules that require a term to be of a safe type, *e.g.*, when a session is delegated. Establishing a precise connection between these two notions may pave the way for generalisations of our probabilistic type combinator. On the other hand, probabilistic choices in a system need to be (probabilistically) independent. This problem is connected with the well-known *confusion phenomenon*, in which concurrent (and hence, independent) choices may influence each other (*e.g.*, one choice may enable/disable some branch in another choice). As shown in [1, 32, 11], confusion can be avoided by establishing an order in which choices are executed; essentially, by reducing concurrency. We remark that the session discipline imposed by our language – and rule t-par in particular – makes all probabilistic choices independent (in a probabilistic sense).

Probabilistic languages and analyses. Probabilistic models are frequently used to prove properties that can be expressed as reachability probabilities; they are then verified by model-checking [31]. Our types are also reachability properties related to the probability of successful completion of a session. Besides, our type system guarantees deadlock-freedom. Many approaches have been recently proposed for reasoning on probabilistic programs, *e.g.*, deductive-style approaches based on separation logic [6, 51, 5], probabilistic strategy logic [3], proof of termination [23, 36], static analysis [55], and probabilistic symbolic execution [10]. Typing has been used in the sequential setting to ensure almost-sure termination in a probabilistic lambda calculus [34]. Our type system does not ensure termination, but it could form the basis for a probabilistic termination analysis.

6 Concluding Remarks

In this work we start the study of a type-based static analysis technique for reasoning on probabilistic reachability problems in session-based systems. We relate a probabilistic variant of a session-based calculus (Section 2) with a probabilistic variant of binary session types (Section 3) and establish a correspondence between probability annotations in processes and those in types (Section 4). By breaking down a complex system of communicating processes into sessions, we are able to modularly infer properties concerning the (probable) evolution of the system from the much simpler specifications described by session types.

There are many developments that stem from this work addressing both technical and practical problems. Here we discuss those looking more promising or intriguing.

To make our approach practical, the type system must be supported by suitable type checking and inference algorithms. Indeed, even though the typing rules are syntax directed, the probabilistic type combinator (Definition 3.5) is difficult to deal with because it is not injective (the same type can result from combining types with different probability annotations). We are also considering extensions of the very same operator so that it is applicable to "deep choices" that do not necessarily occur at the top level of a session type. This extension requires a careful balancing with the notion of type safety (Definition 4.2).

Subtyping relations for session types [24] are important for addressing realistic programming scenarios. Given the already established connections between session subtyping and (fair) testing relations [35, 13, 44, 8, 46] and the extensive literature on probabilistic testing relations [14, 43, 21, 20] and behavioral equivalences [39], the investigation of probabilistic variants of session subtyping has solid grounds to build upon. A related problem is that process models that feature both non-deterministic and probabilistic choices are known to be difficult to model and analyze [21]. It could be the case that session-based systems with both non-deterministic and probabilistic choices are easier to address thanks to their simpler structure, as already observed in [53].

Our analysis based on probabilistic session types can be extended in several ways. For example, it would be interesting to quantify the probability of (partial) execution traces rather than (or in addition to) the reachability of "successful states". eMProbability ranges could also be useful in those cases where probability annotations in processes are uncertain, possibly because they have been estimated from execution traces [22]. catia We think that probability annotations in session types may also support forms of static analysis aimed at quantifying the termination probability of session-based programs. Known type systems that ensure progress, deadlock and livelock freedom are often quite constraining on the structure of well-typed programs [45, 15, 4]. It could be the case that switching to a probabilistic setting broadens substantially the range of addressable programs.

References

1 Samy Abbes and Albert Benveniste. True-concurrency probabilistic models: Branching cells and distributed probabilities for event structures. *Information and Computation*, 204(2):231–274, 2006. doi:10.1016/j.ic.2005.10.001.

2 Bogdan Aman and Gabriel Ciobanu. Probabilities in session types. In Mircea Marin and Adrian Craciun, editors, *Proceedings Third Symposium on Working Formal Methods, FROM 2019, Timişoara, Romania, 3-5 September 2019*, volume 303 of *EPTCS*, pages 92–106, 2019. doi:10.4204/EPTCS.303.7.

3 Benjamin Aminof, Marta Kwiatkowska, Bastien Maubert, Aniello Murano, and Sasha Rubin. Probabilistic strategy logic. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 32–38, 2019. doi:10.24963/ijcai.2019/5.

4 Stephanie Balzer, Bernardo Toninho, and Frank Pfenning. Manifest deadlock-freedom for shared session types. In *Proceedings of the European Symposium on Programming Languages (ESOP)*, volume 11423, pages 611–639. Springer, 2019. doi:10.1007/978-3-030-17184-1_22.

5 Gilles Barthe, Justin Hsu, and Kevin Liao. A probabilistic separation logic. *Proc. ACM Program. Lang.*, 4(POPL):55:1–55:30, 2020. doi:10.1145/3371123.

6 Kevin Batz, Benjamin Lucien Kaminski, Joost-Pieter Katoen, Christoph Matheja, and Thomas Noll. Quantitative separation logic: a logic for reasoning about probabilistic pointer programs. *Proc. ACM Program. Lang.*, 3(POPL):34:1–34:29, 2019. doi:10.1145/3290347.

7 Yakov Ben-Haim. *Info-gap decision theory: decisions under severe uncertainty.* Academic Press, 2006.

8 Giovanni Bernardi and Matthew Hennessy. Using higher-order contracts to model session types. *Logical Methods in Computer Science*, 12(2), 2016. doi:10.2168/LMCS-12(2:10)2016.

9 Rémi Bonnet, Stefan Kiefer, and Anthony Widjaja Lin. Analysis of probabilistic basic parallel processes. In *Proceedings of the International Conference on Foundations of Software Science and Computation Structures (FoSSaCS)*, volume 8412, pages 43–57. Springer, 2014. doi:10.1007/978-3-642-54830-7_3.

10 Mateus Borges, Antonio Filieri, Marcelo d'Amorim, and Corina S. Pasareanu. Iterative distribution-aware sampling for probabilistic symbolic execution. In *Proceedings of the Joint Meeting on Foundations of Software Engineering (ESEC/FSE)*, pages 866–877, 2015. doi:10.1145/2786805.2786832.

11 Roberto Bruni, Hernán C. Melgratti, and Ugo Montanari. Concurrency and probability: Removing confusion, compositionally. *Log. Methods Comput. Sci.*, 15(4), 2019. doi:10.23638/LMCS-15(4:17)2019.

12 Luís Caires, Frank Pfenning, and Bernardo Toninho. Linear logic propositions as session types. *Math. Struct. Comput. Sci.*, 26(3):367–423, 2016. doi:10.1017/S0960129514000218.

13 Giuseppe Castagna, Mariangiola Dezani-Ciancaglini, Elena Giachino, and Luca Padovani. Foundations of session types. In António Porto and Francisco Javier López-Fraguas, editors, *Proceedings of the International Conference on Principles and Practice of Declarative Programming (PPDP)*, pages 219–230. ACM, 2009. doi:10.1145/1599410.1599437.

14 Rance Cleaveland, Zeynep Dayar, Scott A. Smolka, and Shoji Yuen. Testing preorders for probabilistic processes. *Inf. Comput.*, 154(2):93–148, 1999. doi:10.1006/inco.1999.2808.

15 Mario Coppo, Mariangiola Dezani-Ciancaglini, Nobuko Yoshida, and Luca Padovani. Global progress for dynamically interleaved multiparty sessions. *Math. Struct. Comput. Sci.*, 26(2):238–302, 2016. doi:10.1017/S0960129514000188.

16 Bruno Courcelle. Fundamental properties of infinite trees. *Theor. Comput. Sci.*, 25:95–169, 1983. doi:10.1016/0304-3975(83)90059-2.

17 David Darais, Ian Sweet, Chang Liu, and Michael Hicks. A language for probabilistically oblivious computation. *Proc. ACM Program. Lang.*, 4(Proceedings of the ACM SIGPLAN Symposium on Principles of Programming Languages (POPL)):50:1–50:31, 2020. doi:10.1145/3371118.

18 Ornela Dardha and Simon J. Gay. A new linear logic for deadlock-free session-typed processes. In Christel Baier and Ugo Dal Lago, editors, *Foundations of Software Science and Computation Structures - 21st International Conference, FOSSACS 2018, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2018, Thessaloniki, Greece, April 14-20, 2018, Proceedings*, volume 10803 of *Lecture Notes in Computer Science*, pages 91–109. Springer, 2018. doi:10.1007/978-3-319-89366-2_5.

19 Rocco De Nicola, Diego Latella, and Mieke Massink. Formal modeling and quantitative analysis of klaim-based mobile systems. In *Proceedings of the ACM symposium on Applied computing (SAC)*, pages 428–435, 2005. doi:10.1145/1066677.1066777.

20 Yuxin Deng, Rob Van Glabbeek, Matthew Hennessy, and Carroll Morgan. Testing finitary probabilistic processes. In *Proceedings of the International Conference on Concurrency Theory (CONCUR)*, pages 274–288. Springer, 2009. doi:10.1007/978-3-642-04081-8_19.

21 Yuxin Deng, Rob J. van Glabbeek, Matthew Hennessy, Carroll Morgan, and Chenyi Zhang. Remarks on testing probabilistic processes. *Electron. Notes Theor. Comput. Sci.*, 172:359–397, 2007. doi:10.1016/j.entcs.2007.02.013.

22 Seyedeh Sepideh Emam and James Miller. Inferring extended probabilistic finite-state automaton models from software executions. *ACM Trans. Softw. Eng. Methodol.*, 27(1):4:1–4:39, 2018. doi:10.1145/3196883.

23 Luis María Ferrer Fioriti and Holger Hermanns. Probabilistic termination: Soundness, completeness, and compositionality. In *Proceedings of the ACM SIGPLAN Symposium on Principles of Programming Languages (POPL)*, pages 489–501. ACM, 2015. doi:10.1145/2775051.2677001.

24 Simon J. Gay and Malcolm Hole. Subtyping for session types in the pi calculus. *Acta Inf.*, 42(2-3):191–225, 2005. doi:10.1007/s00236-005-0177-z.

25 Sonja Georgievska and Suzana Andova. Probabilistic CSP: preserving the laws via restricted schedulers. In *Proceedings of the International GI/ITG Conference on Measurement, Modelling, and Evaluation of Computing Systems and Dependability and Fault Tolerance (MMB/DFT)*, volume 7201, pages 136–150. Springer, 2012. doi:10.1007/978-3-642-28540-0_10.

26 Jean Goubault-Larrecq, Catuscia Palamidessi, and Angelo Troina. A probabilistic applied pi-calculus. In Zhong Shao, editor, *Programming Languages and Systems, 5th Asian Symposium, APLAS 2007, Singapore, November 29-December 1, 2007, Proceedings*, volume 4807 of *Lecture Notes in Computer Science*, pages 175–190. Springer, 2007. doi:10.1007/978-3-540-76637-7_12.

27 Hans A. Hansson. Time and probabilities in specification and verification of real-time systems. In *Proceedings of the Euromicro workshop on Real-Time Systems (RTS)*, pages 92–97, 1992. doi:10.1109/EMWRT.1992.637477.

28 Oltea Mihaela Herescu and Catuscia Palamidessi. Probabilistic asynchronous π-calculus. In *Proceedings of the International Conference on Foundations of Software Science and Computation Structures (FoSSaCS)*, volume 1784, pages 146–160. Springer, 2000. doi:10.1007/3-540-46432-8_10.

29 Kohei Honda. Types for dyadic interaction. In Eike Best, editor, *Proceedings of the International Conference on Concurrency Theory (CONCUR)*, volume 715, pages 509–523. Springer, 1993. doi:10.1007/3-540-57208-2_35.

30 Hans Hüttel, Ivan Lanese, Vasco T. Vasconcelos, Luís Caires, Marco Carbone, Pierre-Malo Deniélou, Dimitris Mostrous, Luca Padovani, António Ravara, Emilio Tuosto, Hugo Torres Vieira, and Gianluigi Zavattaro. Foundations of session types and behavioural contracts. *ACM Comput. Surv.*, 49(1):3:1–3:36, 2016. doi:10.1145/2873052.

31 Joost-Pieter Katoen. The probabilistic model checking landscape. In Martin Grohe, Eric Koskinen, and Natarajan Shankar, editors, *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '16, New York, NY, USA, July 5-8, 2016*, pages 31–45. ACM, 2016. doi:10.1145/2933575.2934574.

32 Joost-Pieter Katoen and Doron A. Peled. Taming confusion for modeling and implementing probabilistic concurrent systems. In Matthias Felleisen and Philippa Gardner, editors, *Programming Languages and Systems - 22nd European Symposium on Programming, ESOP 2013, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2013, Rome, Italy, March 16-24, 2013. Proceedings*, volume 7792 of *Lecture Notes in Computer Science*, pages 411–430. Springer, 2013. doi:10.1007/978-3-642-37036-6_23.

33 John G. Kemeny and J. Laurie Snell. *Finite Markov Chains*. Springer-Verlag, 1976.

34 Ugo Dal Lago and Charles Grellois. Probabilistic termination by monadic affine sized typing. *ACM Trans. Program. Lang. Syst.*, 41(2):10:1–10:65, 2019. doi:10.1145/3293605.

35 Cosimo Laneve and Luca Padovani. The pairing of contracts and session types. In *Concurrency, Graphs and Models, Essays Dedicated to Ugo Montanari on the Occasion of His 65th Birthday*, volume 5065, pages 681–700. Springer, 2008. doi:10.1007/978-3-540-68679-8_42.

36 Ondrej Lengál, Anthony Widjaja Lin, Rupak Majumdar, and Philipp Rümmer. Fair termination for parameterized probabilistic concurrent systems. In *Proceedings of the International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 10205, pages 499–517, 2017. doi:10.1007/978-3-662-54577-5_29.

37 Thomas Leventis. A deterministic rewrite system for the probabilistic λ-calculus. *Math. Struct. Comput. Sci.*, 29(10):1479–1512, 2019. doi:10.1017/S0960129519000045.

38 Alexander K. Lew, Marco F. Cusumano-Towner, Benjamin Sherman, Michael Carbin, and Vikash K. Mansinghka. Trace types and denotational semantics for sound programmable inference in probabilistic languages. *Proc. ACM Program. Lang.*, 4(POPL):19:1–19:32, 2020. doi:10.1145/3371087.

39 Natalia López and Manuel Núñez. An overview of probabilistic process algebras and their equivalences. In *Validation of Stochastic Systems - A Guide to Current Research*, volume 2925, pages 89–123. Springer, 2004. doi:10.1007/978-3-540-24611-4_3.

40 Gavin Lowe. Probabilistic and prioritized models of timed CSP. *Theor. Comput. Sci.*, 138(2):315–352, 1995. doi:10.1016/0304-3975(94)00171-E.

41 Ramon E. Moore, R. Baker Kearfott, and Michael J. Cloud. *Introduction to Interval Analysis*. SIAM, 2009. doi:10.1137/1.9780898717716.

42 Gethin Norman, Catuscia Palamidessi, David Parker, and Peng Wu. Model checking the probabilistic pi-calculus. In *Fourth International Conference on the Quantitative Evaluaiton of Systems (QEST 2007), 17-19 September 2007, Edinburgh, Scotland, UK*, pages 169–178. IEEE Computer Society, 2007. doi:10.1109/QEST.2007.31.

43 Manuel Núñez and David Rupérez. Fair testing through probabilistic testing. In *Proceedings of the Joint International Conference on Formal Description Techniques for Distributed Systems and Communication Protocols and Protocol Specification, Testing and Verification (PSTV)*, volume 156, pages 135–150. Kluwer, 1999. doi:10.1007/978-0-387-35578-8_8.

44 Luca Padovani. Fair subtyping for open session types. In *Proceedings of the International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 7966, pages 373–384. Springer, 2013. doi:10.1007/978-3-642-39212-2_34.

45 Luca Padovani. Deadlock and lock freedom in the linear π-calculus. In *Proceedings of the Joint Meeting of the EACSL Annual Conference on Computer Science Logic and the Annual ACM/IEEE Symposium on Logic in Computer Science (CSL-LICS)*, pages 72:1–72:10. ACM, 2014. doi:10.1145/2603088.2603116.

46 Luca Padovani. Fair subtyping for multi-party session types. *Math. Struct. Comput. Sci.*, 26(3):424–464, 2016. doi:10.1017/S096012951400022X.

47 Benjamin C. Pierce. *Types and programming languages*. MIT Press, 2002.

48 Jan J. M. M. Rutten, Marta Z. Kwiatkowska, Gethin Norman, David Parker, and Prakash Panangaden. *Mathematical techniques for analyzing concurrent and probabilistic systems*, volume 23 of *CRM monograph series*. American Mathematical Society, 2004.

49 Roberto Segala and Nancy Lynch. Probabilistic simulations for probabilistic processes. *Nordic Journal of Computing*, 2(2):250–273, 1995.

50 Ana Sokolova and Erik P. de Vink. Probabilistic automata: System types, parallel composition and comparison. In Christel Baier, Boudewijn R. Haverkort, Holger Hermanns, Joost-Pieter Katoen, and Markus Siegle, editors, *Validation of Stochastic Systems - A Guide to Current Research*, volume 2925 of *Lecture Notes in Computer Science*, pages 1–43. Springer, 2004. doi:10.1007/978-3-540-24611-4_1.

51 Joseph Tassarotti and Robert Harper. A separation logic for concurrent randomized programs. *Proc. ACM Program. Lang.*, 3(POPL):64:1–64:30, 2019. doi:10.1145/3290377.

52 Daniele Varacca and Glynn Winskel. Distributing probability over non-determinism. *Math. Struct. Comput. Sci.*, 16(1):87–113, 2006. doi:10.1017/S0960129505005074.

53 Daniele Varacca and Nobuko Yoshida. Probabilistic π-calculus and event structures. *Electronic Notes in Theoretical Computer Science*, 190(3):147–166, 2007. doi:10.1016/j.entcs.2007.07.009.

54 Moshe Y. Vardi. Automatic verification of probabilistic concurrent finite-state programs. In *26th Annual Symposium on Foundations of Computer Science, Portland, Oregon, USA, 21-23 October 1985*, pages 327–338. IEEE Computer Society, 1985. doi:10.1109/SFCS.1985.12.

55 Di Wang, Jan Hoffmann, and Thomas W. Reps. PMAF: an algebraic framework for static analysis of probabilistic programs. In *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*, pages 513–528, 2018. doi:10.1145/3296979.3192408.

56 Lotfi A. Zadeh. Fuzzy sets. *Inf. Control.*, 8(3):338–353, 1965. doi:10.1016/S0019-9958(65)90241-X.

A Supplement to Section 3

▶ **Example A.1.** Consider the type T in Example 3.4. The transition matrix $P = [p_{ij}]$ of its associated DTMC is shown below:

$$P = \left[\begin{array}{cc|cccc} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 & 0 & 0 \\ \frac{1}{4} & 0 & 0 & 0 & \frac{3}{4} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & \frac{2}{3} & \frac{1}{3} & 0 & 0 & 0 \end{array}\right] \quad \textit{where} \quad \begin{aligned} S_0 &= \bullet \\ S_1 &= \circ \\ S_2 &= T \\ S_3 &= \bullet \,{}_{\frac{1}{4}}\& \; ?\mathsf{int}.(\circ \,{}_{\frac{2}{3}}\oplus T) \\ S_4 &= ?\mathsf{int}.(\circ \,{}_{\frac{2}{3}}\oplus T) \\ S_5 &= \circ \,{}_{\frac{2}{3}}\oplus T \end{aligned}$$

Note that we have given P in its *canonical form* [33], in which we have partitioned P in four submatrices with the names and meaning described below in clockwise order, starting from the top-left corner of P:

- S is the 2-by-2 identity matrix giving the probability transitions among the absorbing states. By definition of absorbing state, this is an identity matrix.
- O is the 2-by-4 matrix giving the probability transitions from the absorbing states to the transient states. By definition, these probabilities are all zeros.
- Q is the 4-by-4 matrix giving the probability transitions among the transient states.
- R is the 4-by-2 matrix giving the probability transitions from the transient states to the absorbing states.

Now, the probability of S_2 being absorbed by S_1, *i.e.*, $[\![T]\!]$, can be obtained from the matrix $B = [b_{ij}]$ which is computed as follows:

$$
\begin{aligned}
B &= (I - Q)^{-1} R \\
&= \begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 1 & -\frac{3}{4} & 0 \\ 0 & 0 & 1 & -1 \\ -\frac{1}{3} & 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} 0 & 0 \\ \frac{1}{4} & 0 \\ 0 & 0 \\ 0 & \frac{2}{3} \end{bmatrix} = \begin{bmatrix} \frac{4}{3} & \frac{4}{3} & 1 & 1 \\ \frac{1}{3} & \frac{4}{3} & 1 & 1 \\ \frac{4}{9} & \frac{4}{9} & \frac{4}{3} & \frac{4}{3} \\ \frac{4}{9} & \frac{4}{9} & \frac{1}{3} & \frac{4}{3} \end{bmatrix} \begin{bmatrix} 0 & 0 \\ \frac{1}{4} & 0 \\ 0 & 0 \\ 0 & \frac{2}{3} \end{bmatrix} = \begin{bmatrix} \frac{1}{3} & \frac{2}{3} \\ \frac{1}{3} & \frac{2}{3} \\ \frac{1}{9} & \frac{8}{9} \\ \frac{1}{9} & \frac{8}{9} \end{bmatrix}
\end{aligned}
$$

Then, the probability of absorption for $S_2 = T$ is b_{00}. Hence, $[\![T]\!] = \frac{1}{3}$. ■

▶ **Theorem A.2** ([33]). *Let P be the transition matrix of an absorbing DTMC and B^* be the matrix of the absorption probabilities. Then, $PB^* = B^*$.*

Note that the column l of B^*, *i.e.*, $[b_{il}]$ contains the probabilities of s_i being absorbed by s_l. Consequently, $b_{ll} = 1$ and $b_{il} = 0$ for all absorbing states $s_i \neq s_l$. Also, the probability b_{il} for non-absorbing states s_i can be obtained by solving the system of linear equations corresponding to l-column of B^* in the equality $B^* = PB^*$, *i.e.*,

$$
\begin{array}{ll}
b_{ll} = 1 & \\
b_{ii} = 0 & \text{for all absorbing states } s_i \neq s_l \\
b_{il} = \sum_h p_{ih} \times b_{hl} & \text{for all non-absorbing states } s_i
\end{array}
$$

When considering the DTMCs associated with session types there are exactly two absorbing states, namely $\bullet$ and $\circ$. Moreover, we are interested in computing the column in B^* associated with $\bullet$. If we write $[\![S_i]\!]$ in place of b_il when $S_l = \bullet$, then the set of linear equations is

$$
\begin{array}{ll}
[\![\bullet]\!] = 1 & \\
[\![\circ]\!] = 0 & \\
[\![S_i]\!] = \sum_h p_{ih} \times [\![S_h]\!] & \text{for all } S_i \notin \{\circ, \bullet\}
\end{array}
$$

▶ **Example A.3.** The system of equations for the DTMC in Example A.1 is

$$
\begin{aligned}
[\![\bullet]\!] &= 1 \\
[\![\circ]\!] &= 0 \\
[\![T]\!] &= [\![S_3]\!] \\
[\![S_3]\!] &= \tfrac{1}{4}[\![\bullet]\!] + \tfrac{3}{4}[\![S_4]\!] \\
[\![S_4]\!] &= [\![S_5]\!] \\
[\![S_5]\!] &= \tfrac{2}{3}[\![\circ]\!] + \tfrac{1}{3}[\![T]\!]
\end{aligned}
$$

Note in particular that the system of equations corresponds exactly to the one derived from Definition 3.3 and its solution is $[\![T]\!] = \frac{1}{3}$, $[\![S_3]\!] = \frac{1}{3}$, $[\![S_5]\!] = \frac{1}{9}$, $[\![S_5]\!] = \frac{1}{9}$. ■

We conclude this section with the proof of Proposition 3.6.

▶ **Proposition 3.6.** $[\![T_1 \,{}_p\!\boxplus T_2]\!] = p[\![T_1]\!] + (1 - p)[\![T_2]\!]$.

Proof. The only interesting case is when $T_1 = T\ {}_q\!\oplus S$ and $T_2 = T\ {}_r\!\oplus S$. We have

$$\begin{array}{ll}
[\![T_1\ {}_p\!\boxplus T_2]\!] & \\
\quad = [\![(T\ {}_q\!\oplus S)\ {}_p\!\boxplus (T\ {}_r\!\oplus S)]\!] & \text{by definition of } T_1 \text{ and } T_2 \\
\quad = [\![T\ {}_{pq+(1-p)r}\!\oplus S]\!] & \text{by definition of } {}_p\!\boxplus \\
\quad = (pq + (1-p)r)[\![T]\!] + (1 - pq - (1-p)r)[\![S]\!] & \text{by definition of } [\![\cdot]\!] \\
\quad = pq[\![T]\!] + r[\![T]\!] - pr[\![T]\!] + [\![S]\!] - pq[\![S]\!] - r[\![S]\!] + pr[\![S]\!] &
\end{array}$$

$$\begin{array}{ll}
p[\![T_1]\!] + (1-p)[\![T_2]\!] & \\
\quad = p[\![T\ {}_q\!\oplus S]\!] + (1-p)[\![T\ {}_r\!\oplus S]\!] & \text{by definition of } T_1 \text{ and } T_2 \\
\quad = p(q[\![T]\!] + (1-q)[\![S]\!]) + (1-p)(r[\![T]\!] + (1-r)[\![S]\!]) & \text{by definition of } [\![\cdot]\!] \\
\quad = \ pq[\![T]\!] + p[\![S]\!] - pq[\![S]\!] + r[\![T]\!] + [\![S]\!] - r[\![S]\!] - pr[\![T]\!] - p[\![S]\!] + pr[\![S]\!] & \\
\quad = \ pq[\![T]\!] + r[\![T]\!] - pr[\![T]\!] + [\![S]\!] - pq[\![S]\!] - r[\![S]\!] + pr[\![S]\!] &
\end{array}$$

which confirms the statement. ◀

B Examples

B.1 Typing of Example 4.11

1. The derivation below shows that $\mathtt{case}\,x\,[\mathtt{inr}\,y.\mathtt{done}\,x, \mathtt{inl}\,y.\mathtt{done}\,y]$ is well typed in the context $x : \bullet\ {}_p\!\& \circ, y : \bullet\ {}_{1-p}\!\oplus \circ$.

$$\dfrac{\dfrac{\dfrac{}{x : \bullet, y : \circ \vdash \mathtt{done}\,x}\ \textsc{T-Done}}{x : \bullet, y : \bullet\ {}_0\!\oplus \circ \vdash \mathtt{inr}\,y.\mathtt{done}\,x}\ \textsc{T-Right} \qquad \dfrac{\dfrac{}{x : \circ, y : \bullet \vdash \mathtt{done}\,y}\ \textsc{T-Done}}{x : \circ, y : \bullet\ {}_1\!\oplus \circ \vdash \mathtt{inl}\,y.\mathtt{done}\,y}\ \textsc{T-Left}}{x : \bullet\ {}_p\!\& \circ, y : \bullet\ {}_{1-p}\!\oplus \circ \vdash \mathtt{case}\,x\,[\mathtt{inr}\,y.\mathtt{done}\,x, \mathtt{inl}\,y.\mathtt{done}\,y]}\ \textsc{T-Branch}$$

2. The following derivation shows that $\mathtt{case}\,x\,[\mathtt{case}\,y\,[\mathtt{inl}\,z.\mathtt{done}\,z, \mathtt{inr}\,z], \mathtt{case}\,y\,[\mathtt{inr}\,z, \mathtt{inr}\,z]]$ is well typed in the context $x : \circ\ {}_p\!\& \circ, y : \circ\ {}_q\!\& \circ, z : \bullet\ {}_{pq}\!\oplus \circ$.

$$\dfrac{\dfrac{\dfrac{\dfrac{}{x : \circ, y : \circ, z : \bullet \vdash \mathtt{done}\,z}\ \textsc{T-Done}}{x : \circ, y : \circ, z : \bullet\ {}_1\!\oplus \circ \vdash \mathtt{inl}\,z.\mathtt{done}\,z}\ \textsc{T-Left} \qquad \dfrac{\dfrac{}{x : \circ, y : \circ, z : \circ \vdash \mathtt{idle}}\ \textsc{T-Idle}}{x : \circ, y : \circ, z : \bullet\ {}_0\!\oplus \circ \vdash \mathtt{inr}\,z}\ \textsc{T-Left}}{x : \circ, y : \circ\ {}_q\!\& \circ, z : \bullet\ {}_q\!\oplus \circ \vdash \mathtt{case}\,y\,[\mathtt{inl}\,z.\mathtt{done}\,z, \mathtt{inr}\,z]}\ \textsc{T-Branch} \qquad \vdots}{x : \circ\ {}_p\!\& \circ, y : \circ\ {}_q\!\& \circ, z : \bullet\ {}_{pq}\!\oplus \circ \vdash \mathtt{case}\,x\,[\mathtt{case}\,y\,[\mathtt{inl}\,z.\mathtt{done}\,z, \mathtt{inr}\,z], \mathtt{case}\,y\,[\mathtt{inr}\,z, \mathtt{inr}\,z]]}\ \textsc{T-Branch}$$

3. We illustrate below that $\mathtt{inl}\,x.\mathtt{inl}\,x.\mathtt{done}\,x\ {}_{\frac{1}{2}}\!\boxplus \mathtt{inr}\,x.\mathtt{inr}\,x$ cannot be typed with the context $x : (\bullet\ {}_{\frac{1}{2}}\!\oplus \circ)\ {}_{\frac{1}{2}}\!\oplus (\circ\ {}_{\frac{1}{2}}\!\oplus \circ)$.

$$\dfrac{\dfrac{\dfrac{}{x : \bullet\ {}_{\frac{1}{2}}\!\oplus \circ \vdash \mathtt{inl}\,x.\mathtt{done}\,x}\ \skull}{x : (\bullet\ {}_{\frac{1}{2}}\!\oplus \circ)\ {}_1\!\oplus \circ \vdash \mathtt{inl}\,x.\mathtt{inl}\,x.\mathtt{done}\,x}\ \textsc{T-Left} \qquad \dfrac{\dfrac{}{x : \circ \vdash \mathtt{idle}}}{x : (\bullet\ {}_{\frac{1}{2}}\!\oplus \circ)\ {}_0\!\oplus \circ \vdash \mathtt{inr}\,x}\ \textsc{T-Right}}{x : (\bullet\ {}_{\frac{1}{2}}\!\oplus \circ)\ {}_{\frac{1}{2}}\!\oplus \circ \vdash (\mathtt{inl}\,x.\mathtt{inl}\,x.\mathtt{done}\,x)\ {}_{\frac{1}{2}}\!\boxplus \mathtt{inr}\,x}\ \textsc{T-Choice}$$

B.2 Typing of Example 4.12

We first show that the defining equation for the process variable C is well typed, *i.e.*, that the judgement $x : !\mathsf{int}.(\bullet\ {}_r\!\& \circ) \vdash x!\mathit{job}.\mathtt{case}\,x\,[\mathtt{done}\,x, \mathtt{idle}]$ holds (when assuming job is of type int).

$$\dfrac{\dfrac{\dfrac{}{x : \bullet \vdash \mathtt{done}\,x}\,\text{T-DONE} \qquad \dfrac{}{x : \circ \vdash \mathtt{idle}}\,\text{T-IDLE}}{x : \bullet \;{}_r\!\& \circ \vdash \mathtt{case}\,x\,[\mathtt{done}\,x, \mathtt{idle}]}\,\text{T-BRANCH} \qquad \mathsf{safe}(\mathsf{int})}{x : !\mathsf{int}.(\bullet\;{}_r\!\& \circ) \vdash x!\mathit{job}.\mathtt{case}\,x\,[\mathtt{done}\,x, \mathtt{idle}]}\,\text{T-OUT}$$

We now consider the defining equation for the process variable B. For presentation purposes we consider first the derivations for three different subterms corresponding to the alternative choices in the definition. In particular,

- $x : \bullet\;{}_1\!\oplus \circ, y : \circ\;{}_1\!\oplus\, !S.!\mathsf{int}.\overline{T}, \mathit{job} : \mathsf{int} \vdash \mathtt{inl}\,x.\mathtt{inl}\,y.\mathtt{done}\,x$ Equation (B.1);
- $x : \bullet\;{}_0\!\oplus \circ, y : \circ\;{}_1\!\oplus\, !S.!\mathsf{int}.\overline{T}, \mathit{job} : \mathsf{int} \vdash \mathtt{inr}\,x.\mathtt{inl}\,y$ (B.2);
- $x : S, y : \circ\;{}_0\!\oplus\, !S.!\mathsf{int}.\overline{T}, \mathit{job} : \mathsf{int} \vdash \mathtt{inr}\,y.y!x.y!\mathit{job}.A\langle y\rangle$ (B.3).

$$\dfrac{\dfrac{\dfrac{}{x : \bullet, y : \circ, \mathit{job} : \mathsf{int} \vdash \mathtt{done}\,x}\,\text{T-DONE}}{x : \bullet, y : \circ\;{}_1\!\oplus\, !S.!\mathsf{int}.\overline{T}, \mathit{job} : \mathsf{int} \vdash \mathtt{inl}\,y.\mathtt{done}\,x}\,\text{T-LEFT}}{x : \bullet\;{}_1\!\oplus \circ, y : \circ\;{}_1\!\oplus\, !S.!\mathsf{int}.\overline{T}, \mathit{job} : \mathsf{int} \vdash \mathtt{inl}\,x.\mathtt{inl}\,y.\mathtt{done}\,x}\,\text{T-LEFT} \tag{B.1}$$

$$\dfrac{\dfrac{\dfrac{}{x : \circ, y : \circ, \mathit{job} : \mathsf{int} \vdash \mathtt{idle}}\,\text{T-IDLE}}{x : \circ, y : \circ\;{}_1\!\oplus\, !S.!\mathsf{int}.\overline{T}, \mathit{job} : \mathsf{int} \vdash \mathtt{inl}\,y}\,\text{T-LEFT}}{x : \bullet\;{}_0\!\oplus \circ, y : \circ\;{}_1\!\oplus\, !S.!\mathsf{int}.\overline{T}, \mathit{job} : \mathsf{int} \vdash \mathtt{inr}\,x.\mathtt{inl}\,y}\,\text{T-RIGHT} \tag{B.2}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{A : \overline{T} \qquad \mathsf{safe}(\overline{T})}{y : \overline{T} \vdash A\langle y\rangle}\,\text{T-VAR} \qquad \mathsf{safe}(\mathsf{int})}{y : !\mathsf{int}.\overline{T}, \mathit{job} : \mathsf{int} \vdash y!\mathit{job}.A\langle y\rangle}\,\text{T-OUT} \qquad \mathsf{safe}(S)}{x : S, y : !S.!\mathsf{int}.\overline{T}, \mathit{job} : \mathsf{int} \vdash y!x.y!\mathit{job}.A\langle y\rangle}\,\text{T-OUT}}{x : S, y : \circ\;{}_0\!\oplus\, !S.!\mathsf{int}.\overline{T}, \mathit{job} : \mathsf{int} \vdash \mathtt{inr}\,y.y!x.y!\mathit{job}.A\langle y\rangle}\,\text{T-RIGHT} \tag{B.3}$$

Then, the derivation for the right-most probabilistic choice in the definition of B is obtained from Equation (B.2) and Equation (B.3) as follows.

$$\dfrac{\dfrac{\vdots\;\;(\text{B.2})}{x : \bullet\;{}_0\!\oplus \circ, y : \circ\;{}_1\!\oplus\, !S.!\mathsf{int}.\overline{T}, \mathit{job} : \mathsf{int} \vdash \ldots} \qquad \dfrac{\vdots\;\;(\text{B.3})}{x : S, y : \circ\;{}_0\!\oplus\, !S.!\mathsf{int}.\overline{T}, \mathit{job} : \mathsf{int} \vdash \ldots}}{x : \bullet\;{}_{(1-q)r}\!\oplus \circ, y : \circ\;{}_q\!\oplus\, !S.!\mathsf{int}.\overline{T}, \mathit{job} : \mathsf{int} \vdash \mathtt{inr}\,x.\mathtt{inl}\,y\;{}_q\!\boxplus\,\mathtt{inr}\,y.y!x.y!\mathit{job}.A\langle y\rangle}\,\text{T-CHOICE} \tag{B.4}$$

The derivation for the definition of B is obtained as follows.

$$\dfrac{\dfrac{\dfrac{\vdots\;\;(\text{B.1})}{x : \bullet\;{}_1\!\oplus \circ, y : \circ\;{}_1\!\oplus\, !S.!\mathsf{int}.\overline{T}, \mathit{job} : \mathsf{int} \vdash \ldots} \qquad \dfrac{\vdots\;\;(\text{B.3})}{x : \bullet\;{}_{(1-q)r}\!\oplus \circ, y : \circ\;{}_q\!\oplus\, !S.!\mathsf{int}.\overline{T}, \mathit{job} : \mathsf{int} \vdash \ldots}}{x : \bullet\;{}_{p+(1-q)(1-q)r}\!\oplus \circ, y : \circ\;{}_{p+(1-p)q}\!\oplus\, !S.!\mathsf{int}.\overline{T}, \mathit{job} : \mathsf{int} \vdash \ldots\;{}_p\!\boxplus \ldots}\,\text{T-CHOICE}}{x : \bullet\;{}_{p+(1-q)(1-q)r}\!\oplus \circ, y : !\mathsf{unit}.(\circ\;{}_{p+(1-p)q}\!\oplus\, !S.!\mathsf{int}.\overline{T}), \mathit{job} : \mathsf{int} \vdash y!\langle\rangle.\ldots\;{}_p\!\boxplus \ldots}\,\text{T-OUT}$$

(B.5)

The proof is completed by noting that $p + (1-p)(1-q)\frac{p}{p-pq+q} = \frac{p}{p-pq+q} = r$, and $p + (1-p)q = p - pq + q$.

We show that the definition of A is well typed with the derivation below.

$$\dfrac{\dfrac{}{y : \circ \vdash \texttt{idle}}\,\text{T-IDLE} \qquad \dfrac{\dfrac{\dfrac{B : S, T, \mathsf{int} \quad \mathsf{safe}(S, T, \mathsf{int})}{x : S, y : T, z : \mathsf{int} \vdash B\langle x, y, z\rangle}\,\text{T-VAR}}{x : S, y : ?\mathsf{int}.T \vdash y?(z).B\langle x, y, z\rangle}\,\text{T-IN}}{y : ?S.?\mathsf{int}.T \vdash y?(x).y?(z).B\langle x, y, z\rangle}\,\text{T-IN}}{\dfrac{y : \circ\ {}_{p-pq+q}\!\&\ ?S.?\mathsf{int}.T \vdash \texttt{case}\, y\, [\texttt{idle}, y?(x).y?(z).B\langle x, y, z\rangle]}{y : ?\mathsf{unit}.(\circ\ {}_{p-pq+q}\!\&\ ?S.?\mathsf{int}.T) \vdash y?().\texttt{case}\, y\, [\texttt{idle}, y?(x).y?(z).B\langle x, y, z\rangle]}\,\text{T-IN}}\,\text{T-BRANCH}$$

The typing for the composition $C\langle x\rangle \mid x?(z).B\langle x, y, z\rangle \mid A\langle y\rangle$ is obtained as follows.

$$\dfrac{\dfrac{C : !\mathsf{int}.\overline{S} \quad \mathsf{safe}(!\mathsf{int}.\overline{S})}{x : !\mathsf{int}.\overline{S} \vdash C\langle x\rangle}\,\text{T-VAR} \qquad \dfrac{\dfrac{\dfrac{B : S, T, \mathsf{int} \quad \mathsf{safe}(S, T, \mathsf{int})}{x : S, y : T, z : \mathsf{int} \vdash B\langle x, y, z\rangle}\,\text{T-VAR}}{x : ?\mathsf{int}.S, y : T \vdash x?(z).B\langle x, y, z\rangle}\,\text{T-IN} \qquad \dfrac{A : \overline{T} \quad \mathsf{safe}(\overline{T})}{y : \overline{T} \vdash A\langle y\rangle}\,\text{T-VAR}}{x : ?\mathsf{int}.S, y : \langle\llbracket T\rrbracket\rangle \vdash x?(z).B\langle x, y, z\rangle \mid A\langle y\rangle}\,\text{T-PAR}}{x : \langle\llbracket ?\mathsf{int}.S\rrbracket\rangle, y : \langle\llbracket T\rrbracket\rangle \vdash C\langle x\rangle \mid x?(z).B\langle x, y, z\rangle \mid A\langle y\rangle}\,\text{T-PAR}$$

Finally, we compute the success probabilities:

- $\llbracket ?\mathsf{int}.S\rrbracket = \llbracket S\rrbracket = r\llbracket \bullet\rrbracket + (1-r)\llbracket \circ\rrbracket = r$, and
- $\llbracket T\rrbracket = 0$ since T cannot reach $\bullet$. The complete computation is as follows.

$$\begin{aligned}
\llbracket T\rrbracket &= \llbracket \circ\ {}_{p-pq+q}\!\oplus\ !S.!\mathsf{int}.\overline{T}\rrbracket \\
&= (p - pq + q)\llbracket \circ\rrbracket + r\llbracket !S.!\mathsf{int}.\overline{T}\rrbracket && \text{where } r = (1 - (p - pq + q)) \\
&= r\llbracket !S.!\mathsf{int}.\overline{T}\rrbracket && \text{by } \llbracket \circ\rrbracket = 0 \\
&= r\llbracket !\mathsf{int}.\overline{T}\rrbracket \\
&= r\llbracket \overline{T}\rrbracket \\
&= r\llbracket \circ\ {}_{p-pq+q}\!\&\ ?S.?\mathsf{int}.T\rrbracket \\
&= r(p - pq + q)\llbracket \circ\rrbracket + r^2\llbracket ?S.?\mathsf{int}.T\rrbracket \\
&= r^2\llbracket ?S.?\mathsf{int}.T\rrbracket && \text{by } \llbracket \circ\rrbracket = 0 \\
&= r^2\llbracket ?\mathsf{int}.T\rrbracket \\
&= r^2\llbracket T\rrbracket
\end{aligned}$$

whose unique solution is $\llbracket T\rrbracket = 0$ (for $0 < p, q < 1$).

C Proof of Theorem 4.3

▶ **Lemma C.1.** *If $t\ {}_1\!\boxplus\ s$ is defined, then $t\ {}_1\!\boxplus\ s = t$.*

Proof. The only interesting case is when $t \neq s$ and this can happen in two cases only. If $t = \langle p\rangle$ and $s = \langle q\rangle$, then we conclude $t\ {}_1\!\boxplus\ s = \langle p\rangle$. If $t = T\ {}_p\!\oplus\ S$ and $s = T\ {}_q\!\oplus\ S$, then we conclude $t\ {}_1\!\boxplus\ s = T\ {}_p\!\oplus\ S$. ◀

The next result shows that, if the very same process can be typed in two different contexts, then the success probabilities of the session types in the two contexts is the same. In general it is not true that the session types themselves are the same, because T-LEFT and T-RIGHT allow selections to be typed differently as far as the non-selected branch is concerned. Let $\simeq$ be the smallest equivalence relation on types such that $T \simeq S$ if $\llbracket T \rrbracket = \llbracket S \rrbracket$. We write $\Gamma \simeq \Delta$ if $\Gamma(x) \simeq \Delta(x)$ for every $x \in \mathsf{dom}(\Gamma) \cap \mathsf{dom}(\Delta)$.

▶ **Lemma C.2.** *If $\Gamma_i \vdash P$ for $i = 1, 2$ and $\mathsf{dom}(\Gamma_1) = \mathsf{dom}(\Gamma_2)$, then $\Gamma_1 \simeq \Gamma_2$.*

Proof. By induction on the structure of P and by cases on its shape. We only discuss a few representative cases, the others being similar or simpler.

$\boxed{P = \texttt{idle}}$ Then $\mathsf{un}(\Gamma_i)$ for $i = 1, 2$ and we conclude $\Gamma_1 \simeq \Gamma_2$ by observing that types of the form $\langle p \rangle$ are not unrestricted and that the only unrestricted session type is $\circ$.

$\boxed{P = \texttt{done}\, x}$ Then there exist Γ'_1 and Γ'_2 such that $\Gamma_i = \Gamma'_i, x : \bullet$ for $i = 1, 2$. We conclude $\Gamma_1 \simeq \Gamma_2$ by the same observations made in the previous case.

$\boxed{P = A\langle \overline{x} \rangle}$ Then there exist Γ'_1 and Γ'_2 such that $\Gamma_i = \Gamma'_i, \overline{x : t}$ and $\mathsf{un}(\Gamma'_i)$ for $i = 1, 2$ and $A : \overline{t}$. We conclude $\Gamma_1 \simeq \Gamma_2$ by the same observations made in the previous cases.

$\boxed{P = x?(y).Q}$ Then there exist Γ'_1, Γ'_2, t_1, t_2, T_1 and T_2 such that $\Gamma_i = \Gamma'_i, x : ?t_i.T_i$ and $\Gamma'_i, x : T_i, y : t_i \vdash Q$ for $i = 1, 2$. Using the induction hypothesis we deduce $\Gamma'_1 \simeq \Gamma'_2$ and $t_1 \simeq t_2$ and $T_1 \simeq T_2$. We conclude $\Gamma_1 \simeq \Gamma_2$ since $\llbracket ?t_1.T_1 \rrbracket = \llbracket T_1 \rrbracket = \llbracket T_2 \rrbracket = \llbracket ?t_2.T_2 \rrbracket$.

$\boxed{P = P_1 \;{}_p\boxplus\; P_2}$ Then there exist Γ_{ij} for $1 \leq i, j \leq 2$ such that $\Gamma_i = \Gamma_{i1} \;{}_p\boxplus\; \Gamma_{i2}$ and $\Gamma_{ij} \vdash P_j$ for $1 \leq i, j \leq 2$. Using the induction hypothesis we deduce $\Gamma_{1j} \simeq \Gamma_{2j}$ for all $j = 1, 2$. We conclude $\Gamma_1 = \Gamma_{11} \;{}_p\boxplus\; \Gamma_{12} \simeq \Gamma_{21} \;{}_p\boxplus\; \Gamma_{22} = \Gamma_2$.

$\boxed{P = \texttt{inl}\, x.Q}$ Then there exist Γ'_1, Γ'_2, T_1, T_2, S_1 and S_2 such that $\Gamma_i = \Gamma'_i, x : T_i \;{}_1\oplus\; S_i$ and $\Gamma'_i, x : T_i \vdash Q$ for $i = 1, 2$. Using the induction hypothesis we deduce $\Gamma'_1 \simeq \Gamma'_2$ and $T_1 \simeq T_2$. We conclude $\Gamma_1 \simeq \Gamma_2$ by observing that $\llbracket T_1 \;{}_1\oplus\; S_1 \rrbracket = \llbracket T_1 \rrbracket = \llbracket T_2 \rrbracket = \llbracket T_2 \;{}_1\oplus\; S_2 \rrbracket$.

$\boxed{P = P_1 \mid P_2}$ From T-PAR we deduce that there exist Γ_{11}, Γ_{12}, Γ_{21}, Γ_{22}, T_1 and T_2 such that $\Gamma_i = \Gamma_{i1}, \Gamma_{i2}, x : \langle \llbracket T_i \rrbracket \rangle$ and $\Gamma_{i1}, x : T_i \vdash P_1$ and $\Gamma_{i2}, x : \overline{T_i} \vdash P_2$ for $i = 1, 2$. Using the induction hypothesis we deduce $\Gamma_{1j} \simeq \Gamma_{2j}$ for $j = 1, 2$ and $T_1 \simeq T_2$ namely $\llbracket T_1 \rrbracket = \llbracket T_2 \rrbracket$. We conclude $\Gamma_1 \simeq \Gamma_2$. ◀

The next result shows that a process becoming aware of a probabilistic choice can be typed differently so as to account for the probabilistic information transmitted with the choice. This is the key lemma that allows us to deal with S-PAR-CHOICE. Note that, as the process may be connected with other processes through sessions, the information concerning the probabilistic choice may need to propagate along an arbitrary number of sessions.

▶ **Lemma C.3.** *If $\Gamma, x : \overline{T_1 \;{}_r\boxplus\; T_2} \vdash P$, then there exist Γ_1 and Γ_2 such that $\Gamma_1 \;{}_r\boxplus\; \Gamma_2 = \Gamma$ and $\Gamma_i, x : \overline{T_i} \vdash P$ for every $i = 1, 2$.*

Proof. If $T_1 = T_2$ we conclude immediately by taking $\Gamma_1 = \Gamma_2 = \Gamma$, so from now on we assume $T_1 \neq T_2$ which can happen only when T_1 and T_2 are a choice. We proceed by induction on the derivation of $\Gamma, x : \overline{T_1 \;{}_r\boxplus\; T_2} \vdash P$ and by cases on the last rule applied. We discuss only interesting cases, particularly those compatible with the assumption $T_1 \neq T_2$.

$\boxed{\text{T-VAR}}$ Then $P = A\langle \overline{x} \rangle$. From T-VAR we deduce:

- $\Gamma, \overline{x : t} = \Delta, x : \overline{T_1 \;{}_r\boxplus\; T_2}$;
- $\mathsf{un}(\Delta)$;
- $A : \overline{t}$;

- $\mathsf{safe}(\overline{t})$.

Since T_1 and T_2 are choices, they cannot be unrestricted. Therefore, x must be one of the variables in $\overline{x}$ and $\overline{T_1 \,{}_r\boxplus\, T_2}$ is one of the types in $\overline{t}$. But then $\overline{T_1 \,{}_r\boxplus\, T_2}$ is a branch, which is not a safe type according to Definition 4.2. We conclude that this case is impossible.

$\boxed{\text{T-BRANCH when } x \text{ is the endpoint being used for input}}$ Then $P = \mathtt{case}\, x\, [P_1, P_2]$. From T-BRANCH we deduce that there exist Δ_1, Δ_2, S_1 and S_2 such that:

- $\Delta_1 \,{}_p\boxplus\, \Delta_2 = \Gamma$;
- $\overline{T_1 \,{}_r\boxplus\, T_2} = S_1 \,{}_p\&\, S_2$;
- $\Delta_i, x : S_i \vdash P_i$ for $i = 1, 2$.

From Definition 3.5 we deduce that there exist p_1 and p_2 such that $\overline{T_i} = S_1 \,{}_{p_i}\&\, S_2$ and $p = rp_1 + (1-r)p_2$. Let $\Gamma_i \overset{\mathsf{def}}{=} \Delta_1 \,{}_{p_i}\boxplus\, \Delta_2$ and observe that $(\Delta_1 \,{}_{p_1}\boxplus\, \Delta_2) \,{}_p\boxplus\, (\Delta_1 \,{}_{p_2}\boxplus\, \Delta_2) = \Gamma$. We conclude $\Gamma_i, x : S_1 \,{}_{p_i}\&\, S_2 \vdash \mathtt{case}\, x\, [P_1, P_2]$ with an application of T-BRANCH.

$\boxed{\text{T-PAR}}$ Then $P = Q \mid R$. Since $\overline{T_1 \,{}_r\boxplus\, T_2}$ is a session type and not a type of the form $\langle q \rangle$, x cannot be used by both Q and R. We consider only the case in which x is used by Q, the other case being symmetric. From T-PAR we deduce:

- $\Delta_1, y : S, x : \overline{T_1 \,{}_r\boxplus\, T_2} \vdash Q$;
- $\Delta_2, y : \overline{S} \vdash R$;
- $\Gamma = \Delta_1, \Delta_2, y : \langle [\![S]\!] \rangle$.

Using the induction hypothesis we deduce that there exist Δ_{11}, Δ_{12}, S_1 and S_2 such that $(\Delta_{11}, y : S_1) \,{}_r\boxplus\, (\Delta_{12}, y : S_2) = \Delta_1, y : S$ and $\Delta_{1i}, y : S_i, x : \overline{T_i} \vdash Q$ for $i = 1, 2$. In particular, we have $\overline{S} = \overline{S_1 \,{}_r\boxplus\, S_2}$. Using the induction hypothesis once again, we deduce that there exist Δ_{21} and Δ_{22} such that $\Delta_{21} \,{}_r\boxplus\, \Delta_{22} = \Delta_2$ and $\Delta_{2i}, y : \overline{S_i} \vdash$ for $i = 1, 2$. Let $\Gamma_i \overset{\mathsf{def}}{=} \Delta_{1i}, \Delta_{2i}, y : \langle [\![S_i]\!] \rangle$ and observe that $\Gamma_1 \,{}_r\boxplus\, \Gamma_2 = \Gamma$. We conclude $\Gamma_i, x : \overline{T_i} \vdash P$ for $i = 1, 2$ using T-PAR.

$\boxed{\text{T-CHOICE}}$ Then we have:

- $P = P_1 \,{}_p\boxplus\, P_2$ for some P_1 and P_2;
- $\overline{T_1 \,{}_r\boxplus\, T_2} = S_1 \,{}_p\boxplus\, S_2$ for some S_1, S_2 and p;
- $\Delta_1 \,{}_p\boxplus\, \Delta_2 = \Gamma$ for some Δ_1 and Δ_2;
- $\Delta_k, x : S_k \vdash P_k$ for $k = 1, 2$.

Since T_1 and T_2 are choices, S_1 and S_2 must be branches. Since the combination of branches is only defined when they are exactly the same, we deduce $S_1 = S_2 = \overline{T_1 \,{}_r\boxplus\, T_2}$. Using the induction hypothesis, we deduce that for every $k = 1, 2$ there exist Δ_{k1} and Δ_{k2} such that $\Delta_{k1} \,{}_r\boxplus\, \Delta_{k2} = \Delta_k$ and $\Delta_{ki}, x : \overline{T_i} \vdash P_k$ for $i = 1, 2$. Let $\Gamma_i \overset{\mathsf{def}}{=} \Delta_{1i} \,{}_p\boxplus\, \Delta_{2i}$ for $i = 1, 2$ and observe that $\Gamma_1 \,{}_r\boxplus\, \Gamma_2 = \Gamma$. We conclude $\Gamma_i, x : \overline{T_i} \vdash P$ for $i = 1, 2$ using T-CHOICE. ◀

We now have all the ingredients to show that typing is preserved by structural pre-congruence.

▶ **Lemma C.4.** *If* $\Gamma \vdash P$ *and* $P \preccurlyeq Q$*, then* $\Gamma \vdash Q$.

Proof. By induction on the derivation of $P \preccurlyeq Q$ and by cases on the last rule applied. We only discuss a few selected cases, the others being simpler or trivial.

$\boxed{\text{S-NO-CHOICE}}$ Then we have $P = Q \,{}_1\boxplus\, R$. From T-CHOICE we deduce that there exist Γ_1 and Γ_2 such that $\Gamma = \Gamma_1 \,{}_1\boxplus\, \Gamma_2$ and $\Gamma_1 \vdash Q$ and $\Gamma_2 \vdash R$. Using Lemma C.1 we conclude $\Gamma = \Gamma_1$.

$\boxed{\textsc{s-choice-idem}}$ Then we have $P = Q \;{}_p\!\boxplus\; Q$. From $\textsc{t-choice}$ we deduce that there exist Γ_1 and Γ_2 such that $\Gamma = \Gamma_1 \;{}_p\!\boxplus\; \Gamma_2$ and $\Gamma_i \vdash Q$ for $i = 1, 2$. By Lemma C.2 we deduce $\Gamma_1 \simeq \Gamma_2$. It is a simple exercise to show that $\Gamma = \Gamma_1 \;{}_p\!\boxplus\; \Gamma_2$ and $\Gamma_1 \simeq \Gamma_2$ imply $\Gamma = \Gamma_1 = \Gamma_2$, which suffices to conclude.

$\boxed{\textsc{s-par-choice}}$ Then we have:

- $P = (P_1 \;{}_p\!\boxplus\; P_2) \mid R$;
- $Q = (P_1 \mid R) \;{}_p\!\boxplus\; (P_2 \mid R)$.

From $\textsc{t-par}$ and $\textsc{t-choice}$ we deduce:

- $\Gamma = (\Gamma_1 \;{}_p\!\boxplus\; \Gamma_2), \Delta, x : \langle\llbracket T_1 \;{}_p\!\boxplus\; T_2\rrbracket\rangle$;
- $\Gamma_i, x : T_i \vdash P_i$ for $i = 1, 2$;
- $\Delta, x : \overline{T_1 \;{}_p\!\boxplus\; T_2} \vdash R$.

Using Lemma C.3 we deduce that there exist Δ_1 and Δ_2 such that $\Delta_1 \;{}_p\!\boxplus\; \Delta_2 = \Delta$ and $\Delta_i, x : \overline{T_i} \vdash R$ for every $i = 1, 2$. We derive $\Gamma_i, \Delta_i, x : \langle\llbracket T_i\rrbracket\rangle \vdash P_i \mid R$ for $i = 1, 2$ using $\textsc{t-par}$. We conclude $(\Gamma_1 \;{}_p\!\boxplus\; \Gamma_2), (\Delta_1 \;{}_p\!\boxplus\; \Delta_2), x : \langle\llbracket T_1\rrbracket\rangle \;{}_p\!\boxplus\; \langle\llbracket T_2\rrbracket\rangle \vdash Q$ observing that

$$\begin{aligned}
\langle\llbracket T_1\rrbracket\rangle \;{}_p\!\boxplus\; \langle\llbracket T_2\rrbracket\rangle &= \langle p\llbracket T_1\rrbracket + (1-p)\llbracket T_2\rrbracket\rangle && \text{by Definition 3.5} \\
&= \langle\llbracket T_1 \;{}_p\!\boxplus\; T_2\rrbracket\rangle && \text{by Proposition 3.6}
\end{aligned}$$

$\boxed{\textsc{s-par-assoc}}$ Then we have $P = (P_1 \mid P_2) \mid P_3$ and $Q = P_1 \mid (P_2 \mid P_3)$ and $\mathsf{fn}(P_2) \cap \mathsf{fn}(P_3) \neq \emptyset$. From $\textsc{t-par}$ we deduce:

- $\Gamma = \Delta, \Gamma_3, x : \langle\llbracket T\rrbracket\rangle$;
- $\Delta, x : T \vdash P_1 \mid P_2$;
- $\Gamma_3, x : \overline{T} \vdash P_3$.

From $\mathsf{fn}(P_2) \cap \mathsf{fn}(P_3) \neq \emptyset$ and $\mathsf{dom}(\Delta) \cap \Gamma_3 = \emptyset$ we deduce $x \in \mathsf{fn}(P_2)$. Hence, from $\textsc{t-par}$ we deduce:

- $\Delta = \Gamma_1, \Gamma_2, y : \langle\llbracket S\rrbracket\rangle$;
- $\Gamma_1, y : S \vdash P_1$;
- $\Gamma_2, x : T, y : \overline{S} \vdash P_2$.

We derive $\Gamma_2, \Gamma_3, x : T, y : \langle\llbracket S\rrbracket\rangle \vdash P_2 \mid P_3$ with one application of $\textsc{t-par}$ and we conclude $\Gamma_1, \Gamma_2, \Gamma_3, x : \langle\llbracket T\rrbracket\rangle, y : \langle\llbracket S\rrbracket\rangle \vdash P_1 \mid (P_2 \mid P_3)$ with another application of $\textsc{t-par}$. ◀

▶ **Theorem 4.3** (subject reduction). *If $\Gamma \vdash P$ and $P \to Q$, then $\Gamma \vdash Q$.*

Proof. By induction on the derivation of $P \to Q$ and by cases on the last rule applied. Since typing is syntax directed, in each case we can use the typing rule corresponding to the shape of the term under consideration.

$\boxed{\textsc{r-com}}$ Then there exist x, y, P_1 and P_2 such that:

- $P = x!y.P_1 \mid x?(y).P_2$;
- $Q = P_1 \mid P_2$.

From $\textsc{t-par}$, $\textsc{t-out}$ and $\textsc{t-in}$ we deduce that there exist Γ_1, Γ_2, t and T such that:

- $\Gamma = \Gamma_1, \Gamma_2, x : \langle\llbracket !t.T\rrbracket\rangle, y : t$;
- $\Gamma_1, x : T \vdash P_1$;
- $\Gamma_2, x : \overline{T}, y : t \vdash P_2$.

We conclude $\Gamma \vdash P_1 \mid P_2$ with one application of $\textsc{t-par}$ and observing that $\llbracket !t.T\rrbracket = \llbracket T\rrbracket$ by Definition 3.3.

$\boxed{\textsc{r-left}}$ Then there exist x, P_1, Q_1 and Q_2 such that:

- $P = \texttt{inl}\, x.P_1 \mid \texttt{case}\, x\, [Q_1, Q_2]$;
- $Q = P_1 \mid Q_1$.

From T-PAR, T-LEFT and T-BRANCH we deduce that there exist Γ_1, Δ, T and S such that:

- $\Gamma = \Gamma_1, \Delta, x : \langle[\![T \;{}_1\!\oplus S]\!]\rangle$;
- $\Gamma_1, x : T \vdash P_1$;
- $\Delta, x : \overline{T} \vdash Q_1$.

We conclude $\Gamma \vdash Q$ with one application of T-PAR and observing that $[\![T \;{}_1\!\oplus S]\!] = [\![T]\!]$ by Definition 3.3.

$\boxed{\text{R-PAR}}$ Then there exist P_1, P_1' and P_2 such that:

- $P = P_1 \mid P_2$ for some P_1 and P_2;
- $P_1 \to P_1'$;
- $Q = P_1' \mid P_2$.

From T-PAR we deduce that there exist Γ_1, Γ_2, x and T such that:

- $\Gamma = \Gamma_1, \Gamma_2, x : \langle[\![T]\!]\rangle$;
- $\Gamma_1, x : T \vdash P_1$;
- $\Gamma_2, x : \overline{T} \vdash P_2$.

Using the induction hypothesis we deduce $\Gamma_1, x : T \vdash P_1'$. We conclude $\Gamma \vdash Q$ with one application of T-PAR.

$\boxed{\text{R-NEW}}$ Then there exist x, R and R' such that:

- $P = (x)R$;
- $R \to R'$;
- $Q = (x)R'$.

From T-NEW we deduce that there exist Δ and p such that:

- $\Gamma = \Delta, x : \langle p\rangle$;
- $\Delta, x : \langle p\rangle \vdash R$.

Using the induction hypothesis we deduce that $\Delta, x : \langle p\rangle \vdash R'$ and we conclude $\Gamma \vdash Q$ with one application of T-NEW.

$\boxed{\text{R-CHOICE}}$ Then there exist P_1, P_1', P_2 and p such that:

- $P = P_1 \;{}_p\!\boxplus P_2$;
- $P_1 \to P_1'$;
- $Q = P_1' \;{}_p\!\boxplus P_2$.

From T-CHOICE we deduce that there exist Γ_1 and Γ_2 such that:

- $\Gamma = \Gamma_1 \;{}_p\!\boxplus \Gamma_2$;
- $\Gamma_i \vdash P_i$ for all $i = 1, 2$.

Using the induction hypothesis we deduce $\Gamma_1 \vdash P_1'$ and we conclude with one application of T-CHOICE.

$\boxed{\text{R-STRUCT}}$ Then we have $P \preccurlyeq R \to R' \preccurlyeq Q$ for some R and R'. From $\Gamma \vdash P$ and Lemma C.4 we deduce $\Gamma \vdash R$. Using the induction hypothesis we deduce that $\Gamma \vdash R'$. From Lemma C.4 we conclude $\Gamma \vdash Q$. ◀

D Proof of Theorem 4.5

In this appendix we develop the proof that well-typed processes are deadlock free. First of all, we introduce the auxiliary notion of *hyper-context* which will be useful in the proof of Theorem 4.5. An hypercontext $\mathcal{H}$ is a non-empty multiset of contexts written $\Gamma_1 \fatsemi \dots \fatsemi \Gamma_n$. We write $\mathsf{dom}(\mathcal{H})$ for the union of the domains of the contexts in $\mathcal{H}$ and $\mathcal{H} \fatsemi \mathcal{H}'$ for the multiset union of $\mathcal{H}$ and $\mathcal{H}'$.

If we think of a context as of the abstraction of well-typed process, then an hyper-context intuitively represents a parallel composition of such processes and a well-formed hyper-context is one that represents a *well-typed parallel composition* of the same processes. Formally:

▶ **Definition D.1** (well-formed hyper-context). *We say that $\mathcal{H}$ is* well formed *if there exists Γ such that $\mathcal{H} \vdash \Gamma$ is derivable using the following axiom and rule:*

$$\Gamma \vdash \Gamma \qquad \frac{\mathcal{H} \vdash \Gamma, x : T \qquad \mathcal{H}' \vdash \Delta, x : \overline{T}}{\mathcal{H} \fatsemi \mathcal{H}' \vdash \Gamma, \Delta, x : \langle [\![T]\!] \rangle}$$

Note that the rightmost rule establishing the well formedness of an hyper-context corresponds to T-PAR in the typing of processes. A simple induction on the derivation of $\mathcal{H} \vdash \Gamma$ suffices to establish that $\mathsf{dom}(\mathcal{H}) = \mathsf{dom}(\Gamma)$.

We now show that there is a relationship between well-formed hyper-contexts and the absence of cycles in the (contexts of the) processes that are composed in parallel.

▶ **Definition D.2** (acyclic hyper-context). *We say that $\mathcal{H}$ has a* cycle *if there exist n pairwise distinct $x_1, \dots, x_n$ and n pairwise distinct $\Gamma_1, \dots, \Gamma_n \in \mathcal{H}$ with $n \geq 2$ such that $x_i \in \mathsf{dom}(\Gamma_i) \cap \mathsf{dom}(\Gamma_{(i \bmod n)+1})$ for very $1 \leq i \leq n$. We say that $\mathcal{H}$ is* acyclic *if it has no cycle.*

▶ **Proposition D.3.** *If $\mathcal{H}$ is well formed, then it is acyclic.*

Proof. We prove a more general result, namely that $\mathcal{H} \vdash \Gamma$ implies that $\mathcal{H}$ is acyclic. We proceed by induction on the derivation of $\mathcal{H} \vdash \Gamma$. In the base case we have $\mathcal{H} = \Gamma$, hence $\mathcal{H}$ is acyclic because a cycle requires two or more contexts. Suppose $\mathcal{H} = \mathcal{H}_1 \fatsemi \mathcal{H}_2$ and $\mathcal{H}_1 \vdash \Gamma_1, x : T$ and $\mathcal{H}_2 \vdash \Gamma_2, x : \overline{T}$ and $\Gamma = \Gamma_1, \Gamma_2, x : \langle [\![T]\!] \rangle$. By induction hypothesis both $\mathcal{H}_1$ and $\mathcal{H}_2$ are acyclic. Hence, any cycle of $\mathcal{H}$ must involve two distinct names $x_1 \in \mathsf{dom}(\Gamma_1, x : T)$ and $x_2 \in \mathsf{dom}(\Gamma_2, x : \overline{T})$ that connect a context in $\mathcal{H}_1$ and a context in $\mathcal{H}_2$. However, $\mathcal{H}_1$ and $\mathcal{H}_2$ share just the name x because $\mathsf{dom}(\Gamma_1) \cap \mathsf{dom}(\Gamma_2) = \emptyset$. Therefore, $\mathcal{H}$ is acyclic. ◀

The next step towards the proof of deadlock freedom is to prove a *proximity lemma* showing that, whenever two well-typed processes share a name – that is, when they are connected by a session – it is always possible to rearrange them using structural pre-congruence and respecting typing in such a way that they sit next to each other and can possibly reduce. To do so, we introduce some standard notation for process contexts:

▶ **Definition D.4** (process context). *A* process context *is a process containing a finite number of unguarded "holes" [ ]. Formally, it is a term generated by the following grammar:*

$$\mathcal{C}, \mathcal{D} ::= [\,] \mid P \mid \mathcal{C} \mid \mathcal{D} \mid \mathcal{C} \;{}_p\!\boxplus \mathcal{D} \mid (x)\mathcal{C}$$

If $\mathcal{C}$ is a context with n holes numbered from left to right according to the syntax of $\mathcal{C}$, we write $\mathcal{C}[P_1] \cdots [P_n]$ for the process obtained by filling the i-th hole with P_i. Note that filling a hole differs from substitution in that it may capture names, for example if P_i is inserted in the scope of a binder. By writing $\mathcal{C}[P_1] \cdots [P_n]$, we implicitly assume that $\mathcal{C}$ has n holes.

Here is the proximity lemma. The hypothesis $x \in (\mathsf{fn}(P) \setminus \mathsf{bn}(\mathcal{C})) \cap \mathsf{fn}(Q)$ makes sure that the name x showing up in the context $\Gamma, x : T$ is the very same x that occurs free in P.

▶ **Lemma D.5** (proximity lemma). *If* $\Gamma, x : T \vdash \mathcal{C}[P]$ *and* $\Delta, x : \overline{T} \vdash Q$ *and* $x \in (\mathsf{fn}(P) \setminus \mathsf{bn}(\mathcal{C})) \cap \mathsf{fn}(Q)$ *and* $\mathsf{dom}(\Gamma) \cap \mathsf{dom}(\Delta) = \emptyset$*, then there exists* $\mathcal{D}$ *such that* $\mathcal{C}[P] \mid Q \preccurlyeq \mathcal{D}[P \mid Q]$ *and* $\Gamma, \Delta, x : \langle [\![T]\!] \rangle \vdash \mathcal{D}[P \mid Q]$*.*

Proof. By induction on $\mathcal{C}$. We omit symmetric cases.

$\boxed{\mathcal{C} = [\,]}$ We conclude by taking $\mathcal{D} \overset{\text{def}}{=} [\,]$ with one application of T-PAR.

$\boxed{\mathcal{C} = R \mid \mathcal{C}'}$ From T-PAR we deduce $\Gamma = \Gamma_1, \Gamma_2, y : \langle [\![S]\!] \rangle$ and $\Gamma_1, y : S \vdash R$ and $\Gamma_2, y : \overline{S}, x : T \vdash \mathcal{C}'[P]$. Note that $x \neq y$, because the type of x in the context used for typing $\mathcal{C}[P]$ is a session type and not a type of the form $\langle r \rangle$. Using the induction hypothesis we deduce that there exists $\mathcal{D}'$ such that $\mathcal{C}'[P] \mid Q \preccurlyeq \mathcal{D}'[P \mid Q]$ and $\Gamma_2, y : \overline{S}, \Delta, x : \langle [\![T]\!] \rangle \vdash \mathcal{D}'[P \mid Q]$. Let $\mathcal{D} \overset{\text{def}}{=} R \mid \mathcal{D}'$. We derive

$$\begin{aligned}
\mathcal{C}[P] \mid Q &= (R \mid \mathcal{C}'[P]) \mid Q && \text{by definition of } \mathcal{C} \\
&\preccurlyeq R \mid (\mathcal{C}'[P] \mid Q) && \text{by S-PAR-ASSOC using } x \in \mathsf{fn}(\mathcal{C}'[P]) \cap \mathsf{fn}(Q) \\
&\preccurlyeq R \mid \mathcal{D}'[P \mid Q] && \text{by property of } \mathcal{D}' \\
&= \mathcal{D}[P \mid Q] && \text{by definition of } \mathcal{D}
\end{aligned}$$

and we conclude with one application of T-PAR.

$\boxed{\mathcal{C} = R \mathbin{{}_{p}\boxplus} \mathcal{C}'}$ From T-CHOICE we deduce $\Gamma, x : T = (\Gamma_1, x : T_1) \mathbin{{}_{p}\boxplus} (\Gamma_2, x : T_2)$ and $\Gamma_1, x : T_1 \vdash R$ and $\Gamma_2, x : T_2 \vdash \mathcal{C}'[P]$. In particular, $\overline{T} = \overline{T_1 \mathbin{{}_{p}\boxplus} T_2}$. By Lemma C.3 we deduce that there exist Δ_1 and Δ_2 such that $\Delta = \Delta_1 \mathbin{{}_{p}\boxplus} \Delta_2$ and $\Delta_i, x : \overline{T_i} \vdash Q$ for $i = 1, 2$. Using the induction hypothesis we deduce that there exists $\mathcal{D}'$ such that $\mathcal{C}'[P] \mid Q \preccurlyeq \mathcal{D}'[P \mid Q]$ and $\Gamma_2, \Delta_2, x : \langle [\![T_2]\!] \rangle \vdash \mathcal{D}'[P \mid Q]$. Let $\mathcal{D} \overset{\text{def}}{=} (R \mid Q) \mathbin{{}_{p}\boxplus} \mathcal{D}'$. We derive

$$\begin{aligned}
\mathcal{C}[P] \mid Q &= (R \mathbin{{}_{p}\boxplus} \mathcal{C}'[P]) \mid Q && \text{by definition of } \mathcal{C} \\
&\preccurlyeq (R \mid Q) \mathbin{{}_{p}\boxplus} (\mathcal{C}'[P] \mid Q) && \text{by S-PAR-CHOICE} \\
&\preccurlyeq (R \mid Q) \mathbin{{}_{p}\boxplus} \mathcal{D}'[P \mid Q] && \text{by property of } \mathcal{D}' \\
&= \mathcal{D}[P \mid Q] && \text{by definition of } \mathcal{D}
\end{aligned}$$

We derive $\Gamma_1, \Delta_1, x : \langle [\![T_1]\!] \rangle \vdash R \mid Q$ using T-PAR and we conclude with one application of T-CHOICE, observing that $\langle [\![T]\!] \rangle = \langle [\![T_1 \mathbin{{}_{p}\boxplus} T_2]\!] \rangle = \langle [\![T_1]\!] \rangle \mathbin{{}_{p}\boxplus} \langle [\![T_2]\!] \rangle$ by Proposition 3.6.

$\boxed{\mathcal{C} = (y)\mathcal{C}'}$ From T-NEW we deduce $\Gamma, y : \langle [\![S]\!] \rangle, x : T \vdash \mathcal{C}'[P]$. Since y is bound we may assume, without loss of generality, that $y \notin \mathsf{fn}(Q)$. Using the induction hypothesis we deduce that there exists $\mathcal{D}'$ such that $\mathcal{C}'[P] \mid Q \preccurlyeq \mathcal{D}'[P \mid Q]$ and $\Gamma, y : \langle [\![S]\!] \rangle, x : \langle [\![T]\!] \rangle \vdash \mathcal{D}'[P \mid Q]$. Let $\mathcal{D} \overset{\text{def}}{=} (y)\mathcal{D}'$. We derive

$$\begin{aligned}
\mathcal{C}[P] \mid Q &= (y)\mathcal{C}'[P] \mid Q && \text{by definition of } \mathcal{C} \\
&\preccurlyeq (y)(\mathcal{C}'[P] \mid Q) && \text{by S-PAR-NEW} \\
&\preccurlyeq (y)\mathcal{D}'[P \mid Q] && \text{by property of } \mathcal{D}' \\
&= \mathcal{D}[P \mid Q] && \text{by definition of } \mathcal{D}
\end{aligned}$$

and we conclude with one application of T-NEW. ◀

We now show that well-typed processes can be rewritten in a *normal form* in which all the restrictions and probabilistic choices have been "pushed outwards", so that all the parallel compositions concern sequential processes.

▶ **Definition D.6** (prefixed, sequential and exposed process). *A process is* prefixed *if it has the form $x?(y).P$ or $x!y.P$ or $\texttt{inl}\,x.P$ or $\texttt{inr}\,x.P$ or $\texttt{case}\,x\,[P,Q]$. A process is* sequential *if it is either prefixed or it has the form $\texttt{idle}$ or $\texttt{done}\,x$ or $A\langle\overline{x}\rangle$. A process is* exposed *if it is a parallel composition of sequential processes.*

▶ **Definition D.7** (process normal form). *A process is in* normal form *if it is generated by the grammar*

$$P_{nf} \;::=\; P \;\mid\; (x)P_{nf} \;\mid\; P_{nf}\; {}_p\boxplus\, Q_{nf}$$

where P is an exposed process.

▶ **Lemma D.8.** *If P_1 is in normal form and P_2 is exposed and $\Gamma_1, x : T \vdash P_1$ and $\Gamma_2, x : \overline{T} \vdash P_2$ and $\mathsf{dom}(\Gamma_1) \cap \mathsf{dom}(\Gamma_2) = \emptyset$, then there exists P in normal form such that $P_1 \mid P_2 \preccurlyeq P$ and $\Gamma_1, \Gamma_2, x : \langle[\![T]\!]\rangle \vdash P$.*

Proof. A simple induction on the structure of P_1 recalling that it is in normal form. In the base case, when P_1 is exposed, $P_1 \mid P_2$ is already in normal form and the result follows by reflexivity of $\preccurlyeq$ and one application of T-PAR. The inductive cases are analogous to the ones discussed in the proof of Lemma D.5. ◀

▶ **Lemma D.9.** *If P_1 and P_2 are in normal form and $\Gamma_1, x : T \vdash P_1$ and $\Gamma_2, x : \overline{T} \vdash P_2$ and $\mathsf{dom}(\Gamma_1) \cap \mathsf{dom}(\Gamma_2) = \emptyset$, then there exists P in normal form such that $P_1 \mid P_2 \preccurlyeq P$ and $\Gamma_1, \Gamma_2, x : \langle[\![T]\!]\rangle \vdash P$.*

Proof. A simple induction on P_2 recalling that it is in normal form. In the base case, when P_2 is exposed, the result follows from Lemma D.8. ◀

▶ **Lemma D.10** (normal form). *If $\Gamma \vdash P$, then there exists Q in normal form such that $P \preccurlyeq Q$ and $\Gamma \vdash Q$.*

Proof. By induction on P and by cases on its shape.

$\boxed{P \text{ is sequential}}$ Then P is already in normal form and there is nothing left to prove.

$\boxed{P = P_1\; {}_p\boxplus\, P_2}$ From T-CHOICE we deduce $\Gamma = \Gamma_1\; {}_p\boxplus\, \Gamma_2$ and $\Gamma_i \vdash P_i$ for $i = 1, 2$. Using the induction hypothesis we deduce that there exist Q_1 and Q_2 in normal form such that $P_i \preccurlyeq Q_i$ and $\Gamma_i \vdash Q_i$ for $i = 1, 2$. Let $Q \overset{\mathsf{def}}{=} Q_1\; {}_p\boxplus\, Q_2$ and observe that Q is in normal form. Now $P = P_1\; {}_p\boxplus\, P_2 \preccurlyeq Q_1\; {}_p\boxplus\, Q_2 = Q$ and we conclude $\Gamma \vdash Q$ with one application of T-CHOICE.

$\boxed{P = (x)P'}$ From T-NEW we deduce $\Gamma, x : \langle p\rangle \vdash P'$ for some p. Using the induction hypothesis we deduce that there exists Q' in normal form such that $P' \preccurlyeq Q'$ and $\Gamma, x : \langle p\rangle \vdash Q'$. Let $Q \overset{\mathsf{def}}{=} (x)Q'$ and observe that Q is in normal form. Now $P = (x)P' \preccurlyeq (x)Q' = Q$ and we conclude $\Gamma \vdash Q$ with one application of T-NEW.

$\boxed{P = P_1 \mid P_2}$ From T-PAR we deduce $\Gamma = \Gamma_1, \Gamma_2, x : \langle[\![T]\!]\rangle$ and $\Gamma_1, x : T \vdash P_1$ and $\Gamma_2, x : \overline{T} \vdash P_2$. Using the induction hypothesis we deduce that there exist Q_1 and Q_2 in normal form such that $P_i \preccurlyeq Q_i$ for $i = 1, 2$ and $\Gamma_1, x : T \vdash Q_1$ and $\Gamma_2, x : \overline{T} \vdash Q_2$. We conclude using Lemma D.9. ◀

We now have almost all the ingredients for proving Theorem 4.5. The only aspect we have to consider is that the proof will be an induction on the structure of the typing derivation, hence the property that the process is well typed in the empty context is not general enough to apply the induction hypothesis. We generalize Theorem 4.5 by considering processes that are well typed in *balanced* contexts, assuring us that all the session endpoints are used.

▶ **Definition D.11** (balanced type). *We say that t is* balanced *and we write* $\mathsf{bal}(t)$ *if either* $\mathsf{un}(t)$ *or t has the form $\langle p\rangle$ for some p. We write* $\mathsf{bal}(\Gamma)$ *if* $\mathsf{bal}(\Gamma(x))$ *for every* $x \in \mathsf{dom}(\Gamma)$*.*

▶ **Lemma D.12.** *If* $\mathsf{bal}(\Gamma)$ *and* $\Gamma \vdash P$ *and* $P \not\rightarrow$*, then* $P\downarrow$*.*

Proof. Without loss of generality, we may assume that P is an exposed process. Indeed:

- If P is not in normal form, then Lemma D.10 allows us to rewrite P into a normal form process that is well typed in the same Γ.
- If P is in normal form but not exposed, then it consists of top-level session restrictions and process distributions containing exposed processes, each of which is well typed in a balanced context and none of which reduces.

From the hypothesis $P \not\rightarrow$ we deduce that none of the sequential processes in P is a process invocation. Therefore, P is a parallel composition of $P_1, \dots, P_n, Q_1, \dots, Q_m$ where the P_i are prefixed processes and the Q_j are either `idle` or of the form `done` x. From $\Gamma \vdash P$ and T-PAR we deduce that there exist $\Gamma_1, \dots, \Gamma_n, \Delta_1, \dots, \Delta_m$ such that $\Gamma_i \vdash P_i$ for every $1 \le i \le n$ and $\Delta_j \vdash Q_j$ for every $1 \le j \le m$. Also, we let x_i be the channel that occurs in the prefix of P_i. Clearly, $x_i \in \mathsf{dom}(\Gamma_i)$. We proceed by contradiction, assuming that $n \neq 0$. It must be the case that the x_i are pairwise distinct. Indeed, if $x_i = x_j$, then x_i and x_j would be the two peer endpoints of the same session performing complementary actions, by Lemma D.5 we would be able to move the two processes using x_i and x_j next to each other and P would be able to reduce, thus contradicting the hypothesis $P \not\rightarrow$. Also, from the derivation of $\Gamma \vdash P$ we can build a derivation of $\Gamma_1 \fatsemi \dots \fatsemi \Gamma_n \fatsemi \Delta_1 \fatsemi \dots \Delta_m \vdash \Gamma$ according to Definition D.1.

The sub-structural nature of the type system and the hypothesis $\mathsf{bal}(\Gamma)$ ensure that each session name occurs exactly twice. Therefore, each x_i must also occur free in some other P_j with $j \neq i$. We let $f : [1, n] \to [1, n]$ be the function that maps i to the index of the process in which x_i occurs free. That is, $x_i \in \mathsf{fn}(P_{f(i)})$ for every $1 \le i \le n$. Note that $f(i) \neq i$ by definition of f. Now we build the following infinite sequence of names

$$x_1, x_{f(1)}, x_{f(f(1))}, x_{f(f(f(1)))}, \dots$$

Since there are n distinct names x_i and $f(i) \neq i$, there are at least two names that occur infinitely often in this sequence. Consequently, the hyper-context $\Gamma_1 \fatsemi \dots \fatsemi \Gamma_n \fatsemi \Delta_1 \fatsemi \dots \fatsemi \Delta_m$ must have a cycle in the sense of Definition D.2, which contradicts Proposition D.3. ◀

▶ **Theorem 4.5** (deadlock freedom). *If* $\emptyset \vdash P$ *and* $P \Rightarrow Q$*, then either* $Q \rightarrow$ *or* $Q\downarrow$*.*

Proof. Immediate consequence of Theorem 4.3 and Lemma D.12. ◀

E Proof of Theorem 4.8

▶ **Lemma E.1.** *If* $\Gamma, x : T \vdash P$ *and* $P\downarrow$*, then* $P\uparrow^{x}_{[\![T]\!]}$*.*

Proof. By induction on the derivation of $\Gamma, x : T \vdash P$ and by cases on the last rule applied. We only discuss those cases that are compatible with the hypothesis $P\downarrow$.

$\boxed{\text{T-IDLE}}$ Then $P =$ `idle` and $\mathsf{un}(T)$, hence $T = \circ$. We conclude $P\uparrow^{x}_{0}$ noting that $[\![T]\!] = 0$.

$\boxed{\text{T-DONE}}$ Then $P =$ `done` y. We distinguish two subcases. If $x = y$, then $T = \bullet$ and we conclude $P\uparrow^{x}_{1}$ noting that $[\![T]\!] = 1$. If $x \neq y$, then we have $\mathsf{un}(T)$, hence $T = \circ$ and we conclude as in the case of rule T-IDLE.

$\boxed{\text{T-PAR}}$ Then $P = P_1 \mid P_2$ and $\Gamma, x : T = \Gamma_1, \Gamma_2, y : \langle \llbracket S \rrbracket \rangle$ and $\Gamma_1, y : S \vdash P_1$ and $\Gamma_2, y : \overline{S} \vdash P_2$. It must be the case that $x \in \mathsf{dom}(\Gamma_i)$ for some $i \in \{1, 2\}$. We conclude using the induction hypothesis on P_i.

$\boxed{\text{T-CHOICE}}$ Then there exist P_1, P_2, Γ_1, Γ_2, T_1 and T_2 such that $P = P_1 \;{}_{p}\!\boxplus\; P_2$ and $\Gamma, x : T = \Gamma_1, x : T_1 \;{}_{p}\!\boxplus\; \Gamma_2, x : T_2$ and $\Gamma_i, x : T_i \vdash P_i$ for $i = 1, 2$. From $P\!\downarrow$ we deduce $P_i\!\downarrow$ for $i = 1, 2$. Using the induction hypothesis we deduce $P_i \uparrow^{x}_{\llbracket T_i \rrbracket}$ for $i = 1, 2$, hence $P \uparrow^{x}_{p\llbracket T_1 \rrbracket + (1-p)\llbracket T_2 \rrbracket}$ by Definition 4.6. We conclude $P \uparrow^{x}_{\llbracket T \rrbracket}$ using Proposition 3.6.

$\boxed{\text{T-NEW}}$ Then there exist Q, y and p such that $P = (y)Q$ and $\Gamma, y : \langle p \rangle, x : T \vdash Q$. From $P\!\downarrow$ we deduce $Q\!\downarrow$. We conclude using the induction hypothesis. ◀

▶ **Theorem 4.8.** *If $x : \langle p \rangle \vdash P$ and $P \not\rightarrow$, then $P \uparrow^{x}_{p}$.*

Proof. From Lemma D.12 we deduce $P\!\downarrow$. We prove that $\Gamma, x : \langle p \rangle \vdash P$ and $P\!\downarrow$ imply $P \uparrow^{x}_{p}$ by induction on the derivation of $\Gamma, x : \langle p \rangle \vdash P$ and by cases on the last rule applied. We only consider those cases that are compatible with the assumption $x : \langle p \rangle$.

$\boxed{\text{T-PAR when the name being split is } x}$ Then there exist P_1, P_2, Γ_1, Γ_2 and T such that $P = P_1 \mid P_2$ and $\Gamma = \Gamma_1, \Gamma_2$ and $\Gamma_1, x : T \vdash P_1$ and $\Gamma_2, x : \overline{T} \vdash P_2$ and $p = \llbracket T \rrbracket$. From $P\!\downarrow$ we deduce $P_1\!\downarrow$. We conclude using Lemma E.1.

$\boxed{\text{T-PAR when the name being split is some } y \neq x}$ Then there exist P_1, P_2, Γ_1, Γ_2 and T such that $P = P_1 \mid P_2$ and $\Gamma, x : \langle p \rangle = \Gamma_1, \Gamma_2, y : \langle \llbracket T \rrbracket \rangle$ and $\Gamma_1, y : T \vdash P_1$ and $\Gamma_2, y : \overline{T} \vdash P_2$. We only discuss the case $x \in \mathsf{dom}(\Gamma_1)$, the other being analogous. Then $\Gamma_1 = \Gamma_1', x : \langle p \rangle$ for some Γ_1'. From $P\!\downarrow$ we deduce $P_1\!\downarrow$. We conclude using the induction hypothesis.

$\boxed{\text{T-CHOICE}}$ Then there exist P_1, P_2, Γ_1, Γ_2, q, p_1 and p_2 such that $P = P_1 \;{}_{q}\!\boxplus\; P_2$ and $\Gamma, x : \langle p \rangle = (\Gamma_1, x : \langle p_1 \rangle) \;{}_{q}\!\boxplus\; (\Gamma_2, x : \langle p_2 \rangle)$ and $\Gamma_i, x : \langle p_i \rangle \vdash P_i$ for $i = 1, 2$. In particular, $p = qp_1 + (1-q)p_2$. From $P\!\downarrow$ we deduce $P_i\!\downarrow$ for $i = 1, 2$. Using the induction hypothesis we deduce $P_i \uparrow^{x}_{p_i}$ for $i = 1, 2$, hence we conclude $P \uparrow^{x}_{qp_1 + (1-q)p_2}$.

$\boxed{\text{T-NEW}}$ Then there exist y, p, Q such that $P = (y)Q$ and $\Gamma, y : \langle q \rangle, x : \langle p \rangle \vdash Q$. From $P\!\downarrow$ we deduce $Q\!\downarrow$. We conclude using the induction hypothesis. ◀

▶ **Corollary 4.9** (relative success). *Let $P \Uparrow^{x}_{p}$ if there exist (P_n) and (p_n) such that $P \Rightarrow P_n$ and $P_n \uparrow^{x}_{p_n}$ for all $n \in \mathbb{N}$ and $\lim_{n\to\infty} p_n = p$. Then (1) $x : \langle 1 \rangle \vdash P$ and $P \Downarrow_p$ imply $P \Uparrow^{x}_{p}$ and (2) $x : \langle p \rangle \vdash P$ and $P \Downarrow_1$ imply $P \Uparrow^{x}_{p}$.*

Proof. We prove the two items separately.

$\boxed{\text{Item 1}}$ From the hypothesis $P \Downarrow_p$ we know that there exist (Q_n), (R_n) and (p_n) such that $P \Rightarrow Q_n \;{}_{p_n}\!\boxplus\; R_n$ and $Q_n\!\downarrow$ for every $n \in \mathbb{N}$ and $\lim_{n\to\infty} p_n = p$. From the hypothesis $x : \langle 1 \rangle \vdash P$ and Theorem 4.3 we deduce $x : \langle 1 \rangle \vdash Q_n \;{}_{p_n}\!\boxplus\; R_n$ for every $n \in \mathbb{N}$. From T-CHOICE and Definition 3.5 we deduce $x : \langle 1 \rangle \vdash Q_n$ for every $n \in \mathbb{N}$. From $Q_n\!\downarrow$ and Theorem 4.8 we deduce $Q_n \uparrow^{x}_{1}$. Using Definition 4.6 we derive $Q_n \;{}_{p_n}\!\boxplus\; R_n \uparrow^{x}_{p_n}$ for every $n \in \mathbb{N}$, hence $P \Uparrow^{x}_{p}$.

$\boxed{\text{Item 2}}$ From the hypothesis $P \Downarrow_1$ we know that there exist (Q_n), (R_n) and (p_n) such that $P \Rightarrow Q_n \;{}_{p_n}\!\boxplus\; R_n$ and $Q_n\!\downarrow$ for every $n \in \mathbb{N}$ and $\lim_{n\to\infty} p_n = p$. That is, for every $\varepsilon > 0$, there exists N such that, for every $n \geq N$, we have $1 - p_n < \varepsilon$. From the hypothesis $x : \langle p \rangle \vdash P$ and Theorem 4.3 we deduce $x : \langle p \rangle \vdash Q_n \;{}_{p_n}\!\boxplus\; R_n$ for every $n \in \mathbb{N}$. From T-CHOICE and Definition 3.5 we deduce that, for every $n \in \mathbb{N}$, there exist q_n and r_n such that $p = p_n q_n + (1-p_n) r_n$ and $x : \langle q_n \rangle \vdash Q_n$ and $x : \langle r_n \rangle \vdash R_n$. From $Q_n\!\downarrow$ and Theorem 4.8 we deduce $Q_n \uparrow^{x}_{q_n}$ for every $n \in \mathbb{N}$, hence $(Q_n \;{}_{p_n}\!\boxplus\; R_n) \uparrow^{x}_{p_n q_n}$ for every $n \in \mathbb{N}$. Now $p - p_n q_n = p_n q_n + (1-p_n) r_n - p_n q_n = (1-p_n) r_n < \varepsilon$, hence $\lim_{n\to\infty} p_n q_n = p$. ◀