Probabilities in Session Types

Bogdan Aman Gabriel Ciobanu
Romanian Academy, Institute of Computer Science, Iaşi, Romania
Alexandru Ioan Cuza University of Iaşi, Faculty of Computer Science, Romania
bogdan.aman@iit.academiaromana-is.ro gabriel@info.uaic.ro

This paper deals with the probabilistic behaviours of distributed systems described by a process calculus considering both probabilistic internal choices and nondeterministic external choices. For this calculus we define and study a typing system which extends the multiparty session types in order to deal also with probabilistic behaviours. The calculus and its typing system are motivated and illustrated by a running example.

1 Introduction

Probabilities allow uncertainty to be described in quantitative terms. If there are no uncertainties about how a system behaves, then its expected behaviour has a 100% chance of occurring, while any other behaviour would have no chance (i.e., 0% chance). Regarding the possible behaviours of a system, people working in artificial intelligence have used probability distributions over a set of events [9]. In such an approach, the probabilities assigned to behaviours are real numbers from $[0,1]$ rather than values in $\{0,1\}$. In [5], the authors made explicitly the assumption that probabilities are distributed over a restricted set of events, each of them corresponding to an equivalence class of events. We adapt these ideas to the framework of multiparty session types, and introduce probabilities assigned to actions and label selections.

An important feature of a probabilistic model is given by the distinction between nondeterministic and probabilistic choices [18]. The nondeterministic choices refer to the choices made by an external process, while probabilistic choices are choices made internally by the process (not under control of an external process). Intuitively, a probabilistic choice is given by a set of alternative transitions, where each transition has a certain probability of being selected; moreover, the sum of all these probabilities (for each choice) is 1. To clarify the difference between nondeterministic and probabilistic choices, we consider a variant of the *two-buyers-seller* protocol [14] depicted in Figure 1. Two buyers (*Alice* and *Bob*) wish to buy an expensive book (out of several possible ones) from a *Seller* by combining their money (in various amounts depending on the amount of cash *Alice* is willing to pay). The communications between them can be described in several steps. Firstly, *Alice* sends (out of several choices) a book *title* (a string) or an *ISBN* (a number) to the *Seller*. The fact that *Alice* chooses which book she wants to buy by sending the book title or book *ISBN* is an example of a probabilistic choice, because it is under her control and preference (this is why in Figure 1 we added probabilities to the possible choices of *Alice* regarding the book). Then, *Alice* waits for an answer regarding the quote of the book. This is a nondeterministic choice, because the choice of the answer received by *Alice* is out of her control. This is due to the fact that the *Seller* may provide different quotes depending on the buying history of *Alice* and existing discounts. Next, *Seller* sends back a *quote* (an integer) to *Alice* and *Bob*. *Alice* tells *Bob* how much she can contribute (an integer). Depending on the contribution of *Alice*, *Bob* notifies *Seller* whether it accepts the quote or not. If *Bob* accepts, he sends his home or office address (a string), and awaits from the *Seller* a delivery date when the requested book will be received.

M. Marin, A. Crăciun (Eds.):
Working Formal Methods Symposium 2019 (FROM 2019)
EPTCS 303, 2019, pp. 92–106, doi:10.4204/EPTCS.303.7

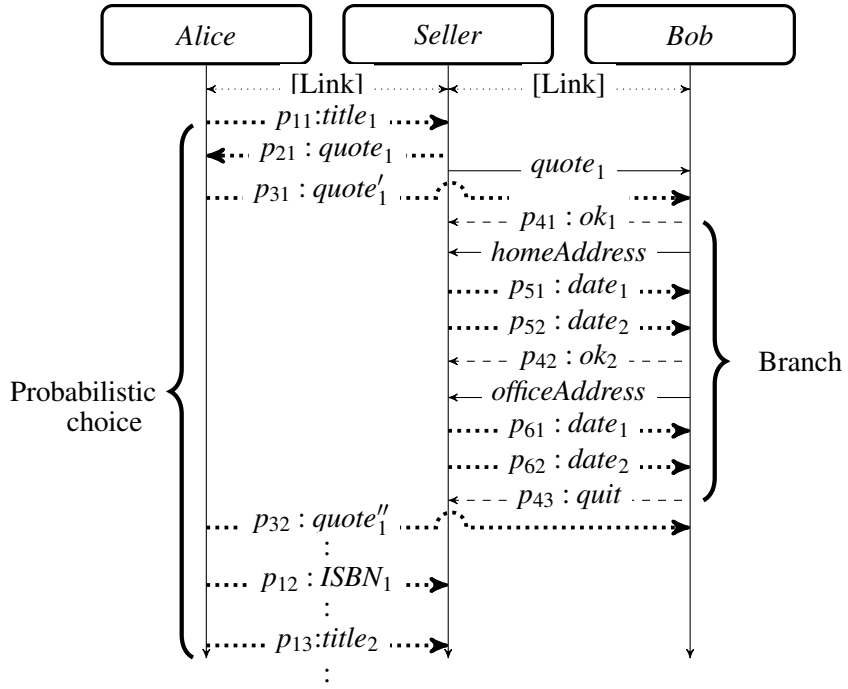

Figure 1: Dotted lines stand for probabilistic choices p_{ij}, dashed lines for branching, solid lines for deterministic choices, while double headed dotted lines for session initialization.

One goal of the current research lines is to use a formal approach to describe in a rigorous way how distributed systems should behave, and then to design these systems properly in order to satisfy the behavioural constraints. In the last few years the focus has moved towards the quantitative study of the distributed systems behaviour to be able to solve problems that are not solvable by deterministic approaches (e.g., leader election problem [7]).

Probabilistic modelling is usually used to represent and quantify uncertainty in the study of distributed systems. Several probabilistic process calculi have been considered in the literature: probabilistic CCS [10], probabilistic CSP [15], probabilistic ACP [2], probabilistic asynchronous π-calculus [11], PEPA [12]. The basic idea of these probabilistic process calculi is to include a probabilistic choice operator. Essentially, there are two possibilities of extending such an approach: either to replace nondeterministic choices by probabilistic choices, or to allow both probabilistic and nondeterministic choices.

In this paper we consider the second alternative, and allow probabilistic choices made internally by the communicating processes (sending a value or a label), and also nondeterministic choices controlled by an external process (receiving a value or a label). Notice that in our operational semantics we impose that for each received value/label, the continuation of a nondeterministic choice is unique; thus, the corresponding execution turns out to be completely deterministic. We use a probabilistic extension of the process calculus presented in [14], a calculus which is also an extension of the π-calculus [16] for which the papers [11, 20] present a probabilistic approach. For this calculus we define and study a typing system by extending the multiparty session types with both nondeterministic and probabilistic behaviours.

Session types [13, 19] and multiparty session types [14] provide a typed foundation for the design of communication-based systems. The main intuition behind session types is that a communication-based application exhibits a structured sequence of interactions. Such a structure is abstracted as a type through an intuitive syntax which is used to validate programs. Session types are terms of a process algebra that also contains a selection construct (an internal choice among a set of branches), a branching construct

(an external choice offered to the environment) and recursion. Session types are able to guarantee several properties in a session: (i) interactions never lead to a communication error (communication safety); (ii) channels are used linearly (linearity) and are deadlock-free (progress); (iii) the communication sequence follows a declared scenario (session fidelity, predictability).

While many communication patterns can be captured through such sessions, there are cases where basic multiparty session types are not able to capture interactions which involve internal probabilistic choices of the participants. Probabilities are used in the design and verification of complex systems in order to quantify unreliable or unpredictable behaviour, but also taken also into account when analyzing quantitative properties (measuring somehow the success level of the protocol). Overall, we study the nondeterministic and probabilistic choices in the framework of multiparty session types in order to understand better the quantitative aspects of uncertainty that might arise in communicating processes.

In the following, Section 2 presents the syntax and semantics of our probabilistic process calculus, and motivates the key ideas by using the two-buyers-seller protocol. Section 3 explains the global and local types, and the connection between them. Section 4 describes the new typing system and presents the main results. Section 5 concludes and discusses some related probabilistic approaches involving typing systems.

2 Probabilistic Multiparty Session Processes

The most natural way to define a probabilistic extension of a process calculus consists of adding probabilistic information to some actions [8]. Probabilities are not attached to some actions, while others have probabilities (see [20]). When modelling the probabilistic behaviour of a distributed system, we should be able to model the fact that either the system or the environment chooses between several alternative behaviours. Moreover, when modelling such a system we should avoid to 'approximate' the nondeterministic choice by a probabilistic distribution (very often a uniform distribution is used). For these reasons, we define a probabilistic extension of the process calculus used in [14] that combines both nondeterministic and probabilistic behaviours. We actually define a calculus that puts together probabilistic internal choices (sending a value and selecting a label) with nondeterministic external choices (receiving a value and branching a process by using a selected value). In this setting, the nondeterministic actions of a process use information (values and labels) provided only by probability actions. The type system for this calculus is inspired from the synchronous multiparty session types [3]. As far as we know, our approach is new among the existing models used to formalize multiparty processes in the framework of multiparty session types.

2.1 Syntax

In what follows we use our variant of the two-buyers-seller protocol to illustrate some of the syntactic constructs defined afterwards.

Example 1. *Let us note that the book to buy represents the choice of Alice, and so she sends the title of a book (a string) or an ISBN (a ten digit number). Since this is under her control and preference, it represents an example of a probabilistic choice.*

$$Alice = 0.3 : as!\langle \text{“}War\ and\ Peace''\rangle; Alice_1 + 0.5 : as!\langle \text{“}The\ Art\ of\ War''\rangle; Alice_2 + 0.2 : as!\langle 0195014766\rangle; Alice_3.$$

Here 'as' denotes the channel used for the communication between Alice and the Seller. Actually, channels 'as' and 'ab' are used by Alice to communicate with Seller and with Bob, while channel 'bs' is used

by Bob to communicate with Seller. We denote by $Alice_i$ ($1 \leq i \leq 3$) the different behaviours of Alice after she sent her book choice. We use this index notation to keep track of the behaviours for each participant, to simplify the syntax and make it easier to read. The detailed description of all participants can be found in Example 3.

When receiving the book orders, the Seller expects the buyers sending him either a string representing a title of the book or a number representing an ISBN. This behaviour is nondeterministic depending on the received information: $Seller = as?(title : string); Seller_1 + as?(ISBN : nat); Seller_2$.

Informally, a session is a series of interactions between multiple parties serving as a unit of conversation. A session is established via a shared name representing a public interaction point, and consists of series of communication actions performed on fresh session channels. The syntax for processes is based on user-defined processes [14] extended with probabilistic choices. The syntax is presented in Table 1, where we use: probabilities p_1, p_2, ...; shared names a, b, n, ... and session names x, y, ...; channels s, t, ...; expressions e, e_i, ...; labels l, l_i, ..., participants q, ... We use symbols q to name participants despite the fact that they are in reality numbers.

Processes	P	::=	$\overline{a}[\mathrm{n}](\tilde{s}).P$		(multicast session request)
		|	$a[q](\tilde{s}).P$		(session acceptance)
		|	$\sum_{i\in I} p_i : s!\langle \tilde{e}_i\rangle; P_i$		(value sending)
		|	$\sum_{i\in I} s?(\tilde{x}_i : \tilde{S}_i); P_i$	$(\tilde{S}_k \neq \tilde{S}_t$, for $k,t \in I, k \neq t)$	(value reception)
		|	$s!\langle\langle \tilde{s}\rangle\rangle; P$		(session delegation)
		|	$s?((\tilde{s})); P$		(session reception)
		|	$\sum_{i\in I} p_i : s \lhd l_i; P_i$	$(l_k \neq l_t$, for $k,t \in I, k \neq t)$	(label selection)
		|	$s \rhd \{l_i : P_i\}_{i\in I}$	$(l_k \neq l_t$, for $k,t \in I, k \neq t)$	(label branching)
		|	if e then P else Q		(conditional branch)
		|	$P \mid Q$		(parallel)
		|	$\mathbf{0}$		(inaction)
		|	$(\nu n)P$		(hiding)
		|	$\mu X.P$		(recursion)
		|	X		(variable)
Expressions	e	::=	v | e and e' | not e | ...		
Values	v	::=	a | $true$ | $false$ | ...		
Sorts	S	::=	$bool \mid nat \mid \ldots$		(value types)

Table 1: Syntax

Excepting the primitives for value sending, value receiving and label selection, all the other constructs are from [14]. The process $\overline{a}[\mathrm{n}](\tilde{s}).P$ sends along channel a a request to start a new session using the channels $\tilde{s}$ with participants 1 ... n, where it participates as 1 and continues as P. Its dual $a[q](\tilde{s}).P$ engages in a new session as participant q. The communications taking place inside an established session are performed using the next six primitives: sending/receiving a value, session delegation/reception, and selection/branching. By using the delegation/reception pair, a process delegates to another one the capability to participate in a session by passing the channels associated with the session. The conditional branching establishes the continuation of an evolution based on the truth value of an expression e. It is worth mentioning that the internal choices (sending a value and selecting a label) are probabilistically chosen, while the receiving values represent a nondeterministic choice (as external choice). The conditional branch, parallel and inaction are standard. A sequence of parallel composition is written $\Pi_i P_i$. The

syntax $(\nu n)P$ makes the name n local to P. Interaction which can be repeated unboundedly is realized by recursion; as in [4], we do not use arguments when defining recursion. We often omit writing $\mathbf{0}$ at the end of processes (e.g., $s!\langle\tilde{e}_i\rangle;\mathbf{0}$ is written as $s!\langle\tilde{e}_i\rangle$).

The notions of identifiers (bound and free), process variables (bound and free), channels, alpha equivalence $\equiv_\alpha$ and substitution are standard. The bound identifiers are $\tilde{s}$ in multicast session request, session acceptance and session reception, $\tilde{x}_j$ in value reception and n in hiding, while the bound process variable is X in recursion. $fv(P)$ and $fn(P)$ denote the sets of free process variables and free identifiers of P, respectively.

2.2 Operational Semantics

Structural equivalence for processes is the least equivalence relation satisfying the following equations:

$$P \mid \mathbf{0} \equiv P \qquad P \mid Q \equiv Q \mid P \qquad (P \mid Q) \mid R \equiv P \mid (Q \mid R)$$
$$(\nu n)P \mid Q \equiv (\nu n)(P \mid Q) \text{ if } n \notin fn(Q) \quad (\nu n)(\nu n')P \equiv (\nu n')(\nu n)P$$
$$(\nu n)\mathbf{0} \equiv \mathbf{0} \qquad \mu X.\mathbf{0} \equiv \mathbf{0} \qquad p_i : P + p_j : Q \equiv p_j : Q + p_i : P \qquad P + Q \equiv Q + P.$$

We define the operational semantics in such a way that it distinguishes between probabilistic choices made internally by a process and nondeterministic choices made externally. This distinction allows us to reason about the evolutions of the system in which the nondeterministic actions of a process use only the data send by the probability actions. The operational semantics is given by a reduction relation denoted by $P \to_r Q$ (meaning P reduces to Q with probability r) representing the smallest relation generated by the rules of Table 2 (where $e \downarrow v$ means that expression e is evaluated to value v).

$\overline{a}[\mathrm{n}](\tilde{s}).P_1 \mid \Pi_{q\in\{2..\mathrm{n}\}} a[q](\tilde{s}).P_q \to_1 (\nu\tilde{s})\Pi_{q\in\{1..\mathrm{n}\}} P_q$	(LINK)
$\sum_{i\in I} p_i : s!\langle\tilde{e}_i\rangle;P_i \mid \sum_{j\in J} s?(\tilde{x}_j : \tilde{S}_j);P_j \to_{p_i} P_i \mid P_j\{\tilde{v}_i/\tilde{x}_j\}\ (\tilde{e}_i \downarrow \tilde{v}_i,\ \tilde{v}_j : \tilde{S}_j)$	(COM)
$s!\langle\langle\tilde{s}\rangle\rangle;P \mid s?((\tilde{s}));Q \to_1 P \mid Q$	(DELEG)
$\sum_{i\in I} p_i : s \lhd l_i;P_i \mid s \rhd \{l_j : P_j\}_{j\in J} \to_{p_i} P_i \mid P_j\ (j \in J)$	(LABEL)
if e then P else $Q \to_1 P$ $(e \downarrow true)$	(IFT)
if e then P else $Q \to_1 Q$ $(e \downarrow false)$	(IFF)
$\mu X.P \to_1 P\{\mu X.P/X\}$	(CALL)
$P \to_p P'$ implies $(\nu n)P \to_p (\nu n)P'$	(SCOPE)
$P \to_p P'$ and $Q \not\to$ implies $P \mid Q \to_p P' \mid Q$	(PAR1)
$P \to_p P'$ and $Q \to_q Q'$ implies $P \mid Q \to_{p\cdot q} P' \mid Q'$	(PAR2)
$P \equiv P'$ and $P' \to_p Q'$ and $Q' \equiv Q'$ implies $P \to_p Q$	(STRUCT)

Table 2: Operational Semantics

Rule (LINK) describes a session initiation among n parties, generating $|\tilde{s}|$ fresh multiparty session channels. For simplicity, we consider that this rule has probability 1; a normalization based on the possible reachable processes in one step is eventually needed (as done in [8]). Rules (COM), (DELEG) and (LABEL) are used to communicate values, session channels and labels. The values to be communicated in the rules (COM) and (LABEL) are chosen probabilistically, a fact illustrated by adding the probability of the consumed action to the transition of the reduction relation. In both rules (COM) and (LABEL), the choice of the continuation process to be executed after sending or selecting is probabilistic, while when receiving or branching is nondeterministic. Inspired by [1], we add the conditions $\tilde{S}_k \neq \tilde{S}_t$ (meaning that the types of $\tilde{x}_k$ and $\tilde{x}_t$ are different) and $l_k \neq l_t$ (meaning that l_k and l_t are different) in the rules (COM) and (LABEL) to indicate that each value and label leads to a unique continuation. This means that a process

of the form $s?(x:S);P+s?(x:S).Q$ is not possible in our syntax. Thus, the probability of the transition is equal with the probability of the sending/selecting process. It could be noticed from (LINK), (COM) and (LABEL) that the calculus is synchronous; this choice is made in order to simplify the presentation.

The rules (IFT) and (IFF) choose which branch to take depending on the truth value of e_i. The rules (SCOPE) and (STRUCT) are standard. Rule (PAR2) is used to compose the evolutions of parallel processes, while rule (PAR1) is used to compose concurrent processes that are able to evolve with processes that are not able to evolve. In rule (PAR1), $Q \not\rightarrow$ means that the process Q is not able to evolve by means of any rule (we say that Q is a stuck process). Negative premises are used to denote the fact that passing to a new step is performed based on the absence of actions. The use of negative premises does not lead to an inconsistent set of rules. The following example illustrates how and when the rule (PAR1) is used.

Example 2 (cont.)**.** *Let us consider the process $P = Alice \mid Seller \mid Bob$, where Alice and Seller have the definitions from Example 1, while Bob can have any form. By applying a (*COM*) rule, we could have*

$$Alice \mid Seller \rightarrow_{0.2} Alice_3 \mid Seller_2\{0195014766/ISBN\}$$

*In order to illustrate the evolution of P, we need to add also Bob to the above reduction. Notice that during this step Bob is not able to interact neither with Alice nor with Seller. This is done by using the rule (*PAR1*), and so obtaining* $P \rightarrow_{0.2} Alice_3 \mid Seller_2\{0195014766/ISBN\} \mid Bob$.

Example 3 (cont.)**.** *Let us consider an instance of the two-buyers-seller protocol in which Alice wants to buy one of the following two books:*

- *Title: "War and Peace" / ISBN: 0140447938;*
- *Title: "The Art of War" / ISBN: 0195014766.*

Firstly, Alice sends to Sellera book identifier (title or ISBN), namely with probability 0.3 *the book title "War and Peace", with probability* 0.5 *the book title "The Art of War", and with probability* 0.2 *the ISBN 0195014766 of the latter book. Then Alice waits for Seller to send a quote to both her and Bob. Alice tells Bob how much she can contribute (based on certain probabilities and the book she actually wants). For example, for the book "War and Peace" she is willing to participate with either quote*$/2$ *or quote*$/3$ *with the same probability* 0.5*. We now describe formally the behaviour of Alice as a process:*

$$\begin{aligned} Alice \stackrel{def}{=} {} & \overline{a}[3](ab,as,bs). \\ & 0.3 : as!\langle \text{“}War\ and\ Peace''\rangle; as?(quote:nat); \\ & \quad 0.5 : ab!\langle quote/2\rangle.P_1 + 0.5 : ab!\langle quote/3\rangle.P_1 \\ & + 0.5 : as!\langle \text{“}The\ Art\ of\ War''\rangle; as?(quote:nat); \\ & \quad 0.4 : ab!\langle quote/2\rangle.P_1 + 0.2 : ab!\langle quote/3\rangle.P_1 + 0.4 : ab!\langle quote/4\rangle.P_1 \\ & + 0.2 : as!\langle 0195014766\rangle; as?(quote:nat); \\ & \quad 0.4 : ab!\langle quote/2\rangle.P_1 + 0.2 : ab!\langle quote/3\rangle.P_1 + 0.4 : ab!\langle quote/4\rangle.P_1 \end{aligned}$$

Notice that the price options for the second book (searched either by title or ISBN) are the same; however, this is just a coincidence and not a requirement in our calculus. By using probabilities, it is possible to describe executions that may return different prices for the same title sold by the same Seller, but possibly printed by different publishers.

Using this process (behaviour) of Alice, we can find answers to questions like:

- *What is the probability that Alice buys "The Art of War" with quote*$/3$*?*

  This means that Alice needs to execute

  $$0.5 : as!\langle \text{“}The\ Art\ of\ War''\rangle; as?(quote:nat); 0.2 : ab!\langle quote/3\rangle.P_1$$

  with probability $0.5 \times 0.2 = 0.1$*, or to execute*

  $$0.2 : as!\langle 0195014766\rangle; as?(quote:nat); 0.2 : ab!\langle quote/3\rangle.P_1$$

  with probability $0.2 \times 0.2 = 0.04$*. Thus we get:*

- *Answer:* $0.5 \times 0.2 + 0.2 \times 0.2 = 0.14$.

- *What is the most probable choice made by Alice?*
  - *Answer: "The Art of War" with quote*$/2$ *and quote*$/4$*, with probability* $0.7 \times 0.4 = 0.28$.

Alice is willing to contribute partially to the quote, contribution that is probabilistically chosen out of several possibilities, depending on the book Alice intends to purchase. In process P_1*, Alice may perform the remaining transactions with Seller and Bob.*

3 Global and Local Types

In what follows, the notion of probability already presented in the previous section scales up to the global types. Since the probabilities are static, the global types just need to check if the probabilities to execute certain actions are the desired ones. Usually session types lead to a unique description of a distributed system by means of processes. If we would simply incorporate probabilities in the session types as done for processes, this would be too restrictive as the slightest perturbation of the probabilities in the processes can make the system failing the prescribed behaviour. This is why in what follows we use probabilistic intervals in session types, allowing for several processes to be considered behavioural equivalent by having the same type.

3.1 Global Types

The global types $G, G', \ldots$ presented in Table 3 describe the global behaviour of a probabilistic multiparty session process. In what follows we use probabilistic intervals δ having one of the following forms (c,d), $[c,d]$, $(c,d]$ or $[c,d]$, where $c,d \in [0,1]$ and $c \leq d$. For simplicity, we write $\delta = \lfloor c,d \rfloor$ with $\lfloor \in \{[,(\}$ and $\rfloor \in \{],)\}$. In what follows, we use also the addition of intervals defined as: if $\delta_1 = \lfloor c_1,d_1 \rfloor$ and $\delta_2 = \lfloor c_2,d_2 \rfloor$ then $\delta_1 + \delta_2 = \lfloor min(c_1+c_2,1), min(d_1+d_2,1) \rfloor$. If $\delta = [c,c]$, we use the shorthand notation $\delta = c$.

Global	G	::=	$\sum_{i \in I} q \to_{\delta_i} q' : k\langle S_i \rangle.G_i$	(probValues)
		|	$q \to_1 q' : k\langle T@p \rangle.G'$	(delegation)
		|	$\sum_{i \in I} q \to_{\delta_i} q' : k\{l_i : G_i\}$	(probBranching)
		|	G, G'	(parallel)
		|	$\mu t.G$	(recursive)
		|	t	(variable)
		|	end	(end)
Sorts	S	::=	$bool \mid nat \mid \ldots$	(value types)

Table 3: Syntax of Global Types

Type $\sum_{i \in I} q \to_{\delta_i} q' : k\langle S_i \rangle.G_i$ states that a participant q sends with a probability in the interval δ_i a message of type S_i to a participant q' through the channel k, and then the interactions described by G_i take place. We assume that in each communication $q \to q'$ we have $q \neq q'$, i.e. we prohibit reflexive interactions. Type $q \to_1 q' : k\langle T@p \rangle.G'$ denotes the delegation of a session channel of type T (called local type) with role p (written as $T@p$). The local types are discussed in detail later.

Type $\sum_{i \in I} q \to_{\delta_i} q' : k\{l_i : G_i\}$ says that participant q sends with a probability in the interval δ_i one label on channel k to another participant q'. If l_i is sent, evolution described by type G_i takes place. Type G, G'

represents concurrent runs of processes specified by G and G'. Type $\mu t.G$ is a recursive type, where type variable t is guarded in the standard way (they only appear under some prefix). Similar to the approach presented in [4], we overload the notation μ as it is easy to see from the context if it precedes a process or a type. Type end represents the termination of a process; we identify both G,end and end,G with G.

In a probabilistic choice, identically behaved branches can be replaced by a single branch with a behaviour having the sum of the probabilities of the individual branches.

Remark 1. *If all the possible interactions communicate the same types (all S_i are identical), all select the same branch (all l_i are identical), and the continuations after communications respect the same global type (all G_i are identical), then the global systems can be simplified by using the following rules:*

- $\sum_{i\in I} q \to_{\delta_i} q' : k\langle S_i\rangle.G_i$ *is the same as* $q \to_{\sum_{i\in I}\delta_i} q' : k\langle S\rangle.G$ *whenever* $S = S_i$ *and* $G = G_i$ *for all* i*;*
- $\sum_{i\in I} q \to_{\delta_i} q' : k\{l_i : G_i\}$ *is the same as* $q \to_{\sum_{i\in I}\delta_i} q' : k\{l_i : G_i\}$ *whenever all* l_i *are equal.*

This means that if $\sum_{i\in I}\delta_i = 1$, then global types may contain only probabilities equal to 1, namely a form similar to the global types in multiparty session types from [14]. Therefore, for the processes of this particular type, all the results presented in [14] hold.

Example 4 (cont.)**.** *Using the previous remark, the following is a global type of the two-buyers-seller protocol of Example 1:*

$$Alice \to_{\lfloor 0.7,0.9\rfloor} Seller : as\langle string\rangle.G1 + Alice \to_{\lfloor 0.15,0.25\rfloor} Seller : as\langle nat\rangle.G1, \text{ where}$$

$$G_1 = Seller \to_1 Alice : as\langle int\rangle.Seller \to_1 Bob : bs\langle int\rangle.$$
$$Alice \to_1 Bob : ab\langle int\rangle.\ Bob \to_{\lfloor 0.18,0.22\rfloor} Seller : bs\{ok_1 : Bob \to_1 Seller : bs\langle string\rangle.$$
$$Seller \to_1 Bob : bs\langle date\rangle.end\}$$
$$+ Bob \to_{\lfloor 0.27,0.31\rfloor} Seller : bs\{ok_2 : Bob \to_1 Seller : bs\langle string\rangle.$$
$$Seller \to_1 Bob : bs\langle date\rangle.end\}$$
$$+ Bob \to_{\lfloor 0.45,0.52\rfloor} Seller : bs\{quit.end\}.$$

This global type for Alice is due to the fact that even if she has different book titles that she wants to buy, the global type only records the type of the sent value (namely a string). Also, the fact that she behaves in a similar manner after sending the title, the global type can be reduced to a simpler form (according to the above remark).

Example 5. *Let us consider now that Alice decided that, instead of the books "War and Peace" and "The Art of War", she wants the books "Peter Pan" and "Robinson Crusoe", and she is willing to pay different amount from the quote. More exactly,*

$$Alice \stackrel{def}{=} \overline{a}[3](ab, as, bs).$$
$$0.15 : as!\langle \text{“}Peter\ Pan''\rangle; as?(quote : nat);\ 1 : ab!\langle quote/3\rangle.P_1$$
$$+\ 0.65 : as!\langle \text{“}Robinson\ Crusoe''\rangle; as?(quote : nat);\ 0.35 : ab!\langle quote/3\rangle.P_1 + 0.65 : ab!\langle quote/4\rangle.P_1$$
$$+\ 0.2 : as!\langle 1593080115\rangle; as?(quote : nat);\ 0.45 : ab!\langle quote/2\rangle.P_1 + 0.55 : ab!\langle quote/4\rangle.P_1\ .$$

It is worth mentioning that the two-buyers-seller protocol in which Alice is described by this definition is well-typed using the same global type (the one from Example 4) as the initial protocol of Example 1. Therefore, several different processes may have the same global type.

3.2 Local Types

Local types $T, T', \ldots$ presented in Table 4 describe the local behaviour of processes, acting as a link between global types and processes.

Local	T	::=	$\sum_{i\in I} \delta_i : k!\langle S_i\rangle.T_i$	(send)
		|	$\sum_{i\in I} k?(S_i).T_i$	(receive)
		|	$k!\langle T@q\rangle.T'$	(sessionDelegation)
		|	$k?(T@q).T'$	(sessionReceive)
		|	$k \oplus \{\delta_i : (l_i : T_i)\}_{i\in I}$	(selection)
		|	$k\&\{l_i : T_i\}_{i\in I}$	(branching)
		|	$\mu t.T$	(recursive)
		|	t	(variable)
		|	end	(end)
Sorts	S	::=	*bool* | *nat* | ...	(value types)

Table 4: Syntax of Local Types

Type $\sum_{i\in I} \delta_i : k!\langle S_i\rangle.T_i$ represents the behaviour of sending with probability in the interval δ_i a value of type S_i, and then behaving as described by type T_i. Similarly, $\sum_{i\in I} k?(S_i).T_i$ is for nondeterministic receiving, and then continuing as described by local type T_i. The type $k!\langle T@q\rangle.T'$ represents the behaviour of delegating a session of type $T@q$, while $k?(T@q).T'$ describes the behaviour of receiving a session of type $T@q$. Type $k \oplus \{\delta_i : (l_i : T_i)\}_{i\in I}$ describes a branching: it waits for $|I|$ options, and behaves as type T_i if the i-th label is selected with probability in the interval δ_i. Type $k\&\{l_i : T_i\}_{i\in I}$ represents the behaviour which nondeterministically selects one of the tags (say l_i), and then behaves as T_i. The rest is the same as for the global types, demanding type variables to occur guarded by a prefix. For simplicity, as done in [14], the local types do not contain the parallel composition.

Example 6. *The following is a local type for the process Alice presented in Example 3:*

$\lfloor 0.7, 0.9\rfloor : as!\langle string\rangle.as?(int).1 : ab!\langle int\rangle.T_1 + \lfloor 0.15, 0.25\rfloor : as!\langle nat\rangle.as?(int).1 : ab!\langle int\rangle.T_1$ *,*
where T_1 is the local type of process P_1 from the definition of Alice.

We define the projection of a global type to a local type for each participant.

Definition 1. *The projection for a participant q appearing in a global type G, written $G \upharpoonright q$, is inductively given as:*

- $(q_1 \to_1 q_2 : k\langle T@p\rangle.G') \upharpoonright q = \begin{cases} k!\langle T@p\rangle.(G' \upharpoonright q) & \text{if } q = q_1 \neq q_2 \\ k?(T@p).(G' \upharpoonright q) & \text{if } q = q_2 \neq q_1 \\ G' \upharpoonright q & \text{if } q \neq q_1 \text{ and } q \neq q_2 \end{cases}$;
- $(\sum_{i\in I} q_1 \to_{\delta_i} q_2 : k\langle S_i\rangle.G_i) \upharpoonright q = \begin{cases} \sum_{i\in I} \delta_i : k!\langle S_i\rangle.(G_i \upharpoonright q) & \text{if } q = q_1 \neq q_2 \\ \sum_{i\in I} k?(S_i).(G_i \upharpoonright q) & \text{if } q = q_2 \neq q_1 \\ G_1 \upharpoonright q & \text{if } q \neq q_1 \text{ and } q \neq q_2 \\ & \forall i, j \in J,\ G_i \upharpoonright q = G_j \upharpoonright q \end{cases}$;
- $(\sum_{i\in I} q_1 \to_{\delta_i} q_2 : k\{l_i : G_i\}) \upharpoonright q = \begin{cases} k \oplus \{\delta_i : (l_i : G_i \upharpoonright q)\}_{i\in I} & \text{if } q = q_1 \neq q_2 \\ k\&\{l_i : G_i \upharpoonright q\}_{i\in I} & \text{if } q = q_2 \neq q_1 \\ G_1 \upharpoonright q & \text{if } q \neq q_1 \text{ and } q \neq q_2 \\ & \forall i, j \in J,\ G_i \upharpoonright q = G_j \upharpoonright q \end{cases}$;

- $(G_1, G_2) \upharpoonright q = \begin{cases} G_i \upharpoonright q & \text{if } q \in G_i \text{ and } q \notin G_j, i \neq j \in \{1,2\} \\ \mathsf{end} & \text{if } q \notin G_1 \text{ and } q \notin G_2 \end{cases}$;
- $(\mu t.G) \upharpoonright q = \begin{cases} \mu t.(G \upharpoonright q) & \text{if } G \upharpoonright q \neq \mathsf{end} \text{ or } G \upharpoonright q \neq t \\ \mathsf{end} & \text{otherwise} \end{cases}$;
- $t \upharpoonright q = t$ • $\mathsf{end} \upharpoonright q = \mathsf{end}$.

When none of the side conditions hold, the projection is undefined.

Remark 2. *Regarding the check of linear usage of channels, the verification is similar to the one performed in [14], noting that the probabilistic and nondeterministic choices are treated similar to the branching in [14]. However, due to the use of synchronous communications, the sequence of interactions follows more strictly the one of the global behaviour description, resulting in a simpler linear property than in [14]. It should be said that in the branching clause, the projections of those participants different from q_1 and q_2 should generate an identical local type (otherwise undefined).*

Hereafter we assume that global types are well-formed, i.e. $G \upharpoonright q$ is defined for all q occurring in G.

4 Probabilistic Multiparty Session Types

We introduce a typing system with the purpose of typing efficiently the probabilistic behaviours of our processes. This typing system uses a map from shared names to either their sorts $(S, S', \ldots)$, or to a special sort $\langle G \rangle$ used to type sessions. Since a type is inferred for each participant, we use notation $T@q$ (called located type) to represent a local type T assigned to a participant q. Using these, we define

$$\Gamma ::= \emptyset \mid \Gamma, x : S \mid \Gamma, a : \langle G \rangle \mid \Gamma, X : \Delta \qquad \Delta ::= \emptyset \mid \Delta, \tilde{s} : \{T@q\}_{q \in I} .$$

A sorting $(\Gamma, \Gamma', \ldots)$ is a finite map from names to sorts, and from process variables to sequences of sorts and types. Typing $(\Delta, \Delta', \ldots)$ records linear usage of session channels by assigning a family of located types to a vector of session channels. $pid(G)$ stands for the set of participants occurring in G, while $sid(G)$ stands for the number of session channels in G. We write $\tilde{s} : T@q$ for a singleton typing $\tilde{s} : \{T@q\}$. Given two typings Δ and Δ', their disjoint union is denoted by Δ, Δ' (by assuming that their domains contain disjoint sets of session channels).

The type assignment system for processes is given in Table 5. We use the judgement $\Gamma \vdash P \rhd \Delta$ saying that "under the environment Γ, process P has typing Δ". The rules (TNAME),(TBOOL) and (TOR) are for typing names and expressions. The rules (TMCAST) and (TMACC) are for typing the session request and session accept, respectively. The type for $\tilde{s}$ is the projection on participant q of the declared global type G for a in Γ. It could be noticed that in rule (TMCAST) the projection is made on the participant requesting the session, while in (TMACC) the projection is made for each of the $(n-1)$ accepting participants. The local type $(G \upharpoonright q)@q$ means that the participant q has $G \upharpoonright q$ (namely the projection of G onto q) as its local type. The condition $|\tilde{s}| = sid(G)$ ensures that the number of session channels meets those in G.

The rules (TSEND) and (TRECEIVE) are for sending and receiving values, respectively. As these rules require probabilistic and nondeterministic choices, the rules should check all the possible choices with respect to Γ. Since one of the channels appearing in $\tilde{s}$ (say k) is used for communication, we record k by using the name $s[k]$ as part of the typed process. In both rules, q in $\tilde{s} : T_i@q$ ensures that each P_i represents (being inferred as) the behaviour for participant q, and its domain should be $\tilde{s}$. Then the relevant type prefixes $\sum_{i \in I} \delta_i : k!\langle \tilde{S}_i \rangle; T_i@q$ for the output and $\sum_{i \in I} k?(\tilde{S}_i); T_i@q$ for the input are composed in the session environment (as conclusion). The rules (TSDELEG) and (TSRECEIVE) are for delegation of

$$\Gamma, x : S \vdash x : S \qquad \Gamma \vdash \mathit{true}, \mathit{false} : \mathit{bool} \qquad \text{(TNAME), (TBOOL)}$$

$$\frac{\Delta \text{ end } \mathit{only}}{\Gamma \vdash \mathbf{0} \triangleright \Delta} \qquad \frac{\Gamma \vdash e_i : \mathit{bool}}{\Gamma \vdash e_1 \text{ or } e_2 : \mathit{bool}} \qquad \text{(TEND), (TOR)}$$

$$\frac{\Gamma \vdash a : \langle G \rangle \quad \Gamma \vdash P \triangleright \Delta, \tilde{s} : (G \upharpoonright 1)@1 \quad \{1, \ldots, \mathrm{n}\} = pid(G) \quad |\tilde{s}| = sid(G)}{\Gamma \vdash \overline{a}[\mathrm{n}](\tilde{s}).P \triangleright \Delta} \qquad \text{(TMCAST)}$$

$$\frac{\Gamma \vdash a : \langle G \rangle \quad \Gamma \vdash P \triangleright \Delta, \tilde{s} : (G \upharpoonright q)@q \quad q \in pid(G) \quad q \neq 1 \quad |\tilde{s}| = sid(G)}{\Gamma \vdash a[q](\tilde{s}).P \triangleright \Delta} \qquad \text{(TMACCEPT)}$$

$$\frac{\forall i.\Gamma \vdash \tilde{e_i} : \tilde{S_i} \quad \forall i.\Gamma \vdash P_i \triangleright \Delta, \tilde{s} : T_i@q \quad \sum_{i \in I} p_i = 1 \quad p_i \in \delta_i}{\Gamma \vdash \sum_{i \in I} p_i : s[k]!\langle \tilde{e_i} \rangle; P_i \triangleright \Delta, \tilde{s} : \sum_{i \in I} \delta_i : k!\langle \tilde{S_i} \rangle; T_i@q} \qquad \text{(TSEND)}$$

$$\frac{\forall i.\Gamma, \tilde{x_i} : \tilde{S_i} \vdash P_i \triangleright \Delta, \tilde{s} : T_i@q}{\Gamma \vdash \sum_{i \in I} s[k]?(\tilde{x_i} : \tilde{S_i}); P_i \triangleright \Delta, \tilde{s} : \sum_{i \in I} k?(\tilde{S_i}); T_i@q} \qquad \text{(TRECEIVE)}$$

$$\frac{\Gamma \vdash P \triangleright \Delta, \tilde{s} : T@q}{\Gamma \vdash s[k]!\langle \langle \tilde{t} \rangle \rangle; P \triangleright \Delta, \tilde{s} : k!\langle T'\tilde{@}q' \rangle; T@q, \tilde{t} : T'@q'} \qquad \text{(TSDELEG)}$$

$$\frac{\Gamma \vdash P \triangleright \Delta, \tilde{s} : T@q, \tilde{t} : T'@q'}{\Gamma \vdash s[k]?((\tilde{t})); P \triangleright \Delta, \tilde{s} : k?(T'@q'); T@q} \qquad \text{(TSRECEIVE)}$$

$$\frac{\forall i.\Gamma \vdash P_i \triangleright \Delta, \tilde{s} : T_i@q \quad \sum_{i \in I} p_i = 1 \quad p_i \in \delta_i}{\Gamma \vdash \sum_{i \in I} p_i : s[k] \lhd l_i; P_i \triangleright \Delta, \tilde{s} : k \oplus \{\delta_i : (l_i : T_i)\}_{i \in I}@q} \qquad \text{(TSELECT)}$$

$$\frac{\forall j.\Gamma \vdash P_j \triangleright \Delta, \tilde{s} : T_j@q}{\Gamma \vdash s[k] \rhd \{l_j; P_j\}_{j \in J} \triangleright \Delta, \tilde{s} : k\&\{l_j : T_j\}_{j \in J}@q} \qquad \text{(TBRANCH)}$$

$$\frac{\Gamma \vdash P \triangleright \Delta \qquad \Gamma \vdash Q \triangleright \Delta'}{\Gamma \vdash P \mid Q \triangleright \Delta, \Delta'} \qquad \frac{\Gamma \vdash e \triangleright \mathit{bool} \qquad \Gamma \vdash P \triangleright \Delta \qquad \Gamma \vdash Q \triangleright \Delta}{\Gamma \vdash \text{if } e \text{ then } P \text{ else } Q \triangleright \Delta} \qquad \text{(TCONC), (TIF)}$$

$$\frac{\Gamma, a : \langle G \rangle \vdash P \triangleright \Delta}{\Gamma \vdash (\nu a)P \triangleright \Delta} \qquad \frac{\Gamma \vdash P \triangleright \Delta, \tilde{s} : \{T_i@i\}_{i \in I}}{\Gamma \vdash (\nu \tilde{s})P \triangleright \Delta} \qquad \text{(TNRES), (TCRES)}$$

$$\frac{\Delta' \text{ end } \mathit{only}}{\Gamma, X : \Delta \vdash X \triangleright \Delta, \Delta'} \qquad \frac{\Gamma, X : \Delta \vdash P \triangleright \Delta}{\Gamma \vdash \mu X.P \triangleright \Delta} \qquad \text{(TVAR), (TREC)}$$

Table 5: Typing System

a session and its dual. They are similar to the rules (TSEND) and (TRECEIVE), except that here a vector of session channels is communicated instead of values. The carried type T' is located, making sure that the receiver takes the role of a specific participant (here q') in the delegated multiparty session. It should be noticed that in rule (TSDELEG) the type of $\tilde{t} : T'@q'$ does not appear in the type of P, while it appears in rule (TSRECEIVE) meaning that it uses the channels of P. The rules (TSELECT) and (TBRANCH) are for typing selection and branching, respectively. Similar to (TSEND) and (TRECEIVE), these rules employ probabilistic and nondeterministic choices, respectively. This means that the rules should check all the possible choices with respect to Γ.

The rule (TCONC) composes two processes if their local types are disjoint. The rules (TIF), (TEND), (TREC) and (TVAR) are standard. The rules (TNRES) and (TCRES) represent the restriction rules for shared names and channel names, respectively. In (TEND), "Δ end *only*" means that Δ contains only end types.

As processes interact, their dynamics is formalized as in [14] by a reduction relation $\Rightarrow$ on typing Δ:

- $\tilde{s} : \{\sum_{i\in I} \delta_i : k!\langle \tilde{S}_i\rangle; T_i@q_1, \sum_{j\in J} k?(\tilde{S}_j); T_j@q_2\} \Rightarrow_{\delta_{k_1}}$
  $\tilde{s} : \{T_{k_1}@q_1, T_{k_2}@q_2, \ldots\}$, for $k_1 \in I$, $k_2 \in J$ and $S_{k_1} = S_{k_2}$;
- $\tilde{s} : \{k!\langle T'@q'\rangle; T@q, , k?(T'@q'); T''@q''\} \Rightarrow_1 \tilde{s} : \{T@q, T''@q''\}$
- $\tilde{s} : \{k \oplus \{\delta_i : (l_i : T_i)\}_{i\in I}@q_1, k\&\{l_j : T_j\}_{j\in J}@q_2, \ldots\} \Rightarrow_{\delta_{k_1}}$
  $\tilde{s} : \{T_{k_1}@q_1, T_{k_2}@q_2, \ldots\}$, for $k_1 \in I$, $k_2 \in J$ and $S_{k_1} = S_{k_2}$;
- $\Delta, \Delta' \Rightarrow_p \Delta, \Delta''$ if $\Delta' \Rightarrow_p \Delta''$.

The first rule corresponds to sending/receiving a value of type $\tilde{S}_j$ by the participant q, while the second rule corresponds to session delegation. The third rule illustrates the choice and reception of a label l_j by the participant q. The last rule is used to compose typings when only a part of a typing changes.

We present two basic properties of our type system: substitution and weakening. The substitution plays a central role in proving type preservation, while weakening allows introducing new entries in a typing.

Lemma 1.

(1) (substitution) $\Gamma, \tilde{x} : S \vdash P \rhd \Delta$ *and* $\Gamma \vdash \tilde{v} : S$ *imply* $\Gamma \vdash P\{\tilde{v}/\tilde{x}\} \rhd \Delta$.

(2) (type weakening) Whenever $\Gamma \vdash P \rhd \Delta$ *is derivable, then its weakening is also derivable, namely* $\Gamma \vdash P \rhd \Delta, \Delta'$ *for disjoint* Δ'*, where* Δ' *contains only end.*

Proof. The proof is rather standard, similar to that presented in [14]. □

We now prove that our probabilistic typing system is sound, namely its type-checking rules prove only terms that are valid with respect to both structural congruence and operational semantics. In what follows, by inverting a rule we describe how the (sub)processes of a well-typed process can be typed. This is a basic property that is used in some papers when reasoning by induction on the structure of processes (see [4] and [14], for instance).

Theorem 1 (type preservation under equivalence)**.** $\Gamma \vdash P \rhd \Delta$ *and* $P \equiv P'$ *imply* $\Gamma \vdash P' \rhd \Delta$.

Proof. The proof is by induction on $\equiv$, showing (in both ways) that if one side has a typing, then the other side has the same typing.

- Case $P \mid \mathbf{0} \equiv P$.
  $\Rightarrow$ Assume $\Gamma \vdash P \mid \mathbf{0} \rhd \Delta$. By inverting the rule (TConc), we obtain $\Gamma \vdash P \rhd \Delta_1$ and $\Gamma \vdash \mathbf{0} \rhd \Delta_2$, where $\Delta_1, \Delta_2 = \Delta$. By inverting the rule (TEnd), Δ_2 is only end and Δ_2 is such that $dom(\Delta_1) \cap dom(\Delta_2) = \emptyset$. Then, by weakening, we get that $\Gamma \vdash P \rhd \Delta$, where $\Delta = \Delta_1, \Delta_2$.
  $\Leftarrow$ Assume $\Gamma \vdash P \rhd \Delta$. By rule (TEnd), it holds that $\Gamma \vdash \mathbf{0} \rhd \Delta'$, where Δ' is only end and $dom(\Delta) \cap dom(\Delta') = \emptyset$. By applying rule (TConc), we obtain $\Gamma \vdash P \mid \mathbf{0} \rhd \Delta, \Delta'$, and for $\Delta' = \emptyset$ we obtain $\Gamma \vdash P \mid \mathbf{0} \rhd \Delta$, as required.

The remaining cases are proved in a similar manner. □

According to the following theorem, if a well-typed process takes a reduction step of any kind, the resulting process is also well-typed.

Theorem 2 (type preservation under reduction)**.**
$\Gamma \vdash P \rhd \Delta$ *and* $P \to_{p_i} P'$ *imply* $\Gamma \vdash P' \rhd \Delta'$*, where* $\Delta = \Delta'$ *or* $\Delta \Rightarrow_{\delta_i} \Delta'$ *with* $p_i \in \delta_i$.

Proof. By induction on the derivation of $P \rightarrow_{p_i} P'$. There is a case for each operational semantics rule, and for each operational semantics rule we consider each typing system rule generating $\Gamma \vdash P \rhd \Delta$.

- Case (COM): $\sum_{i\in I} p_i : s!\langle \tilde{e_i}\rangle;P_i \mid \sum_{j\in J} s?(\tilde{x_j});P_j \rightarrow_{p_i} P_i \mid P_j\{\tilde{v_i}/\tilde{x_j}\}$.

  By assumption, $\Gamma \vdash \sum_{i\in I} p_i : s!\langle \tilde{e_i}\rangle;P_i \mid \sum_{j\in J} s?(\tilde{x_j});P_j \rhd \Delta$. By inverting the rule (TCONC), we get $\Gamma \vdash \sum_{i\in I} p_i : s!\langle \tilde{e_i}\rangle;P_i \rhd \Delta_1$, $\Gamma \vdash \sum_{j\in J} s?(\tilde{x_j});P_j \rhd \Delta_2$ with $\Delta = \Delta_1,\Delta_2$. Since these can be inferred only from (TSEND) and (TRECEIVE), we know that $\Delta_1 = \Delta_1', \tilde{s} : \sum_{i\in I} \delta_i : k!\langle \tilde{S_i}\rangle;T_i@q_1$ and $\Delta_2 = \Delta_2', \tilde{s} : \sum_{j\in J} k?\langle \tilde{S_j}\rangle;T_j@q_2$. By inverting the rules (TSEND) and (TRECEIVE), we get that $\forall i.\Gamma \vdash \tilde{e_i} : \tilde{S_i}$, $\sum_{i\in I} p_i = 1$, $p_i \in \delta_i$, $\forall i.\Gamma \vdash P_i \rhd \Delta_1', \tilde{s} : T_i@q_1$ and $\forall j.\Gamma, \tilde{x_j} : \tilde{S_j} \vdash P_j \rhd \Delta_2', \tilde{s} : T_j@q_2$. Assuming that $e_i \downarrow v_i$ and knowing that $\forall i.\Gamma \vdash \tilde{e_i} : \tilde{S_i}$, it implies that $\forall i.\Gamma \vdash \tilde{v_i} : \tilde{S_i}$. From $\Gamma \vdash \tilde{v_i} : \tilde{S_i}$ and $\Gamma, \tilde{x_j} : \tilde{S_j} \vdash P_j \rhd \Delta_2', \tilde{s} : T_j@q_2$, by applying the substitution part of Lemma 1, we get that $\Gamma \vdash P_j\{v_i/x_j\} \rhd \Delta_2', \tilde{s} : T_j@q_2$. By applying the rule (TCONC), we get $\Gamma \vdash P_i \mid P_j\{v_i/x_j\} \rhd \Delta_1', \tilde{s} : T_i@q_1, \Delta_2', \tilde{s} : T_j@q_2$. Using the reduction on types, we get $\Delta \Rightarrow_{\delta_i} \Delta'$, where $\Delta' = \Delta_1', \tilde{s} : T_i@q_1, \Delta_2', \tilde{s} : T_j@q_2$ and $p_i \in \delta_i$.

- Case (DELEG): $s!\langle\langle \tilde{s}\rangle\rangle;P \mid s?((\tilde{s}));Q \rightarrow_1 P \mid Q$.

  By assumption, $\Gamma \vdash s!\langle\langle \tilde{s}\rangle\rangle;P \mid s?((\tilde{s}));Q \rhd \Delta$. By inverting the rule (TCONC), we get that $\Gamma \vdash s!\langle\langle \tilde{s}\rangle\rangle;P \rhd \Delta_1$, $\Gamma \vdash s?((\tilde{s}));Q \rhd \Delta_2$ with $\Delta = \Delta_1,\Delta_2$. Since these can be inferred only from (TSDELEG) and (TSRECEIVE), we know that $\Delta_1 = \Delta_1', \tilde{s} : k!\langle T'@q'\rangle;T@q, \tilde{t} : T'@q'$ and $\Delta_2 = \Delta_2', \tilde{s} : k?(T'@q');T''@q''$. By inverting the rules (TSDELEG) and (TSRECEIVE), we get that $\Gamma \vdash P \rhd \Delta_1', \tilde{s} : T@q$ and $\Gamma \vdash Q \rhd \Delta_2', \tilde{s} : T''@q'', \tilde{t} : T'@q'$. By applying the rule (TCONC), we get $\Gamma \vdash P \mid Q \rhd \Delta_1', \tilde{s} : T@q, \Delta_2', \tilde{s} : T''@q', \tilde{t} : T'@q''$. By using the reduction on types, we get that $\Delta \Rightarrow_1 \Delta'$, where $\Delta' = \Delta_1', \tilde{s} : T@q, \Delta_2', \tilde{s} : T''@q'', \tilde{t} : T'@q'$.

The remaining cases are proved in a similar manner. □

A corollary of the type preservation result is the probabilistic-error freedom. An error is reached when a process performs an action that violates the constraints prescribed by its type. To formulate this property of probabilistic-error freedom, we extend the syntax by including a process error, while the reduction rules for processes are extended as below. This is done to accommodate the fact that the processes with value sending and label selection in which the sum of all probabilities is different from 1 generate an error.

$\sum_{i\in I} p_i : s!\langle \tilde{e_i}\rangle;P_i \mid \sum_{j\in J} s?(\tilde{x_j} : \tilde{S_j});P_j \rightarrow_1 \mathsf{error}$	(if $\sum_{i\in I} p_i \neq 1$)	(ECOM)
$\sum_{i\in I} p_i : s \lhd l_i;P_i \mid s \rhd \{l_j : P_j\}_{j\in J} \rightarrow_1 \mathsf{error}$	(if $\sum_{i\in I} p_i \neq 1$)	(ELABEL)

Table 6: Extending Operational Semantics with Rules for error

Theorem 3 (probabilistic-error freedom)**.** *If* $\Gamma \vdash P \rhd \Delta$ *and* $P \rightarrow_{p_i} P'$*, then* $P' \neq \mathsf{error}$.

Proof. We assume that $P \neq \mathsf{error}$, and proceed by case analysis on the reduction $P \rightarrow_{p_i} P'$. If the last reduction is by one of the rules of Table 2 then $P' \neq \mathsf{error}$ since these rules do not introduce error processes. Also, by using Theorem 2, we are able to show that $\Gamma \vdash P' \rhd \Delta'$ for some Δ' (obtained by some reduction from Δ).

The only reductions introducing error processes are provided by the rules of Table 6. We consider only one case (as the other is treated in a similar manner). Consider the rule (ECOM) applied to P having the form $\sum_{i\in I} p_i : s!\langle \tilde{e}_i\rangle; P_i \mid \sum_{j\in J} s?(\tilde{x_j} : \tilde{S_j}); P_j$. Then by (ECOM) we have $\sum_{i\in I} p_i \neq 1$. By hypothesis, P is well-typed. By using the typing rules (TSEND) and (TRECEIVE) of Table 5, process P can be typed by using the condition $\sum_{i\in I} p_i = 1$ which contradicts the fact that rule (ECOM) can be applied. The fact that none of the reductions introducing errors can be applied means that the result holds. □

By the correspondence between local types and global types given in Section 3.2, these results guarantee that interactions between typed processes follow exactly the interactions specified in a global type.

5 Conclusion

We have defined and studied a typing system extending the (synchronous version of) multiparty session types to deal also with probabilistic and nondeterministic choices. We proposed a process calculus considering both the probabilistic internal choices (sending a value and selecting a label) with the nondeterministic external choices (receiving a value and branching a process by using a selected value). We used a system inspired from the synchronous calculus presented in [3], but avoiding the use (and typing) of queues presented in [3]. The calculus from [3] has been modified in [6] and [17] by using channels with roles, and so eliminating the need to use the notation $T@q$ for delegation. However, we feel that this notation for delegation makes the rules easier to read; thus, we keep it in our typing system.

The approach presented in this paper has attractive properties and features. It retains the classical approach (type system), and it is specified in such a way to satisfy the axioms of a standard probability theory for computing the probability of a behaviour. As far as we know, in the field of session types there is no other related work.

Several formal tools have been proposed for probabilistic reasoning. Some approaches concern the use of probabilistic logics. In [5], terms are assigned probabilistically to types via probabilistic type judgements, and from an intuitionistic typing system is derived a probabilistic logic as a subsytem [21].

In [20] there are proposed two semantics of a probabilistic variant of the π-calculus. For these, the types are used to identify a class of nondeterministic probabilistic behaviours which can preserve the compositionality of the parallel operator in the framework of event structures. The authors claim to perform an initial step towards a good typing discipline for probabilistic name passing by employing Segala automata [18] and probabilistic event structures. In comparison with them, we simplify the approach and work directly with processes, giving a probabilistic typing in the context of multiparty session types.

References

[1] A. Aldini & M. Bravetti (2000): *An Asynchronous Calculus for Generative-Reactive Probabilistic Systems*. Technical Report UBLCS-2000-3, University of Bologna. Available at `https://disi.unibo.it/it/ricerca/technical-report/2000/pdfs/2000-03.ps.gz`.

[2] S. Andova (1999): *Process Algebra with Probabilistic Choice*. In: *Formal Methods for Real-Time and Probabilistic Systems, 5th International AMAST Workshop, ARTS'99, Bamberg, Germany, May 26-28, 1999. Proceedings*, pp. 111–129, doi:10.1007/3-540-48778-6_7.

[3] A. Bejleri & N. Yoshida (2009): *Synchronous Multiparty Session Types*. *Electronic Notes in Theoretical Computer Science* 241, pp. 3–33, doi:10.1016/j.entcs.2009.06.002.

[4] L. Bocchi, W. Yang & N. Yoshida (2014): *Timed Multiparty Session Types*. In: *CONCUR 2014 - Concurrency Theory - 25th International Conference, Rome, Italy, September 2-5, 2014. Proceedings*, pp. 419–434, doi:10.1007/978-3-662-44584-6_29.

[5] R. Cooper, S. Dobnik, S. Lappin & S. Larsson (2014): *A Probabilistic Rich Type Theory for Semantic Interpretation*. In: *Proceedings of the EACL 2014 Workshop on Type Theory and Natural Language Semantics (TTNLS)*, Association for Computational Linguistics, Gothenburg, Sweden, pp. 72–79, doi:10.3115/v1/W14-1409.

[6] M. Coppo, M. Dezani-Ciancaglini, N. Yoshida & L. Padovani (2016): *Global Progress for Dynamically Interleaved Multiparty Sessions*. *Mathematical Structures in Computer Science* 26(2), pp. 238–302, doi:10.1017/S0960129514000188.

[7] Y. Deng (2015): *Semantics of Probabilistic Processes: An Operational Approach*. Springer Publishing Company, Incorporated, doi:10.1007/978-3-662-45198-4.

[8] R. J. van Glabbeek, S. A. Smolka & B. Steffen (1995): *Reactive, Generative and Stratified Models of Probabilistic Processes*. *Information and Computation* 121(1), pp. 59–80, doi:10.1006/inco.1995.1123.

[9] J. Y. Halpern (2003): *Reasoning About Uncertainty*. MIT Press, Cambridge, MA, USA.

[10] H. A. Hansson (1994): *Time and Probability in Formal Design of Distributed Systems*. Elsevier Science Inc., New York, NY, USA.

[11] O. M. Herescu & C. Palamidessi (2000): *Probabilistic Asynchronous π-Calculus*. In: *Foundations of Software Science and Computation Structures, Third International Conference, FOSSACS 2000, Held as Part of the Joint European Conferences on Theory and Practice of Software,ETAPS 2000, Berlin, Germany, March 25 - April 2, 2000, Proceedings*, pp. 146–160, doi:10.1007/3-540-46432-8_10.

[12] J. Hillston (1996): *A Compositional Approach to Performance Modelling*. Cambridge University Press, New York, NY, USA, doi:10.1017/CBO9780511569951.

[13] K. Honda (1993): *Types for Dyadic Interaction*. In: *CONCUR '93, 4th International Conference on Concurrency Theory, Hildesheim, Germany, August 23-26, 1993, Proceedings*, pp. 509–523, doi:10.1007/3-540-57208-2_35.

[14] K. Honda, N. Yoshida & M. Carbone (2016): *Multiparty Asynchronous Session Types*. *Journal of the ACM* 63(1), pp. 9:1–9:67, doi:10.1145/2827695.

[15] G. Lowe (1993): *Probabilities and Priorities in Timed CSP*. Ph.D. thesis, University of Oxford, UK. Available at http://ora.ox.ac.uk/objects/uuid:cfec28d9-aa50-46f3-a664-eb5fbe97b261.

[16] R. Milner (1999): *Communicating and Mobile Systems - the π-calculus*. Cambridge University Press.

[17] A. Scalas, O. Dardha, R. Hu & N. Yoshida (2017): *A Linear Decomposition of Multiparty Sessions for Safe Distributed Programming*. In: *31st European Conference on Object-Oriented Programming, ECOOP 2017, June 19-23, 2017, Barcelona, Spain*, pp. 24:1–24:31, doi:10.4230/LIPIcs.ECOOP.2017.24.

[18] R. Segala & N. A. Lynch (1995): *Probabilistic Simulations for Probabilistic Processes*. *Nordic Journal of Computing* 2(2), pp. 250–273.

[19] K. Takeuchi, K. Honda & M. Kubo (1994): *An Interaction-based Language and its Typing System*. In: *PARLE '94: Parallel Architectures and Languages Europe, 6th International PARLE Conference, Athens, Greece, July 4-8, 1994, Proceedings*, pp. 398–413, doi:10.1007/3-540-58184-7_118.

[20] D. Varacca & N. Yoshida (2007): *Probabilistic π-Calculus and Event Structures*. *Electronic Notes in Theoretical Computer Science* 190(3), pp. 147–166, doi:10.1016/j.entcs.2007.07.009.

[21] J. H. Warrell (2016): *A Probabilistic Dependent Type System based on Non-Deterministic Beta Reduction*. *CoRR* abs/1602.06420. Available at http://arxiv.org/abs/1602.06420.