Available online at www.sciencedirect.com
ScienceDirect
ELSEVIER
Electronic Notes in Theoretical Computer Science 229 (2009) 11–26
Electronic Notes in Theoretical Computer Science
www.elsevier.com/locate/entcs

MarCaSPiS: a Markovian Extension of a Calculus for Services

Rocco De Nicola

Dipartimento di Sistemi e Informatica - Università di Firenze

Diego Latella

Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo"- Consiglio Nazionale delle Ricerche

Michele Loreti

Dipartimento di Sistemi e Informatica - Università di Firenze

Mieke Massink

Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo"- Consiglio Nazionale delle Ricerche

Abstract

Service Oriented Computing (SOC) is a design paradigm that has evolved from earlier paradigms including object-orientation and component-based software engineering. Important features of services are compositionality, context-independence, encapsulation and re-usability. To support the formal design and analysis of SOC applications recently a number of Service Oriented Calculi have been proposed. Most of them are based on process algebras enriched with primitives specific of service orientation such as operators for manipulating semi-structured data, mechanisms for describing safe client-service interactions, constructors for composing possibly unreliable services and techniques for services query and discovery. In this paper we show a versatile technique for the definition of Structural Operational Semantics of MarCaSPiS, a Markovian extension of one of such calculi, namely the Calculus of Sessions and Pipelines, CaSPiS. The semantics deals in an elegant way with a stochastic version of two-party synchronisation, typical of a service-oriented approach, and with the problem of transition multiplicity while preserving highly desirable mathematical properties such as associativity and commutativity of parallel composition.
We also show how the proposed semantics can be naturally used for defining a bisimulation-based behavioural equivalence for MarCaSPiS terms that induces the same equalities as those obtained via Strong Markovian Equivalence.

Keywords: Calculi for Service Oriented Computing, Quantitative Analysis of Systems, Stochastic Process Algebras

1 Introduction

Service Oriented Computing (SOC) is a design paradigm that has evolved from earlier paradigms including object-orientation and component-based software engi-

doi:10.1016/j.entcs.2009.07.071

neering. Important features of services are compositionality, context-independence, encapsulation and re-usability. To support the formal design and analysis of SOC applications recently a number of Service Oriented Calculi have been proposed. Most of them are based on process algebras enriched with primitives specific of service orientation such as operators for manipulating semi-structured data, mechanisms for describing safe client-service interactions, constructors for composing possibly unreliable services and techniques for query and discovery of services, see for example [2,19,6,23,5,21,7,20,3]. These calculi provide clean mathematical foundations for modeling typical aspects of service oriented computing like safe service composition, service interaction and service orchestration. Besides qualitative aspects of service oriented systems it is also important that phenomena related to performance and dependability are addressed to deal with issues related to quality of service. These aspects are particularly relevant for this class of systems because often the latter rely on components interacting over networks where failures are likely and congestion may cause unpredictable delays.

Traditional performance modeling techniques such as queuing networks and stochastic variants of Petri nets are widely adopted to describe performance and dependability models. Their lack of compositionality, however, hinders their usability when dealing with complex systems consisting of many components: it turns out to be difficult, or even impossible, to "glue" together specifications of system components such that an overall model of the system is obtained. This observation has given rise to the development of stochastic extensions of *process algebras* (see e.g. [4]), since these formalisms assign great importance to compositionality.

The main contribution of this paper is the introduction of a versatile technique for the definition of structured operational semantics (SOS) of stochastic extensions of service oriented calculi that handles the problem of transition multiplicity and considers two-party CCS-like interaction which is typical for service oriented approaches. We illustrate the technique providing a Markovian extension for one of the service oriented calculi namely the *Calculus of Sessions and Pipelines* [3] (CaSPiS). CaSPiS is a core calculus where *sessions* and *pipelines* are viewed as natural constructors for structuring client-service interaction and service orchestration. The stochastic extension of CaSPiS, that we shall call MarCaSPiS, permits an integrated analysis of both qualitative and quantitative aspects of formal specifications of services. We define a formal structural operational semantics that is used as the basis for associating a Continuous Time Markov Chain (CTMC) with each MarCaSPiS specification while handling appropriately and naturally the problem of dealing with multiple transitions leading via the same action to the same state. In our approach, the transition relation associates each pair of MarCaSPiS terms and actions to a *function* from MarCaSPiS terms to transition rates. This function maps each MarCaSPiS term into the rate with which it can be reached from the term at the source of the transition, via the action.

Another aspect that differentiates our work from that, e.g., by [25,24], is the specific choice in the adaptation of the apparent rate approach proposed for a process algebra with multi-party (CSP-like) synchronisation by Hillston [16] to a calcu-

lus with *two-party* (CCS-like) synchronisation while guaranteeing *associativity and commutativity* of parallel composition.

We also show how the proposed semantics can be naturally used for defining a bisimulation-based behavioural equivalence on MarCaSPiS terms that induces the same equalities as those obtained via Strong Markovian Equivalence. We do expect that our approach is not limited to MarCaSPiS but can be used in many similar situations where stochastic extensions of calculi are needed.

The rest of the paper is organised as follows. In Section 2, we briefly recall CaSPiS and its informal semantics while in Section 3 we introduce MarCaSPiS and its stochastic semantics. In Section 4 we introduce Rate Aware Bisimulation Equivalence. Section 5 concludes the paper. Due to page limitations, for the formal proofs of theorems and lemmata of Section 3 and Section 4 we refer to [10].

2 CaSPiS: A Calculus of Sessions and Pipelines

CaSPiS *(Calculus of Sessions and Pipelines)* [3] is a core calculus where *sessions* and *pipelines* are viewed as natural tools for structuring client-service interaction and service orchestration. In the following we give a flavour of CaSPiS by means of a small running example. In CaSPiS, service definitions and invocations are written like (nullary) input and output prefixes in the π-calculus [22]. *Service definition* and *service invocation* are respectively rendered as $s.P$ and $\overline{s}.Q$, where s is the name of the service. However, differently from π-calculus, P and Q are not continuations but identify the protocols governing the interaction between client (service invoker) and server (service provider). For example:

$$\texttt{news}.\langle \text{“}news\ item\text{”}\rangle \mathbf{0} \qquad \overline{\texttt{news}}.(?x)\langle x\rangle^{\uparrow}\mathbf{0} \tag{1}$$

defines respectively a service that, once activated, sends the latest news item and a client invoking the news service, reading the news item and returning it as a result to the enclosing environment.

Synchronisation of $s.P$ and $\overline{s}.Q$ leads to the creation of a *new* session, identified by a fresh name r that can be viewed as a private, synchronous channel binding caller and callee. Since client and service may be far apart, a session naturally comes with two sides, written $r \triangleright P$ and $r \triangleright Q$. Synchronisation of client and service in (1) leads to:

$$(\nu r)\left(r \triangleright \langle \text{“}news\ item\text{”}\rangle \mathbf{0} | r \triangleright (?x)\langle x\rangle^{\uparrow}\mathbf{0}\right) \tag{2}$$

In CaSPiS, like in π-calculus, names can be restricted. Indeed, $(\nu n)P$ identifies a process P that uses a *private* name n. Standard rules for extrusion of private names are used for handling passing of private names among processes.

Processes at the two sides of a session can interact with each other by means of *concretions* ($\langle V\rangle P$) and *abstractions* ($(F)P$): the former *produce* a value V while the latter *read* a value matching pattern F. Values produced by P via concretions can be consumed (synchronously) by abstractions in Q, and vice-versa: this permits the description of interaction patterns more complex than the usual *one-way* and

request-response (provided e.g. by Orc [23] and Web Service technologies). Rules governing creation and scoping of sessions are based on those of the restriction operator in the π-calculus. Values consist of expressions composed of *constructors*, from a given signature Σ, and *names* $x, y, ..., u, v, ...$ Constructors $f, f', ...$ are equipped with an integer arity, while names play the role of variables or basic values depending on the context. Richer languages of expressions, comprising specific data values and evaluation mechanisms, are easy to accommodate. Patterns (F) can also contain variables $(?x)$, i.e. $(F)P$ can only read those values V matching pattern F. This leads to a substitution σ such that $match(F, V) = \sigma$. Here, the *pattern-matching* function *match* is defined as expected: $match(F, V) = \sigma$, if σ is the (only) substitution such that $\mathrm{dom}(\sigma) = \mathrm{bn}(F)$ and $F\sigma = V$ (where $?x\sigma = \sigma(x)$).

For the process in (2) the expression $(\nu r)\left(r \rhd \langle \text{“}news\ item\text{”}\rangle\mathbf{0} | r \rhd (?x)\langle x\rangle^{\uparrow}\mathbf{0}\right)$ eveolves to $(\nu r)\left(r \rhd \mathbf{0} | r \rhd \langle \text{“}news\ item\text{”}\rangle^{\uparrow}\mathbf{0}\right)$. The remaining activity will then be performed by the client-side of the session: $r \rhd \langle \text{“}news\ item\text{”}\rangle^{\uparrow}\mathbf{0}$ will emit “*news item*” outside the session on the client side, becoming the inert process $r \rhd \mathbf{0}$ (as already happened to the service side). In fact, values can be returned outside a session to the enclosing environment using the return operator, $\langle\cdot\rangle^{\uparrow}$. These values can be sent over sessions, or used to invoke other services, to start new activities. This is achieved using the *pipeline* operator, written:

$$P > Q\,.$$

Here, a *new* instance of process Q is activated each time P emits a value that Q can consume. Such new instance runs in parallel with $P' > Q$, where P' is the continuation of P. For instance, what follows is a client that invokes the service `news` and then uses the obtained news to invoke the service `emailMe`:

$$\overline{\texttt{news}}.(?y)\langle y\rangle^{\uparrow}\mathbf{0} > (?z)\overline{\texttt{emailMe}}.\langle z\rangle\mathbf{0}$$

When a next news item V is received from service `news` and it is returned by the client as above, the following configuration will be reached:

$$\left(r \rhd \mathbf{0} > (?z)\overline{\texttt{emailMe}}.\langle z\rangle\mathbf{0}\right) | \overline{\texttt{emailMe}}.\langle V\rangle\mathbf{0}$$

Within sessions, it is also possible to invoke services, thus giving rise to hierarchies of nested sessions, like $r_1 \rhd (r_2 \rhd P_2 | r_3 \rhd P_3)$.

Iterative behaviour of CaSPiS processes is modelled by means of *recursion* [1]. Intuitively, process **rec** $X.P$ behaves like $P[\mathbf{rec}\ X.P/X]$. The example above can be rewritten as follows:

$$\texttt{news}.\mathbf{rec}\ X.\langle \text{“}news\ item\text{”}\rangle X \qquad \overline{\texttt{news}}.\mathbf{rec}\ Y.(?y)\langle y\rangle^{\uparrow}Y > (?z)\overline{\texttt{emailMe}}.\langle z\rangle \quad (3)$$

After its invocation service `news` continuously sends “*news items*” to the client that

[1] In the original version of CaSPiS presented in [3] standard π calculus replication is used for modelling iteration. It is a common practice to use recursion for modeling iteration in stochastic and probabilistic calculi, see for instance [14] and [25]

will send a mail for each of the received elements. The formal syntax and semantics of CaSPiS can be found in [3].

3 MarCaSPiS: Markovian CaSPiS

In this section we propose a Markovian extension of CaSPiS (called MarCaSPiS). In MarCaSPiS, each *output activity* (service invocation, concretion and return) is equipped with a parameter (a *rate*, $\lambda \in \mathbb{R}^+$) characterising a random variable with a negative exponential distribution, modeling the duration of the activity. Furthermore, each *input activity* (service definition and abstractions) is annotated with a *weight* ($\omega \in \mathbb{N}^+$): a positive integer that will be used for determining the probability that the specific input is selected when a complementary output is executed.

This choice assigns an *active* role to output actions and a *passive* one to input actions. Passive actions have been introduced in stochastic process algebras (see e.g. [16]) to model server activities which can handle any request rate. In this paper we follow the above mentioned approach; in Section 5 we shall see that other choices can be easily accommodated using our approach. The one we are presenting in this paper has been chosen because, in our opinion, it naturally fits with the SOC paradigm.

The stochastic operational semantics of MarCaSPiS is defined by a transition relation associating to each process P and transition label α a function (denoted by $\mathscr{P}$, $\mathscr{Q}$,...) that maps each MarCaSPiS process to a non negative real number. Basically, $P \xrightarrow{\alpha} \mathscr{P}$ and $\mathscr{P}(Q) = v \in \mathbb{R}_{>0}$ means that Q is reachable from P via the execution of α with rate/weight v; on the other hand, $\mathscr{P}(Q) = 0$ means that Q is not reachable from P via α. In the sequel, we will use $[P_1 \mapsto v_1, \ldots, P_n \mapsto v_n]$ to denote a function associating v_i to P_i and 0 to all the other processes.

As an appetiser to our approach to stochastic semantics, let us consider the parallel composition of processes $P = \overline{s}^{\lambda_1}.P_1 + \overline{s}^{\lambda_2}.P_2$ and $Q = s^{\omega_1}.Q_1 + s^{\omega_2}.Q_2$. P models a process that can invoke service s either with protocol P_1 or with protocol P_2 and the duration of these invocations is determined by rates λ_1 and λ_2. In Q there are two definitions for service s: one with weight ω_1 and the other with weight ω_2. These weights model the probability of selecting each of the available definitions when service s is invoked.

The transitions corresponding to the possible synchronisations of $P|Q$ on service s, labeled by $\overleftrightarrow{s}$, are the following:

$$P|Q \xrightarrow{\overleftrightarrow{s}} \left[\begin{array}{l} ((\nu r) r \rhd P_1 | r \rhd Q_1) \mapsto \frac{\omega_1}{\omega_1+\omega_2} \cdot \lambda_1, ((\nu r) r \rhd P_1 | r \rhd Q_2) \mapsto \frac{\omega_2}{\omega_1+\omega_2} \cdot \lambda_1, \\ ((\nu r) r \rhd P_2 | r \rhd Q_1) \mapsto \frac{\omega_1}{\omega_1+\omega_2} \cdot \lambda_2, ((\nu r) r \rhd P_2 | r \rhd Q_2) \mapsto \frac{\omega_2}{\omega_1+\omega_2} \cdot \lambda_2 \end{array} \right] \quad (4)$$

The sum of the rates of the above transitions is $\lambda_1 + \lambda_2$, which is the *total rate of invocation* of s in $P|Q$, while the actual rate of each transition is determined by multiplying the rate of output actions (λ_1 and λ_2) by the probability of selecting the pairing input action. This probability is defined as the weight of the selected action divided by the total weight of the same actions in the system ($\frac{\omega_1}{\omega_1+\omega_2}$ and $\frac{\omega_2}{\omega_1+\omega_2}$).

$$
\begin{array}{rcll}
P,Q & ::= & A & \text{Guarded Sum} \\
 & | & D & \text{Service Definitions} \\
 & | & I & \text{Service Invocations} \\
 & | & r \rhd P & \text{Session} \\
 & | & P > Q & \text{Pipeline} \\
 & | & P|Q & \text{Parallel Composition} \\
 & | & (\nu n)P & \text{Restriction} \\
 & | & \mathbf{rec}\ X.P & \text{Recursion} \\
 & | & X & \text{Process Variable} \\
V & ::= & u \mid f(\tilde{V}) & (f \in \Sigma)
\end{array}
\qquad
\begin{array}{rcll}
\pi & ::= & (F)_\omega & \text{Abstraction} \\
 & | & \langle V \rangle_\lambda & \text{Concretion} \\
 & | & \langle V \rangle_\lambda^\uparrow & \text{Return} \\
A & ::= & \pi P \mid A + A & \\
D & ::= & s^\omega.P \mid D + D & \\
I & ::= & \overline{s}^\lambda.P \mid I + I & \\
 & & & \\
F & ::= & ?x \mid u \mid f(\tilde{F}) & (f \in \Sigma)
\end{array}
$$

Fig. 1. Syntax of MarCaSPiS processes.

It is important to notice that, when $P|Q$ is combined in parallel with a process R, that might provide new definitions for s, the synchronisations rates occurring in $P|Q$ have to be updated to take into account the possibility of new synchronisations. This is essential to guarantee standard properties of parallel composition like, for instance, associativity. Indeed, the rules for parallel composition will be defined in such a way that the rates of each action are always updated to consider the role of each partner of the parallel composition. Finally, the adopted semantics will guarantee that if $P \xrightarrow{\alpha} \mathscr{P}$, then total rate/weight of α in P is sum of all values in the codomain of $\mathscr{P}$ ($\sum_Q \mathscr{P}(Q)$), which will be proved finite for each α and P.

3.1 Syntax and Stochastic semantics of MarCaSPiS

The syntax of MarCaSPiS is presented in Figure 1, where operators are listed in decreasing order of precedence while $\tilde{V}$ and $\tilde{F}$ denote a sequence of values and patterns respectively.

Let $\mathcal{N}$ be a countable set of *names* ranged over by $n, n', ...$ Set $\mathcal{N}$ contains two disjoint countable sets $\mathcal{N}_{\text{srv}}$ of *service* names $s, s', ...$ and $\mathcal{N}_{\text{sess}}$ of *session* names $r, r' ...$, such that $\mathcal{N} \setminus (\mathcal{N}_{\text{srv}} \cup \mathcal{N}_{\text{sess}})$ is infinite. The set $\mathcal{N} \setminus \mathcal{N}_{\text{sess}}$ is ranged over by $x, y, ..., u, v....$

The syntax of MarCaSPiS is similar to that of CaSPiS except for CaSPiS output activities, which are enriched with by the rates of exponential distributions, and input activities, which are enriched with weights. We shall also assume that for each process $\mathbf{rec}\ X.P$, variable X is not bound in P and occurs only guarded, i.e. prefixed by π, s^ω or $\overline{s}^\lambda$. Moreover, for each process P we assume that each session name r occurs in P at most twice and that for each process $r \rhd Q$, r does not occur in Q. The set of all MarCaSPiS processes will be denoted by $\mathcal{C}$.

Structural congruence $\equiv$ is defined as the least congruence relation induced by the laws in Figure 2. This set of laws contains the structural rules for restriction from the π-calculus, plus the obvious extension of the restriction's scope extrusion law to pipelines and sessions. For each $P \in \mathcal{C}$ we let $[P]$ denote its structural congruence class and $rep([P])$ the associated representative. We abstract from the particular definition of $rep([P])$. We usually write $rep[P]$ or even P instead of $rep([P])$, the intended meaning being clear from the context.

$$
\begin{array}{rcl}
P|(\nu n)Q & \equiv & (\nu n)(P|Q) \text{ if } n \notin \mathrm{fn}(P) \\
(\nu n)(\nu m)P & \equiv & (\nu m)(\nu n)P \\
((\nu n)Q) > P & \equiv & (\nu n)(Q > P) \text{ if } n \notin \mathrm{fn}(P) \\
r \rhd (\nu n)P & \equiv & (\nu n)(r \rhd P) \text{ if } r \neq n \\
(\nu m)P & \equiv & (\nu n)P[n/m] \text{ if } n \notin \mathrm{fn}(P)
\end{array}
$$

Fig. 2. Structural congruence laws.

The operational semantics of MarCaSPiS processes is defined in terms of the labelled transition relation defined by the rules of Table 1 and Table 2. For each process P and transition label α the relation yields a function $\mathscr{P}$, called the *next state function*, associating a non-negative real value to each process in $\mathcal{C}$. If the transition label refers to an *output* activity or a *synchronisation* activity, this value has to be intended as the transition execution rate. In this case the next state function can be thought of as the row of the rate matrix that will be used for defining the Continuous Time Markov Chain associated to a process P. If the transition label refers to an *input* activity, the next state function identifies weights used to compute the probability of reaching one of the possible next configurations.

We find it convenient to introduce the following notations and operations on next state functions.

Definition 3.1 For each $\mathscr{P}, \mathscr{Q} : \mathcal{C} \to \mathbb{R}_{\geq 0}$ and P, Q in $\mathcal{C}$, $C \subseteq \mathcal{C}$, we let:

- $\mathscr{P}(C) = \sum_{P \in C} \mathscr{P}(P)$
- $\emptyset$ denote the constant 0, for each $Q \in \mathcal{C}$: $\emptyset(Q) = 0$
- $[P_0 \mapsto \lambda_0, \ldots, P_n \mapsto \lambda_n]$ $(i \neq j \to P_i \neq P_j)$ denote the function $\mathscr{P} : \mathcal{C} \to \mathbb{R}_{\geq 0}$ such that: $\mathscr{P}(P) =$ **if** $P = P_i$ **then** λ_i **else** 0.
- $(\mathscr{P} + \mathscr{Q})(P) = \mathscr{P}(P) + \mathscr{Q}(P)$
- $(\mathscr{P}|\mathscr{Q})(R) =$ **if** $R = P|Q$ **then** $\mathscr{P}(P) \cdot \mathscr{Q}(Q)$ **else** 0
- $(\frac{\mathscr{P} \cdot \omega_1}{\omega_2})(P) =$ **if** $\omega_2 \neq 0$ **then** $\frac{\mathscr{P}(P) \cdot \omega_1}{\omega_2}$ **else** 0, where $\omega_1, \omega_2 \in \mathbb{R}_{\geq 0}$
- $\oplus(\mathscr{P}) = \mathscr{P}(\mathcal{C})$
- $\mathscr{P}_{/\equiv}(P) =$ **if** $P = rep[P]$ **then** $\mathscr{P}([P])$ **else** 0

In the spirit of SOS, the transitions of a process are determined by composing transitions of its sub-processes. For this reason, a specific notation will be used for describing this composition. If *op* is a process algebra operator, and Q a process, we let $\mathscr{P}$ *op* Q be the function $\mathscr{R} : \mathcal{C} \to \mathbb{R}_{\geq 0}$ such that: $\mathscr{R}(R) =$ **if** $(R = P \; op \; Q)$ **then** $\mathscr{P}(P)$ **else** 0. The same for Q *op* $\mathscr{P}$ and *op* $\mathscr{P}$. Similarly, we let $\mathscr{P}[Q/X](R) = \sum_{\{P : P[Q/X] = R\}} \mathscr{P}(P)$.

We let $\mathcal{L}$ be the set of transition labels α having the following syntax and names:

$$
\begin{array}{rrcll}
\alpha ::= & s(r) & \mid \overline{s}(r) & & \text{(service definition/invocation)} \\
\mid & (V) & \mid \langle V \rangle & & \text{(value consumption/production)} \\
\mid & r:(V) & \mid r:\langle V \rangle & & \text{(consumption/production within session } r\text{)} \\
& & \mid \uparrow V & & \text{(value return)} \\
& & \mid \overleftrightarrow{s} & & \text{(service synchronization)} \\
& & \mid \overleftrightarrow{r} & & \text{(session synchronization)} \\
& & \mid \tau & & \text{(silent step)} \\
& & \mid (\nu n)\alpha & & \text{(open).}
\end{array}
$$

We define names $\mathrm{n}(\alpha)$, free names $\mathrm{fn}(\alpha)$ and bound names $\mathrm{bn}(\alpha)$ as expected; in particular $\mathrm{bn}(s(r)) = \mathrm{bn}(\overline{s}(r)) = \{r\}$ while $\mathrm{bn}((\nu\tilde{n})\alpha) = \{\tilde{n}\} \cup \mathrm{bn}(\alpha)$.

Rules (Out),(In), (Ret), (Inv) and (Def) describe the behaviour of concretion, abstraction, return, service definition and service invocation. For instance, rule (Out) states that expression $\langle V \rangle_\lambda P$ evolves to process P with rate λ, while (In) states that if $\sigma = match(V, F)$ then $(F)_\omega P$ evolves to $P\sigma$ with weight ω.

Rule (Sum) states that $P + Q$ can behave either like P or like Q. Moreover, rates/weights of each transition are added to take multiplicity into account. For instance:

$$
\dfrac{\dfrac{}{\langle 3 \rangle_{.5}\mathbf{0} \xrightarrow{\langle 3 \rangle} [\mathbf{0} \mapsto .5]}\,(\textsc{Out}) \qquad \dfrac{}{\langle 3 \rangle_{.75}\mathbf{0} \xrightarrow{\langle 3 \rangle} [\mathbf{0} \mapsto .75]}\,(\textsc{Out})}{\langle 3 \rangle_{.5}\mathbf{0} + \langle 3 \rangle_{.75}\mathbf{0} \xrightarrow{\langle 3 \rangle} [\mathbf{0} \mapsto 1.25]}\,(\textsc{Sum})
$$

Rules (S-Out), (S-In) and (S-Ret) respectively state that each output ($\langle V \rangle$), input ((V)) and return ($\uparrow V$) performed within a session becomes a session output ($r:\langle V \rangle$), a session input ($r:(V)$) and an output.

Rule (S-Pass) states that if P performs a transition labelled α leading to $\mathscr{P}$ and α is not an input, an output or a return, and does not contain name r, the same transition is performed by $r \rhd P$ leading to $r \rhd \mathscr{P}$. Rule (P-Pass) behaves similarly.

Rule (P-Sync) governs synchronisation within pipelines. A state R can be reached from $P > Q$ P evolves with τ to P' and $R = P' > Q$ or P produces a value V (leading to P') that Q can consume (leading to Q') and $R = (P' > Q)|Q'$. All the transitions not involving communication along the pipe are captured by $P \xrightarrow{\tau} \mathscr{P}$ while the other ones are determined by considering, for each value V, $\mathscr{P}_V$ and $\mathscr{Q}_V$ such that $P \xrightarrow{\langle V \rangle} \mathscr{P}_V$ and $Q \xrightarrow{(V)} \mathscr{Q}_V$. Hence, $(\mathscr{P}_V > Q)|\mathscr{Q}_V$ characterises all the processes reachable from $P > Q$ after a synchronisation on value V while $\oplus\mathscr{Q}_V$ is the total weight of input of V in Q. Thus, $\frac{(\mathscr{P}_V > Q)|\mathscr{Q}_V}{\oplus(\mathscr{Q}_V)}$ characterises the synchronisation rates on V. The complete synchronisation rate is finally obtained by considering all possible values V that P can generate as an output: $\Sigma_V \frac{(\mathscr{P}_V > Q)|\mathscr{Q}_V}{\oplus(\mathscr{Q}_V)}$. This value is finite and computable. Indeed, for each process P the set of values V such that $P \xrightarrow{\langle V \rangle} \mathscr{P}$ and $\mathscr{P} \neq \emptyset$ is finite (see Lemma 3.2).

Rule (Par) states that if α is not a synchronisation action ($\overleftrightarrow{s}$ or $\overleftrightarrow{r}$), the states

$$\frac{}{\langle V\rangle_\lambda P \xrightarrow{\langle V\rangle} [P \mapsto \lambda]}\ (\textsc{Out}) \qquad \frac{\sigma = match(V,F)}{(F)_\omega P \xrightarrow{(V)} [P\sigma \mapsto \omega]}\ (\textsc{In})$$

$$\frac{}{\langle V\rangle^\uparrow_\lambda P \xrightarrow{\uparrow V} [P \mapsto \lambda]}\ (\textsc{Ret}) \qquad \frac{r \notin \mathrm{fn}(P)}{\overline{s}^\lambda.P \xrightarrow{\overline{s}(r)} [r \rhd P \mapsto \lambda]}\ (\textsc{Inv})$$

$$\frac{r \notin \mathrm{fn}(P)}{s^\omega.P \xrightarrow{s(r)} [r \rhd P \mapsto \omega]}\ (\textsc{Def}) \qquad \frac{P \xrightarrow{\alpha} \mathscr{P} \quad Q \xrightarrow{\alpha} \mathscr{Q}}{P+Q \xrightarrow{\alpha} \mathscr{P}+\mathscr{Q}}\ (\textsc{Sum})$$

$$\frac{P \xrightarrow{\langle V\rangle} \mathscr{P}}{r \rhd P \xrightarrow{r:\langle V\rangle} r \rhd \mathscr{P}}\ (\text{S-}\textsc{Out}) \qquad \frac{P \xrightarrow{(V)} \mathscr{P}}{r \rhd P \xrightarrow{r:(V)} r \rhd \mathscr{P}}\ (\text{S-}\textsc{In})$$

$$\frac{P \xrightarrow{\uparrow V} \mathscr{P}}{r \rhd P \xrightarrow{\langle V\rangle} r \rhd \mathscr{P}}\ (\text{S-}\textsc{Ret}) \qquad \frac{P \xrightarrow{\alpha} \mathscr{P} \quad \alpha \neq \langle V\rangle, (V), \uparrow V \quad r \notin \mathrm{fn}(\alpha)}{r \rhd P \xrightarrow{\alpha} r \rhd \mathscr{P}}\ (\text{S-}\textsc{Pass})$$

$$\frac{P \xrightarrow{\alpha} \mathscr{P} \quad \alpha \neq \langle V\rangle, \tau}{P > Q \xrightarrow{\alpha} \mathscr{P} > Q}\ (\text{P-}\textsc{Pass}) \qquad \frac{P \xrightarrow{\tau} \mathscr{P} \quad \forall V.\left(P \xrightarrow{\langle V\rangle} \mathscr{P}_V, Q \xrightarrow{(V)} \mathscr{Q}_V\right)}{P > Q \xrightarrow{\tau} (\mathscr{P} > Q) + \Sigma_V \frac{(\mathscr{P}_V > Q) \mid \mathscr{Q}_V}{\oplus \mathscr{Q}_V}}\ (\text{P-}\textsc{Sync})$$

$$\frac{P \xrightarrow{\alpha} \mathscr{P} \quad Q \xrightarrow{\alpha} \mathscr{Q} \quad \alpha \neq \overleftarrow{s}, \overrightarrow{r}, (\nu n)\alpha'}{P|Q \xrightarrow{\alpha} \mathscr{P}|Q + P|\mathscr{Q}}\ (\textsc{Par})$$

$$\frac{P \xrightarrow{\overleftarrow{s}} \mathscr{P} \quad P \xrightarrow{s(r)} \mathscr{P}_d \quad P \xrightarrow{\overline{s}(r)} \mathscr{P}_i \quad Q \xrightarrow{\overleftarrow{s}} \mathscr{Q} \quad Q \xrightarrow{\overline{s}(r)} \mathscr{Q}_i \quad Q \xrightarrow{s(r)} \mathscr{Q}_d}{P|Q \xrightarrow{\overleftarrow{s}} \frac{\mathscr{P}|Q \cdot \oplus \mathscr{P}_d}{\oplus(\mathscr{P}_d+\mathscr{Q}_d)} + \frac{P|\mathscr{Q} \cdot \oplus \mathscr{Q}_d}{\oplus(\mathscr{P}_d+\mathscr{Q}_d)} + \frac{(\nu r)(\mathscr{P}_d|\mathscr{Q}_i)}{\oplus(\mathscr{P}_d+\mathscr{Q}_d)} + \frac{(\nu r)(\mathscr{P}_i|\mathscr{Q}_d)}{\oplus(\mathscr{P}_d+\mathscr{Q}_d)}}\ (\textsc{Call})$$

$$\frac{P \xrightarrow{\overleftarrow{r}} \mathscr{P} \quad Q \xrightarrow{\overleftarrow{r}} \mathscr{Q} \quad \forall V.\left(P \xrightarrow{r:\langle V\rangle} \mathscr{P}_{\langle V\rangle} \quad P \xrightarrow{r:(V)} \mathscr{P}_{(V)} \quad Q \xrightarrow{r:\langle V\rangle} \mathscr{Q}_{\langle V\rangle} \quad Q \xrightarrow{r:(V)} \mathscr{Q}_{(V)}\right)}{P|Q \xrightarrow{\overleftarrow{r}} \mathscr{P}|Q + P|\mathscr{Q} + \Sigma_V \frac{\mathscr{P}_{\langle V\rangle}|\mathscr{Q}_{(V)}}{\oplus \mathscr{Q}_{(V)}} + \Sigma_V \frac{\mathscr{P}_{(V)}|\mathscr{Q}_{\langle V\rangle}}{\oplus \mathscr{P}_{(V)}}}\ (\text{S-}\textsc{Sync})$$

$$\frac{P \xrightarrow{\alpha} \mathscr{P} \quad n \notin \mathrm{n}(\alpha) \quad \alpha \neq \tau}{(\nu n)P \xrightarrow{\alpha} (\nu n)\mathscr{P}}\ (\text{R-}\textsc{Pass}) \qquad \frac{P \xrightarrow{\overleftarrow{n}} \mathscr{P}_n \quad P \xrightarrow{\tau} \mathscr{P}}{(\nu n)P \xrightarrow{\tau} (\nu n)(\mathscr{P}_n + \mathscr{P})}\ (\textsc{Hide})$$

$$\frac{P \xrightarrow{\alpha} \mathscr{P} \quad \alpha = (\nu\tilde{n})\langle V\rangle, (\nu\tilde{n}) \uparrow V, (\nu\tilde{n})r : \langle V\rangle \quad n \notin \{\tilde{n}\} \quad n \in \mathrm{fn}(V)}{(\nu n)P \xrightarrow{(\nu n)\alpha} \mathscr{P}}\ (\textsc{Open})$$

$$\frac{P[\mathbf{rec}\ X.P/X] \xrightarrow{\alpha} \mathscr{P}}{\mathbf{rec}\ X.P \xrightarrow{\alpha} \mathscr{P}}\ (\textsc{Rec}) \qquad \frac{P \equiv Q \quad Q \xrightarrow{\alpha} \mathscr{P}}{P \xrightarrow{\alpha} \mathscr{P}_{/\equiv}}\ (\textsc{Struct})$$

Table 1
Stochastic Labelled Transition Relation (Part 1)

reachable from $P|Q$ via α are those reachable from P composed in parallel with Q, and the states reachable from Q, composed in parallel with P. Notice that it is crucial to have specific rules for for proving the $\emptyset$ derivation when a side of the parallel composition is not able to perform a given action (see below). The rules defined in Table 2) are then introduced for modeling the impossibility of a process to perform a given action. They state that no process is reachable from concretions, abstractions, returns, service definitions and service invocations by a transition that is not labelled by one of the following: $\langle V\rangle$, (V), $\uparrow V$, $s(r)$ or $\overline{s}(r)$ respectively. Rule (Nil) states that process $\mathbf{0}$ cannot evolve to any other process while rule (F-Res)

$$\frac{\alpha \neq \langle V\rangle \quad \mathrm{bn}(\alpha)\cap \mathrm{fn}(\langle V\rangle_\lambda P)=\emptyset}{\langle V\rangle_\lambda P \xrightarrow{\alpha} \emptyset}\ (\text{F-Out}) \qquad \frac{\alpha \neq\uparrow V \quad \mathrm{bn}(\alpha)\cap \mathrm{fn}(\langle V\rangle_\lambda P)=\emptyset}{\langle V\rangle^{\uparrow}_\lambda P \xrightarrow{\alpha} \emptyset}\ (\text{F-Ret})$$

$$\frac{\alpha = (V) \rightarrow \neg match(V,F) \quad \mathrm{bn}(\alpha)\cap \mathrm{fn}(\langle V\rangle_\lambda P)=\emptyset}{(F)_\omega P \xrightarrow{\alpha} \emptyset}\ (\text{F-In})$$

$$\frac{\alpha \neq \overline{s}(r) \quad \mathrm{bn}(\alpha)\cap \mathrm{fn}(\overline{s}^\lambda.P)=\emptyset}{\overline{s}^\lambda.P \xrightarrow{\alpha} \emptyset}\ (\text{F-Inv}) \qquad \frac{\alpha \neq s(r) \quad \mathrm{bn}(\alpha)\cap \mathrm{fn}(s^\omega.P)=\emptyset}{s^\omega.P \xrightarrow{\alpha} \emptyset}\ (\text{F-Def})$$

$$\frac{}{\mathbf{0} \xrightarrow{\alpha} \emptyset}\ (\text{Nil}) \qquad \frac{}{(\nu n)P \xrightarrow{\overleftrightarrow{n}} \emptyset}\ (\text{F-Res})$$

Table 2
Stochastic Labelled Transition Relation (Part 2)

states that no synchronisation on n can occur in $(\nu n)P$. It is easy to show that the operational semantics rules guarantee that for each $P \in \mathcal{C}$ and for each transition label α, if $fn(P) \cap bn(\alpha) = \emptyset$, then there exists $\mathscr{P}$ such that $P \xrightarrow{\alpha} \mathscr{P}$. Moreover, side conditions on free names permit avoiding unexpected binding by bound names in α.

Rules (Call) and (S-Sync) model synchronisation of parallel components. These rules are similar to (P-Sync), but they have to take into account local synchronisations occurring in P and Q in order to build the global next state function. Let us consider (Call) for the activation of service s. We first consider the simplified scenario in which no activation of service s can be performed by P or Q alone, i.e. $P \xrightarrow{\overleftrightarrow{s}} \emptyset$ and $Q \xrightarrow{\overleftrightarrow{s}} \emptyset$. In this case, as a simple generalisation of example (4) we get the rule below:

$$\frac{P \xrightarrow{s(r)} \mathscr{P}_d \quad P \xrightarrow{\overline{s}(r)} \mathscr{P}_i \quad Q \xrightarrow{\overline{s}(r)} \mathscr{Q}_i \quad Q \xrightarrow{s(r)} \mathscr{Q}_d}{P|Q \xrightarrow{\overleftrightarrow{s}} \frac{(\nu r)(\mathscr{P}_d|\mathscr{Q}_i)+(\nu r)(\mathscr{P}_i|\mathscr{Q}_d)}{\oplus(\mathscr{P}_d+\mathscr{Q}_d)}}$$

Notice that in the above rule the probabilities obtained from the weights associated to the specific synchronisations (i.e. $\mathscr{P}_d$ and $\mathscr{Q}_d$) and the total weight of the definitions for service s (i.e. $\oplus(\mathscr{P}_d + \mathscr{Q}_d)$), are combined with the relevant rates in the resulting next state function $\frac{(\nu r)(\mathscr{P}_d|\mathscr{Q}_i)+(\nu r)(\mathscr{P}_i|\mathscr{Q}_d)}{\oplus(\mathscr{P}_d+\mathscr{Q}_d)}$.

In the general case, when activation of service s can also be performed (locally) by P, i.e. $P \xrightarrow{\overleftrightarrow{s}} \mathscr{P} \neq \emptyset$, or by Q, i.e. $Q \xrightarrow{\overleftrightarrow{s}} \mathscr{Q} \neq \emptyset$, the *total* weight to be used for computing transition probabilities, *including* those modeling activations of service s local to P (Q respectively), is $\oplus(\mathscr{P}_d + \mathscr{Q}_d)$. This implies that the next state function for such local activation of service s, namely $\mathscr{P}|Q$ ($P|\mathscr{Q}$, respectively) must be first "cleaned up" of (total) weight $\oplus(\mathscr{P}_d)$ ($\oplus(\mathscr{Q}_d)$, respectively), relative to P (Q, respectively) alone.

Rule (S-Sync) is similar. However, local synchronisations over session r within P (resp. within Q) cannot take place. Rules (R-Pass), (Hide), and (Open) handle name restrictions. The first rule states that if α does not contain the restricted name, then the next state function of $(\nu n)P$ is obtained as the *restriction* of the

next state function of P. Rule (HIDE) handles synchronisations in the presence of a private name while (OPEN) handles extrusion of private names. Finally, rules (REC) and (STRUCT) are standard rules for recursion and for handling structural equivalent terms, where $\mathscr{P}/\equiv$ is the next state function obtained by summation of the rates of structurally congruent terms. Notice that Lemma 3.2 and Lemma 3.3 below guarantee finiteness and uniqueness of calculated rates.

We now present some results that guarantee tractability of the proposed operational semantics. Indeed, we show that each process can emit only a finite set of values. Moreover, each process can only evolve into a finite set of processes.

Lemma 3.2 *For each P, the set $\{V|\exists \mathscr{P}.P \xrightarrow{\langle V\rangle} \mathscr{P} \vee P \xrightarrow{\uparrow V} \mathscr{P} \wedge \mathscr{P} \neq \emptyset\}$ is finite.*

Proof. The claim easily follows by induction on the syntax of P. □

Lemma 3.3 *If $P \xrightarrow{\alpha} \mathscr{P}$ then the set $\{Q|\mathscr{P}(Q) > 0\}$ is finite.*

Proof. The claim easily follows by induction on the definition of $\longrightarrow$ by using Lemma 3.2. □

The following theorem guarantees that if we can prove two different behaviours for P, namely there exists α such that $P \xrightarrow{\alpha} \mathscr{P}$ and $P \xrightarrow{\alpha} \mathscr{Q}$, then these are the same when one only considers *representatives* of the classes of equivalent processes ($\mathscr{P}_{/\equiv} = \mathscr{Q}_{/\equiv}$).

Theorem 3.4 *For each P and α, if there exist $\mathscr{P}$ and $\mathscr{Q}$ such that $P \xrightarrow{\alpha} \mathscr{P}$ and $P \xrightarrow{\alpha} \mathscr{Q}$ then $\mathscr{P}_{/\equiv} = \mathscr{Q}_{/\equiv}$.*

Proof. The statement follows by induction on the syntax of P □

3.2 Generating Continuous Time Markov Chains

As in previous approaches to enhance calculi with stochastic aspects, see e.g. [25,24,8,9], we use transition relation $\longrightarrow$ to associate a CTMC to each MarCaSPiS specification.

A continuous-time Markov chain (CTMC) is a tuple $(S, \mathcal{R})$ where S is a countable set of states and $\mathcal{R}$ a *rate matrix* assigning non-negative values to pairs of states, such that $\sum_{s' \in S} \mathcal{R}[s, s']$ converges [2]. Intuitively, $(S, \mathcal{R})$ models a stochastic process where, for any state $s \in S$, whenever $\sum_{s' \in S} \mathcal{R}[s, s'] > 0$, the probability to take an outgoing transition from s by (continuous) time t is $1 - e^{-\sum_{s' \in S} \mathcal{R}[s,s'] \cdot t}$, i.e. the s-residence time is exponentially distributed with rate $\sum_{s' \in S} \mathcal{R}[s, s']$, and the probability to take a transition from state s to state s', given that s is left, is $\frac{\mathcal{R}(s,s')}{\sum_{s'' \in S} \mathcal{R}[s,s'']}$. If $\sum_{s' \in S} \mathcal{R}[s, s'] = 0$, then s is said to be *absorbing*, i.e. if the process enters state s, it remains in s forever. In what follows, the rate matrix function $\mathcal{R}$ of any CTMC $(S, \mathcal{R})$ is lifted to sets of states $C \subseteq S$ in the natural way: $\mathcal{R}[s, C] \stackrel{def}{=} \sum_{s' \in C} \mathcal{R}[s, s']$.

[2] Notice that this definition allows self loops in CTMC, i.e. $\mathcal{R}[s, s] > 0$ is allowed; we refer to [1] for details.

For each set of processes C, the CTMC associated to C ($CTMC[C]$) is obtained by considering the processes reachable from C by means of internal actions (τ) or service interactions ($\overleftrightarrow{s}$), where structural equivalent terms are identified.

Definition 3.5 For set $C \subseteq \mathcal{C}$, the set of derivatives of C, denoted as $Der(C)$, is the smallest set such that:

- $C \subseteq Der(C)$, and if $Q \in Der(C)$ and there exists $\mathscr{Q}$ such that $Q \xrightarrow{\tau} \mathscr{Q}$ or $\exists s.Q \xrightarrow{\overleftrightarrow{s}} \mathscr{Q}$: then $\{Q' \mid \mathscr{Q}_{/\equiv}(Q') > 0\} \subseteq Der(C)$

Definition 3.6 For $C \subseteq \mathcal{C}$, the CTMC of C, is defined as $CTMC[C] \stackrel{def}{=} (S, \mathcal{R})$ where

- $S \stackrel{def}{=} \{rep[Q] \mid Q \in Der(C)\}$
- For all $P, Q \in S$ and $s \in \mathcal{N}_{\text{srv}}$, if $\mathscr{P}^\tau$ is such that $P \xrightarrow{\tau} \mathscr{P}^\tau$ and, for each s, $\mathscr{P}^s$ is such that $P \xrightarrow{\overleftrightarrow{s}} \mathscr{P}^s$; then: $\mathcal{R}[P, Q] \stackrel{def}{=} \mathscr{P}^\tau{}_{/\equiv}(Q) + \sum_{s \in \mathcal{N}_{\text{srv}}} \mathscr{P}^s{}_{/\equiv}(Q)$

Notice that Theorem 3.4 guarantees well-definedness of $CTMC[C]$.

4 Rate aware bisimulation

One of the key concepts in the theory of process algebras is the notion of *behavioural equivalence and congruence* which permits identifying *different* processes exhibiting *similar* behaviour. These notions have been very useful for formal reasoning about processes, for minimising process representations and for replacing equivalent components with "better" ones according to specific quality criteria without changing the overall behaviour. For example, one could say that Q is "better" than P if it has fewer states, lower complexity, fewer components, clearer structure, etc.

In the literature, many behavioural equivalences have been proposed which differ in what are considered to be the essential aspects of *observable* behaviour. More recently, such behavioural equivalences have been extended to *Markovian* process algebras where "better" would refer to better performance relative to specific criteria. In this paper we focus on *Strong Markovian Bisimulation Equivalence* [16], which has a direct correspondence to the notion of *lumpability*—a successful minimisation technique—of CTMCs [16,17], and for which efficient algorithms have been devised for computing the best possible lumping [15]. We introduce *Rate Aware Bisimulation Equivalence* as the natural equivalence induced by the next state function and we show that this equivalence implies Strong Markovian Bisimulation Equivalence. We point out that the semantic approach we used in this paper makes the definition of the Rate Aware Bisimulation Equivalence a very simple one. The following definition is recalled from [4].

Definition 4.1 [Strong Markovian bisimilarity] Given generic CTMC $(S, \mathcal{R})$

- An equivalence relation $\mathcal{E}$ on S is a Markovian bisimulation on S if and only if for all $(s, s') \in \mathcal{E}$ and for all equivalence classes $C \in S_{/\mathcal{E}}$ the following condition holds: $\mathcal{R}[s, C] \leq \mathcal{R}[s', C]$.

- Two states $s, s' \in S$ are strong Markovian bisimilar, written $s \sim_M s'$, if and only if there exists a Markovian bisimulation $\mathcal{E}$ on S with $(s, s') \in \mathcal{E}$.

The definition of Rate Aware Bisimulation Equivalence follows:

Definition 4.2 [Rate Aware Bisimilarity]

- An equivalence relation $\mathcal{E}$ on $\mathcal{C}$ is a *rate aware* bisimulation if and only if, for each $(P, Q) \in \mathcal{E}$ and for each α, $\mathscr{P}$, $\mathscr{Q}$ and for all equivalence classes $C \in \mathcal{C}_{/\mathcal{E}}$:

$$P \xrightarrow{\alpha} \mathscr{P} \wedge Q \xrightarrow{\alpha} \mathscr{Q} \Longrightarrow \mathscr{P}(C) = \mathscr{Q}(C)$$

- Two processes $P, Q \in \mathcal{C}$ are *rate aware bisimilar* $(P \sim Q)$ if there exists a rate aware bisimulation $\mathcal{E}$ such that $(P, Q) \in \mathcal{E}$.

The following theorem guarantees that the proposed equivalence is a congruence for all the operators but for *abstractions*.

Theorem 4.3 *For each $P, Q \in \mathcal{C}$, if $P \sim Q$ then for each V, s, r, n and R: $\overline{s}^{\lambda}.P \sim \overline{s}^{\lambda}.Q$, $s^{\omega}.P \sim s^{\omega}.Q$; $\langle V \rangle_{\lambda} P \sim \langle V \rangle_{\lambda} Q$; $\langle V \rangle_{\lambda}^{\uparrow} P \sim \langle V \rangle_{\lambda}^{\uparrow} Q$; $r \rhd P \sim r \rhd Q$; $P + R \sim Q + R$; $(\nu n)P \sim (\nu n)Q$; $P|R \sim Q|R$.*

The following theorem guarantees that if two processes are *rate aware* equivalent processes, then the corresponding states in the generated CTMC are *strong Markovian equivalent*. Notice that the reverse is not true.

Theorem 4.4 *Let $P, Q \in \mathcal{C}$ and $(S, \mathcal{R}) = CTMC[\{P, Q\}]$: $P \sim Q \Longrightarrow rep[P] \sim_M rep[Q]$*

Proof. The theorem follows by proving that $\mathcal{E} = \{(rep[P], rep[Q]) | P \sim Q\}$ is a Markovian bisimulation. □

Rate aware bisimilarity can be used for proving associativity of parallel composition. Indeed, one has to prove that for each P, Q and R, $(P|Q)|R \sim P|(Q|R)$.

Proposition 4.5 *For each P, Q and R, $(P|Q)|R \sim P|(Q|R)$.*

Proof. The statement follows by proving that $\mathcal{E} = \{((\nu\tilde{n})((P, Q)|R), (\nu\tilde{n})(P|(Q|R)) | P, Q, R \in \mathcal{C}\}$ is a rate aware bisimulation. □

5 Variations and Conclusions

In this paper we have presented MarCaSPiS, a stochastic extension of CaSPiS (*Calculus of Sessions and Pipelines*), together with a structured operational semantics that uses a novel technique to deal with the problem of transition multiplicity. The proposed approach is somewhat reminiscent to that of Deng et al. [12] where terms of a CSP-like probabilistic process algebra are associated to a discrete probability distribution over such terms. However, differently from [12], we consider a CCS-like synchronisation in a stochastic context.

In MarCaSPiS, each *output activity* has been equipped with a rate modeling the duration of the activity. Moreover, *input activities* have been considered passive, and each of them has been assigned a weight. These weights are used in order to compute the relative probabilities of the involved input activities (service definitions and abstractions). Indeed, the actual probability of an input activity depends on the context in which it is executed. On the other hand, quantitative evaluation normally can be applied only to completely specified models where probabilities are completely none.

Even if in the present paper we have considered a synchronisation mechanism implicitly based on *active* and *passive* actions, other synchronisation patterns proposed in the literature can be easily dealt with using our approach. For instance, one could associate proper rates both to output and input actions and define the synchronisation rate as a suitable function of such rates.

For instance, if one assumes the synchronisation policy of TIPP [13], where the rate of the synchronisation of two activities with rate r_1 and r_2 is defined as the product $r_1 \cdot r_2$, then the rule (CALL) for service activation reduces to:

$$\frac{P \xrightarrow{\overleftrightarrow{s}} \mathscr{P} \quad P \xrightarrow{s(r)} \mathscr{P}_d \quad P \xrightarrow{\overline{s}(r)} \mathscr{P}_i \quad Q \xrightarrow{\overleftrightarrow{s}} \mathscr{Q} \quad Q \xrightarrow{s(r)} \mathscr{Q}_d \quad Q \xrightarrow{\overline{s}(r)} \mathscr{Q}_i}{P|Q \xrightarrow{\overleftrightarrow{s}} \mathscr{P}|Q + (\nu r)(\mathscr{P}_d|\mathscr{Q}_i) + (\nu r)(\mathscr{P}_i|\mathscr{Q}_d) + P|\mathscr{Q}}$$

Similarly, one can obtain the synchronisation policy used in PEPA [16], which is based on a CSP-like interaction pattern and on the notions of apparent and minimal rates. The adaptation consists mainly in replacing in the PEPA synchronisation rule the apparent rates relative to the two instances of the synchronisation action with the total-input and total-output rates of the two complementary actions, while keeping the principle of minimal rate. The resulting rule for service synchronisation is:

$$\frac{P \xrightarrow{\overleftrightarrow{s}} \mathscr{P} \quad P \xrightarrow{s(r)} \mathscr{P}_d \quad P \xrightarrow{\overline{s}(r)} \mathscr{P}_i \quad Q \xrightarrow{\overleftrightarrow{s}} \mathscr{Q} \quad Q \xrightarrow{s(r)} \mathscr{Q}_d \quad Q \xrightarrow{\overline{s}(r)} \mathscr{Q}_i}{P|Q \xrightarrow{\tau} \left(\frac{\mathscr{P}|Q}{w_P} + \frac{P|\mathscr{Q}}{w_Q} + \mathscr{P}_d|\mathscr{Q}_i + \mathscr{P}_i|\mathscr{Q}_d\right) \cdot w_{PQ}}$$

where $w_P = \mathcal{W}[\mathscr{P}_d, \mathscr{P}_i]$, $w_Q = \mathcal{W}[\mathscr{Q}_d, \mathscr{Q}_i]$, $w_{PQ} = \mathcal{W}[\mathscr{P}_d + \mathscr{Q}_d, \mathscr{P}_i + \mathscr{Q}_i]$, and

$$\mathcal{W}[\mathscr{F}, \mathscr{G}] \stackrel{def}{=} \frac{min(\oplus\mathscr{F}, \oplus\mathscr{G})}{\oplus\mathscr{F} \cdot \oplus\mathscr{G}}$$

We would like to emphasize that the synchronisation introduced above guarantees associativity of parallel composition, and that this is instead lost if a more direct adaptation of Hillston's approach is used and the apparent rate approach is directly applied to CCS-like interaction [18]. This is, for instance, the case of stochastic π-calculus by Priami [25]. Also in this calculus the two-party synchronisation paradigm is considered and its semantics is directly inspired by the apparent rate approach of [16]. The stochastic semantics of [25] relies on so-called proved transition systems to deal with transition multiplicity and takes into account also

rates of non-internal actions. Instead, we consider only the rates of τ-transitions for the construction of a CTMC. To a certain extent, such a line has been followed also in later works by Priami when evolutions of π-calculus such as [26,11] were considered for the stochastic modelling of biological systems. However, in that work the synchronisation mechanism differs considerably from the one proposed in our work.

Recently, a stochastic extension of COWS [24], Calculus for Orchestration of Web Services has been developed. This calculus is based on correlation sets rather than sessions and pipelines. Also stochastic semantics of COWS is based on proved transition systems.

The work presented in this paper is a first step towards definition of a set of formal tools for specifying and verifying quantitative properties of service oriented architectures. Indeed, we plan to study stochastic modal logics that permit characterising interesting properties of services and that simplifying the formalisation of stochastic properties of MarCaSPiS systems. Moreover, we also plan to develop tools that, in the spirit of [9], permit automatically verifying stochastic behaviours of MarCaSPiS processes by relying on powerful and well studied automatic tools like MRMC and PRISM.

Acknowledgement

This work has been partially funded by the EU project Sensoria (IST-2005-016004), by the EU project Resist/Faerus (IST-2006-026764), by the CNR/RSTL XXL project and by the Italian MUR project FIRB tocai.it.

References

[1] C. Baier, B. Haverkort, H. Hermanns, and J.-P. Katoen. Model-Checking Algorithms for Continuous-Time Markov Chains. *IEEE Transactions on Software Engineering*, 29(6):524–541, 2003.

[2] M. Boreale, R. Bruni, L. Caires, R. De Nicola, I. Lanese, M. Loreti, F. Martins, U. Montanari, A. Ravara, D. Sangiorgi, V. Vasconcelos, and G. Zavattaro. SCC: a service centered calculus. In *Proc. of WS-FM 2006*, volume 4184 of *Lect. Notes in Comput. Sci.*, pages 38–57. Springer, 2006.

[3] M. Boreale, R. Bruni, R. De Nicola, and M. Loreti. Sessions and pipelines for structured service programming. In *Proc. of FMOODS'08*, *Lect. Notes in Comput. Sci.*, 2008. To appear.

[4] E. Brinksma and H. Hermanns. Process algebra and markov chains. In E. Brinksma, H. Hermanns, and J.-P. Katoen, editors, *Euro Summer School on Trends in Computer Science*, volume 2090 of *Lect. Notes in Comput. Sci.*, pages 183–231. Springer, 2001.

[5] N. Busi, R. Gorrieri, C. Guidi, R. Lucchi, and G. Zavattaro. Choreography and orchestration conformance for system design. In *Proc. of COORDINATION'06*, volume 4038 of *Lect. Notes in Comput. Sci.*, pages 63–81. Springer, 2006.

[6] L. Caires and H. Viera. The conversation calculus: A model for service oriented computation. In *Proc. of ESOP'08*, volume 4960 of *Lect. Notes in Comput. Sci.*, pages 269–283. Springer, 2008.

[7] M. Carbone, K. Honda, and N. Yoshida. Structured communication-centred programming for web serices. In *ESOP '07*, volume 4421 of *Lect. Notes in Comput. Sci.*, pages 2–17. Springer, 2007.

[8] R. De Nicola, J.-P. Katoen, D. Latella, M. Loreti, and M. Massink. MoSL: A Stochastic Logic for StoKlaim. Technical report, ISTI, 2006. Available at `http://www1.isti.cnr.it/~Latella/MoSL.pdf`.

[9] R. De Nicola, J.-P. Katoen, D. Latella, M. Loreti, and M. Massink. Model checking mobile stochastic logic. *Theoretical Computer Science*, 382(1):42–70, 2007.

[10] R. De Nicola, D. Latella, M. Loreti, and M. Massink. Marcaspis: a markovian extension of a calculus for services. Technical report, 2008.

[11] P. Degano, D. Prandi, C. Priami, and P. Quaglia. Beta-binders for biological quantitative experiments. *Electronic Notes in Theoretical Computer Science - Proc. of QAPL 2006*, 164(3):101–117, 2006.

[12] Y. Deng, R. van Glabbeek, M. Hennessy, C. Morgan, and C. Zhang. Characterising testing preorders for finite probabilistic processes. In *Proc. of LICS'07*, volume 313-325. IEEE Computer Society, 2007.

[13] N. Glotz, U. Herzog, and M. Rettelbach. Multiprocessor and distributed systems design: The integration of functional specification and performance analysis using stochastic process algebras. In L. Donatiello and R. Nelson, editors, *Performaance Evaluation of Computer and Communication Systems*, volume 729 of *Lect. Notes in Comput. Sci.* Springer, 1993.

[14] O. Herescu and C. Palamidessi. Probabilistic Asynchronous π-Calculus. In J. Tiuryn, editor, *FoSSaCS 2000*, volume 1784 of *Lect. Notes in Comput. Sci.*, pages 146–160. Springer, 2000.

[15] H. Hermanns and M. Siegle. Bisimulation Algorithms for Stochastic Process Algebras and Their BDD-Based Implementation. In *ARTS*, *Lect. Notes in Comput. Sci.*, pages 244–264. Springer, 1999.

[16] J. Hillston. A compositional approach to performance modelling, 1996. Distinguished Dissertation in Computer Science. Cambridge University Press.

[17] J. Kemeny and J. Snell. *Finite Markov Chains*. Springer, 1976.

[18] B. Klin and V. Sassone. Structural operational semantics for stochastic process calculi. In *Proc. of FOSSACS 2008*, volume 4968 of *Lecture Notes in Computer Science*. Springer, 2008.

[19] I. Lanese, F. Martins, A. Ravara, and V. Vasconcelos. Disciplining orchestration and conversation in service-oriented computing. In Proc. of *SEFM '07*, pages 305–314. IEEE Computer Society Press, 2007.

[20] C. Laneve and L. Padovani. Smooth orchestrators. In *Proc. of FoSSaCS '06*, volume 3921 of *Lect. Notes in Comput. Sci.*, pages 32–46. Springer, 2006.

[21] A. Lapadula, R. Pugliese, and F. Tiezzi. A calculus for orchestration of web services. In *ESOP '07*, volume 4421 of *Lect. Notes in Comput. Sci.*, pages 33–47. Springer, 2007.

[22] R. Milner, J. Parrow, and J. Walker. A Calculus of Mobile Processes, I and II. *Information and Computation*, 100(1):1–40, 41–77, 1992.

[23] J. Misra and W. R. Cook. Computation orchestration: A basis for wide-area computing. *Journal of Software and Systems Modeling*, 6(1):83–110, 2007.

[24] D. Prandi and P. Quaglia. Stochastic COWS. In *Proc. of ICSOC '07*, volume 4749 of *LNCS*, 2007.

[25] C. Priami. Stochastic π-Calculus. *The Computer Journal*, 38(7):578–589, 1995.

[26] C. Priami and P. Quaglia. Beta binders for biological interactions. In V. Danos and V. Schächter, editors, *Proc. of CMSB 2004*, volume 3082 of *Lecture Notes in Computer Science*, pages 20–33. Springer, 2005.