Uniform Labeled Transition Systems for Nondeterministic, Probabilistic, and Stochastic Process Calculi

Marco Bernardo
Dipartimento di Scienze di Base e Fondamenti – Università di Urbino – Italy

Rocco De Nicola
IMT – Institute for Advanced Studies Lucca – Italy
Dipartimento di Sistemi e Informatica – Università di Firenze – Italy

Michele Loreti
Dipartimento di Sistemi e Informatica – Università di Firenze – Italy

Labeled transition systems are typically used to represent the behavior of nondeterministic processes, with labeled transitions defining a one-step state-to-state reachability relation. This model has been recently made more general by modifying the transition relation in such a way that it associates with any source state and transition label a reachability distribution, i.e., a function mapping each possible target state to a value of some domain that expresses the degree of one-step reachability of that target state. In this extended abstract, we show how the resulting model, called ULTRAS from Uniform Labeled TRAnsition System, can be naturally used to give semantics to a fully nondeterministic, a fully probabilistic, and a fully stochastic variant of a CSP-like process language.

1 Introduction

Process algebras are one of the most successful formalisms for modeling concurrent systems and proving their properties such as correctness, liveness or safety. After their initial success in this respect, they have also been extended to deal with properties related to performance and quality of service. Thus, process algebras have been enriched with quantitative notions of time and probabilities and integrated theories have been considered; for a comprehensive description of this approach, the reader is referred to [1]. Moreover, due to the growing interest in the analysis of shared-resource systems, stochastic variants of process algebras have also been proposed. The main aim being the integration of qualitative descriptions with those relative to performance in a single mathematical framework by building on th ecombination of labeled transition systems (LTS) and continuous-time Markov chains (CTMC).

In [9], two of the authors of the present paper, together with D. Latella and M. Massink, proposed a variant of LTS, namely *rate transition systems* (RTS), as a tool for providing semantics to some of the most representative stochastic process languages. Within LTS, the transition relation describes the evolution of a system from one state to another as determined by the execution of specific actions, thus it is a set of triples $(state, action, state)$. In contrast, within RTS the transition relation $\rightarrowtail$ associates with a given state P and a given transition label (action) a a function, say $\mathscr{P}$, mapping each term into a non-negative real number. The transition $P \overset{a}{\rightarrowtail} \mathscr{P}$ has the following meaning: if $\mathscr{P}(Q) = v$ with $v \neq 0$, then Q is reachable from P by executing a, the duration of such an execution being exponentially distributed with rate v; if $\mathscr{P}(Q) = 0$, then Q is not reachable from P via a.

RTSs have been used for providing a uniform semantic framework for modeling many of the different stochastic process languages, facilitating reasoning about them, and throwing light on their similarities as well as on their differences. In [8], we considered a limited, but representative, number of stochastic process calculi and provided the RTS semantics for (fully) stochastic process languages both based on the CSP-like, multipart interaction paradigm and on the CCS-like, two-ways interaction paradigm. Then, in [10], RTSs were extended by requiring that the domain of $\mathscr{P}$ be a generic semiring and other variants of stochastic process algebras are studied, in particular it is shown that also languages, like IML [13], that mix stochasticity and nondeterminism can be easily modeled.

In [4], we performed a further step in the direction of providing a uniform characterization of the semantics of different process calculi and introduced a more general framework than RTS, which could be instantiated to model not only stochastic process algebras but also classical process algebras, usually modelled via LTS, and other quantitative variants of process algebras that would consider time, probabilities, resources, etc.; we thus introduced ULTRAS (*Uniform Labeled* TRA*nsition Systems*). The transition relation of ULTRAS associates with a state and a given transition label a function mapping each state into an element of a generic domain D. An ULTRAS transition $(s, a, \mathscr{D})$ is written $s \xrightarrow{a} \mathscr{D}$, with $\mathscr{D}(s')$ being a D-value quantifying the degree of reachability of s' from s via the execution of a and $\mathscr{D}(s') = \bot$ meaning that s' is not reachable from s via a. By appropriately changing the domain D, different models of concurrent systems can be captured. For example, if D is the set $\mathbb{B}$ consisting of the two Boolean values $true$ and $false$ we can capture classical LTSs, while if D is the set $\mathbb{R}_{[0,1]}$ we do capture probabilistic models, and when D is the set $\mathbb{R}_{\geq 0}$ we do capture stochastically timed models.

Of course, modeling state transitions and their annotations is one of the key ingredients; however, one has also to combine single transitions to obtain computations and find out ways for determining when two states give rise to "equivalent" computation trees. To this aim, in [4] we introduced the notions of trace equivalence and bisimulation equivalence over ULTRAS. An important component of the equivalences definition is a *measure function* $\mathscr{M}_M(s, \alpha, S')$ that computes the degree of multi-step reachability of a set of target states S' from a source state s when performing computations labeled with trace α. For instance, to capture classical equivalences over nondeterministic systems, the measure yields $\top$ if there exists a computation from s to S' labeled with α and $\bot$ otherwise. As another example, to capture probabilistic equivalences, the measure yields a value in $\mathbb{R}_{[0,1]}$ that represents the probability of the set of computations labeled with α to reach a state in S' from s.

In this note, we put ULTRAS at work and use them to provide a uniform semantical description for a few (qualitative and quantitative) variants of a very simple process algebra. For the sake of simplicity, we limit our attention to a purely nondeterministic, a fully probabilistic, and a fully stochastic calculus, without allowing any interplay between nondeterminism and quantitative aspects. In our view, the three (very compact) resulting sets of operational rules give evidence of the expressive power of our approach and help in appreciating similarities and differences among the three variants of the considered process algebra.

The rest of the paper is organized as follows. In Sect. 2, we recap the basic notions of ULTRAS introduced in [4] and define three different types of behavioral equivalences over them. To the definition of trace and bisimulation equivalences already present in [4], we add the definition of testing equivalence together with the set up of the necessary testing framework that we have introduced in [5]. In Sect. 3, we show how ULTRAS can be used to provide the operational semantics of classical CSP [6] and of two of its probabilistic [17, 2] and stochastic [14] variants. Finally, Sect. 4 reports on some future work.

2 Uniform Labeled Transition Systems

The behavior of sequential, concurrent, and distributed processes can be described by means of the so called labeled transition system (LTS) model [16]. It consists of a set of states, a set of transition labels, and a transition relation. States correspond to the operational modes that processes can pass through. Labels describe the activities that processes can perform internally or use to interact with the environment. The transition relation defines process evolution as determined by the execution of specific activities and is formalized as a *state-to-state* reachability relation.

In this section, we recall from [4] a generalization of the LTS model that aims at providing a uniform framework that can be employed for defining and comparing the behavior of different types of process. In the new model, named ULTRAS from Uniform Labeled TRAnsition System, the transition relation associates with any source state and transition label a function mapping each possible target state to an element of a domain D. In other words, the state-to-state reachability relation typical of the LTS model is replaced by a *state-to-state-distribution* reachability relation. This is a consequence of the fact that the concept of next state is generalized via a function that represents a one-step reachability distribution, which expresses the degree of reachability from the current state of every possible next state.

As shown in [4], by appropriately changing the domain D we can capture different process models, in particular quantitative ones like Markov chains [18]. For example:

- If D is the support set $\mathbb{B} = \{\bot, \top\}$ of the Boolean algebra with the standard conjunction ($\wedge$) and disjunction ($\vee$) operators, then we capture classical LTS models.
- If $D = \mathbb{R}_{[0,1]}$, then we capture fully probabilistic models in the form of action-labeled discrete-time Markov chains (ADTMC).
- If $D = \mathbb{R}_{\geq 0}$, then we capture fully stochastic models in the form of action-labeled continuous-time Markov chains (ACTMC).

2.1 Definition of the Uniform Process Model

The definition of our uniform model is parameterized with respect to a complete partial order $(D, \sqsubseteq)$ whose elements express the degree of *one-step* reachability of a state. In the following, we denote by $\bot$ the $\sqsubseteq$-least element of D and by $[S \rightarrow D]$ the set of functions from a set S to D, which is ranged over by $\mathcal{D}$.

Definition 2.1 Let $(D, \sqsubseteq)$ be a complete partial order. A uniform labeled transition system on $(D, \sqsubseteq)$, or D-ULTRAS for short, is a triple $\mathcal{U} = (S, A, \longrightarrow)$ where:

- S is an at most countable set of states.
- A is a countable set of transition-labeling actions.
- $\longrightarrow \subseteq S \times A \times [S \rightarrow D]$ is a transition relation.

We say that the D-ULTRAS $\mathcal{U}$ is functional iff $\longrightarrow$ is a function from $S \times A$ to $[S \rightarrow D]$. ■

Every transition $(s, a, \mathcal{D})$ is written $s \xrightarrow{a} \mathcal{D}$, with $\mathcal{D}(s')$ being a D-value quantifying the degree of reachability of s' from s via the execution of a and $\mathcal{D}(s') = \bot$ meaning that s' is not reachable from s via a. When considering a functional ULTRAS, we will often write $\mathcal{D}_{s,a}(s')$ to denote the same D-value.

2.2 Behavioral Equivalences for the ULTRAS Model

LTS-based models come equipped with equivalences through which it is possible to compare processes on the basis of their behavior and reduce the state space of a process before analyzing its properties. These behavioral equivalences result in a linear-time/branching-time spectrum [11, 15, 3, 1] including several variants of three major approaches: bisimulation [12], trace [6], and testing [7]. We now recall how bisimulation, trace, and testing equivalences can be uniformly defined over the ULTRAS model. Their definition is parameterized with respect to a measure function that expresses the degree of *multi-step* reachability of a set of states. Similar to the one-step reachability encoded within an ULTRAS, in which we consider individual actions, multi-step reachability relies on sequences of actions commonly called traces, which are the observable effects of the computations performed by an ULTRAS.

Definition 2.2 Let A be a countable set of transition-labeling actions. A trace α is an element of A^*, where $\alpha = \varepsilon$ denotes the empty trace. ■

Definition 2.3 Let $\mathscr{U} = (S, A, \longrightarrow)$ be a D-ULTRAS and $(M, \oplus, \otimes)$ be a lattice. An M-measure function for $\mathscr{U}$ is a function $\mathscr{M}_M : S \times A^* \times 2^S \to M$. ■

Note that different measure functions can induce different variants of a behavioral equivalence on the same D-ULTRAS depending on the support set and the operations of $(M, \oplus, \otimes)$. Although D and M may be the same support set, this is not necessarily the case: while D-values are related to one-step reachability, M-values – especially those of the form $\mathscr{M}_M(s, \alpha, S')$ – are computed on the basis of D-values to quantify multi-step reachability.

2.2.1 Trace Equivalence

Trace equivalence is straightforward: two states are trace equivalent if every trace has the same measure with respect to the entire set of states when starting from those two states.

Definition 2.4 Let $\mathscr{U} = (S, A, \longrightarrow)$ be a D-ULTRAS and $\mathscr{M}_M$ be an M-measure function for $\mathscr{U}$. We say that $s_1, s_2 \in S$ are $\mathscr{M}_M$-trace equivalent, written $s_1 \sim_{\mathrm{Tr}, \mathscr{M}_M} s_2$, iff for all traces $\alpha \in A^*$:

$$\mathscr{M}_M(s_1, \alpha, S) = \mathscr{M}_M(s_2, \alpha, S)$$

■

2.2.2 Bisimulation Equivalence

While trace equivalence simply compares any two states without taking into account the states reached at the end of the trace, bisimulation equivalence also poses constraints on the reached states.

Definition 2.5 Let $\mathscr{U} = (S, A, \longrightarrow)$ be a D-ULTRAS and $\mathscr{M}_M$ be an M-measure function for $\mathscr{U}$. An equivalence relation $\mathscr{B}$ over S is an $\mathscr{M}_M$-bisimulation iff, whenever $(s_1, s_2) \in \mathscr{B}$, then for all traces $\alpha \in A^*$ and equivalence classes $C \in S/\mathscr{B}$:

$$\mathscr{M}_M(s_1, \alpha, C) = \mathscr{M}_M(s_2, \alpha, C)$$

We say that $s_1, s_2 \in S$ are $\mathscr{M}_M$-bisimilar, written $s_1 \sim_{\mathrm{B}, \mathscr{M}_M} s_2$, iff there exists an $\mathscr{M}_M$-bisimulation $\mathscr{B}$ over S such that $(s_1, s_2) \in \mathscr{B}$. ■

2.2.3 Testing Equivalence

The definition of testing equivalence requires the formalization of the notion of test and the consideration of configurations rather than simple states. A test specifies which actions of a process are permitted at each step and can be expressed as some suitable ULTRAS that includes a success state, which is used to determine which ones are the successful computations.

Definition 2.6 Let $(D, \sqsubseteq)$ be a complete partial order. A D-ULTRAS $\mathcal{O} = (O, A, \longrightarrow)$ where O contains a distinguished success state denoted by ω such that, whenever $\omega \xrightarrow{a} \mathcal{D}$, then $\mathcal{D}(o) = \bot$ for all $o \in O$. We say that a computation of $\mathcal{O}$ is successful iff its length is finite and its last state is ω. ■

A D-ULTRAS can be tested only through a D-observation system by running them in parallel and enforcing synchronization on any action. The states of the resulting D-ULTRAS are called configurations and are pairs each formed by a state of the D-ULTRAS under test and a state of the D-observation system. A configuration can evolve to a new configuration only through the synchronization of two transitions – leaving the two states constituting the configuration – that are labeled with the same action and reach at least one state, i.e., two identically labeled transitions whose target functions are not identically equal to $\bot$.

For each such pair of synchronizing transitions, the target function of the resulting transition is obtained from the two original target functions by means of some D-valued function δ, which computes the degree of one-step reachability of every possible target configuration. Since $\bot$ represents unreachability, the only constraint on δ is that it is $\bot$-preserving, i.e., that it yields $\bot$ iff at least one of its arguments is $\bot$. As a consequence of this constraint, in the case of nondeterministic processes δ boils down to logical conjunction, whereas several alternative options are available in the case of probabilistic and stochastic processes.

Definition 2.7 Let $\mathcal{U} = (S, A, \longrightarrow_{\mathcal{U}})$ be a D-ULTRAS, $\mathcal{O} = (O, A, \longrightarrow_{\mathcal{O}})$ be a D-observation system, and δ be a $\bot$-preserving D-valued function. The interaction system of $\mathcal{U}$ and $\mathcal{O}$ with respect to δ is the D-ULTRAS $\mathcal{I}^{\delta}(\mathcal{U}, \mathcal{O}) = (S \times O, A, \longrightarrow)$ where:

- Every element $(s, o) \in S \times O$ is called a configuration and is said to be successful iff $o = \omega$. We denote by $\mathcal{S}^{\delta}(\mathcal{U}, \mathcal{O})$ the set of successful configurations of $\mathcal{I}^{\delta}(\mathcal{U}, \mathcal{O})$.
- The transition relation $\longrightarrow \subseteq (S \times O) \times A \times [(S \times O) \to D]$ is such that $(s, o) \xrightarrow{a} \mathcal{D}$ iff $s \xrightarrow{a}_{\mathcal{U}} \mathcal{D}_1$ and $o \xrightarrow{a}_{\mathcal{O}} \mathcal{D}_2$ with $\mathcal{D}(s', o')$ being obtained from $\mathcal{D}_1(s')$ and $\mathcal{D}_2(o')$ by applying δ. We say that a computation of $\mathcal{I}^{\delta}(\mathcal{U}, \mathcal{O})$ is successful iff its length is finite and its last configuration is successful. ■

Definition 2.8 Let $\mathcal{U} = (S, A, \longrightarrow_{\mathcal{U}})$ be a D-ULTRAS, $\mathcal{M}_M$ be an M-measure function for $\mathcal{U}$, δ be a $\bot$-preserving D-valued function, and $\mathcal{O} = (O, A, \longrightarrow_{\mathcal{O}})$ be a D-observation system. The extension of $\mathcal{M}_M$ to $\mathcal{I}^{\delta}(\mathcal{U}, \mathcal{O})$ is the function $\mathcal{M}_M^{\delta,\mathcal{O}} : (S \times O) \times A^* \times 2^{S \times O} \to M$ whose definition is obtained from that of $\mathcal{M}_M$ by replacing states and transitions of $\mathcal{U}$ with configurations and transitions of $\mathcal{I}^{\delta}(\mathcal{U}, \mathcal{O})$. ■

Definition 2.9 Let $\mathcal{U} = (S, A, \longrightarrow_{\mathcal{U}})$ be a D-ULTRAS, $\mathcal{M}_M$ be an M-measure function for $\mathcal{U}$, and δ be a $\bot$-preserving D-valued function. We say that $s_1, s_2 \in S$ are $\mathcal{M}_M^{\delta}$-testing equivalent, written $s_1 \sim_{\mathrm{T}, \mathcal{M}_M^{\delta}} s_2$, iff for all D-observation systems $\mathcal{O} = (O, A, \longrightarrow_{\mathcal{O}})$ with initial state $o \in O$ and for all traces $\alpha \in A^*$:

$$\mathcal{M}_M^{\delta,\mathcal{O}}((s_1, o), \alpha, \mathcal{S}^{\delta}(\mathcal{U}, \mathcal{O})) = \mathcal{M}_M^{\delta,\mathcal{O}}((s_2, o), \alpha, \mathcal{S}^{\delta}(\mathcal{U}, \mathcal{O}))$$

■

3 ULTRAS in Use: Three Experiments with CSP

In this section, we show that the ULTRAS formalism can be used for providing operational models of different kinds of process algebra. In particular, we will see how operational semantics of the language of Communicating Sequential Processes (CSP) [6] and two of its variants, which respectively extend the calculus with probabilistic binary operators and exponentially timed actions, can be easily described within the ULTRAS model by appropriately instantiating the domain D.

First, we introduce the syntax of the nondeterministic language and its operational semantics in terms of ULTRAS. For the sake of simplicity, we only consider a kernel of CSP and omit some operators, like hiding and renaming, because their treatment would add very little to the message we wish to convey. Then, we focus on the probabilistic and stochastic variants of the kernel of CSP by exhibiting a suitable ULTRAS-based operational semantics for each of them.

3.1 $\mathbb{B}$-ULTRAS Semantics for a Kernel of CSP

In CSP, systems are described as interactions of components that may engage in activities. Components reflect the behavior of the important parts of a system, while activities capture the actions that the components perform. The choice among the activities that are enabled in each system state is nondeterministic.

Let A be a countable set of activities. We denote by $\mathbb{P}_{\mathrm{CSP}}$ the set of process terms defined according to the following grammar:

$$\boxed{P ::= a.P \mid P + P \mid P \|_L P \mid B}$$

where $a \in A$, $L \subseteq A$, and B is a behavioral constant defined by an appropriate equation of the form $A \overset{\Delta}{=} P$ for some process term P in which constants occur only guarded in P, i.e., inside the scope of an action prefix. Component $a.P$ models a process that performs activity a and then behaves like P. Component $P_1 + P_2$ models a process that may behave either as P_1 or as P_2. The operator $P_1 \|_L P_2$ models instead the parallel execution of P_1 and P_2, which synchronize (or cooperate) on every activity in L and proceed independently on every activity not in L. The behavior of constant B is the same as that of the process term P on the right-hand side of its defining equation.

The semantics for the considered kernel of CSP can be described in terms of the following functional $\mathbb{B}$-ULTRAS:

$$(\mathbb{P}_{\mathrm{CSP}}, A, \longrightarrow)$$

whose transition relation $\longrightarrow$ is defined in Table 1. Given a transition $P \xrightarrow{a} \mathscr{D}$, intuitively we have that $\mathscr{D}(Q) = \top$ means that Q is reachable from P via an a-transition, while $\mathscr{D}(Q) = \bot$ means that it is not possible to reach Q from P by executing a.

Rule ACT states that $a.P$ evolves via a to $[P \mapsto \top]$, with the latter being the function associating $\top$ with P and $\bot$ with all the other process terms. On the contrary, $\emptyset$-ACT establishes that no state is reachable from $a.P$ by performing any action $b \neq a$. This is formalized by letting $a.P$ evolve via $b \neq a$ to $[]$, the function associating $\bot$ with each process term. Rule SUM describes nondeterministic choice: the states reachable from $P_1 + P_2$ via a are all those that can be reached either by P_1 or by P_2. Indeed, $\mathscr{D}_1 \vee \mathscr{D}_2$ denotes the function $\mathscr{D}$ such that $\mathscr{D}(Q) = \mathscr{D}_1(Q) \vee \mathscr{D}_2(Q)$ for all process terms Q.

Rules COOP and INT govern parallel composition. Rule COOP is used for computing the next-state function when a synchronization between P_1 and P_2 occurs. Whenever $P_1 \xrightarrow{a} \mathscr{D}_1$ and $P_2 \xrightarrow{a} \mathscr{D}_2$ with $a \in L$, then $P_1 \|_L P_2$ evolves via a to $\mathscr{D}_1 \|_L \mathscr{D}_2$, where $(\mathscr{D}_1 \|_L \mathscr{D}_2)(Q)$ is $\mathscr{D}_1(Q_1) \wedge \mathscr{D}_2(Q_2)$ if $Q = Q_1 \|_L Q_2$ and $\bot$ otherwise. Rule INT deals with $a \notin L$. In that case, if $P_1 \xrightarrow{a} \mathscr{D}_1$ and $P_2 \xrightarrow{a} \mathscr{D}_2$, then $P_1 \|_L P_2$ evolves via a to $(\mathscr{D}_1 \|_L P_2) \vee (P_1 \|_L \mathscr{D}_2)$, where $\mathscr{D}_1 \|_L P_2$ (resp. $P_1 \|_L \mathscr{D}_2$) denotes the function $\mathscr{D}$ such that $\mathscr{D}(Q)$ is $\mathscr{D}_1(P_1')$ (resp. $\mathscr{D}_2(P_2')$) if $Q = P_1' \|_L P_2$ (resp. $Q = P_1 \|_L P_2'$) and $\bot$ otherwise.

$$\frac{}{a.P \xrightarrow{a} [P \mapsto \top]}\ \textsc{Act} \qquad \frac{b \neq a}{a.P \xrightarrow{b} [\,]}\ \emptyset\text{-}\textsc{Act} \qquad \frac{B \triangleq P \quad P \xrightarrow{a} \mathscr{D}}{B \xrightarrow{a} \mathscr{D}}\ \textsc{Call}$$

$$\frac{P_1 \xrightarrow{a} \mathscr{D}_1 \quad P_2 \xrightarrow{a} \mathscr{D}_2}{P_1 + P_2 \xrightarrow{a} \mathscr{D}_1 \vee \mathscr{D}_2}\ \textsc{Sum}$$

$$\frac{P_1 \xrightarrow{a} \mathscr{D}_1 \quad P_2 \xrightarrow{a} \mathscr{D}_2 \quad a \in L}{P_1 \,\|_L\, P_2 \xrightarrow{a} \mathscr{D}_1 \,\|_L\, \mathscr{D}_2}\ \textsc{Coop}$$

$$\frac{P_1 \xrightarrow{a} \mathscr{D}_1 \quad P_2 \xrightarrow{a} \mathscr{D}_2 \quad a \notin L}{P_1 \,\|_L\, P_2 \xrightarrow{a} (\mathscr{D}_1 \,\|_L\, P_2) \vee (P_1 \,\|_L\, \mathscr{D}_2)}\ \textsc{Int}$$

Table 1: ULTRAS-based operational semantic rules for CSP

3.2 $\mathbb{R}_{[0,1]}$-ULTRAS Semantics for PCSP

We now consider a probabilistic variant of CSP that we call PCSP. While in CSP the next action to execute is selected nondeterministically, in PCSP it is selected according to some discrete probability distribution that can be different from state to state. Taking inspiration from [17, 2], the probabilistic calculus PCSP is obtained from CSP by decorating the alternative and parallel composition operators with a probability value $p \in \mathbb{R}_{[0,1]}$.

We denote by $\mathbb{P}_{\text{PCSP}}$ the set of process terms defined according to the following grammar:

$$\boxed{P ::= a.P \mid P +_p P \mid P \|_L{}^p P \mid B}$$

Component $P_1 +_p P_2$ models a process that, after performing an action, behaves as the continuation of P_1 with probability p or the continuation of P_2 with probability $1-p$. Similarly, in $P_1 \|_L{}^p P_2$ the value p is used to regulate the interleaving of P_1 and P_2.

The semantics for PCSP can be described in terms of the following functional $\mathbb{R}_{[0,1]}$-ULTRAS:

$$(\mathbb{P}_{\text{PCSP}}, A, \longrightarrow)$$

whose transition relation $\longrightarrow$ is defined in Table 2. Given a transition $P \xrightarrow{a} \mathscr{D}$, intuitively we have that $\mathscr{D}(Q) > 0$ means that Q is reachable from P via an a-transition with probability $\mathscr{D}(Q)$, while $\mathscr{D}(Q) = 0$ means that it is not possible to reach Q from P by executing a. Note that $\sum_Q \mathscr{D}(Q) \in \{0,1\}$.

The first three rules are identical to the first three rules of Table 1, with the difference that $[P \mapsto 1]$ denotes the function associating 1 with P and 0 with all the other process terms, while $[\,]$ denotes the function associating 0 with each process term. Rule SUM relies on the following notation:

- $\mathscr{D}_1 + \mathscr{D}_2$ denotes the function $\mathscr{D}$ such that $\mathscr{D}(Q) = \mathscr{D}_1(Q) + \mathscr{D}_2(Q)$ for all process terms Q.
- $\oplus\mathscr{D} = \sum_Q \mathscr{D}(Q)$.
- $\frac{x}{y} \cdot \mathscr{D}$ denotes the function $\mathscr{D}'$ such that $\mathscr{D}'(Q) = \frac{x}{y} \cdot \mathscr{D}'(Q)$ if $y \neq 0$ and 0 otherwise.

This rule asserts that the states reachable from $P_1 +_p P_2$ via a are obtained by aggregating according to p the probability distributions associated with P_1 and P_2 after a. When both P_1 and P_2 can perform a, i.e., $P_1 \xrightarrow{a} \mathscr{D}_1$ and $P_2 \xrightarrow{a} \mathscr{D}_2$ with $\mathscr{D}_1$ and $\mathscr{D}_2$ both different from $[\,]$, then $\oplus\mathscr{D}_1 = \oplus\mathscr{D}_2 = 1$ and hence the

$$\frac{}{a.P \xrightarrow{a} [P \mapsto 1]}\ \textsc{Act} \qquad \frac{b \neq a}{a.P \xrightarrow{b} [\,]}\ \emptyset\text{-}\textsc{Act} \qquad \frac{B \overset{\Delta}{=} P \quad P \xrightarrow{a} \mathscr{D}}{B \xrightarrow{a} \mathscr{D}}\ \textsc{Call}$$

$$\frac{P_1 \xrightarrow{a} \mathscr{D}_1 \quad P_2 \xrightarrow{a} \mathscr{D}_2}{P_1 +_p P_2 \xrightarrow{a} \frac{p \cdot \oplus \mathscr{D}_1}{p \cdot \oplus \mathscr{D}_1 + (1-p) \cdot \oplus \mathscr{D}_2} \cdot \mathscr{D}_1 + \frac{(1-p) \cdot \oplus \mathscr{D}_2}{p \cdot \oplus \mathscr{D}_1 + (1-p) \cdot \oplus \mathscr{D}_2} \cdot \mathscr{D}_2}\ \textsc{Sum}$$

$$\frac{P_1 \xrightarrow{a} \mathscr{D}_1 \quad P_2 \xrightarrow{a} \mathscr{D}_2 \quad a \in L}{P_1 \,\|_L{}^p\, P_2 \xrightarrow{a} \mathscr{D}_1 \,\|_L\, \mathscr{D}_2}\ \textsc{Coop}$$

$$\frac{P_1 \xrightarrow{a} \mathscr{D}_1 \quad P_2 \xrightarrow{a} \mathscr{D}_2 \quad a \notin L}{P_1 \,\|_L{}^p\, P_2 \xrightarrow{a} \frac{p \cdot \oplus \mathscr{D}_1}{p \cdot \oplus \mathscr{D}_1 + (1-p) \cdot \oplus \mathscr{D}_2} \cdot (\mathscr{D}_1 \,\|_L\, P_2) + \frac{(1-p) \cdot \oplus \mathscr{D}_2}{p \oplus \mathscr{D}_1 + (1-p) \cdot \oplus \mathscr{D}_2} \cdot (P_1 \,\|_L\, \mathscr{D}_2)}\ \textsc{Int}$$

Table 2: ULTRAS-based operational semantic rules for PCSP

aggregate probability distribution reduces to $p \cdot \mathscr{D}_1 + (1-p) \cdot \mathscr{D}_2$. In contrast, when $\mathscr{D}_1$ (resp. $\mathscr{D}_2$) is equal to $[\,]$, then $\oplus \mathscr{D}_1 = 0$ (resp. $\oplus \mathscr{D}_2 = 0$) and hence the aggregate probability distribution reduces to $\mathscr{D}_2$ (resp. $\mathscr{D}_1$).

Rules COOP and INT govern parallel composition. They are similar to the two corresponding rules of Table 1, with the differences that (i) in the synchronization case $(\mathscr{D}_1 \,\|_L\, \mathscr{D}_2)(Q)$ is $\mathscr{D}_1(Q_1) \cdot \mathscr{D}_2(Q_2)$ if $Q = Q_1 \,\|_L{}^p\, Q_2$ and 0 otherwise, while (ii) in the interleaving case a SUM-like aggregation based on p of the probability distributions associated with P_1 and P_2 after a comes into play.

3.3 $\mathbb{R}_{\geq 0}$-ULTRAS Semantics for PEPA

Building on [9, 8], we finally consider a stochastically timed variant of CSP called Performance Evaluation Process Algebra (PEPA) [14]. In this calculus, every action is equipped with a rate $\lambda \in \mathbb{R}_{>0}$ that uniquely characterizes the exponentially distributed random variable quantifying the duration of the action itself (the expected duration is $1/\lambda$). The choice among the actions that are enabled in each state is governed by the race policy: the action to execute is the one that samples the least duration. Therefore, (i) the sojourn time in each state is exponentially distributed with rate given by the sum of the rates of the transitions departing from that state, (ii) the execution probability of each transition is proportional to its rate, and (iii) the alternative and parallel composition operators are implicitly probabilistic.

We denote by $\mathbb{P}_{\text{PEPA}}$ the set of process terms defined according to the following grammar:

$$\boxed{P ::= (a,\lambda).P \mid P + P \mid P \|_L P \mid B}$$

Component $(a,\lambda).P$ models a process that can perform action a at rate λ and then behaves like P.

The semantics for PEPA can be described in terms of the following functional $\mathbb{R}_{\geq 0}$-ULTRAS:

$$(\mathbb{P}_{\text{PEPA}}, A, \longrightarrow)$$

whose transition relation $\longrightarrow$ is defined in Table 3. Given a transition $P \xrightarrow{a} \mathscr{D}$, intuitively we have that $\mathscr{D}(Q) > 0$ means that Q is reachable from P via an a-transition at rate $\mathscr{D}(Q)$, while $\mathscr{D}(Q) = 0$ means that it is not possible to reach Q from P by executing a.

The rules of Table 3 are similar to those of Table 2, with the differences that (i) $[P \mapsto \lambda]$ denotes the function associating λ with P and 0 with all the other process terms, (ii) no normalization is needed in

$$\frac{}{(a,\lambda).P \xrightarrow{a} [P \mapsto \lambda]}\ \textsc{Act} \qquad \frac{a \neq b}{(a,\lambda).P \xrightarrow{b} [\,]}\ \emptyset\text{-}\textsc{Act} \qquad \frac{B \overset{\Delta}{=} P \quad P \xrightarrow{a} \mathscr{D}}{B \xrightarrow{a} \mathscr{D}}\ \textsc{Call}$$

$$\frac{P_1 \xrightarrow{a} \mathscr{D}_1 \quad P_2 \xrightarrow{a} \mathscr{D}_2}{P_1 + P_2 \xrightarrow{a} \mathscr{D}_1 + \mathscr{D}_2}\ \textsc{Sum}$$

$$\frac{P_1 \xrightarrow{a} \mathscr{D}_1 \quad P_2 \xrightarrow{a} \mathscr{D}_2 \quad a \in L}{P_1 \parallel_L P_2 \xrightarrow{a} \frac{\min\{\oplus\mathscr{D}_1,\oplus\mathscr{D}_2\}}{\oplus\mathscr{D}_1\cdot\oplus\mathscr{D}_2}\cdot(\mathscr{D}_1 \parallel_L \mathscr{D}_2)}\ \textsc{Coop}$$

$$\frac{P_1 \xrightarrow{a} \mathscr{D}_1 \quad P_2 \xrightarrow{a} \mathscr{D}_2 \quad a \notin L}{P_1 \parallel_L P_2 \xrightarrow{a} (\mathscr{D}_1 \parallel_L P_2) + (P_1 \parallel_L \mathscr{D}_2)}\ \textsc{Int}$$

Table 3: ULTRAS-based operational semantic rules for PEPA

rules SUM and INT because transition rates simply sum up due to the race policy, and (iii) the multiplicative factor in rule COOP is specific to the PEPA cooperation discipline based on the slowest component.

4 Conclusions and Future Work

After recalling the ULTRAS model from [4, 5], in this paper we have extended the scope of the work done in [9, 8, 10] by exhibiting the ULTRAS-based operational semantic rules for CSP and two of its probabilistic and stochastically timed variants. These three experiments seem to indicate that the ULTRAS model naturally lends itself to be used as a compact and uniform semantic framework for different classes of process calculi.

With respect to future work, we plan to continue our experiments by using the ULTRAS model for describing the operational semantics of other process description languages of nondeterministic, probabilistic, or stochastic nature, as well as process calculi combining nondeterminism and probability or stochasticity. This should help to assess the relative expressiveness of their operators and establish general properties for the various languages. Moreover, the uniform characterization of the equivalences might help in evaluating and discerning among the many relations proposed in the literature. It would be, indeed, interesting to determine which of the existing relations can be obtained as instances of the general framework.

This study may also lead to the definition of a uniform process calculus with an ULTRAS-based operational semantics and the development of uniform axiomatizations of bisimulation, trace, and testing equivalences. From this calculus, it should be possible to retrieve the originally proposed calculi by varying the target domain and the behavioral operators. We shall also consider further options related to quantitative aspects like including quantities within actions (*integrated quantity approach*) or attaching them to traditional operators or providing specific operators for them (*orthogonal quantity approach*).

Finally, it would be interesting to see whether is is possible to build generic tools for supporting verifications that are based on the uniform model and have only to be instantiated to deal with the specific calculi.

Acknowledgment: This work has been partially supported by the EU project ASCENS 257414.

References

[1] A. Aldini, M. Bernardo & F. Corradini (2010): *A Process Algebraic Approach to Software Architecture Design*. Springer, doi:10.1007/978-1-84800-223-4.

[2] J.C.M. Baeten, J.A. Bergstra & S.A. Smolka (1995): *Axiomatizing Probabilistic Processes: ACP with Generative Probabilities*. Information and Computation 121, pp. 234–255, doi:10.1006/inco.1995.1135.

[3] C. Baier, J.-P. Katoen, H. Hermanns & V. Wolf (2005): *Comparative Branching-Time Semantics for Markov Chains*. Information and Computation 200, pp. 149–214, doi:10.1016/j.ic.2005.03.001.

[4] M. Bernardo, R. De Nicola & M. Loreti (2010): *Uniform Labeled Transition Systems for Nondeterministic, Probabilistic, and Stochastic Processes*. In: Proc. of the 5th Int. Symp. on Trustworthy Global Computing (TGC 2010), LNCS 6084, Springer, pp. 35–56, doi:10.1007/978-3-642-15640-3.

[5] M. Bernardo, R. De Nicola & M. Loreti (2011): *A Uniform Framework for Process Models and Behavioral Equivalences of Nondeterministic, Probabilistic, Stochastic, or Mixed Nature*. Submitted for journal publication.

[6] S.D. Brookes, C.A.R. Hoare & A.W. Roscoe (1984): *A Theory of Communicating Sequential Processes*. Journal of the ACM 31, pp. 560–599, doi:10.1145/828.833.

[7] R. De Nicola & M. Hennessy (1984): *Testing Equivalences for Processes*. Theoretical Computer Science 34, pp. 83–133, doi:10.1016/0304-3975(84)90113-0.

[8] R. De Nicola, D. Latella, M. Loreti & M. Massink (2009): *On a Uniform Framework for the Definition of Stochastic Process Languages*. In: Proc. of the 14th Int. Workshop on Formal Methods for Industrial Critical Systems (FMICS 2009), LNCS 5825, Springer, pp. 9–25, doi:10.1007/978-3-642-04570-7_2.

[9] R. De Nicola, D. Latella, M. Loreti & M. Massink (2009): *Rate-Based Transition Systems for Stochastic Process Calculi*. In: Proc. of the 36th Int. Coll. on Automata, Languages and Programming (ICALP 2009), LNCS 5556, Springer, pp. 435–446, doi:10.1007/978-3-642-02930-1_36.

[10] R. De Nicola, D. Latella, M. Loreti & M. Massink (2011): *State to Function Labelled Transition Systems: A Uniform Framework for Defining Stochastic Process Calculi*. Technical Report, CNR-ISTI. Available at http://puma.isti.cnr.it/download.php?DocFile=2011-TR-012_0.pdf&idcode=2011-TR-012&authority=cnr.isti&collection=cnr.isti.

[11] R.J. van Glabbeek (2001): *The Linear Time – Branching Time Spectrum I*. In: Handbook of Process Algebra, Elsevier, pp. 3–99, doi:10.1007/BFb0039066.

[12] M. Hennessy & R. Milner (1985): *Algebraic Laws for Nondeterminism and Concurrency*. Journal of the ACM 32, pp. 137–162, doi:10.1145/2455.2460.

[13] H. Hermanns (2002): *Interactive Markov Chains*. Springer, doi:10.1007/3-540-45804-2. Volume 2428 of LNCS.

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

[15] C.-C. Jou & S.A. Smolka (1990): *Equivalences, Congruences, and Complete Axiomatizations for Probabilistic Processes*. In: Proc. of the 1st Int. Conf. on Concurrency Theory (CONCUR 1990), LNCS 458, Springer, pp. 367–383, doi:10.1007/BFb0039071.

[16] R.M. Keller (1976): *Formal Verification of Parallel Programs*. Communications of the ACM 19, pp. 371–384, doi:10.1145/360248.360251.

[17] K. Seidel (1995): *Probabilistic Communicating Processes*. Theoretical Computer Science 152, pp. 219–249, doi:10.1016/0304-3975(94)00286-0.

[18] W.J. Stewart (1994): *Introduction to the Numerical Solution of Markov Chains*. Princeton University Press. Available at http://press.princeton.edu/titles/5640.html.