UNIVERSITY
OF TRENTO

DIPARTIMENTO DI INGEGNERIA E SCIENZA DELL'INFORMAZIONE

38050 Povo – Trento (Italy), Via Sommarive 14
http://www.disi.unitn.it

REVIEW ON STOCHASTIC COMPARISON RELATIONS

Stefano Schivo

February 2008

Technical Report # DISI-08-034

Review on stochastic comparison relations*

Stefano Schivo

Abstract

The report offers an overview of some stochastic simulation and bisimulation relations based on continuous-time Markov chains and on stochastic (possibly Markovian) process algebras.

1 Introduction

In the design of some systems (like communication protocols, embedded systems or multimedia systems) an equally important role is played by the functional aspects and the performance constraints. Indeed, it would be worth nothing having developed the most elegant VoIP protocol only to find out that it has high values in delay and jitter. In these cases, performance has to be taken into account as early as possible in the development process. Therefore, researchers have been evolving modelling languages which take into account both qualitative and quantitative aspects.

The most widespread languages for stochastic modelling allow for the production of Continuous Time Markov Chains (CTMCs), on which interesting properties can be easily verified. To have some degree of certainty that a model corresponds to what it is meant to represent, model checking is often used. A great problem with model checking is that it often requires to run across (nearly) all the states of a CTMC, thus requiring a lot of computational resources. In order to reduce the effort required by model checking, a common technique is to reduce CTMC state space by "lumping". Lumped states represent equivalence classes generated by the application of a partitioning relation on the state space. *Exactly lumpable* partitions have the advantage to preserve the Markov property also in the resulting aggregated state spaces, thus keeping the model simple to be solved. Still more important, models resulting from lumping can be solved in place of their respective larger models, obtaining exactly the same solutions.

Definition 1 *χ is a* strongly lumpable *partition of Markov process $X(t)$ with state space $\mathcal{S} = \{s_0, \ldots, s_N\}$ if for any $X_{[l]}, X_{[k]} \in \chi$, $s_i, s_j \in X_{[k]}$*

$$q(s_i, X_{[l]}) = q(s_j, X_{[l]})$$

where $q(s, X)$ denotes the aggregated transition rate from state s to partition X.

*This work has been partially sponsored by the project SENSORIA, IST-2005-016004.

The objective of this paper is to give an overview of some important bisimulation relations which can be applied to stochastic models in order to reduce their state space, while retaining the Markov property.

After a brief recall on Markov chains, we will start presenting some famous results by Hillston [9] regarding the kinds of comparison methods between LTS states. Then, we will show the work of Baier et al. [2], in which attention is put into characterising some notions of simulation and bisimulation on DTMC and CTMC, relating these relations with a notion of satisfiability of PCTL and CSL formulas respectively. Afterwards, we will provide an overview on what has been done on the language $\spadesuit$ [6, 7], presenting the most interesting relations defined for it. Finally, we will give a quick look over a stochastic barbed bisimulation for Stochastic Ambient Calculus [12].

2 Markov Chains

Markov chains can be divided into two main groups, depending whether they abstract on the concept of time or not: discrete-time and continuous-time. The former are also called "probabilistic", while the latter are said "stochastic".

2.1 Discrete-time Markov chains

Definition 2 *A fully probabilistic system (FPS) is a tuple $\mathcal{D} = (S, \mathbf{P}, L)$ where:*

- *S is a countable set of states*
- *$\mathbf{P} : S \times S \to [0, 1]$ is a probability matrix satisfying $\sum_{s' \in S} \mathbf{P}(s, s') \in [0, 1] \; \forall s \in S$*
- *$L : S \to 2^{AP}$ is a labelling function, assigning a set of atomic propositions $\{a_1, a_2, \ldots, a_n\} \subseteq AP$ to each state $s \in S$. In other words, we assume that the atomic propositions $a_1, a_2, \ldots, a_n$ hold in s.*

A state s is called *stochastic* if $\sum_{s' \in S} \mathbf{P}(s, s') = 1$. When on the contrary $\sum_{s' \in S} \mathbf{P}(s, s') = 0$ s is called *absorbing*. In all other cases, s is called *sub-stochastic*.

Definition 3 *A labelled discrete time Markov chain (DTMC) is an FPS where each state is either stochastic or absorbing.*

2.2 Continuous-time Markov chains

DTMCs are time-abstract, so they are appropriate when duration of changes between states is not being considered. When time is considered, we use continuous time Markov chains:

Definition 4 *A labelled continuous time Markov chain (CTMC) is a tuple $C = (S, \mathbf{R}, L)$ where*

- *S is a countable set of states*
- *$\mathbf{R} : S \times S \to \mathbb{R}_{\geqslant 0}$ such that $\sum_{s' \in S} \mathbf{R}(s, s')$ converges is a* rate matrix

- $L : S \to 2^{AP}$ *is a labelling function as before*

Taken a CTMC $C = (S, \mathbf{R}, L)$, the *exit rate* for state $s \in S$ is

$$E(s) = \sum_{s' \in S} \mathbf{R}(s, s') < \infty$$

We can compute the probability to move from state $s \in S$ to a state $s' \in S$ (such that $\mathbf{R}(s, s') > 0$) before t time units using the exponential distribution function:

$$P[s \to s' \text{ by } t \text{ time units}] = 1 - e^{-\lambda t}$$

where $\lambda = \mathbf{R}(s, s')$.
The probability that the transition $s \to s'$ is chosen between all transitions exiting from s (that is, the probability that the delay for going from s to s' "finishes before" any other delay of going from s to another state $s'' \neq s'$) is

$$P[s \to s'] = \frac{\mathbf{R}(s, s')}{E(s)}$$

From this kind of probabilities, the time-abstract (discrete) version of a CTMC can be derived:

Definition 5 *The* embedded DTMC *of CTMC* $C = (S, \mathbf{R}, L)$ *is* $emb(C) = (S, \mathbf{P}, L)$*, where*

$$\mathbf{P}(s, s') = \begin{cases} \frac{\mathbf{R}(s,s')}{E(s)} & \text{if } E(s) > 0 \\ 0 & \text{otherwise} \end{cases}$$

Definition 6 *A CTMC* C *is* uniformized *if* $E(s) = E(s') \; \forall s, s' \in S$

Any CTMC can be transformed into its uniformized version simply by adding self-loops (i.e., transitions such that $\mathbf{R}(s, s) > 0$):

Definition 7 *Let* $C = (S, \mathbf{R}, L)$ *be a CTMC and let* $\mathbf{e}$ *be a real such that* $\mathbf{e} \geqslant max_{s \in S} E(s)$*. Then* $unif(C) = (S, \mathbf{R}', L)$ *is a uniformized CTMC in which*

$$\mathbf{R}'(s, s') = \begin{cases} \mathbf{R}(s, s') & \text{if } s \neq s' \\ \mathbf{R}(s, s) + \mathbf{e} - E(s) & \text{otherwise} \end{cases}$$

3 PEPA: isomorphism, strong bisimilarity and strong equivalence

The PEPA language [9] has been developed with the intention to add the power and user-friendliness of compositional modelling (typical of process algebra) to the widely known tools for performance evaluation: (unlabelled) continuous time Markov chains (CTMC). From PEPA models CTMCs can be generated, and on these all classical methods for performance evaluation can be applied.

Any non-trivial model has tends to generate a large number of states in the Markov chain: for this reason, PEPA has been provided with some lumping relations. We will analyze all four relations introduced in [9], dwelling more on the most important two, strong bisimilarity and strong equivalence.

3.1 Strong and weak isomorphism

Definition 8 *Given two PEPA processes P and Q, a function $\mathcal{F} : ds(P) \to ds(Q)$ is a* component isomorphism *between P and Q if $\mathcal{F}$ is an injective function, and for any component P', $\mathcal{A}ct(P') = \mathcal{A}ct(\mathcal{F}(P'))$, and for all $a \in \mathcal{A}ct$, the set of a-derivatives of $\mathcal{F}(P')$ is the same as the set of $\mathcal{F}$-images the a-derivatives of P', i.e.*

$$\{Q' \mid \mathcal{F}(P') \xrightarrow{a} Q'\} = \{\mathcal{F}(P'') \mid P \xrightarrow{a} P''\}$$

where $\mathcal{A}ct(P)$ represents the activities enabled in P and $ds(P)$ is the derivative set (i.e. all one-step successors) of P.

Definition 9 *Two PEPA processes P and Q are* isomorphic*, denoted $P = Q$, if there exists a component isomorphism $\mathcal{F}$ between them such that $\mathcal{D}(\mathcal{F}(P)) = \mathcal{D}(Q)$, where $\mathcal{D}(P)$ is the full derivation graph of P (all n-step successors of P, with $n \in \mathbb{N}$).*

Isomorphism can be seen as a rather "strong" relation because it only groups processes showing *exactly the same* behaviour, and which differ only in the naming of derivatives. Still, it can be applied directly on the syntax of the language, thanks to some equational laws, providing a tool for model transformation. Moreover, two isomorphic PEPA processes generate equivalent Markov processes. It is also argued that strong isomorphism implies all other relations.

Weak isomorphism, while still allowing us to join equally behaving processes, provides also a more useful model simplification tool. This is achieved by considering only "visible" actions, while ignoring internal actions (as any classical weak version of a relation does). Unfortunately, being not preserved by the choice operator, weak isomorphism is not a congruence, so not all processes can be reduced via weak isomorphism.
The precise definition of weak isomorphism is a bit more complex of the one for its strong counterpart, and out of the scope of this document.

3.2 Strong bisimilarity

Strong bisimilarity is defined on the labelled transition system obtained from PEPA processes (or components) as a classical CCS-strong bisimulation taking into account also rates of actions.

Definition 10 *A binary relation $\mathcal{R} \subseteq C \times C$ over components is a* strong bisimulation *if $(P, Q) \in \mathcal{R}$ implies, for all action types $\alpha \in \mathcal{A}$,*

1. *$r_\alpha(P) = r_\alpha(Q)$;*

   and for all $a \in \mathcal{A}ct$,

2. *whenever $P \xrightarrow{a} P'$ then, for some Q', $Q \xrightarrow{a} Q'$ and $(P', Q') \in \mathcal{R}$;*

3. *whenever $Q \xrightarrow{a} Q'$ then, for some P', $P \xrightarrow{a} P'$ and $(P', Q') \in \mathcal{R}$.*

where $r_\alpha(P)$ is the rate at which action α is performed in process P.

Definition 11 *Components P and Q are* strong bisimilar*, written $P \sim Q$, if there exists a strong bisimulation $\mathcal{R}$ such that $(P, Q) \in \mathcal{R}$.*

It is interesting to note that strong bisimilarity is a congruence for PEPA, as it preserves all contexts, and so, thanks to an axiomatisation, it can be directly applied on the syntactic level. However, strong bisimilarity shows some flaws in the stochastic field: it does not guarantee that processes will behave in the same way from the probabilistic point of view. Consider the processes

$$\begin{array}{lcllcl} P & = & a.P + a.P + a.P' & Q & = & a.Q + a.Q' + a.Q' \\ P' & = & b.P & Q' & = & b.Q \end{array}$$

where $a = (\alpha, r_a)$ and $b = (\beta, r_b)$.
We can see that, even if $P \sim Q$ (the strong bisimulation is $\mathcal{R} = \{(P, Q), (P', Q')\}$), their transition rates are different:

$$\begin{array}{lcllcllcl} q(P,P) & = & 2r_a & q(P,P') & = & r_a & q(P',P) & = & r_b \\ q(Q,Q) & = & 2_a & q(Q,Q') & = & 2r_a & q(Q',Q) & = & r_b \end{array}$$

Thus, at steady-state the two systems will not show the same action frequencies. For this reason, strong bisimilar components do not always generate equivalent Markov processes[1]. A relation weaker than equivalence between Markov processes is *lumpable equivalence*:

Definition 12 *Two Markov processes $\{X_i\}$ and $\{Y_j\}$ are* lumpably equivalent *if there is a lumpable partition $\{X_{[i]}\}$ of $\{X_i\}$ and a lumpable partition $\{Y_{[j]}\}$ of $\{Y_j\}$ such that there is an injective function f satisfying*

$$q(X_{[k]}, X_{[l]}) = q(Y_{f([k])}, Y_{f([l])})$$

Strong bisimilar processes do not always produce equivalent nor lumpably equivalent Markov processes; thus strong bisimilarity comes out only partly useful. We will now see that this problem can be overcome with strong equivalence.

3.3 Strong equivalence

Strong equivalence uses a different approach towards the stochastic rates of actions by taking inspiration from probabilistic bisimulation [10].

Definition 13 *An equivalence relation $\mathcal{R} \subseteq \mathcal{C} \times \mathcal{C}$ is a* strong equivalence *if whenever $(P, Q) \in \mathcal{R}$ then for all $\alpha \in \mathcal{A}$ and for all $S \in \mathcal{C}/\mathcal{R}$,*

$$q[P, S, \alpha] = q[Q, S, \alpha]$$

where $\mathcal{A}$ is the set of all action types.

Definition 14 *Components P and Q are* strong equivalent*, written $P \cong Q$, if there exists a strong equivalence $\mathcal{R}$ such that $(P, Q) \in \mathcal{R}$.*

[1] Two Markov processes are equivalent if they have the same number of states and the same transition rates between these states

Differently from strong bisimilarity, strong equivalence allows for the distinction of long-run behaviour: in fact, the two processes P and Q shown before cannot be strong equivalent, as $q[P, \{P'\}, \alpha] = r_a$ while $q[Q, \{Q'\}, \alpha] = 2r_a$. Moreover, the Markov processes underlying two strong equivalent processes are lumpably equivalent: this allows us to effectively reduce state spaces for model solution. Model reduction is greatly helped by the strong equivalence being a congruence (and thus applicable directly at syntactical level, thanks to the axiomatisation of the relation).

4 Simulations and bisimulations for CTMCs

The work in [2] presents a number of comparative semantics for both time-abstract (discrete-time) and time-aware (continuous-time) models. Here we focus only on the relations defined on continuous-time setting, presenting simulations and bisimulations (both in weak and strong form) defined on CTMCs.

4.1 Strong bisimulation

Definition 15 *Let $C = (S, \mathbf{R}, L)$ be a CTMC and R an equivalence relation on S. R is a* strong bisimulation *on C if for $s_1 R s_2$*

$$L(s_1) = L(s_2) \;\; and \;\; \mathbf{R}(s_1, C) = \mathbf{R}(s_2, C) \, \forall C \in S/R$$

Two states s_1 and s_2 are *strongly bisimilar* (denoted $s_1 \sim s_2$) if there exists a strong bisimulation R on C such that $s_1 R s_2$.

As we can see, this is the "classical" probabilistic bisimulation to which an identification on state labelling has been added (in order to get logical characterisations on formula satisfiability: see later on).

4.2 Strong simulation

As transitions between states in probabilistic transition systems connect states with probability distributions, strong simulation is based on asymmetric "mappings" between distribution functions.

Definition 16 *A* distribution *on set S is a function $\mu : S \to [0, 1]$ with $\sum_{s \in S} \mu(s) \leqslant 1$.*

Let $Distr(S)$ denote the set of all distributions on S and $S_\perp = S \cup \{\perp\}$, where $\perp$ can be regarded to as a deadlock "non-state".

Definition 17 *Let S be a set, $R \subseteq S \times S$, and $\mu, \mu' \in Distr(S)$. A* weight function *for μ and μ' with respect to R is a function $\Delta : S_\perp \times S_\perp \to [0, 1]$ such that:*

1. *$\Delta(s, s') > 0$ implies sRs' or $s = \perp$,*
2. *$\mu(s) = \sum_{s' \in S_\perp} \Delta(s, s')$ for any $s \in S_\perp$.*
3. *$\mu'(s') = \sum_{s \in S_\perp} \Delta(s, s')$ for any $s' \in S_\perp$.*

We write $\mu \sqsubseteq_R \mu'$ iff there exists a weight function for μ and μ' with respect to R.

Definition 18 *Let $C = (S, \mathbf{R}, L)$ be a CTMC and $R \subseteq S \times S$. R is a* strong simulation *on C if for all $s_1 R s_2$:*

$$L(s_1) = L(s_2), \quad \mathbf{P}(s_1, \cdot) \sqsubseteq_R \mathbf{P}(s_2, \cdot), \quad \textit{and} \quad E(s_1) \leqslant E(s_2)$$

where $\mathbf{P}$ is such that $(S, \mathbf{P}, L) = emb(C)$.

Definition 19 *State s_2 strongly simulates state s_1 in C, written $s_1 \precsim s_2$, iff there exists a strong simulation R on C such that $s_1 R s_2$.*

Note that when $s_1 \precsim s_2$, s_2 can also be "faster" than s_1 ($E(s_2) > E(s_1)$), as long as its transition probabilities can be mapped on those of s_1.

4.3 Weak bisimulation

Weak bisimulation is inspired by branching bisimulation [11], which considers as stutter-steps all transitions ending in the same equivalence class as they start. Two states are equivalent if they have the same labels, and they reach the same equivalence classes (except their own) with the same rates:

Definition 20 *Let $C = (S, \mathbf{R}, L)$ be a CTMC and R and equivalence relation on S. R is a* weak bisimulation *on C if for all $s_1 R s_2$:*

$$L(s_1) = L(s_2) \;\; \textit{and} \;\; \mathbf{R}(s_1, C) = \mathbf{R}(s_2, C) \;\; \textit{for all } C \in S/R \textit{ with } C \neq [s_1]_R.$$

Definition 21 *s_1 and s_2 in C are* weakly bisimilar*, denoted $s_1 \approx s_2$, iff there exists a weak bisimulation R on C such that $s_1 R s_2$.*

4.4 Weak simulation

Weak simulation is based on more complex concepts. Successors of state s are splitted into two sets: V and U. The former representing a sort of "stutter-equivalence class" of s (so that all stutter steps will end into this set) and the latter containing all other successors (in which all "real steps" end). A scheme representing the idea underlying weak simulation is shown in Figure 1, where s_2 weakly simulates s_1, V_i are the stutter-step successors of s_i and U_i are the visible-step successors of s_i. Notice the simulation relations that must hold between V_i and s_i, while we use the weight relation from strong simulation to relate U_i and s_i.

Definition 22 *Let $C = (S, \mathbf{R}, L)$ be a CTMC and $R \subseteq S \times S$. R is a* weak simulation *on C iff for $s_1 R s_2$:*

- *$L(s_1) = L(s_2)$;*
- *there exist $\delta_i : S \to [0, 1]$ and $U_i, V_i \subseteq S$ $(i = 1, 2)$ satisfying the following conditions:*

1. (a) $v_1 R s_2$ *for all* $v_1 \in V_1$ *and*
   (b) $s_1 R v_2$ *for all* $v_2 \in V_2$*;*
2. *there exists a function* $\Delta : S \times S \to [0,1]$ *such that:*
   (a) $\Delta(u_1, u_2) > 0$ *implies* $u_1 \in U_1$, $u_2 \in U_2$ *and* $u_1 R u_2$,
   (b) *if* $K_1 > 0$ *and* $K_2 > 0$ *then for all states* $w \in S$*:*

$$K_1 \cdot \sum_{u_2 \in U_2} \Delta(w, u_2) = \delta_1(w) \cdot \mathbf{P}(s_1, w), \;\; K_2 \cdot \sum_{u_1 \in U_1} \Delta(u_1, w) = \delta_2(w) \cdot \mathbf{P}(s_2, w),$$

   where $K_i = \sum_{u_i \in U_i} \delta_i(u_i) \cdot \mathbf{P}(s_i, u_i)$ *for* $i = 1, 2$.
3. $\sum_{u_1 \in U_1} \delta_1(u_1) \cdot \mathbf{R}(s_1, u_1) \leqslant \sum_{s_2 \in U_2} \delta(u_2) \cdot \mathbf{R}(s_2, u_2)$.

Definition 23 s_2 weakly simulates s_1, *denoted* $s_1 \precsim s_2$, *iff there exists a weak simulation* R *on* C *such that* $s_1 R s_2$.

Here δ is a function which divides the successors of s_1 and s_2 into their subsets U_i and V_i: note that a state can partially belong to both sets.

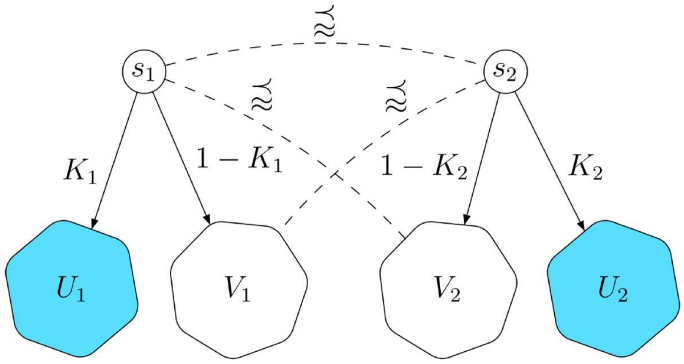

Figure 1: A schema of the relations occurring among s_1, s_2 and their successors when $s_1 \precsim s_2$.

4.5 Remarkable properties of the relations

The main relations holding between the above presented simulations and bisimulations are resumed in Figure 2. The meaning of the arrows is shown below:

$R \to R'$	R is finer than R' ($R \subseteq R'$)
$R \dashrightarrow R'$	R is finer than R' if the CTMC is uniformized
$R \not\to R'$	R is not finer than R'

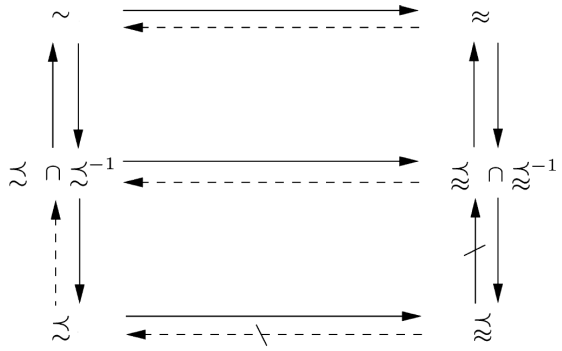

Figure 2: The relations between the bisimulation and simulation relations presented in this section.

4.6 Logical characterisations

The relations previously defined are now enriched with some interesting properties regarding the temporal logic CSL (continuous stochastic logic). An important result is that the previously defined bisimulations coincide with (a fragment of) CSL, thus allowing for the application of verification techniques directly on reduced models.

4.6.1 CSL

CSL is defined as a variant of the logic introduced in [1], to which steady-state and next-state operators are added together with weak variants of next and until operators. Its syntax is defined below:

state formulae : $\Phi ::= \mathrm{tt} \mid a \mid \neg a \mid \Phi \wedge \Phi \mid \Phi \vee \Phi \mid \mathcal{S}_{\trianglelefteq p}(\Phi) \mid \mathcal{P}_{\trianglelefteq p}(\varphi)$

path formulae : $\varphi ::= X^{\leqslant t}\Phi \mid \widetilde{X}^{\leqslant t}\Phi \mid \Phi \mathcal{U}^{\leqslant t}\Phi \mid \Phi \widetilde{\mathcal{U}}^{\leqslant t}\Phi.$

where $p \in [0,1]$, $\trianglelefteq \in \{<, \leqslant, >, \geqslant\}$ and $t \in \mathbb{R} \cup \{\infty\}$.
The intuitive meaning of the non-trivial operators is presented below:

$s \models \mathcal{S}_{\trianglelefteq p}(\Phi)$ iff the probability that, at steady state, a state satisfying Φ is reached starting from s is $\trianglelefteq p$;

$s \models \mathcal{P}_{\trianglelefteq p}(\varphi)$ iff the probability of all paths satisfying φ and starting in s is $\trianglelefteq p$;

$\sigma \models X^{\leqslant t}\Phi$ iff the second state of path σ satisfies Φ and is reached before t time units have elapsed (the sojourn time in the first state of the path is $\leqslant t$);

$\widetilde{X}^{\leqslant t}\Phi$ is the weak variant of $X^{\leqslant t}\Phi$, and allows for the path to contain only one state, or the sojourn time on the first state to be $> t$;

$\sigma \models \Phi \mathcal{U}^{\leqslant t}\Psi$ iff a state satisfying Ψ is reached in σ before t time units have elapsed and all states along path σ satisfy Φ;

$\Phi \widetilde{\mathcal{U}}^{\leqslant t} \Psi$ is the weak variant of $\Phi \mathcal{U}^{\leqslant t} \Psi$, and permits to never reach a state satisfying Ψ before t time units have elapsed, provided that all states along the path satisfy Φ.

4.6.2 Characterising bisimulations

Definition 24 *Given a CTMC $C = (S, \mathbf{R}, L)$, two states $s_1, s_2 \in S$ are* CSL-equivalent*, written $s_1 \equiv_{CSL} s_2$, if they satisfy exactly the same set of CSL formulae, i.e.:*

$$\text{for each CSL state-formula } \Phi \quad s_1 \models \Phi \Leftrightarrow s_2 \models \Phi$$

Strong bisimulation is shown to coincide with $\equiv_{CSL}$. As for weak bisimulation, the coincidence cannot hold for the complete CSL, as the *next* (and *weak next*) operator would generate a conflict with the stutter-step abstraction made by the weak bisimulation, as next operators are not stutter-invariant; thus, $\approx$ is shown to coincide with $\equiv_{CSL\setminus X}$.

As one would expect from what we have shown so far, in an uniformized CTMC $\equiv_{CSL}$ coincides with $\equiv_{CSL\setminus X}$.

4.6.3 Characterising simulations

For the characterisation of simulation relations, a distinction is made between safety ("something bad never happens") and liveness ("something good will eventually happen") properties. Here we show the *CSL-safety formulae*:

$$\Phi ::= \mathrm{tt} \mid a \mid \neg a \mid \Phi \wedge \Phi \mid \Phi \vee \Phi \mid \mathcal{P}_{\geqslant p}(\widetilde{X}^{\leqslant t} \Phi) \mid \mathcal{P}_{\geqslant p}(\Phi \widetilde{\mathcal{U}}^{\leqslant t} \Phi)$$

CSL-liveness formulae are obtained from the following grammar:

$$\Phi ::= \mathrm{tt} \mid a \mid \neg a \mid \Phi \wedge \Phi \mid \Phi \vee \Phi \mid \mathcal{P}_{\geqslant p}(X^{\leqslant t} \Phi) \mid \mathcal{P}_{\geqslant p}(\Phi \mathcal{U}^{\leqslant t} \Phi)$$

Notice as the weak version of next and until operator is present only in the safety fragment of CSL, as next step could seen as "dangerous" for the safety (as in "he who leaves the old path for the new one, knows what he is leaving, but not what he is going to find"), so if the system does not move from a safe position, it remains safe. Notice also that, since the objective is the creation of a preorder relation based on the safety/liveliness of a process, the steady-state operator $\mathcal{S}_{\trianglelefteq p}(\Phi)$ is not present: this is because CTMCs cannot be ordered according to their steady state performance [4].

Definition 25 *Given a CTMC $C = (S, \mathbf{R}, L)$ and two states $s, s' \in S$, we say that s'* is safer than *s, written $s \precsim^{safe}_{CSL} s'$, iff for each CSL-safe formula Φ: $s' \models \Phi$ implies $s \models \Phi$.*

The other relations $\precsim^{live}_{CSL}$, $\precsim^{safe}_{CSL\setminus X}$ and $\precsim^{live}_{CSL\setminus X}$ are defined in a similar way.

It is shown that $\precsim$ coincides with $\precsim^{safe}_{CSL}$ and $\precsim^{live}_{CSL}$. Unfortunately, the same type of coincidence does not apply in the weak case, where only one inclusion can be demonstrated (i.e., $\precsim \subseteq \precsim^{safe}_{CSL\setminus X}$ and $\precsim \subseteq \precsim^{live}_{CSL\setminus X}$), while the other is only conjectured.

5 Congruences for generally-distributed process algebra ♤.

Although the formalisms presented in [6, 7] are not based on CTMCs, the approach used by the authors could come out interesting for the general topic of stochastic equivalence relations. The concepts around ♤ are rather large, so here we will only deal with what is necessary to have an understanding of the main equivalences defined for the language.

5.1 Probabilistic transition systems

Probabilistic transition systems are a more general type of transition systems, where states are divided in two disjoint subsets: *probabilistic states*, each of which make exactly one probabilistic step, and *non-deterministic states*, which may have any number of outgoing non-deterministic transitions.

This kind of transition system is particularly useful, as it can handle general distributions: this way, a representation of non deterministically distributed time constraints is possible.

To each arc exiting from a probabilistic state is associated a probability, and the sum of such probabilities is 1. Arcs exiting from non-deterministic states are also labelled; each of these labels is composed by an element of the set $\mathcal{A} \times \mathbb{R}_{\geqslant 0}$, where $\mathcal{A}$ is the set of all possible actions. A label in the form a, d (also written as $a(d)$) should be intended as "action a happens right after the system has been idle for d time units".

5.1.1 Definition

Definition 26 *A probabilistic transition system is a tuple* $PTS = (\Sigma, \Sigma', \mathcal{L}, T, \longrightarrow)$*, where:*

1. Σ *is the set of probabilistic states*
2. Σ' *is the set of non-deterministic states such that* $\Sigma \cap \Sigma' = \emptyset$
3. $\mathcal{L}$ *is the set of labels for the transitions exiting from non-deterministic states; as we already stated,* $\mathcal{L} = \mathcal{A} \times \mathbb{R}_{\geqslant 0}$
4. T *is a function that connects each probabilistic state with a set of non-deterministic states with a probabilistic transition*[2]
5. $\longrightarrow \subseteq \Sigma' \times \mathcal{L} \times \Sigma$ *is the labelled (or non-deterministic) transition relation.*

A *rooted PTS* is a pair (PTS, σ_0), where σ_0 is the initial probabilistic state. The writing $\sigma \overset{p}{\dashrightarrow} \sigma'$ stands for $P(\sigma') = p$, where $T(\sigma) = (\Omega, \mathcal{F}, P)$; and $\sigma' \overset{\ell}{\longrightarrow} \sigma$ abbreviates $\langle \sigma', \ell, \sigma \rangle \in \longrightarrow$. For some examples of probabilistic transition systems, we refer to [6].

[2] More formally, let $Prob(H)$ be the set of probability spaces $(\Omega, \mathcal{F}, P)$ such that $\Omega \subseteq H$. We have that $T : \Sigma \longrightarrow Prob(\Sigma')$ (cf. [6], Appendix A).

5.1.2 Probabilistic bisimulation

Basically, this is the classical probabilistic bisimulation adapted to the case of probabilistic transition systems.

Definition 27 *Let* $(\Sigma, \Sigma', \mathcal{L}, T, \longrightarrow)$ *be a PTS and* $\mu : \Sigma \times \mathcal{P}_{fin}(\Sigma') \to [0,1]$ *be defined by*

$$\mu(\sigma, S) \stackrel{def}{=} \begin{cases} P(S \cap \Omega) & \text{if } S \cap \Omega \in \mathcal{F} \\ 0 & \text{otherwise} \end{cases}$$

provided that $T(\sigma) = (\Omega, \mathcal{F}, P)$. *Let* $R \subseteq (\Sigma \times \Sigma) \cup (\Sigma' \times \Sigma')$ *be an equivalence, and* Σ'/R *be the set of equivalence classes in* Σ' *induced by* R. R *is a* probabilistic bisimulation *if for any* $\langle \sigma_1, \sigma_2 \rangle \in R$*:*

1. *for all* $S \subseteq \Sigma'/R, \mu(\sigma_1, \cup S) = \mu(\sigma_2, \cup S)$*, whenever* $\sigma_1, \sigma_2 \in \Sigma$*; and*
2. *for all* $\ell \in \mathcal{L}, \sigma_1 \stackrel{\ell}{\longrightarrow} \sigma_1'$ *implies* $\sigma_2 \stackrel{\ell}{\longrightarrow} \sigma_2'$ *and* $\langle \sigma_1', \sigma_2' \rangle \in R$*, for some* σ_2' *whenever* $\sigma_1, \sigma_2 \in \Sigma'$.

The notation $\sigma_1 \sim_p \sigma_2$ is read "states σ_1 and σ_2 are probabilistically bisimilar", and means that there exists a probabilistic bisimulation R such that $\langle \sigma_1, \sigma_2 \rangle \in R$. The same notation can be extended to (rooted) PTS the obvious way.

5.2 Stochastic automata

Probabilistic transition systems can be useful in modelling various stochastic settings, but they are always infinite (both in the number of states and transitions) when using random setting for timing (either by relying on infinite-sample space discrete probabilistic spaces, or by using any continuous probabilistic space). Thus, a proper tractability of PTS can become troublesome. A solution to this problem is the use of stochastic automata, which allow to represent random timing in a finite way.

In order to represent random timing, stochastic automata rely on the so-called *random clock variables*. A clock can be set in accord with its distribution function, and from that point on it will be counting backwards at the same pace as the other clocks. When the value of a clock is no more positive, the clock has *expired*: in that case, all actions associated with that clock can be enabled.

To obtain the value of a clock, we rely on a valuation function $v : C \to \mathbb{R}$, where C is the set of all clocks. The function $v - d(x)$ gives the valuation of clock x obtained d time units after v was observed, and is defined as $(v - d)(x) \stackrel{\text{def}}{=} v(x) - d$. We call $\mathbf{V}$ the set of all valuations.

5.2.1 Definition

Definition 28 *A stochastic automaton is a tuple* $SA = (\mathcal{S}, \mathcal{A}, C, \rightarrow, \kappa)$*, where:*

1. $\mathcal{S}$ *is a set of locations;*
2. $\mathcal{A}$ *is a set of actions;*

3. *C is a set of clocks (each clock $x \in C$ has associated its own distribution function F_x);*

4. *$\rightarrow \subseteq \mathcal{S} \times (\mathcal{A} \times \mathcal{P}_{fin}(C)) \times \mathcal{S}$ is the set of edges;*

5. *$\kappa : \mathcal{S} \rightarrow \mathcal{P}_{fin}(C)$ is the clock setting function.*

When an initial location $\sigma_0 \in \mathcal{S}$ is associated to SA, we call (SA, σ_0) a *rooted* stochastic automaton. When writing $s \xrightarrow{a,C} s'$ we intend that $(s, a, C, s') \in \rightarrow$; C is called the *trigger set* of the edge.

The clock setting function specifies which clocks are to be set when the execution enters a particular state: upon entering state s, for all $x \in \kappa(s)$ the valuation of x is set according with distribution function F_x. This done, all clocks start decreasing their value. If $s \xrightarrow{a,C} s'$ then, when all clocks in C are expired, action a can be performed: the transition is thus called *enabled*. When the transition is performed, the execution advances to state s'. In case more than one transition is enabled at the same time, one is chosen non-deterministically.

5.2.2 Semantics

The semantics of stochastic automata is defined in terms of probabilistic transition systems, and particularly two semantics are adopted: *closed* and *open* system. The former allows the property of *maximal progress* to apply: whenever a transition becomes enabled it immediately takes place. This approach models the eventuality that the system has no external influences, hence no other transition could be executed before the "rightful" one. With open semantics the system is expected to interact with external components, so maximal progress property does not hold.

Closed system behaviour Given a stochastic automaton $SA = (\mathcal{S}, \mathcal{A}, C, \rightarrow, \kappa)$, the closed system behaviour of SA is defined by the probabilistic transition system

$$PTS_c(SA) \stackrel{\text{def}}{=} ((\mathcal{S} \times \mathbf{V}), [\mathcal{S} \times \mathbf{V}], \mathcal{A} \times \mathbb{R}_{\geqslant 0}, T, \rightarrow)$$

which is composed as follows.

The states are obtained by associating to each location of the stochastic automaton all the possible valuations for all timers. In order to distinguish between probabilistic and non-deterministic states in the PTS, we write (s, v) to intend the probabilistic state associated to location s and valuation v, and $[s, v]$ to indicate the corresponding non-deterministic state. Non-deterministic transitions are labelled as one would expect to, and functions T and $\rightarrow$ are defined by the following rules:

$$\textbf{Prob}\quad \frac{\overrightarrow{\kappa(s)} = (x_1, \ldots, x_n)}{T(s,v) = \mathcal{D}_v^s(\mathcal{R}(F_{x_1}, \ldots, F_{x_n}))}$$

$$\textbf{Closed}\quad \frac{s \xrightarrow{a,C} s' \quad \mathsf{exp}_d(v, C) \quad \mathsf{mpr}_d(s, v)}{[s, v] \xrightarrow{a(d)} (s', (v - d))}$$

Intuitively, probabilistic states are exploited to reset timers. In **Prob** rule, we can read $\mathcal{D}_v^s(\mathcal{R}(F_{x_1}, \ldots, F_{x_n}))$ as "*For each $i \in \{1, \ldots, n\}$, use the distribution function F_{x_i} to calculate a starting value for timer x_i and update accordingly the valuation v. The resulting non-deterministic state is [s, v].*".

As for rule **Closed**, non-deterministic states are used to "let time pass". In particular, as system observes maximal progress property, the predicates exp and mpr are used to express this property: $\mathsf{exp}_d(v, C)$ means that all timers in C have expired d time units after v was osserved, and $\mathsf{mpr}_d(s, v)$ says that there is no possibility to leave s before d time units have elapsed.

Open system behaviour The open system behaviour of a stochastic automaton SA is defined by

$$PTS_o(SA) \stackrel{\text{def}}{=} ((\mathcal{S} \times \mathbf{V}), [\mathcal{S} \times \mathbf{V}], \mathcal{A} \times \mathbb{R}_{\geqslant 0}, T, \rightarrow)$$

The only remarkable difference with the closed system behaviour is that function $\rightarrow$ is obtained from the following rule **Open**, which replaces rule **Closed**:

$$\textbf{Open}\quad \frac{s \xrightarrow{a,C} s' \quad \mathsf{exp}_d(v, C)}{[s, v] \xrightarrow{a(d)} (s', (v - d))}$$

Notice that the rule does not request for the mpr predicate to be true.

5.2.3 Equivalences on stochastic automata

Now we show a set of equivalences for stochastic automata, briefly describing the main features of each one.

Open and closed p-similarity Given two rooted stochastic automata (SA_1, s_1) and (SA_2, s_2), they are:

- *closed p-similar* (written $(SA_1, s_1) \sim_c (SA_2, s_2)$) if

$$(PTS_c(SA_1), (s_1, v_0)) \sim_p (PTS_c(SA_2), (s_2, v_0)) \quad \text{for every } v_0 \in \mathbf{V}$$

- *open p-similar* (written $(SA_1, s_1) \sim_o (SA_2, s_2)$) if

$$(PTS_o(SA_1), (s_1, v_0)) \sim_p (PTS_o(SA_2), (s_2, v_0)) \quad \text{for every } v_0 \in \mathbf{V}$$

Being directly based on the (open/closed) PTS semantics of stochastic automata, the two relations $\sim_c$ and $\sim_o$ deal with infinite state spaces. In order to have a more tractable model, we bring forth some weaker relations which allow for symbolic reasoning.

Structural bisimulation Structural bisimulation preserves both actions and sets of clocks. Given a stochastic automaton $SA = (\mathcal{S}, \mathcal{A}, C, \rightarrow, \kappa)$, $R \subseteq \mathcal{S} \times \mathcal{S}$ is a *structural bisimulation* if R is symmetric and, for all $a \in \mathcal{A}, C \in C$, whenever $\langle s_1, s_2 \rangle \in R$ the following properties hold:

1. $s_1 \xrightarrow{a,C} s_1'$ implies $s_2 \xrightarrow{a,C} s_2'$ and $\langle s_1', s_2' \rangle \in R$ for some $s_2' \in \mathcal{S}$ and
2. $\kappa(s_1) = \kappa(s_2)$

Two locations $s_1, s_2 \in \mathcal{S}$ are structurally bisimilar (written $s_1 \sim_s s_2$) if there is a structural bisimulation R such that $\langle s_1, s_2 \rangle \in R$. The definition trivially extends to stochastic automata.

Symbolic bisimulation Although structural bisimulation is more tractable than probabilistic bisimulations, it does not properly handle stochastic information. In particular, trivial syntactical changes in the expression of clocks which leave unaltered their use result in non structurally bisimilar processes. In order to consider probabilistic information while still maintaining tractability, we resort on a symbolic bisimulation on stochastic automata.

The first step is clearly identifying which clocks should be labelled as "relevant", and then should be considered in the definition of the symbolic bisimulation.

Given a stochastic automaton $SA = (\mathcal{S}, C, \mathcal{A}, \rightarrow, \kappa)$,

- the *free clock variables* of location $s \in \mathcal{S}$ (written $\mathsf{Fv}(s)$) is the smallest set satisfying
$$\mathsf{Fv}(s) = \left\{ x \mid s \xrightarrow{a,C} s' \wedge x \in C \cup \mathsf{Fv}(s') \right\} - \kappa(s)$$
- the *relevant clock variables* of s is the smallest set satisfying
$$\mathsf{Rel}(s) = \left\{ x \mid s \xrightarrow{a,C} s' \wedge x \in C \cup \mathsf{Fv}(s') \right\}$$

The symbolic bisimulation relates relevant clocks when they are set in the same instant and with the same distribution. This kind of clock correlation is obtained resorting on *synchronisation relations*.

Let C be a set of clocks, and $C_1, C_2, C_1', C_2' \subseteq C$. A relation $SR \subseteq \mathcal{P}(C) \times \mathcal{P}(C)$ is a synchronisation relation if

$$\langle C_1, C_2 \rangle, \langle C_1', C_2' \rangle \in SR \Rightarrow \langle C_1, C_2 \rangle \trianglelefteq \langle C_1', C_2' \rangle$$

where the relation $\trianglelefteq$, which can be read as "is compatible with", is defined as follows:

$$\langle C_1, C_2 \rangle \trianglelefteq \langle C_1', C_2' \rangle \Longleftrightarrow (C_1 \cap C_1') \cup (C_2 \cap C_2') = \emptyset \vee (C_1 \subseteq C_1' \wedge C_2 \subseteq C_2')$$

Note that the compatibility relation should hold between all couples in SR. So, for each $\langle C_1, C_2\rangle, \langle C'_1, C'_2\rangle \in SR$ we have either $(C_1 \cap C'_1) \cup (C_2 \cap C'_2) = \emptyset$ or $\langle C_1, C_2\rangle = \langle C'_1, C'_2\rangle$

Definition 29 *Given a stochastic automaton $SA = (\mathcal{S}, \mathcal{C}, \mathcal{A}, \rightarrow, \kappa)$, a* symbolic bisimulation *is a relation $R \subseteq \mathcal{S}\times\mathcal{S}\times\mathcal{P}(\mathcal{P}(C)\times\mathcal{P}(C))$ that, for each $\langle s_1, s_2, SR\rangle \in R$, satisfies the following properties:*

1. *SR is a synchronisation relation such that:*
   (a) *for $i = 1, 2$, $\bigcup\{C_i \mid \langle C_1, C_2\rangle \in SR\} = \mathsf{Rel}(s_i)$;*
   (b) *$\langle C_1, C_2\rangle \in SR$ implies $\langle C_1, C_2\rangle \trianglelefteq \langle \kappa(s_1), \kappa(s_2)\rangle$;*
   (c) *if $\langle C_1, C_2\rangle \in SR$ and $C_i \subseteq \kappa(s_i), i = 1, 2$, then, for all $t \in \mathbb{R}$, $\prod_{x\in C_1} F_x(t) = \prod_{y\in C_2} F_y(t)$*
2. *$s_1 \overset{a,C_1^\star}{\rightarrow} s'_1$, then there are s'_2 and $C_2^\star$ such that $s_2 \overset{a,C_2^\star}{\rightarrow}$, and*
   (a) *$\langle C_1, C_2\rangle \in SR$ implies $\langle C_1, C_2\rangle \trianglelefteq \langle C_1^\star, C_2^\star\rangle$; and*
   (b) *$\langle s'_1, s'_2, SR'\rangle \in R$ for some SR' which is* forward compatible *with SR, i.e.:*
   $\left\{\langle C_1, C_2\rangle \in SR \mid (C_1 \cap \mathsf{Fv}(s'_1)) \cup (C_2 \cap \mathsf{Fv}(s'_2)) \neq \emptyset\right\}-\left(\mathcal{P}(C_1^\star)\times\mathcal{P}(C_2^\star)\right) =$
   $\left\{\langle C'_1, C'_2\rangle \in SR' \mid (C'_1 \cap \mathsf{Fv}(s'_1)) \cup (C'_2 \cap \mathsf{Fv}(s'_2)) \neq \emptyset\right\}-\left(\mathcal{P}(C_1^\star)\times\mathcal{P}(C_2^\star)\right)$
3. *$s_2 \overset{a,C_2^\star}{\rightarrow} s'_2$, then there are s'_1 and $C_1^\star$ such that $s_1 \overset{a,C_1^\star}{\rightarrow} s'_1$, and*
   (a) *$\langle C_1, C_2\rangle \in SR$ implies $\langle C_1, C_2\rangle \trianglelefteq \langle C_1^\star, C_2^\star\rangle$; and*
   (b) *$\langle s'_1, s'_2, SR'\rangle \in R$ for some SR' which is forward compatible with SR.*

Two locations $s_1, s_2 \in \mathcal{S}$ are *symbolically bisimilar* (written $s_1 \sim_\& s_2$) if there exists a symbolic bisimulation R such that $\langle s_1, s_2, SR\rangle \in R$ for some SR satisfying

$$\text{if } \langle C_1, C_2\rangle \in SR \text{ and } x \in (C_1 \cap \mathsf{Fv}(s_1)) \cup (C_2 \cap \mathsf{Fv}(s_2)) \text{ then } C_1 = C_2 = \{x\}$$

The rooted stochastic automata (SA_1, s_1) and (SA_2, s_2) are symbolically bisimilar if $s_1 \sim_\& s_2$ in the (disjoint) union of SA_1 and SA_2.

It can be proved that $\sim_\&$ is an equivalence over stochastic automata locations, but it is not a congruence.

Finally, we show the relations subsisting between all equivalence relations introduced so far:

$$\sim_s \subset \sim_\& \subset \sim_o \subset \sim_c$$

Note that symbolic bisimulation is coarser than structural bisimulation but finer than open p-bisimulation.

5.3 ♤ - Stochastic process algebra for discrete event systems

The language ♤ is born with the purpose to deal with discrete and continuous *general* probability distributions. However, this strong assumption has a drawback: indeed, the use of non-memoryless distributions implies that *the expansion law does no longer hold in general*. This can be easily seen in the following example:

$$a_F; p \parallel_\emptyset b_G; q \quad \neq \quad a_F; (p \parallel_\emptyset b_G; q) + b_G; (a_F; p \parallel_\emptyset q)$$

where general distributions F and G are exploited to get delays for actions a and b, respectively. When e.g. action a of the left-hand process is performed, the computation for the remaining time before firing action b has to take into account the time which has already elapsed. The same cannot be straightforwardly obtained in the right-hand process.

The idea which allows to overcome such problem is to syntactically separate the three fundamental concepts encased in a_F:

- start of delay
- end of delay
- instant action

In order to keep track of delays, the language resorts on *clocks*. At the start of a delay, the corresponding clock is initialised according to the associated distribution function, and starts counting backwards. When a clock is no more positive, the delay is finished and the corresponding (instant) action can be performed. Let C be a set of clocks, a an action, and p a process. The symbols representing the aforementioned concepts are the following:

- clock setting: $\{\!|\, C \,|\!\}$
- wait for clock expiration: $\{\, C \,\}\mapsto$
- instant action $a; p$

This allows to express the previous example in the following way:

$$\{\!|\, x \,|\!\}\, p' \parallel_\emptyset \{\!|\, y \,|\!\}\, q' \quad = \quad \{\!|\, x, y \,|\!\}\, (\{\, x \,\}\mapsto a; (p \parallel_\emptyset q') + \{\, y \,\}\mapsto b; (p' \parallel_\emptyset q))$$

where $p' = \{\, x \,\}\mapsto a; p$ and $q' = \{\, y \,\}\mapsto b; q$. This time, in both left-hand and right-hand processes, after the expiration of the first clock (e.g. x) and the execution of the corresponding action (a), the second clock is still counting with the remaining time: thus, the other action can happen after a correct amount of time. The schema shown here can be applied in a general way to obtain a general expansion law.

The most common type of synchronisation, known as patient communication [8], can be represented directly as follows:

$$(\{\!|\, x \,|\!\}\, \{\, x \,\}\mapsto a; p) \parallel_a (\{\!|\, y \,|\!\}\, \{\, y \,\}\mapsto a; q) \quad = \quad \{\!|\, x, y \,|\!\}\, \{\, x, y \,\}\mapsto a; (p \parallel_a q)$$

As can be seen, the time for the whole communication to take place is equal to the time required by the "slowest" process.

5.3.1 Syntax

The syntax of $\spadesuit$ is defined according to the grammar in Table 1, where $a \in \mathcal{A}$ is an action name, $C \in \mathcal{P}_{\text{fin}}(C)$ is a trigger/clock-setting set of random clocks, $A \subseteq \mathcal{A}$ is a synchronisation set, $f : \mathcal{A} \to \mathcal{A}$ is a renaming function, and $X \in \mathcal{V}$ is a process variable. Process variables are defined by recursive equations of the form $X = p$, where $p \in \spadesuit$. A set E of recursive equations defines a recursive specification.

$\mathbf{0}$	*Nil* or *stop*	$p \parallel_A p$	*Parallel composition*
$a; p$	*Action prefix*	$p \mathbin{\underline{\parallel}}_A p$	*Left merge*
$C \mapsto p$	*Triggering condition*	$p \mid_A p$	*Communication merge*
$\{\!\| C \|\!\} p$	*Clock setting*	$p[f]$	*Renaming*
$p + p$	*Choice*	X	*Process instantiation*

Table 1: Syntax of $\spadesuit$.

Intuitively, clock variables are bound in a process p whenever they are set. For instance, free variables in $p \equiv (\{\!| x |\!\} \{ x, y \} \mapsto a; \mathbf{0}) + \{ x \} \mapsto b; \mathbf{0}$ are $\mathsf{fv}(p) = \{x, y\}$, and bound variables are $\mathsf{bv}(p) = \{x\}$.

5.3.2 Intuition of semantics

The semantics of $\spadesuit$ is defined in terms of stochastic automata. As the definition of the semantics is rather complicated and goes beyond the scope of this paper, we only state that the mapping from $\spadesuit$ to stochastic automata can be meaningfully done for a particular "well formed" subset of the whole language.

It is important to notice that $\sim_s$ (structural bisimulation) and $\sim_o$ (open p-bisimulation) are congruences for $\spadesuit$, and an equational theory for the language can be derived from this fact.

5.3.3 Equational theory

Structural bisimulation Restricting $\spadesuit$ to its basic version $\spadesuit^b$ (by removing the operators for parallel composition, renaming, and process definition), an equational theory is created for the language and proved to be sound and complete with respect to $\sim_s$.

Open p-bisimulation Extending the previous set of axioms for the equational theory in $\spadesuit^b$, we obtain a sound (but not complete) equational theory for open p-bisimulation ($\sim_o$).

Static operators In order to include static operators (renaming and parallel compositions) in the equational theory, can be conveniently extended. The resulting set of equational laws is proven to be sound and complete and to allow for the expression of an expansion law for $\spadesuit$.

6 Barbed bisimulation for Stochastic Ambient Calculus

The authors of [12] have created a stochastic extension of Mobile Ambients process algebra [5], adding rates to actions, as usual. What distinguishes this work among the others is the *strong Markovian bisimulation* defined therein, which is presented as a stochastic barbed bisimulation. The authors claim that it is the first barbed stochastic bisimulation to be applied to a process algebra, and it comes useful in their case because a more "classical" stochastic bisimulation (as the ones shown in the sections before) could have shown difficult to deal with, as the semantics for Stochastic Ambient Calculus produces second order labelled transition systems (i.e., labelled transition systems which contain also processes in the labels).

6.1 Stochastic Ambient Calculus

The syntax of the language is shown in Table 2.

$P, Q :=$		process			
	$\mathbf{0}$	nil			
|	$\sum_{i \in I} M_i.\, P_i$	local sum	$M, N :=$		capability
|	$n[P]^f$	stochastic ambient		$\mathsf{in}(n, \lambda)$	enter
|	$P \mid Q$	composition	|	$\mathsf{out}(n, \lambda)$	exit
|	$(\nu n)P$	restriction	|	$\mathsf{open}(n, \lambda)$	open
|	$(fix_A.\, P)$	recursion			
|	A	identifier			

Table 2: Syntax of the Stochastic Ambient Calculus.

The language uses a particular box structure called *stochastic ambient*, in which processes are contained. Each ambient $n[P]^f$ is equipped with its own rate calculation function $f : \mathbb{R} \to \mathbb{R}$, which allows the ambient to influence the rates at which processes interact within it.

In order to obtain the effect of ambients moving into or out from other ambients, the semantics for Stochastic Ambient Calculus needs to generate a *second order labelled transition system*, that is a labelled transition system in which processes are put on transition labels. An example of what we intend is the rule *OUT*, which allows us to deal with a process willing to exit from an ambient:

$$\frac{P \xrightarrow[r]{exit\ n} P'}{m[P]^f \xrightarrow[r]{out\ n(m[P']^f)} \mathbf{0}}$$

6.2 Barbed Stochastic Bisimulation

In order to have a bisimulation relation allowing us to deal with the problems arising from having both a second order labelled transition system and restriction operator, the authors of the calculus have created a barbed Markovian bisimulation. This kind of relation is based on the concept of *barb*, intended in this case as *capability to exhibit a certain ambient name n*: $P \downarrow n$ (see Table 3).

$$n[P]^f \downarrow n$$

$$\frac{P \downarrow n}{P \mid Q \downarrow n} \qquad \frac{P \downarrow n}{Q \mid P \downarrow n}$$

$$\frac{P \downarrow n \quad m \neq n}{(\nu m)P \downarrow n} \qquad \frac{P \downarrow n}{(fix_A.\, P) \downarrow n}$$

Table 3: The conditions for which $P \downarrow n$.

Definition 30 *An equivalence binary relation $\mathcal{S}$ on Stochastic Mobile Ambients processes is a* strong Markovian bisimulation *if for all $P, Q \in \mathcal{P}$, $P\mathcal{S}Q$ implies that for all equivalence classes $E \in \mathcal{P}_{/\mathcal{S}}$ and names $n \in \mathcal{N}$:*

i) if $P \downarrow n$ then $Q \downarrow n$;

ii) the sum of all rates of actions going from $C[P]$ to R is equal to the same sum for $C[Q]$ for all contexts C and all processes $R \in E$.

Definition 31 *Two processes P and Q are called* strongly Markovian bisimilar*, written $P \sim_M Q$, if there is a strong Markovian bisimulation $\mathcal{S}$ such that $P\mathcal{S}Q$.*

Being preserved by all context by definition, the relation is automatically a congruence over Stochastic Mobile Ambients processes.

7 Related work

A work similar in purposes to the present one has been recently carried out by Bernardo in [3].

References

[1] Adnan Aziz, Kumud Sanwal, Vigyan Singhal, and Robert Brayton. Model-checking continuous-time markov chains. *ACM Trans. Comput. Logic*, 1(1):162–170, 2000.

[2] Christel Baier, Joost-Pieter Katoen, Holger Hermanns, and Verena Wolf. Comparative branching-time semantics for markov chains. *Inf. Comput.*, 200(2):149–214, 2005.

[3] Marco Bernardo. A survey of markovian behavioral equivalences. In M. Bernardo and J. Hillston, editors, *Formal Methods for the Design of Computer, Communication and Software Systems: Performance Evaluation (SFM'07)*, pages 180–219, 2007.

[4] Marco Bernardo and Rance Cleaveland. A theory of testing for Markovian processes. *Lecture Notes in Computer Science*, 1877:305–319, 2000.

[5] Luca Cardelli and Andrew D. Gordon. Mobile ambients. In *Foundations of Software Science and Computation Structures: First International Conference, FOSSACS '98*. Springer-Verlag, Berlin Germany, 1998.

[6] Pedro R. D'Argenio and Joost-Pieter Katoen. A theory of stochastic systems. Part I: Stochastic automata. *Inf. Comput.*, 203(1):1–38, 2005.

[7] Pedro R. D'Argenio and Joost-Pieter Katoen. A theory of stochastic systems. Part II: process algebra. *Inf. Comput.*, 203(1):39–74, 2005.

[8] J. Hillston. The nature of synchronisation. In *PAPM'94, Process Algebra and Performance Modelling*.

[9] Jane Hillston. *A compositional approach to performance modelling*. Cambridge University Press, New York, NY, USA, 1996. ISBN 0-521-57189-8.

[10] Kim G. Larsen and Arne Skou. Bisimulation through probabilistic testing. *Inf. Comput.*, 94(1):1–28, 1991.

[11] Rob J. van Glabbeek and W. Peter Weijland. Branching time and abstraction in bisimulation semantics. *J. ACM*, 43(3):555–600, 1996.

[12] Maria Grazia Vigliotti and Peter G. Harrison. Stochastic ambient calculus. *Electr. Notes Theor. Comput. Sci.*, 164(3):169–186, 2006.