Available online at www.sciencedirect.com
SCIENCE DIRECT®
Information and Computation 203 (2005) 39–74
Information and Computation
www.elsevier.com/locate/ic

A theory of stochastic systems. Part II: Process algebra

Pedro R. D'Argenio [a,b,1], Joost-Pieter Katoen [b,c,*]

[a] *Universidad Nacional de Córdoba, Ciudad Universitaria, 5000 Córdoba, Argentina*
[b] *University of Twente, P.O. Box 217, 7500 AE Enschede, The Netherlands*
[c] *RWTH Aachen, Ahornstraße 55, D-52074 Aachen, Germany*

Received 28 November 2003; revised 9 February 2005
Available online 26 September 2005

Abstract

This paper introduces ♤ (pronounce *spades*), a *stochastic process algebra for discrete-event systems*, that extends traditional process algebra with timed actions whose delay is governed by general (a.o. continuous) probability distributions. The operational semantics is defined in terms of stochastic automata, a model that uses clocks—like in timed automata—to symbolically represent randomly timed systems, cf. the accompanying paper [P.R. D'Argenio, J.-P. Katoen, A theory of stochastic systems. Part I: Stochastic automata. Inf. Comput. (2005), to appear]. We show that stochastic automata and ♤ are equally expressive, and prove that the operational semantics of a term up to α-conversion of clocks, is unique (modulo symbolic bisimulation). (Open) probabilistic and structural bisimulation are proven to be congruences for ♤, and are equipped with an equational theory. The equational theory is shown to be complete for structural bisimulation and allows to derive an expansion law.

Keywords: Axiomatisation; Bisimulation; Operational semantics; Stochastic automaton; Stochastic process algebra

* Corresponding author.
E-mail address: katoen@cs.rwth-aachen.de (J.-P. Katoen).
[1] Partially supported by the NWO visiting Grant B-61-519 and by the ANPCyT project PICT 11-11738.

0890-5401/$ - see front matter © 2005 Elsevier Inc. All rights reserved.
doi:10.1016/j.ic.2005.07.002

1. Introduction

1.1. Functionality and performance

Traditionally, models and methods for the analysis of the functional correctness of reactive systems and those for the analysis of their performance and dependability aspects have been studiedby different research communities. This has resulted in the development of successful, but distinct and largely unrelated modeling and analysis techniques for both domains. In many modern systems, however, the difference between their functional features and their performance properties has become blurred, as relevant functionalities become inextricably linked to performance aspects. The strong relationship between functionality and performance aspects would clearly benefit from a paradigm for the modeling and analysis of systems in which qualitative and quantitative aspects are studied from an integrated perspective. This would allow us to check how changes in functionality affect performance issues, and vice versa. In addition, such a paradigm would yield a tight relationship between the models that are used for qualitative and those used for quantitative analysis, thus avoiding the use of different, mutually incompatible models.

1.2. Stochastic process algebra

During the last decade, the need for an integrated perspective has motivated an increased interest in combining insights and results from process algebra [9,42,46,5,28]—traditionally focussed on functionality—with techniques for performance modeling and analysis. Process algebra provides a formal apparatus for reasoning about structure and behaviour of systems in a compositional way. Abstraction mechanisms provide means to treat system components as black boxes, making their internal structure invisible. Their algebraic nature allows to reason about specifications in an equational way, thus allowing transformation and verification. Stochastic process algebras [37,40,8,7,16,35] are aimed to overcome the lack of hierarchical, compositional facilities in performance modeling. In these process algebras, time and probability are integrated by considering delays of a continuous probabilistic nature. Typically, a non-negative real-valued rate r is associated to an action a that probabilistically determines the delay prior to a; the term a_r; p denotes that action a is offered after a delay governed by a negative exponential distribution of rate r. A formal semantics maps terms onto labelled transition systems where transitions are labelled with pairs of actions and rates. By omitting the action labels—but keeping the rate information—one obtains a (time-homogeneous) continuous-time Markov chain (CTMC) for which steady-state and transient performance metrics can be obtained using traditional techniques [54]. These *Markovian* process algebras thus provide a compositional, integrated specification formalism for describing CTMCs; recent surveys can be found in [14,36,41].

1.3. The need for non-exponential distributions

Although exponential distributions yield analytically tractable models (i.e., CTMCs), and are useful for many applications, they are not realistic for modeling many phenomena in an adequate way. System parameters such as sizes of data files stored on web servers, job service times in general-purpose computing environments, and node degrees of certain graph structures (such as hyperlinks

of web-pages), exhibit heavy-tail distributions, i.e., distributions with a very high variance [18]. If one observes heavy-tailed inter-arrivals, then the longer one has waited, the longer we should expect to wait—the expectation paradox. Instead, for exponential distributions the waiting time does not play any role, due to the memoryless property. In addition, phenomena such as timeouts in communication protocols, hard deadlines in real-time systems, human response times or the variability of the delay of sound and video frames (so-called jitter) in modern multi-media communication systems are typically described by non-memoryless distributions such as deterministic (for timeouts and deadlines), log-normal (for human behaviour), or normal (for jitter) distributions. Finally, in many cases the distribution is only partially known, and appropriate approximations—those with "maximal indeterminacy"—are needed.[2] The exponential distribution is such approximation only in cases where only the mean is known (of a positive random variable). For other cases, normal distributions (if mean and variance are known) or uniform distributions (if only minimum and maximum are known) are appropriate.

1.4. Process algebra with general distributions

This paper presents a stochastic process algebra in which the delays of actions are determined by a continuous or discrete probability distribution of *general* nature. The incorporation of general distributions in a process algebraic framework is non-trivial, primarily because a mapping onto (an extension of) labelled transition systems is not straightforward. In traditional process algebras, parallel composition can be rewritten into the primitive operations choice and prefix. This principle, in full generality known as the expansion law [46], has been widely accepted and is essential to process algebraic verification purposes [5]. Changing the role of prefix into $a_F; p$ where F is a general distribution determining the delay prior to the offering of action a—like in Markovian process algebra—yields a setting in which the expansion law does no longer hold in general, e.g.,

$$a_F; p \;||_\varnothing\; b_G; q \neq a_F; (p \;||_\varnothing\; b_G; q) + b_G; (a_F; p \;||_\varnothing\; q)$$

for arbitrary distributions F and G. After the delay imposed by F in the left-hand process, the residual delay has to be taken into account to correctly determine the remaining delay before process q becomes enabled. Due to the memoryless property of negative exponential distributions, Markovian process algebras do not suffer from this problem.

1.5. Our approach

To overcome these problems we syntactically distinguish:

- the start of a probabilistic delay,
- the completion of a probabilistic delay, and
- the occurrence of immediate actions.

[2] In information-theoretic terminology, such approximations maximise entropy [53].

This idea has first been brought up in [23], and has been extended and refined later in [19,24]. To keep track of delays, *clock* variables are used. A clock is initialised by sampling a probability distribution function, and starts counting down once initialised. All clocks count down at the same pace. For C a finite set of clocks, $C \mapsto p$ denotes the process that after the expiration of all clocks in C behaves like p, and $\{\!|C|\!\}\, p$ denotes the process that behaves like p after any clock x in C has been initialised. The prefix $a_F; p$ is thus written as $\{\!|x|\!\}\, (\{x\} \mapsto a; p)$, where clock x is initialised by sampling from distribution F. Parallel composition is treated using the principle of interleaving, e.g., for $p' = \{x\} \mapsto a; p$ and $q' = \{y\} \mapsto b; q$ we have:

$$\{\!|x|\!\}\, p' \;||_{\varnothing}\; \{\!|y|\!\}\, q' = \{\!|x, y|\!\}\, \big(\{x\} \mapsto a; (p \;||_{\varnothing}\; q') + \{y\} \mapsto b; (p' \;||_{\varnothing}\; q)\big).$$

In the right-hand expression, initially both clocks x and y are initialised simultaneously and start counting down. If clock x expires first, action a happens, and a state is reached in which clock y records the remaining time until action b is enabled. A symmetric scenario appears when clock y expires first. This principle can be applied to parallel composition with synchronisation, thus yielding an expansion law in its full generality. In addition, the aforementioned separation yields an orthogonal extension of traditional process algebra, results in a framework in which stochastic choice (resolved by the race policy) and non-deterministic choice both exist, and provides a natural stochastic interpretation to synchronisation of two random delays. Synchronisation algebraically amounts to

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

That is, synchronisation takes place as soon as all partners are ready to participate, i.e., once both clocks x and y have expired.[3] This type of synchronisation is known as patient communication [39].

1.6. Contributions of this paper

This paper presents the syntax, formal semantics and algebraic theory of our formalism, named *stochastic process algebra for discrete-event simulation* and symbolised by ⌂. The semantics of ⌂ is given by a structured operational semantics that maps terms onto *stochastic automata*. Finite semantic objects are thus obtained in a comparable way to regular processes in traditional process algebra. Stochastic automata have been extensively treated in the accompanying article [22] and are strongly based on timed automata [3]. As generalised semi-Markov processes (GSMPs) [29,57] are a proper subset of stochastic automata (see [22]), the process algebra ⌂ can be considered as high-level specification formalism for GSMPs. We discuss the congruence properties of the equivalence relations—all variants of strong bisimulation [46]—on stochastic automata as introduced in [22]. We show that stochastic automata with denumerable branching and ⌂ are equally expressive up to structural bisimulation, the finest equivalence relation. The semantics of a ⌂ term up to α-conversion, i.e., renaming of clocks, is shown to be unique (modulo symbolic bisimulation). This

[3] The random variable that determines the delay prior to the synchronised action equals the maximum of the random variables in the individual processes.

result entails that the simple and intuitive operational semantics also apply to terms with (clock) name clashes. Axiomatisations are presented for the congruence relations, and for structural bisimulation it is shown that this equation system is complete. As a major result of this axiomatisation, an expansion law is obtained, a result that, with the sole exception of [10], has been limited to Markovian process algebras.

1.7. Organisation of the paper

Section 2 summarises the main concepts introduced in [22]. Section 3 introduces the syntax and semantics of ♤ and discusses some congruence relations. Section 4 provides several axiom systems, discusses soundness and completeness, and studies several derived properties. Section 5 discusses related work and Section 6 concludes the paper. This paper is based on the extended abstract [24] and the dissertation [19].

2. Preliminaries

This section briefly recalls probabilistic transition systems and stochastic automata as fully described in [22].

2.1. Probabilistic transition systems

Let $Prob(H)$ be the set of probability spaces $(\Omega, \mathcal{F}, P)$ such that $\Omega \subseteq H$. A *probabilistic transition system* (PTS, for short) is a structure $PTS = (\Sigma, \Sigma', \mathcal{L}, T, \longrightarrow)$ where:

1. Σ is the set of *probabilistic states.*
2. Σ' is the set of *non-deterministic states* such that $\Sigma \cap \Sigma' = \varnothing$.
3. $\mathcal{L}$ is a set of *labels.*
4. $T : \Sigma \to Prob(\Sigma')$ is a (total) function, called *probabilistic transition relation.*
5. $\longrightarrow \subseteq \Sigma' \times \mathcal{L} \times \Sigma$ is the *labelled* (*or non-deterministic*) *transition relation.*

The pair (PTS, σ_0) with initial probabilistic state $\sigma_0 \in \Sigma$ is called a *rooted PTS*. We use the shorthand notations $\sigma' \xrightarrow{\ell} \sigma$ for $\langle \sigma', \ell, \sigma \rangle \in \longrightarrow$, $\sigma' \xrightarrow{\ell}$ for $\exists \sigma.\ \sigma' \xrightarrow{\ell} \sigma$, and $\sigma' \not\xrightarrow{\ell}$ for $\neg(\sigma' \xrightarrow{\ell})$.

2.1.1. Probabilistic bisimulation

Let $\mu : \Sigma \times \wp(\Sigma') \to [0,1]$ be defined by

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

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 $\langle \sigma_1, \sigma_2 \rangle \in R \cap (\Sigma \times \Sigma)$:

$$\mu(\sigma_1, \cup S) = \mu(\sigma_2, \cup S) \quad \text{for all } S \subseteq \Sigma'/R$$

and for any $\langle \sigma_1, \sigma_2 \rangle \in R \cap (\Sigma' \times \Sigma')$:

$$\text{for all } \ell \in \mathcal{L},\ \sigma_1 \xrightarrow{\ell} \sigma_1' \text{ implies } \sigma_2 \xrightarrow{\ell} \sigma_2' \text{ and } \langle \sigma_1', \sigma_2' \rangle \in R \quad \text{for some } \sigma_2' \in \Sigma.$$

States σ_1 and σ_2 are *probabilistically bisimilar*, notation $\sigma_1 \sim_p \sigma_2$, if there exists a probabilistic bisimulation R with $\langle \sigma_1, \sigma_2 \rangle \in R$. $(PTS_1, \sigma_1) \sim_p (PTS_2, \sigma_2)$ if and only if $\sigma_1 \sim_p \sigma_2$ in the (disjoint) union of PTS_1 and PTS_2. $\sim_p$ can be shown to be the largest probabilistic bisimulation and to be an equivalence.

2.2. Stochastic automata

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

- $\mathcal{S}$ is a set of *locations*.
- $\mathcal{C}$ is a set of (*random*) *clocks*; for each $x \in \mathcal{C}$ there is a *distribution function* F_x.
- $\mathcal{A}$ is a set of *actions*.
- $\rightarrow \subseteq \mathcal{S} \times (\mathcal{A} \times \wp_{\text{fin}}(\mathcal{C})) \times \mathcal{S}$ is the set of *edges*.
- $\kappa : \mathcal{S} \to \wp_{\text{fin}}(\mathcal{C})$ is the *clock setting function*.

A *rooted stochastic automaton* is a tuple (SA, s_0) where $s_0 \in \mathcal{S}$ is the initial location. Let $s \xrightarrow{a,C} s'$ abbreviate $(s, a, C, s') \in \rightarrow$ and $s \xrightarrow{a,C}$ be a shorthand for $\exists s'.\ s \xrightarrow{a,C} s'$.

2.2.1. Semantics

Let $\mathbf{V}$ be the set of valuations $v : \mathcal{C} \to \mathbb{R}$. For valuation v and $d \in \mathbb{R}$, let valuation $v - d$ be defined as $(v-d)(x) \stackrel{\text{def}}{=} v(x) - d$. Assume the set C of clocks C can be ordered and let $\vec{C}$ denote its ordered set. For $|C| = n$ and $\vec{d} \in \mathbb{R}^n$, the valuation $v[\vec{C} \leftarrow \vec{d}]$ is defined by:

$$v[\vec{C} \leftarrow \vec{d}](x) \stackrel{\text{def}}{=} \begin{cases} \vec{d}(i) & \text{if } x = \vec{C}(i), \quad \text{for some } i \in \{1, \ldots, n\}, \\ v(x) & \text{otherwise,} \end{cases}$$

where $\vec{C}(i)$ and $\vec{d}(i)$ denote the ith element of $\vec{C}$ and $\vec{d}$, respectively.

The *closed behaviour* of SA is defined by the probabilistic transition system $PTS_c(SA) \stackrel{\text{def}}{=} ((\mathcal{S} \times \mathbf{V}), [\mathcal{S} \times \mathbf{V}], \mathcal{A} \times \mathbb{R}_{\geq 0}, T, \rightarrow)$, where T is defined by:

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

Here, $\mathcal{R}(F_{x_1}, \ldots, F_{x_n})$ is the probability space on the Borel algebra $\mathcal{B}(\mathbb{R}^n)$ with the probability measure uniquely defined by $F_{x_1}, \ldots, F_{x_n}$, and $\mathcal{D}_v^s : \mathbb{R}^n \to [\{s\} \times \mathbf{V}]$ is the decoration defined by $\mathcal{D}_v^s(\vec{d}) \stackrel{\text{def}}{=} \left[s, v[\overrightarrow{\kappa(s)} \leftarrow \vec{d}]\right]$ for all $\vec{d} \in \mathbb{R}^n$; and $\rightarrow$ is defined by the rule

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

The predicate $\mathsf{exp}_d(v,C)$ is true if and only if all clocks in C have expired in v after d time units, i.e., $\mathsf{exp}_d(v,C)$ is defined as

$$\forall x \in C.\ \ (v-d)(x) \leq 0$$

and the predicate $\mathsf{mpr}_d(s,v)$ is true if and only if there is no possibility to leave s before d time units, i.e., $\mathsf{mpr}_d(s,v)$ is defined as:

$$\forall d' \in \mathbb{R}_{\geq 0}.\ d' < d \Rightarrow \left(\forall b, C. \left(s \xrightarrow{b,C} \Rightarrow \exists y \in C. (v-d')(y) > 0\right)\right).$$

The *open behaviour* of SA is defined by the probabilistic transition system $PTS_o(SA) \stackrel{\text{def}}{=} ((\mathcal{S} \times \mathbf{V}), [\mathcal{S} \times \mathbf{V}], \mathcal{A} \times \mathbb{R}_{\geq 0}, T, \longrightarrow)$, where T is defined by rule **Prob** above, and $\longrightarrow$ is defined by

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

The only difference between the open and closed semantics is that the constraint of maximal progress is present in the inference rule **Closed** but not in **Open**. In the open behaviour, non-deterministic transitions with different time labels may leave the same state, whereas this is impossible in the closed behaviour.

2.2.2. Equivalences

Locations s_1 and s_2 are *closed p-bisimilar*, notation $s_1 \sim_c s_2$, if $(s_1, v) \sim_p (s_2, v)$ for every $v \in \mathbf{V}$, where (s_1, v) and (s_2, v) are probabilistic states in $PTS_c(SA)$. Similarly, s_1 and s_2 are *open p-bisimilar*, notation $s_1 \sim_o s_2$, if $(s_1, v) \sim_p (s_2, v)$ for every $v \in \mathbf{V}$, where (s_1, v) and (s_2, v) are probabilistic states in $PTS_o(SA)$. Symmetric relation $R \subseteq \mathcal{S} \times \mathcal{S}$ is a *structural bisimulation* if for any acti on a and set C of clocks, whenever $\langle s_1, s_2 \rangle \in R$ we have $\kappa(s_1) = \kappa(s_2)$ and

$$s_1 \xrightarrow{a,C} s_1' \text{ implies } \exists s_2' \in \mathcal{S}.\ \ (s_2 \xrightarrow{a,C} s_2' \text{ and } \langle s_1', s_2' \rangle \in R).$$

Locations s_1 and s_2 are *structurally bisimilar*, notation $s_1 \sim_s s_2$, if there exists a structural bisimulation R such that $\langle s_1, s_2 \rangle \in R$. $SA = (\mathcal{S}, \mathcal{A}, \mathcal{C}, \rightarrow, \kappa)$ and $SA' = (\mathcal{S}', \mathcal{A}, \mathcal{C}, \rightarrow_\alpha, \kappa')$ are *isomorphic*, notation $SA \cong SA'$, if there is a bijection $\mathfrak{I} : \mathcal{S} \to \mathcal{S}'$, such that $s \xrightarrow{a,C} s' \iff \mathfrak{I}(s) \xrightarrow{a,C}_\alpha \mathfrak{I}(s')$, and $\kappa(s) = \kappa'(\mathfrak{I}(s))$. In [22] we also defined the notion of symbolic bisimulation and denote with $s_1 \sim_\& s_2$ if two locations s_1 and s_2 are *symbolically bisimilar*. Since the definition of symbolic bisimulation is quite involved, we omit it here and refer the reader to [22,19]. For the understanding of this article, it is only important to know that it is a coarser relation than structural bisimilarity and finer than open p-bisimilarity. In fact, the following (strict) inclusions hold: $\cong \subset \sim_s \subset \sim_\& \subset \sim_o \subset \sim_c$.

Table 1
Syntax of ♤

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

3. ♤–Stochastic process algebra for discrete event systems

This section defines the syntax of ♤ and its structured operational semantics using stochastic automata as underlying model. The semantics is defined for terms that do not have name clashes of clocks and it is proven that any term with a name clash can be expressed by an equivalent name-clash-free term. Finally, congruence relations for ♤ are reported.

3.1. Syntax

Terms in ♤ are constructed using the operators of traditional process algebras like CCS, CSP, or ACP plus two new constructs. Let C be a set of clocks. Process $C \mapsto p$ behaves like process p once all clocks in C have expired, i.e., once they all have a non-positive value. Process $\{\!\{C\}\!\}\, p$ sets all clocks in C randomly according to their respective distribution function. $\{\!\{x_1, \ldots, x_n\}\!\}\, p$ is a shorthand for $\{\!\{\{x_1, \ldots, x_n\}\}\!\}\, p$.

Definition 1. Let $\mathcal{A}$ be a set of *actions* and $\mathcal{C}$ a countable set of random clocks. The syntax of $p \in$ ♤ is defined according to the grammar in Table 1 where $a \in \mathcal{A}$ is an *action name*, $C \in \wp_{\mathrm{fin}}(\mathcal{C})$ is a *trigger set* of random clocks, $C' \in \wp_{\mathrm{fin}}(\mathcal{C})$ is a *clock setting* set, $A \subseteq \mathcal{A}$ is a *synchronisation set*, $f : \mathcal{A} \to \mathcal{A}$ is a *renaming function*, and X is a *process variable* belonging to the set $\mathcal{V}$ of process variables. Process variables are defined by *recursive equations* of the form $X = p$ where p is a ♤ term. A set E of recursive equations defines a *recursive specification*. We occasionally consider a distinguished process variable in the recursive specification E called *root*.

The operators $a;\_$, $C \mapsto \_$, $\{\!\{C\}\!\}\,\_$, and $\_[f]$ have precedence over any other operator, and $+$ is the operation with lowest precedence. As the precedence is not always defined, some terms without parentheses, e.g., $a; \{x\} \mapsto X[f]$ or $a; \mathbf{0} \,||_A\, X \,|\!|_B\, \{\!\{x\}\!\}\, Y$ are ambiguous. The shorthand $a(x); p$ stands for $\{\!\{x\}\!\}\, \{x\} \mapsto a; p$.

Definition 2. An occurrence of process variable X is *guarded* in term p if it occurs in q where $a; q$ is a subterm of p. Term p is guarded if all occurrences of process variables in p are guarded. The recursive specification E is guarded if either $X = p \in E$ implies p is guarded, or E can be rewritten into such a form by unfolding.[4]

[4] That is, whenever $X = p \in E$ and Y occurs unguarded in p, $X = p$ is replaced by $X = p[Y/q]$ provided $Y = q \in E$.

Definition 3. A recursive specification E is *regular*[5] if it is finite, guarded, and for any $X = p \in E$, the operators $||_A$, $||\!\!\lfloor_A$, $|_A$, and $\_[f]$ do not occur in p.

Example 4. Consider an automatic switch that controls a light as it can be found in a staircase or corridor in a hotel. People can arrive at any time and press the *on* button to turn on the light or to reset the timer that controls it. The interarrival time is a Poisson process occurring at a rate of one arrival each 30 min. Hence, the time difference between two persons turning on the light is a random variable with (negative) exponential distribution $F_{e,30}(t) = 1 - \mathrm{e}^{-\frac{t}{30}}$. Besides, suppose the light turns automatically off after 2 min. Therefore, this timing is governed by the deterministic distribution $D_2(t) = \mathbf{if}\,(t < 2)\mathbf{then}\,0\,\mathbf{else}\,1$.

This system can be described in ⌂ as follows. The arrival of persons is controlled by clock x with distribution $F_x = F_{e,30}$. The arrival process is defined as:

$$\mathit{Arrival} = \mathit{on}(x);\ \mathit{Arrival}$$

The switch can be turned on at any moment, however it turns itself off after idling exactly 2 min. Therefore, a clock y with distribution function $F_y = D_2$ controls the "*off*" event. The switch is specified by:

$$\mathit{SwitchOff} = \mathit{on};\ \mathit{SwitchOn}$$
$$\mathit{SwitchOn} = \mathit{on};\ \mathit{SwitchOn} + \mathit{off}(y);\ \mathit{SwitchOff}$$

Notice that the timing of the action *on* is governed externally. It only depends on the context in which *Switch* is placed and thus it is not delayed by the *Switch* itself. This is reasonable since people in different hotels may arrive at different rates, or even according to different distributions. The complete system is described by:

$$\mathit{System} = \mathit{Arrival} \underset{\mathit{on}}{||} \mathit{SwitchOff}$$

Example 5. Consider a simple queuing system in which jobs arrive and wait until they are executed by a single server. Assume that the queue has infinite capacity. Jobs arrive with an interarrival time that is determined by some distribution F_a, and the execution time of a job by the server is determined by distribution function F_c. This system is known as a $G/G/1/\infty$ queue where the G's stand for general distributed arrival and service time, respectively, 1 indicates that there is only one server, and ∞ indicates that the queue has infinite capacity. (The curious reader is referred to, e.g., [43,32,17] for more details). We use ⌂ to obtain a hierarchical representation.

[5] This notion can be relaxed by allowing the occurrence of the listed static operators, but this would complicate the definition without extending the expressiveness.

Table 2
Free clock variables in ⌂ terms

$\mathsf{fv}(\mathbf{0}) = \varnothing$	$\mathsf{fv}(C \mapsto p) = C \cup \mathsf{fv}(p)$	$\mathsf{fv}(\{\!\{C\}\!\}\, p) = \mathsf{fv}(p) - C$
$\mathsf{fv}(\mathsf{op}(p)) = \mathsf{fv}(p)$		$(\mathsf{op} \in \{a;\ \_,\ \_[f]\})$
$\mathsf{fv}(p \oplus q) = \mathsf{fv}(p) \cup \mathsf{fv}(q)$		$(\oplus \in \{+, \|\|_A, \lfloor\!\lfloor_A, \lfloor_A\})$
$\mathsf{fv}(X) = \mathsf{fv}(p)$		(provided $X = p$)

The queue can be described by the following recursive specification:

$$\begin{aligned} \mathit{Queue}_0 &= \mathit{in};\ \mathit{Queue}_1 \\ \mathit{Queue}_{n+1} &= \mathit{in};\ \mathit{Queue}_{n+2} + \mathit{out};\ \mathit{Queue}_n \quad (n \geqslant 0) \end{aligned}$$

The subindex n in the process variable Queue_n indicates the number of jobs in the queue. The process that determines the arrival of jobs is similar to the one used in the switch example above. The interarrival time is controlled by clock x with $F_x = F_a$:

$$\mathit{Arrival} = \mathit{in}(x);\ \mathit{Arrival}$$

The server takes a job out of the queue and processes it. The completion time is controlled by clock y with distribution $F_y = F_c$:

$$\mathit{Server} = \mathit{out};\ \mathit{done}(y);\ \mathit{Server}$$

The complete system is described by the process

$$\mathit{QSystem} = (\mathit{Arrival} \,||_{\varnothing}\, \mathit{Server}) \,||_{Q}\, \mathit{Queue}_0$$

where $Q = \{\mathit{in}, \mathit{out}\}$. Notice that process *Arrival* synchronises with *Queue* on the action *in*, while the *Server* does so on action *out*.

The setting of clocks binds clocks in a term. For $x \in C$, any free occurrence of x in p is bound in $\{\!\{C\}\!\}\, p$. Intuitively, clock x is free in p if p has a subterm $(C \cup \{x\}) \mapsto q$ which does not appear in a context $\{\!\{\ldots x \ldots\}\!\}\ \ldots$. The set $\mathsf{fv}(p)$ of *free* (clock) variables in p is defined as the smallest set satisfying the equations in Table 2. For instance, for $p \equiv (\{\!\{x\}\!\}\, \{x, y\} \mapsto a;\ \mathbf{0}) + \{x\} \mapsto b;\ \mathbf{0}$ we have $\mathsf{fv}(p) = \{x, y\}$; note that x also appears as a bound variable.

3.2. Structured operational semantics

The semantics of ⌂ is defined in terms of stochastic automata by using structured operational semantics [47]. To define the semantics of the parallel composition, the auxiliary function noset is considered. Given a process p, $\mathsf{noset}(p)$ denotes a process that behaves like p except that no clock is

Table 3
Auxiliary clock removal operation

$\mathsf{noset}(\mathbf{0}) = \mathbf{0}$	$\mathsf{noset}(a; p) = a; p$	$\mathsf{noset}(\{\!\{C\}\!\}\, p) = \mathsf{noset}(p)$
$\mathsf{noset}(\mathsf{op}(p)) = \mathsf{op}(\mathsf{noset}(p))$		$(\mathsf{op} \in \{C{\mapsto}\_, \_[f]\})$
$\mathsf{noset}(p \oplus q) = \mathsf{noset}(p) \oplus \mathsf{noset}(q)$		$(\oplus \in \{+, \|\|_A, \lfloor\!\lfloor_A, \lfloor_A\})$
$\mathsf{noset}(X) = X_{\mathsf{noset}}$	(provided $X = p$ and $X_{\mathsf{noset}} = \mathsf{noset}(p)$)	

Table 4
Clock setting

$\kappa(\mathbf{0}) = \kappa(a; p) = \varnothing$	$\kappa(\{\!\{C\}\!\}\, p) = C \cup \kappa(p)$	
$\kappa(\mathsf{op}(p)) = \kappa(p)$		$(\mathsf{op} \in \{C{\mapsto}\_, \_[f]\})$
$\kappa(p \oplus q) = \kappa(p) \cup \kappa(q)$		$(\oplus \in \{+, \|\|_A, \lfloor\!\lfloor_A, \lfloor_A\})$
$\kappa(X) = \kappa(p)$	(provided $X = p$)	

set before the first action, cf. the rules in Table 3. Note that $\mathsf{noset}(X)$ defines a new process variable and that for any p we have $\mathsf{noset}(\mathsf{noset}(p)) = \mathsf{noset}(p)$.

The components of the stochastic automaton which is associated to a $\mathcal{D}_n$ term are defined in the following. The function κ and the relation $\longrightarrow$ are defined as the least relations satisfying the rules in Tables 4 and 5, respectively. Note that

$$\kappa(\mathsf{noset}(p)) = \varnothing \quad \text{and} \quad \frac{p \xrightarrow{a,C} p'}{\mathsf{noset}(p) \xrightarrow{a,C} p'}. \tag{1}$$

Besides, $\mathsf{fv}(p) \subseteq \mathsf{fv}(\mathsf{noset}(p)) \subseteq \mathsf{fv}(p) \cup \kappa(p)$ for any p.

Let us explain the inference rules. There is no rule for the process $\mathbf{0}$ as it cannot perform any action. The process $a; p$ can immediately perform an a while evolving into p. Since a is performed immediately, there is no need to wait for the expiration of a clock. Process $C{\mapsto}p$ can perform any action that p can perform, with the restriction that it has to wait until all clocks in the set C have expired. So, if p has to wait for the expiration of all clocks in C' to perform action a, then process $C{\mapsto}p$ has to wait for all clocks in $C \cup C'$. Process $\{\!\{C\}\!\}\, p$ mimics p; their difference with p is solely in the clock setting function, cf. Table 4. $p + q$ behaves like either p or q. $p[f]$ behaves like p except that all actions are renamed according to f. Process X behaves like p, provided it is defined as $X = p$. For parallel composition two situations are distinguished:

- In case a synchronisation takes place, i.e., some action $a \in A$ is performed, both involved processes must be ready to perform a. So, all clocks needed to perform a in both processes have to be expired.

Table 5
Inference rules

$$a;\, p \xrightarrow{a,\varnothing} p$$

$$\frac{p \xrightarrow{a,C'} p'}{C \mapsto p \xrightarrow{a,C \cup C'} p'} \qquad \frac{p \xrightarrow{a,C'} p'}{\{\!|C|\!\}\, p \xrightarrow{a,C'} p'}$$

$$\frac{p \xrightarrow{a,C} p'}{p + q \xrightarrow{a,C} p'} \qquad \frac{p \xrightarrow{a,C} p'}{q + p \xrightarrow{a,C} p'}$$

$$\frac{p \xrightarrow{a,C} p'}{p \,||_A\, q \xrightarrow{a,C} p' \,||_A\, \mathsf{noset}(q)}\ a \notin A \qquad \frac{p \xrightarrow{a,C} p'}{q \,||_A\, p \xrightarrow{a,C} \mathsf{noset}(q) \,||_A\, p'}\ a \notin A$$

$$\frac{p \xrightarrow{a,C} p' \quad q \xrightarrow{a,C'} q'}{p \,||_A\, q \xrightarrow{a,C \cup C'} p' \,||_A\, q'}\ a \in A$$

$$\frac{p \xrightarrow{a,C} p'}{p \,|\!|_A\, q \xrightarrow{a,C} p' \,||_A\, \mathsf{noset}(q)}\ a \notin A \qquad \frac{p \xrightarrow{a,C} p' \quad q \xrightarrow{a,C'} q'}{p \,|_A\, q \xrightarrow{a,C \cup C'} p' \,||_A\, q'}\ a \in A$$

$$\frac{p \xrightarrow{a,C} p'}{p[f] \xrightarrow{f(a),C} p'[f]} \qquad \frac{p \xrightarrow{a,C} p'}{X \xrightarrow{a,C} p'}\ \text{provided } X = p$$

- If a process carries out an action not in A, it does so autonomously. Naively, this yields the following traditional operational rule:

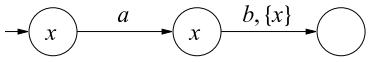
$$\frac{p \xrightarrow{a,C} p'}{p \,||_A\, q \xrightarrow{a,C} p' \,||_A\, q}$$

for $a \notin A$. This would, however, lead to a situation in which all clocks in p' and q are reset in the resulting location. This is incorrect for the clocks in q; by doing so, the elapse of time since the clocks of q were set (when reaching $p \,||_A\, q$) is neglected. To solve this problem, the state $p' \,||_A\, \mathsf{noset}(q)$ is reached instead where the use of $\mathsf{noset}(q)$ avoids the setting of the clocks in q, i.e., $\kappa(\mathsf{noset}(q)) = \varnothing$, cf. Table 4.

The semantics of $\hookuparrow$ is not yet well defined. Due to the binding nature of the clock setting operation, $\hookuparrow$ suffers from the capture of clock variables. Consequently, clocks may be wrongly bound in the derived stochastic automaton. Consider, e.g.,

$$p_1 \equiv \{\!|x|\!\}\, a;\, \{x\} \mapsto \{\!|x|\!\}\, \{x\} \mapsto b;\, \mathbf{0}, \tag{2}$$

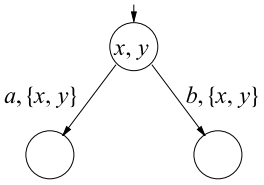
where the second occurrence of x is intended to be bound to the outermost clock setting (as indicated by the grey arrow). The rules in Table 5, however, yield:



in which x is captured by the innermost clock setting (as indicated by the black arrow in Eq. (2)). Clocks that are set at different places are considered to be different (and hence, independent), even when they have the same name. Capture may also occur in summation. A free clock in one of the summands may be wrongly bound to a clock setting in the other summand as shown by the following process:

$$\mathsf{fv}\quad p_2 \equiv \{\!|x, y|\!\}\,\{x, y\}\mapsto a; \mathbf{0} + \{\!|y|\!\}\,\{x, y\}\mapsto b; \mathbf{0} \tag{3}$$

A naive interpretation of process p_2 would yield the following automaton:



which corresponds to process $p_3 \equiv \{\!|x, y|\!\}\,(\{x, y\}\mapsto a; \mathbf{0} + \{x, y\}\mapsto b; \mathbf{0})$. Notice that, in addition, another unexpected phenomenon occurs. Clock y in the left summand of p_2 has been "unified" with clock y in its right summand. Clocks are independent random variables that can take different values, even when they have the same distribution. While in process p_2 actions a and b will usually become enabled at different times—and this happens with probability 1 if x and y are continuously distributed—in process p_3 they are always enabled at the same time. Similar problems arise when replacing $+$ in (3) by $||$.

We also point out the following technical issue. The stochastic automaton is defined such that, for every location s, $\kappa(s)$ is finite. Some unguarded (infinite) recursive specifications, however, may induce infinite sets, e.g.,

$$\{X_n = X_{n+1} + \{\!|x_n|\!\}\,\{x_n\}\mapsto a_n; \mathbf{0} \mid n \in \mathbb{N}\}.$$

Given these complications, we characterise the set of processes whose semantics is definable, i.e., the processes that do not cause such problems. We first syntactically characterise the processes that are in conflict and then generalise this towards a semantic notion. Process p suffers from *local conflict* of (clock) variables if $\mathrm{lcv}(p)$ holds, where lcv is the predicate defined by the rules in Table 6. Otherwise, p is *locally free of conflict*. For the example processes before, $\mathrm{lcv}(p_2)$ and $\neg\mathrm{lcv}(p_1)$. p is *locally definable*, denoted $\mathsf{ldef}(p)$, if and only if $\neg\mathrm{lcv}(p)$ and $\kappa(p)$ is finite. Locally definability does

Table 6
Conflict of clock variables in $\hat{\triangle}$

$$\frac{C \cap \kappa(p) \neq \varnothing}{\mathrm{lcv}(C \mapsto p)} \qquad \frac{\mathrm{lcv}(p) \quad X = p}{\mathrm{lcv}(X)}$$

$$\frac{\mathrm{lcv}(p) \quad \mathsf{op} \in \{C \mapsto \_, \{\!|C|\!\}\_, \_[f]\}}{\mathrm{lcv}(\mathsf{op}(p))} \qquad \frac{\kappa(p) \cap \mathsf{fv}(q) \neq \varnothing \quad \oplus \in \{+, ||_A, \underline{||}_A, |_A\}}{\mathrm{lcv}(p \oplus q) \quad \mathrm{lcv}(q \oplus p)}$$

$$\frac{\mathrm{lcv}(p) \quad \oplus \in \{+, ||_A, \underline{||}_A, |_A\}}{\mathrm{lcv}(p \oplus q) \quad \mathrm{lcv}(q \oplus p)} \qquad \frac{\kappa(p) \cap \kappa(q) \neq \varnothing \quad \oplus \in \{+, ||_A, \underline{||}_A, |_A\}}{\mathrm{lcv}(p \oplus q) \quad \mathrm{lcv}(q \oplus p)}$$

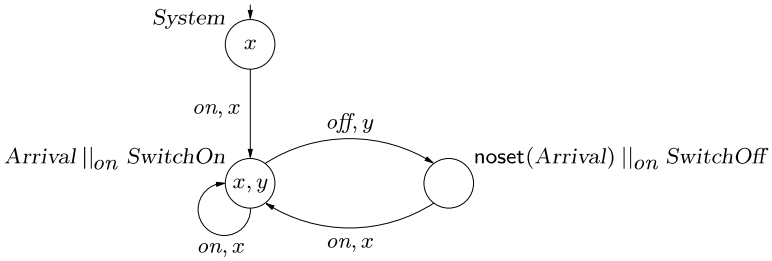


Fig. 1. Stochastic automaton defined by process *System* in Example 4.

not characterise all problems mentioned before (e.g., process p_1). To obtain a full characterisation, all possible derivaties are considered.

Definition 6. $p \in \hat{\triangle}$ is definable, denoted $\mathsf{def}(p)$, if for every derivation sequence $p \equiv p_0 \xrightarrow{a_1, C_1} p_1 \cdots p_{n-1} \xrightarrow{a_n, C_n} p_n$ we have $\mathsf{ldef}(p_i)$ for all $0 \leq i \leq n$.

To check whether process p is (globally) definable, every term in any derivative of p needs to be locally free of conflict and have a finite clock setting. For instance, process p_1 is not definable as a process with local conflict is reached after a single step. Let $\hat{\triangle}^{\mathsf{def}} = \{p \in \hat{\triangle} \mid \mathsf{def}(p)\}$, that is, the set of all definable $\hat{\triangle}$ processes.

Definition 7. The semantics of $p \in \hat{\triangle}^{\mathsf{def}}$ is defined by the rooted stochastic automaton $(SA(\hat{\triangle}^{\mathsf{def}}), p)$ where $SA(\hat{\triangle}^{\mathsf{def}}) = (\hat{\triangle}^{\mathsf{def}}, \mathcal{A}, \mathcal{C}, \rightarrow, \kappa)$ with $\mathcal{A}$ the set of actions, $\mathcal{C}$ the set of clocks, and $\rightarrow$ and κ the least relations satisfying the rules in Tables 4 and 5, respectively.

Example 8. The stochastic automaton defined by the process *System* in Example 4 is given in Fig. 1. The automaton for process *QSystem* of Example 5 is depicted in Fig. 2.

3.3. Clock renaming

Definition 7 provides a semantics for every definable process. This is somewhat restrictive and can be relaxed to terms modulo clock renaming, i.e., α-conversion. The semantics modulo α-conversion in itself is not our interest, but is used to prove that for any process that suffers from conflict of variables, a unique equivalent definable term does exist. As a consequence, the semantics (modulo

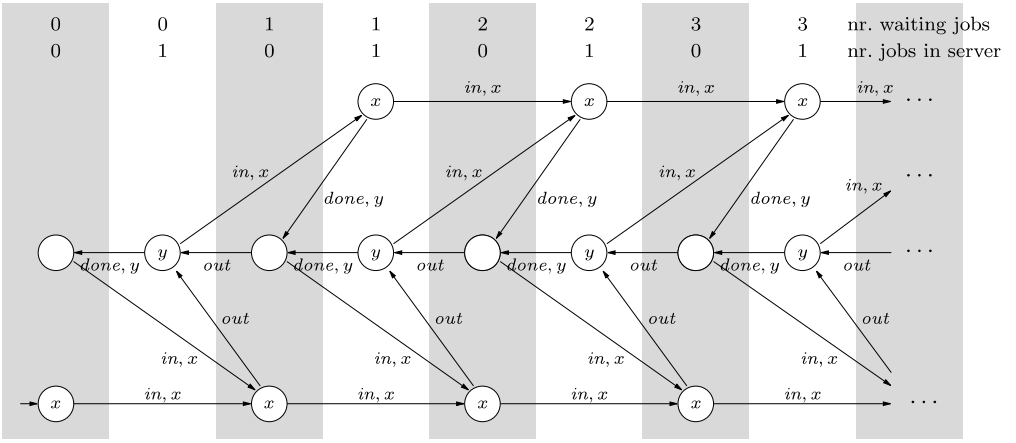


Fig. 2. Stochastic automaton defined by process *QSystem* in Example 5.

symbolic bisimulation) for terms that suffer from the conflict of clock names can be obtained by the recipe of Definition 7. For instance, term $p \equiv \{\!|x, y|\!\}\,(\{x, y\} \mapsto a; \mathbf{0} + \{\!|y|\!\}\,\{x, y\} \mapsto b; \mathbf{0})$ is not definable, but after a simple renaming of clocks, e.g., replace x by w and y by z, the term $q \equiv \{\!|w, z|\!\}\,(\{w, z\} \mapsto a; \mathbf{0} + \{\!|y|\!\}\,\{w, y\} \mapsto b; \mathbf{0})$ is obtained, which exhibits the same behaviour as p provided $F_x = F_w$ and $F_y = F_z$.

For terms $p, q \in \hat{\mathcal{O}}$ we write $p \simeq_\alpha q$ if and only if q can be obtained from p by an *appropriate* renaming of clocks. A renaming of clock x into y is appropriate whenever $F_x = F_y$. Renaming of (sets of) clocks in $\hat{\mathcal{O}}$ terms can be defined in the standard inductive way, e.g., $(p \,||_A\, q)[C'/C] = (p[C'/C]) \,||_A\, (q[C'/C])$. For the new operators we have: $(C \mapsto p)[C'/C] = C' \mapsto p[C'/C]$ and $(\{\!|C|\!\}\,p)[C/C'] = \{\!|C'|\!\}\,(p[C/C'])$ provided $C' \cap (\mathsf{fv}(p) - C) = \varnothing$, that is, the new clock names should not coincide with the free variables in p.

Theorem 9. *For every guarded term $p \in \hat{\mathcal{O}}$, there exists $q \in \hat{\mathcal{O}}$ such that $\mathsf{Idef}(q)$ and $p \simeq_\alpha q$.*

Proof. Using induction on the size of p we first prove that for every guarded p and every finite $C \subseteq \mathcal{C}$, there exists a q such that $p \simeq_\alpha q$, $\neg\mathrm{lcv}(q)$, and $\kappa(q) \cap C = \varnothing$. The idea is that C carries the possible variables that may introduce conflict along the proof tree of $p \simeq_\alpha q$. As an example, we consider the following cases:

- $C' \mapsto p$. By the induction hypothesis, there is a q such that $p \simeq_\alpha q$, $\neg\mathrm{lcv}(q)$, and $\kappa(q) \cap C'' = \varnothing$ for any given finite $C'' \subseteq \mathcal{C}$. In particular, take $C'' = C \cup C'$. Hence, $C' \mapsto p \simeq_\alpha C' \mapsto q$, $\neg\mathrm{lcv}(C' \mapsto q)$ (since $\neg\mathrm{lcv}(q)$ and $\kappa(q) \cap C' = \varnothing$), and $\kappa(C' \mapsto q) \cap C = \kappa(q) \cap C = \varnothing$.
- $\{\!|C'|\!\}\,p$. By the induction hypothesis, $[C^*/C'](p) \simeq_\alpha q$, $\neg\mathrm{lcv}(q)$, and $\kappa(q) \cap C = \varnothing$. W.l.o.g. assume $C^* \cap C = \varnothing$. Then $\{\!|C'|\!\}\,p \simeq_\alpha \{\!|C^*|\!\}\,q$, $\neg\mathrm{lcv}(\{\!|C'|\!\}\,q)$, and $\kappa(\{\!|C^*|\!\}\,q) \cap C = (C^* \cup \kappa(q)) \cap C = \varnothing$.

It remains to show that $\kappa(q)$ is finite and that for every edge $q \xrightarrow{a,C}$, C is finite. This can easily be proved by structural induction on q exploiting that q is guarded (since $p \simeq_\alpha q$ and p is guarded). Consequently, case $q \equiv X$ does not occur. □

The theorem does not hold for unguarded recursion, e.g., process X defined by $X = X + \{\!|x|\!\}\,\{x\}\mapsto a; \mathbf{0}$, suffers from the local conflict of variables and cannot be α-converted into a (locally) definable process. The same applies to X_0 defined by $\{X_n = X_{n+1} + \{\!|x_n|\!\}\,\{x_n\}\mapsto a; \mathbf{0} \mid n \in \mathbb{N}\}$.

Extend the inference rules in Table 5 with the rule:

$$\textbf{alpha}\quad \frac{p \xrightarrow{a,C} p' \quad p' \simeq_\alpha q' \quad \neg\mathrm{lcv}(p) \quad \neg\mathrm{lcv}(q')}{p \xrightarrow{a,C} q'},$$

that adds transitions to target terms that are α-congruent (and locally conflict-free) to existing target terms. Note that this rule requires $\neg\mathrm{lcv}(p)$ and $\neg\mathrm{lcv}(q')$, but not (the stronger) $\mathsf{ldef}(p)$ and $\mathsf{ldef}(q')$, because the fact that $\kappa(p)$ is infinite cannot be changed by α-conversion.

Definition 10. $p \in \mathbb{O}$ is *α-definable*, denoted $\alpha\mathsf{def}(p)$, if $p \simeq_\alpha p_0$ for some $p_0 \in \mathbb{O}$, and whenever $p_0 \xrightarrow{a_1,C_1} p_1 \cdots p_{n-1} \xrightarrow{a_n,C_n} p_n$ with $\mathsf{ldef}(p_i)$ for all $0 \le i < n$, there is $q \in \mathbb{O}$ such that $p_n \simeq_\alpha q$. [6]

Let $\mathbb{O}^\alpha = \{p \in \mathbb{O} \mid \alpha\mathsf{def}(p)\}$ and $\mathbb{O}^{\alpha-} = \{p \in \mathbb{O} \mid \alpha\mathsf{def}(p) \wedge \neg\mathrm{lcv}(p)\}$. Terms in $\mathbb{O}^{\alpha-}$ are α-definable, and, more importantly, are locally free of conflict. These terms are the ones that play a central role in the following definition.

Definition 11. The semantics of $p \in \mathbb{O}^\alpha$ such that $p \simeq_\alpha q \in \mathbb{O}^{\alpha-}$ is defined by the rooted stochastic automaton $(SA(\mathbb{O}^{\alpha-}), q)$ where $SA(\mathbb{O}^{\alpha-}) = (\mathbb{O}^{\alpha-}, \mathcal{A}, \mathcal{C}, \rightarrow_\alpha, \kappa)$ with $\rightarrow_\alpha = \rightarrow \cap\, (\mathbb{O}^{\alpha-} \times \mathcal{A} \times \mathcal{P}_{fin}(\mathcal{C}) \times \mathbb{O}^{\alpha-})$. The other components are as before.

The next theorem states that for every α-definable term its semantics (up to α-conversion) is unique up to symbolic bisimulation:

Theorem 12. *For $p \in \mathbb{O}^\alpha$ with $p \simeq_\alpha q$ and $p \simeq_\alpha r$ for $q, r \in \mathbb{O}^{\alpha-}$: $(SA(\mathbb{O}^{\alpha-}), q) \sim_{\&} (SA(\mathbb{O}^{\alpha-}), r)$.*

Proof. Tedious, see [19]. □

The main result of this section states that the original semantics as given in Definition 7 coincides, modulo symbolic bisimulation, with the semantics up to α-conversion. The proof of this result goes along similar lines as that of Theorem 12 and is omitted here.

Theorem 13. *For $p \in \mathbb{O}^{\mathsf{def}}$. $(SA(\mathbb{O}^{\mathsf{def}}), p) \sim_{\&} (SA(\mathbb{O}^{\alpha-}), p)$.*

3.4. Representability

Any stochastic automaton can be represented by a (definable) $\mathbb{O}$ term p whose semantics is structural bisimilar to it.

Theorem 14. *For any rooted stochastic automaton (SA, s_0) with denumerable branching, there exists a recursive specification E in $\mathbb{O}^{\mathsf{def}}$ with root X_{s_0} such that $(SA, s_0) \sim_s (SA(\mathbb{O}^{\mathsf{def}}), X_{s_0})$.*

[6] The edge-relation $\rightarrow$ is the least relation satisfying the rules in Table 5 and the rule **alpha**.

Proof. Let $SA = (\mathcal{S}, \mathcal{A}, \mathcal{C}, \longrightarrow_\alpha, \kappa')$. Since we cannot represent infinite sums directly, define for each location $s \in \mathcal{S}$ a distinct process variable X_s. Define the recursive specification E_{SA} containing root X_{s_0} and the recursive equations

$$X_s = \{\!|\kappa'(s)|\!\} \left(\sum \{ C \mapsto a; X_{s'} \mid s \xrightarrow{a,C}_\alpha s' \} \right). \tag{4}$$

For $n \geqslant 0$ (possibly infinity), $X_0 \stackrel{\text{def}}{=} \sum\{C_i \mapsto a_i; p_i \mid 0 \leq i < n\}$, provided $X_i = X_{i+1} + C_i \mapsto a_i; p_i$, for $0 \leq i < n$; if n is finite, let $X_n = \mathbf{0}$. By construction, E_{SA} is definable. Consider now the relation $R = \{ \langle s, X_s \rangle \mid s \in \mathcal{S} \}$. It is routine to prove that $R \cup R^{-1}$ is a structural bisimulation. □

Corollary 15. *For every $p \in \mathbb{P}^\alpha$ there exists $q \in \mathbb{P}^{\text{def}}$ such that $p \sim_{\&} q$.*

Due to Theorem 9 and this corollary, it follows that any guarded process can equivalently be written as a conflict-free process.

Example 16. Consider the automaton of Fig. 1. Using the recipe in the above proof, we construct the following recursive specification:

$$\begin{aligned}
X_{System} &= \{\!|x|\!\}\ \{x\} \mapsto on; X_{Arrival||_{on}\, SwitchOn} \\
X_{Arrival||_{on}\, SwitchOn} &= \{\!|x, y|\!\}\ (\{x\} \mapsto on; X_{Arrival||_{on}\, SwitchOn} + \{y\} \mapsto off; X_{\text{noset}(Arrival)||_{on}\, SwitchOff}) \\
X_{\text{noset}(Arrival)||_{on}\, SwitchOff} &= \{\!|\varnothing|\!\}\ \{x\} \mapsto on; X_{Arrival||_{on}\, SwitchOn}.
\end{aligned}$$

The rooted stochastic automaton $(SA(\mathbb{P}^{\text{def}}), X_{System})$ is structurally bisimilar to that of Fig. 1.

We stipulate that the recursive specification E_{SA} defined by the variables in (4) is guarded if SA is finitely branching and regular if SA is finite, i.e., if in addition the number of locations is finite.

3.5. Congruences

To prove that structural bisimilarity is a congruence for the $\mathbb{P}$ operators as well as recursion, we resort to results in structured operational semantics, in particular to the so-called *path* format [4]. Let Rel, Rel_i be *transition relations* (as, for instance, $\longrightarrow$), and let $Pred$, $Pred_i$ be *state predicates*. Let op be an n-ary operation. A rule is in *path* format if it has one of the following two forms:

$$\frac{\{p_i \ Rel_i \quad y_i \mid i \in I\} \cup \{Pred_j\, q_j \mid j \in J\}}{\mathsf{op}(x_1, \ldots, x_n) \ Rel \quad p},$$

$$\frac{\{p_i \ Rel_i \quad y_i \mid i \in I\} \cup \{Pred_j\, q_j \mid j \in J\}}{Pred\ \mathsf{op}(x_1, \ldots, x_n)},$$

where $vars = \{x_1, \ldots, x_n\} \cup \{y_i \mid i \in I\}$ is a set of distinct variables (over terms), and p, p_i, q_j are terms with free variables in $vars$.[7] A set of rules is in *path* format if all of its rules are.

[7] We have actually defined the so-called *pure path format*, which is sufficient for our purposes.

The importance of this format is that bisimulation is a congruence for any operation whose semantics is defined with rules in *path* format [4,31,27]. More recently, Rensink [50] has proved a general congruence theorem for recursion with the only additional requirement that the rules should not have *look-ahead*, that is, the free variables of the p_i's and q_j's can only be those in $\{x_1, \ldots, x_n\}$.

Assume that noset is also an operation of $\hat{\Delta}$. It is not difficult to observe that the rules for $\rightarrow$ given in Table 5 as well as the rule in Eq. (1) are in *path* format. Moreover, the definition of κ can be alternatively defined in terms of a proof system in such a way that, for $x \in \mathcal{C}$, $x \in \kappa(\_)$ is a predicate and it defines the same sets as the equations in Table 5. For instance, the case of $p \,||_A\, q$ is defined by rules

$$\frac{x \in \kappa(p)}{x \in \kappa(p \,||_A\, q)}, \qquad \frac{x \in \kappa(q)}{x \in \kappa(p \,||_A\, q)}.$$

Such set of rules is also in *path* format. Thus the new proof system for κ is in *path* format and defines the same stochastic automata as the one in Table 5. As a consequence, structural bisimulation is a congruence for all the $\hat{\Delta}$ operations including noset. Since in addition none of the rules has look-ahead, it is also a congruence for recursion.

Theorem 17. *Let $p, q \in \hat{\Delta}$ with $p \sim_s q$. For any $\hat{\Delta}$-context* $\mathbf{C}[\,]$, $\mathbf{C}[p] \sim_s \mathbf{C}[q]$. *Moreover, $\sim_s$ is a congruence for recursion in $\hat{\Delta}$.*

Like structural and symbolic bisimulation, the different probabilistic bisimulations also extend to $\hat{\Delta}$: $p \sim_c q$ if there are p' and q' such that their behaviours are α-definable, $p \simeq_\alpha p'$, $q \simeq_\alpha q'$, and $p' \sim_c q'$ where p' and q' are locations in the stochastic automaton $SA(\hat{\Delta}^{\alpha-})$—or $SA(\hat{\Delta}^{\mathsf{def}})$, if applicable—, i.e., $(p', v) \sim_p (q', v)$ for any valuation v, where (p', v) and (q', v) are states in $PTS_c(SA(\hat{\Delta}^{\alpha-}))$ (or equivalently in $PTS_c(SA(\hat{\Delta}^{\mathsf{def}}))$). Similarly, $p \sim_o q$ if there are p' and q' such that their behaviours are α-definable, $p \simeq_\alpha p'$, $q \simeq_\alpha q'$, and $p' \sim_o q'$ where p' and q' are locations in the stochastic automaton $SA(\hat{\Delta}^{\alpha-})$ (or $SA(\hat{\Delta}^{\mathsf{def}})$).

Probabilistic bisimilarity for the closed interpretation of stochastic automata is not a congruence. This is illustrated by the following two examples.

Example 18. *$\sim_c$ is not a congruence for $||$.* Processes $p_1 \equiv a; \mathbf{0} + \{\!|x|\!\}\, \{x\}\!\mapsto\! b; \mathbf{0}$ and $p_2 \equiv a; \mathbf{0} + \{\!|x|\!\}\, \{x\}\!\mapsto\! c; \mathbf{0}$ $(b \neq c)$ are closed p-bisimilar if $F_x(0) = 0$, since in both cases only the action a at time 0 can be performed. However, $p_1 \,||_a\, \mathbf{0}$ and $p_2 \,||_a\, \mathbf{0}$ are not. Because there is no possible synchronisation in this parallel process, the execution of action a is preempted. Therefore, b or c may happen (at a certain time greater than 0). This example is depicted in Fig. 3.

Example 19. *$\sim_c$ is not a congruence for $C\!\mapsto\! p$.* Take p_1 and p_2 as in the previous example. Then, $\{y\}\!\mapsto\! p_1 \not\sim_c \{y\}\!\mapsto\! p_2$. To understand this, assume F_x is a uniform distribution in $[1, 2]$. For any valuation v in which y takes a value greater than 2, all edges become enabled at time $d = v(y)$. In particular, edges $\{y\}\!\mapsto\! p_1 \xrightarrow{b,\{x,y\}} \mathbf{0}$ and $\{y\}\!\mapsto\! p_2 \xrightarrow{c,\{x,y\}} \mathbf{0}$.

The above problems arise because a faster action is pre-empted or slowed down by the context, thus giving the possibility for a slower activity to occur that otherwise (i.e., in the process in isolation) would never occur. Open p-bisimilarity has been devised that prevents these cases. Rather than

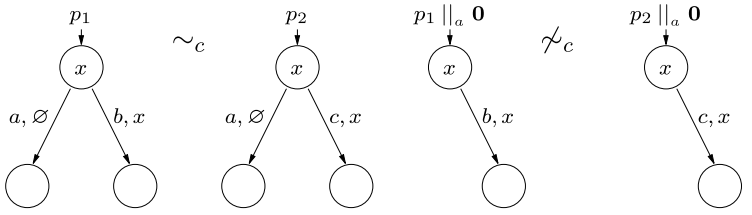


Fig. 3. Closed p-bisimilarity is not a congruence.

taking the fastest action, the open behaviour does not obey the maximal progress condition. In this way, time may pass, allowing slower actions to be observed without the need for interaction to do so. The following theorem shows that this notion indeed is a congruence.

Theorem 20. *Let* $p, q \in \mathcal{O}^{\alpha}$ *such that* $p \sim_o q$. *Let* $\mathbf{C}[]$ *be a* $\mathcal{O}$ *context such that* $\mathbf{C}[p], \mathbf{C}[q] \in \mathcal{O}^{\alpha}$. *Then*, $\mathbf{C}[p] \sim_o \mathbf{C}[q]$.

Proof See [19, Appendix F.2]. □

4. Equational theory for $\mathcal{O}$

This section presents a sound and complete axiomatisation of $\sim_s$ (i.e., structural bisimulation). We first do so for the basic operators of $\mathcal{O}$. An axiomatisation for $\sim_o$ is then given, followed by the axiomatic treatment of the static operators (for $\sim_s$). In particular, we consider a generalisation of the expansion law to $\mathcal{O}$ and discuss the recursive specification principle and the recursive definition principle (as in [6,5]) in $\mathcal{O}$.

4.1. Basic axioms for structural bisimulation

Definition 21. The equational theory for the basic sub-language of $\mathcal{O}$, denoted $\mathcal{O}^b$, is defined by the signature $sig(\mathcal{O}^b)$, containing the constant $\mathbf{0}$, the unary operators $a;\_, C{\mapsto}\_$, and $\{\!|C|\!\}\_$, for every $a \in \mathcal{A}$, and $C \in \wp_{\mathrm{fin}}(\mathcal{C})$, and the binary operation $+$, and the axiom system $ax\ (\mathcal{O}^b)$ as given in Table 7.

The axioms can be explained as follows. The choice is commutative (**A1**) and associative (**A2**). Axiom **A3** states that $+$ is idempotent on action-guarded processes and **A4** states that $\mathbf{0}$ is the neutral element for $+$. In contrast to traditional process algebras, choice is not idempotent in general, e.g., consider $p \equiv \{\!|x|\!\}\,\{x\}{\mapsto}a; \mathbf{0}$ where F_x is a uniform distribution over $[0,2]$. The probability that a occurs in $[0,1]$ in p is $\frac{1}{2}$, whereas in $p + p$ this is $\frac{3}{4}$. Thus, $p \not\sim_s p + p$. Axioms **T1–T5** show how triggering conditions can be simplified. In particular, **T3** defines how to reduce nested triggering conditions into one, and axioms **T4** and **T5** indicate how to shift clock settings and summations out of the scope of a triggering condition. Axiom **CS1** says that it is irrelevant to set an empty set of clocks. **CS2** gathers the clock settings in a single clock setting and **CS3** moves clock settings out of the scope of $+$. The side condition of the latter axiom is needed to avoid a conflict of clock names.

Table 7
Axioms for $\hat{\Diamond}^b$

A1	$p + q = q + p$
A2	$(p + q) + r = p + (q + r)$
A3	$a; p + a; p = a; p$
A4	$p + \mathbf{0} = p$
T1	$C \mapsto \mathbf{0} = \mathbf{0}$
T2	$\varnothing \mapsto p = p$
T3	$C \mapsto C' \mapsto p = C \cup C' \mapsto p$
T4	$C \mapsto \{\!\{C'\}\!\}\, p = \{\!\{C'\}\!\}\, C \mapsto p$ if $C \cap C' = \varnothing$
T5	$C \mapsto (p + q) = C \mapsto p + C \mapsto q$
CS1	$\{\!\{\varnothing\}\!\}\, p = p$
CS2	$\{\!\{C\}\!\}\, \{\!\{C'\}\!\}\, p = \{\!\{C \cup C'\}\!\}\, p$
CS3	$\{\!\{C\}\!\}\, p + \{\!\{C'\}\!\}\, q = \{\!\{C \cup C'\}\!\}\, (p + q)$ if $C \cap (\mathsf{fv}(q) \cup \kappa(q)) = \varnothing$ and $C' \cap (\mathsf{fv}(p) \cup \kappa(p)) = \varnothing$

The following property states that $+$ is idempotent whenever clock settings only occur within the context of a prefix.

Proposition 22. *Let $p \in \hat{\Diamond}^b$ such that every subterm $\{\!\{C\}\!\}\, r$ in p occurs in a subterm of the form $a; q$. Then $ax\ \ (\hat{\Diamond}^b) \vdash p = p + p$.*

Proof. By induction[8] on the structure of p.

Case $\mathbf{0}$. $\mathbf{0} \overset{\text{A4}}{=} \mathbf{0} + \mathbf{0}$

Case $a; p$. $a; p \overset{\text{A3}}{=} a; p + a; p$

Case $C \mapsto p$. $C \mapsto p \overset{\text{ind.}}{=} C \mapsto (p + p) \overset{\text{T5}}{=} C \mapsto p + C \mapsto p$

Case $\{\!\{C\}\!\}\, p$. It does not fall within the hypothesis of the property.

Case $p + q$. $p + q \overset{\text{ind.}}{=} (p + p) + (q + q) \overset{\text{A1,A2}}{=} (p + q) + (p + q)$. □

Theorem 23 (Soundness)**.** *For $p, q \in \hat{\Diamond}^{\mathsf{def}}$. $ax\ \ (\hat{\Diamond}^b) \vdash p = q$ implies $p \sim_s q$.*

Proof. For axiom $p = q$ in $ax\ \ (\hat{\Diamond}^b)$, let $R = \{\langle p, q\rangle, \langle q, p\rangle\} \cup Id$ with Id the identity relation. It can be straightforwardly checked that R is a structural bisimulation for any such axiom. □

Theorem 23 states that $ax\ \ (\hat{\Diamond}^b)$ is sound with respect to $\sim_s$. For establishing completeness of our axiom system, we rely on normal forms, called *basic terms* in the following.

[8] In fact, the proof of this property also requires inductive reasoning which we assume to be standard part of equational reasoning.

Definition 24. The set $B \subseteq \hat{\mathcal{P}}^b$ of *basic terms* is defined inductively using an auxiliary set B' as follows:

- $\mathbf{0} \in B'$
- $p \in B, C \subseteq \wp_{\mathrm{fin}}(\mathcal{C})$ and $a \in \mathcal{A} \quad \Rightarrow \quad C \mapsto a; p \in B'$
- $p, q \in B' \quad \Rightarrow \quad p + q \in B'$
- $p \in B', C \subseteq \wp_{\mathrm{fin}}(\mathcal{C}) \quad \Rightarrow \quad \{\!|C|\!\}\, p \in B.$

A basic term has the following format (modulo the axioms **A1**–**A4**):

$$p = \{\!|C|\!\} \left(\sum_{i \in I} C_i \mapsto a_i; p_i \right),$$

where I is a finite index-set and p_i is a basic term. Terms in B', also called *pre-basic terms*, are like basic terms except that clock settings only occur within the context of an action prefix. They have the following format: $p = \sum_{i \in I} C_i \mapsto a_i; p_i$, with I and p_i as before.

Theorem 25. *For $p \in \hat{\mathcal{P}}^b \cap \hat{\mathcal{P}}^{\mathsf{def}}$, there is a basic term $q \in \hat{\mathcal{P}}^{\mathsf{def}}$ such that ax $(\hat{\mathcal{P}}^b) \vdash p = q$.*

Proof. By structural induction on p. The interesting cases are the triggering condition and choice.

Case $C \mapsto p$. From $\mathsf{def}(C \mapsto p)$ it follows $\mathsf{def}(p)$. From the induction hypothesis, we may thus assume that

$$p = \{\!|C'|\!\} \left(\sum_{i \in I} C_i \mapsto a_i; p_i \right),$$

where p_i is a basic term such that $\mathsf{def}(p_i)$ (for all $i \in I$). As $C \cap C' = \varnothing$ it follows:

$$\begin{array}{lcl} C \mapsto p & \overset{\text{T4}}{=} & \{\!|C'|\!\} \left(C \mapsto \sum_{i \in I} C_i \mapsto a_i; p_i \right) \\ & \overset{\text{T5,T3}}{=} & \{\!|C'|\!\} \left(\sum_{i \in I} (C \cup C_i) \mapsto a_i; p_i \right). \end{array}$$

From the definition of lcv, it directly follows that the obtained term is locally definable. Using $\mathsf{def}(p_i)$ for all $i \in I$, it follows that the term is also definable.

Case $p + q$. From $\mathsf{def}(p + q)$, it follows $\mathsf{def}(p)$ and $\mathsf{def}(q)$. From the induction hypothesis, it may be assumed that:

$$p = \{\!|C|\!\} \left(\sum_{i \in I} C_i \mapsto a_i; p_i \right) \quad \text{and} \quad q = \{\!|C'|\!\} \left(\sum_{j \in J} C'_j \mapsto b_j; q_j \right),$$

where p_i and q_j such that $\mathsf{def}(p_i)$ and $\mathsf{def}(q_j)$, for all $i \in I$ and $j \in J$. Since $C \cap C' = \varnothing$, $C \cap \mathsf{fv}(q) = \varnothing$, and $C' \cap \mathsf{fv}(p) = \varnothing$:

$$\begin{array}{lll} p + q & \overset{\text{ind.}}{=} & \{\!|C|\!\} \left(\sum_{i\in I} C_i \mapsto a_i;\, p_i\right) + \{\!|C'|\!\} \left(\sum_{j\in J} C'_j \mapsto b_j;\, q_j\right) \\ & \overset{\text{CS3}}{=} & \{\!|C \cup C'|\!\} \left(\sum_{i\in I} (C_i \mapsto a_i;\, p_i) + \sum_{j\in J} (C'_j \mapsto b_j;\, q_j)\right). \end{array}$$

The latter term is locally definable as the summands initialise an empty set of clocks, and $C \cap C' = \varnothing$. As $\mathsf{def}(p_i)$ and $\mathsf{def}(q_j)$, it follows that it is also definable. □

Note that for any derivative q of $p \in \mathbb{P}^b$, we have $\kappa(q)$ is finite. This follows from the fact that p does not include process variables, and hence, no infinite clock settings may be obtained. The set of sequential finite terms whose behaviour is definable equals in fact $\mathbb{P}^b \cap \mathbb{P}^{\mathsf{def}}$.

Definition 26. For pre-basic terms p and q: p is a *summand* of q, notation $p \leq q$, if $ax\ \ (\mathbb{P}^b) \vdash q = q + p$.

Reflexivity of $\leq$ follows from Proposition 22 while transitivity follows from axioms **A1**, **A2** (and substitutivity). $\leq$ is thus a partial ordering on pre-basic terms. In addition, from axiom **A4** it follows that **0** is the unique minimal element of $\leqslant$. The notion of summand relates to the operational semantics as stated in the following proposition whose proof is straightforward.

Proposition 27. *For pre-basic term* $p \in \mathbb{P}^{\mathsf{def}}$:

1. $C \mapsto a;\, p' \ \leq\ p$ *implies* $p \xrightarrow{a,C} p^\star$ *and* $ax\ \ (\mathbb{P}^b) \vdash p^\star = p'$.
2. $p \xrightarrow{a,C} p^\star$ *implies* $C \mapsto a;\, p' \ \leq\ p$ *and* $ax\ \ (\mathbb{P}^b) \vdash p^\star = p'$.

Here, $\longrightarrow$ *is the edge-relation as defined in Definition 7.*

Theorem 28 (Completeness)**.** *For* $p, q \in \mathbb{P}^b \cap \mathbb{P}^{\mathsf{def}}$. $p \sim_s q$ *implies* $ax\ \ (\mathbb{P}^b) \vdash p = q$.

Proof. Let $p, q \in \mathbb{P}^b \cap \mathbb{P}^{\mathsf{def}}$ with $p \sim_s q$. From Theorems 25 and 23, it follows that there are basic terms $\hat{p}, \hat{q} \in \mathbb{P}^{\mathsf{def}}$ such that $\hat{p} \sim_s p$ and $q \sim_s \hat{q}$. Let $\hat{p} = \{\!|C|\!\}\, p'$ and $\hat{q} = \{\!|C'|\!\}\, q'$ where p' and q' are pre-basic terms. By definition of $\sim_s$, $\kappa(\hat{p}) = \kappa(\hat{q})$, and since $\kappa(p') = \kappa(q') = \varnothing$, it follows $C = C'$. The remainder of this proof proceeds by induction on the maximum number of nested prefixes in $\hat{p}$ and $\hat{q}$. If this is 0, then $\hat{p} = \{\!|C|\!\}\, \mathbf{0} = \hat{q}$. Otherwise, $\hat{p} = \{\!|C|\!\}\, p'$ and $\hat{q} = \{\!|C|\!\}\, q'$ with $p' = \sum_{i\in I} C_i \mapsto a_i;\, p_i$ and $q' = \sum_{j\in J} C_j \mapsto a_j;\, q_j$. Consider now summand $C_k \mapsto a_k;\, p_k$ of p'. By Proposition 27 (part 1), $p' \xrightarrow{a_k,C_k} p^\star$ for $k \in I$ and $ax\ \ (\mathbb{P}^b) \vdash p_k = p^\star$. By Theorem 23, $p_k \sim_s p^\star$. Since $\hat{p} \sim_s \hat{q}$, we have $p' \sim_s q'$. By definition of $\sim_s$, it follows that $q' \xrightarrow{a_k,C_k} q^\star$ and $p^\star \sim_s q^\star$. By Proposition 27 (part 2), there is $h \in J$ such that, $a_k = a_h$, $C_k = C_h$, $ax\ \ (\mathbb{P}^b) \vdash q^\star = q_h$, and $C_h \mapsto a_h;\, q_h \ \leqslant\ q'$. By Theorem 23, $q_h \sim_s q^\star$. Since $p_k \sim_s p^\star \sim_s q^\star \sim_s q_h$, it follows from the induction hypothesis that $ax\ \ (\mathbb{P}^b) \vdash p_k = q_h$. Hence, $ax\ \ (\mathbb{P}^b) \vdash C_k \mapsto a_k;\, p_k = C_h \mapsto a_h;\, q_h \leqslant q'$. Proceeding in a similar way for every summand of p', we conclude $p' \leqslant q'$. By symmetry $q' \leqslant p'$. Hence, $p' = q'$ and $ax\ \ (\mathbb{P}^b) \vdash p = \hat{p} = \{\!|C|\!\}\, p' = \{\!|C|\!\}\, q' = \hat{q} = q$. □

4.2. Axioms and laws for open p-bisimulation

In this section, we extend the axiomatisation with a set of laws (cf. Table 8) that is sound for open p-bisimulation. The new set of axioms is denoted $ax\ (\hat{\mathcal{O}}^b) + \mathbf{Open}$.

Axioms **Sy1**–**Sy3** respect symbolic bisimulation. **Sy1** eliminates irrelevant clocks, that is, clocks that are set but never used before they are set again. Axiom **Sy2** asserts that clocks that have already been used—i.e., they have expired—are not longer of interest. **Sy3** relates clocks as symbolic bisimulation does. Predicate Lnk, which is defined by rules **Ln1**–**Ln6**, checks whether a set of clocks is *linked* in a process, that is, whether they are set and used together. Axiom **Sy3** states that a set C containing clocks that are linked can be replaced in a process p by a single fresh clock z with the additional requirement that z and C represent the same stochastic value, i.e., the distributions of z and $\max(C)$ are equal. (Recall that $F_{\max(C)}(t) = \prod_{x\in C} F_x(t)$ for any $t \in \mathbb{R}$.)

Axioms **A3′** and **Red** preserve open p-bisimulation, but not symbolic bisimulation. **A3′** gives a weaker notion of idempotency in comparison to axiom **A3**. Notice that the branching process is reduced to only one process that is faster than each of the summands in the original process. This is due to the race policy. Recall that $F_{\min\{x,y\}}(t) = F_x(t) + F_y(t) - F_x(t)F_y(t) = 1 - (1 - F_y(t))(1 - F_x(t))$. Axiom **Red** states that clocks which are stochastically redundant—i.e., they are never set to a positive value—cannot affect the timing of a process. This axiom together with axioms **Sy1** and **Sy2** allows one to eliminate clocks whose support does not contain positive reals.

In the following, we discuss several properties that can be derived from the equational theory $ax\ (\hat{\mathcal{O}}^b) + \mathbf{Open}$. First, notice that axiom **Sy3** implies α-conversion. In fact, if $F_x = F_y$ and $y \notin \mathsf{fv}(p)$, $\{\!|x|\!\}\, p = \{\!|y|\!\}\, [y/x]p$ is a special instance of **Sy3** in which $C = \{x\}$. Moreover, axiom **Sy3** allows us

Table 8
Axioms for open p-bisimulation in $\hat{\mathcal{O}}^b$

Sy1 $\{\!|C|\!\}\, p = p$	if $C \cap \mathsf{fv}(p) = \varnothing$
Sy2 $C{\mapsto}a;\, C{\mapsto}p = C{\mapsto}a;\, p$	
Sy3 $\{\!|C|\!\}\, p = \{\!|z|\!\}\, \{z/C\}p$	if $\mathsf{Lnk}(C,p) \wedge z \notin \mathsf{fv}(p) \wedge F_z = F_{\max(C)}$
A3′ $\{\!|x,y|\!\}\, (\{x\}{\mapsto}a;\, p + \{y\}{\mapsto}a;\, p) = \{\!|z|\!\}\, \{z\}{\mapsto}a;\, p$	if $\{x,y,z\} \cap \mathsf{fv}(p) = \varnothing \wedge F_z = F_{\min\{x,y\}}$
Red $\{\!|x|\!\}\, \{x\}{\mapsto}p = \{\!|x|\!\}\, p$	if $F_x(0) = 1$

$$\textbf{Ln1}\ \frac{C \cap \mathsf{fv}(p) = \varnothing}{\mathsf{Lnk}(C,p)} \qquad \textbf{Ln4}\ \frac{\mathsf{Lnk}(C,p) \quad C \cap C' = \varnothing}{\mathsf{Lnk}(C, C'{\mapsto}p)}$$

$$\textbf{Ln2}\ \frac{\mathsf{Lnk}(C,p)}{\mathsf{Lnk}(C,a;\,p)} \qquad \textbf{Ln5}\ \frac{\mathsf{Lnk}(C,p) \quad C \cap C' = \varnothing}{\mathsf{Lnk}(C, \{\!|C'|\!\}\, p)}$$

$$\textbf{Ln3}\ \frac{C \subseteq C' \quad C \cap \mathsf{fv}(p) = \varnothing}{\mathsf{Lnk}(C, C'{\mapsto}p)} \qquad \textbf{Ln6}\ \frac{\mathsf{Lnk}(C,p) \quad \mathsf{Lnk}(C,q)}{\mathsf{Lnk}(C, p+q)}$$

to eliminate redundant clocks. Suppose $\mathsf{Lnk}(\{x, y\}, p)$ and, for some $t \in \mathbb{R}$, $F_x(t) = 0$ and $F_y(t) = 1$. (Hence $F_x = F_{\max\{x,y\}}$.) Then

$$\{\!|x, y|\!\}\, p \overset{\text{Sy3}}{=} \{\!|z|\!\}\, \{z/\{x, y\}\}p \overset{(\simeq_\alpha)}{=} \{\!|x|\!\}\, [x/z](\{z/x, z/y\}p) = \{\!|x|\!\}\, [x/y]p,$$

provided $z \notin \mathsf{fv}(p)$ and $F_z = F_x$.

For $x \notin \mathsf{fv}(p)$, $a; p + \{\!|x|\!\}\, \{x\}\mapsto a; p = a; p$. To prove this, assume fresh variables $y, z \notin \mathsf{fv}(p)$ such that $F_y(0) = 1$ and $F_z = F_{\min\{x,y\}}$. As a consequence $F_z(0) = 1$. Therefore:

$$\begin{aligned} a; p + \{\!|x|\!\}\, \{x\}\mapsto a; p &\overset{\text{Sy1}}{=} \{\!|y|\!\}\, a; p + \{\!|x|\!\}\, \{x\}\mapsto a; p \\ &\overset{\text{Red}}{=} \{\!|y|\!\}\, \{y\}\mapsto a; p + \{\!|x|\!\}\, \{x\}\mapsto a; p \\ &\overset{\text{CS3,A3}'}{=} \{\!|z|\!\}\, \{z\}\mapsto a; p \\ &\overset{\text{Red,Sy1}}{=} a; p. \end{aligned}$$

In the previous section we argued that $\{\!|x|\!\}\, \{x\}\mapsto a; p + \{\!|x|\!\}\, \{x\}\mapsto a; p \neq \{\!|x|\!\}\, \{x\}\mapsto a; p$. This can be shown as follows. Assume $F_x = F_y$ and $x, y, z \notin \mathsf{fv}(p)$. Then,

$$\begin{aligned} \{\!|x|\!\}\, \{x\}\mapsto a; p + \{\!|x|\!\}\, \{x\}\mapsto a; p &\overset{(\simeq_\alpha)}{=} \{\!|x|\!\}\, \{x\}\mapsto a; p + \{\!|y|\!\}\, [y/x]\{x\}\mapsto a; p \\ &\overset{\text{CS3}}{=} \{\!|x, y|\!\}\, (\{x\}\mapsto a; p + \{y\}\mapsto a; p) \\ &\overset{\text{A3'}}{=} \{\!|z|\!\}\, \{z\}\mapsto a; p. \end{aligned}$$

with $F_z(t) = F_{\min\{x,y\}}(t) = 2F_x(t) - (F_x(t))^2$. Clearly, $\{\!|z|\!\}\, \{z\}\mapsto a; p \neq \{\!|x|\!\}\, \{x\}\mapsto a; p$ for every non-trivial random variable x such that $F_x(0) < 1$. Notice, however, that $\{\!|x|\!\}\, (\{x\}\mapsto a; p + \{x\}\mapsto a; p) = \{\!|x|\!\}\, \{x\}\mapsto (a; p + a; p) = \{\!|x|\!\}\, \{x\}\mapsto a; p$.

The next proposition states that once a clock is terminated (or has expired), it is no longer necessary and can thus be removed. This result can be seen as a generalisation of axiom **Sy2**.

Proposition 29. *For $p \in \mathbb{\Delta}^b$ and $C \subseteq \mathcal{C}$ there is a term q such that $C \cap \mathsf{fv}(q) = \varnothing$ and ax $(\mathbb{\Delta}^b)$ + **Open** $\vdash C\mapsto p = C\mapsto q$.*

Proof. The proof follows by induction on the size of the term p by doing case analysis. Assume $C \cap \mathsf{fv}(q) = \varnothing$ (which follows by inductive reasoning). We only present the most distinctive cases.

Case $C\mapsto C'\mapsto p$.

$$\begin{aligned} C\mapsto C'\mapsto p &\overset{\text{T3}}{=} (C' - C)\mapsto C\mapsto p \overset{\text{ind.}}{=} (C' - C)\mapsto C\mapsto q \\ &\overset{\text{T3}}{=} C\mapsto (C' - C)\mapsto q \end{aligned}$$

Case $C\mapsto\{\!|C'|\!\}\,p$. By α-conversion, assume $\{\!|C'|\!\}\,p = \{\!|C^\star|\!\}\,[C^\star/C']p$ with $C^\star \cap C = \varnothing$. Then,

$$C\mapsto\{\!|C'|\!\}\,p \;=\; C\mapsto\{\!|C^\star|\!\}\,[C^\star/C']p \overset{\text{T4}}{=} \{\!|C^\star|\!\}\,C\mapsto[C^\star/C']p$$
$$\overset{\text{ind.}}{=} \{\!|C^\star|\!\}\,C\mapsto q \overset{\text{T4}}{=} C\mapsto\{\!|C^\star|\!\}\,q. \qquad \square$$

The axiom system $ax\ (\hat{\mathbb{1}}^b) + \mathbf{Open}$ is sound for $\sim_o$; this result is proven in [19, Appendix G].

Theorem 30. *For* $p,q \in \hat{\mathbb{1}}^\alpha$. $ax\ (\hat{\mathbb{1}}^b) + \mathbf{Open} \vdash p = q$ *implies* $p \sim_o q$.

4.3. Axioms for the static operators

The equational theories introduced so far provide axioms only for the basic operations. In this section, the axiomatic treatment is extended to the *static operators*. The new axioms (see Table 9) explain how the static operators are decomposed and eliminated in favour of the basic operations.

Axioms **R1**–**R6** define the renaming operator in terms of the basic operations. Notice that if the axioms are considered as rewrite rules from left to right, the renaming operation is "pushed" inside the term while appropriately renaming actions.

The rest of the axioms concern the different parallel compositions. Axioms **Mrg1** and **Mrg2** move clock settings out of the scope of the parallel composition. This is necessary to avoid the duplication of clocks when expanding parallel composition in terms of summations. Duplicating clocks would transform processes without conflict of variables into (semantically different) processes with conflict of variables. **Mrg3** decomposes the parallel composition in terms of the left merge and the communication merge. To avoid clock duplication, this axiom can be applied provided that clocks in p and q are reset only if an action is previously performed. That is, a reset operator only appears in the scope of a prefix. This requirement is established by predicate csf, which is defined by the rules **csf1**–**csf4** (clock set-free). Observe that if $\mathsf{csf}(p)$, then $\kappa(p) = \varnothing$ and hence $p \equiv \mathsf{noset}(p)$. Notice moreover, that a pre-basic term $p \in B'$ satisfies $\mathsf{csf}(p)$.

The axiomatisation of the parallel composition is completed by axioms **LM1**–**LM7** and **CM0**–**CM7**. Axioms **LM1**–**LM7** define the left merge. Note that in **LM3** it is necessary for the right operand to not reset any clock (predicate $\mathsf{csf}(q)$) to move a prefix in the left operand out of the scope of the left merge. Finally, axioms **CM0**–**CM7** define the communication merge.

Definition 31. The equational theory for $\hat{\mathbb{1}}^p$ is defined by the signature $sig(\hat{\mathbb{1}}^p)$, containing the constant $\mathbf{0}$, the operators a; $\_$, $C\mapsto\_$, $\{\!|C|\!\}\,\_$, and $\_[f]$, and the binary operators $+$, $||_A$, $\underline{||}_A$, and $|_A$, and the axiom system $ax\ (\hat{\mathbb{1}}^p)$ as given in Tables 7 and 9.

Theorem 32 [Soundness of $ax\ (\hat{\mathbb{1}}^p)$]**.**

1. *For* $p,q \in \hat{\mathbb{1}}^{\mathsf{def}}$. $ax\ (\hat{\mathbb{1}}^p) \vdash p = q$ *implies* $p \sim_s q$.
2. *For* $p,q \in \hat{\mathbb{1}}^\alpha$. $ax\ (\hat{\mathbb{1}}^p) + \mathbf{Open} \vdash p = q$ *implies* $p \sim_o q$.

Proof. For both cases, it suffices to check that the axioms in Table 9 preserve $\sim_s$. For both cases it is routine to check that for every axiom $p = q$ in Table 9, the relation $R = \{\,\langle p,q\rangle, \langle q,p\rangle\,\}$ is a structural bisimulation up to $\sim_s$ (see [22] for a definition of this notion). For axioms

Table 9
Axioms for the static operators

R1 $\mathbf{0}[f] = \mathbf{0}$		**R4** $(\{\!|C|\!\}\, p)[f] = \{\!|C|\!\}\, (p[f])$
R2 $(a; p)[f] = f(a); (p[f])$		**R5** $(p + q)[f] = p[f] + q[f]$
R3 $(C \mapsto p)[f] = C \mapsto (p[f])$		
Mrg1 $(\{\!|C|\!\}\, p) \,||_A\, q = \{\!|C|\!\}\, (p \,||_A\, q)$	if $C \cap (\kappa(q) \cup \mathsf{fv}(q)) = \varnothing$	
Mrg2 $p \,||_A\, (\{\!|C|\!\}\, q) = \{\!|C|\!\}\, (p \,||_A\, q)$	if $C \cap (\kappa(p) \cup \mathsf{fv}(p)) = \varnothing$	
Mrg3 $p \,||_A\, q = p \,\underline{|\!|}_A\, q + q \,\underline{|\!|}_A\, p + p \,|_A\, q$	if $\mathsf{csf}(p) \wedge \mathsf{csf}(q)$	
LM1 $\mathbf{0} \,\underline{|\!|}_A\, q = \mathbf{0}$	if $\mathsf{csf}(q)$	
LM2 $a; p \,\underline{|\!|}_A\, q = \mathbf{0}$	if $\mathsf{csf}(q) \wedge a \in A$	
LM3 $a; p \,\underline{|\!|}_A\, q = a; (p \,||_A\, q)$	if $\mathsf{csf}(q) \wedge a \notin A$	
LM4 $(C \mapsto p) \,\underline{|\!|}_A\, q = C \mapsto (p \,\underline{|\!|}_A\, q)$		
LM5 $(\{\!|C|\!\}\, p) \,\underline{|\!|}_A\, q = \{\!|C|\!\}\, (p \,\underline{|\!|}_A\, q)$	if $C \cap (\kappa(q) \cup \mathsf{fv}(q)) = \varnothing$	
LM6 $p \,\underline{|\!|}_A\, (\{\!|C|\!\}\, q) = \{\!|C|\!\}\, (p \,\underline{|\!|}_A\, q)$	if $C \cap (\kappa(p) \cup \mathsf{fv}(p)) = \varnothing$	
LM7 $(p + q) \,\underline{|\!|}_A\, r = (p \,\underline{|\!|}_A\, r) + (q \,\underline{|\!|}_A\, r)$	if $\mathsf{csf}(r)$	
CM0 $p \,|_A\, q = q \,|_A\, p$		
CM1 $\mathbf{0} \,|_A\, \mathbf{0} = \mathbf{0}$		
CM2 $\mathbf{0} \,|_A\, a; q = \mathbf{0}$		
CM3 $a; p \,|_A\, a; q = a; (p \,||_A\, q)$	if $a \in A$	
CM4 $a; p \,|_A\, b; q = \mathbf{0}$	if $a \notin A$	
CM5 $(C \mapsto p) \,|_A\, q = C \mapsto (p \,|_A\, q)$		
CM6 $(\{\!|C|\!\}\, p) \,|_A\, q = \{\!|C|\!\}\, (p \,|_A\, q)$	if $C \cap (\kappa(q) \cup \mathsf{fv}(q)) = \varnothing$	
CM7 $(p + q) \,|_A\, r = (p \,|_A\, r) + (q \,|_A\, r)$	if $\mathsf{csf}(r)$	

csf1 $\mathsf{csf}(\mathbf{0})$ **csf2** $\mathsf{csf}(a; p)$ **csf3** $\dfrac{\mathsf{csf}(p)}{\mathsf{csf}(C \mapsto p)}$ **csf4** $\dfrac{\mathsf{csf}(p) \quad \mathsf{csf}(q)}{\mathsf{csf}(p + q)}$

Mrg3 and **CM0**, it is necessary to know in addition that $p \,||_A\, q \sim_s q \,||_A\, p$ which can be easily verified. □

The rest of this section is devoted to prove completeness of $ax\ \ (\mathbb{C}^p)$. First an elimination theorem is provided stating that for every closed term in $\mathbb{C}^p$ there is a term in $\mathbb{C}^b$ that can be proved equal using the axioms.

Theorem 33. *For $p \in \mathbb{C}^p$, there is $q \in \mathbb{C}^b$ such that $ax\ \ (\mathbb{C}^p) \vdash p = q$.*

Proof [Sketch]**.** For each axiom $p = q$ in Table 9 (except **CM0**) define a rewrite rule $p \to q$. Consider the rewrite system with these rules modulo the axioms in Table 7 and **CM0** (i.e., terms proved equal with these axioms are considered the same term in the rewrite system). It is simple to prove that the normal form in this rewrite system is a term in $\mathbb{C}^b$. □

Because of Theorem 28, completeness of $ax\ (\mathbin{⌂}^p)$ follows as a corollary.

Theorem 34. *For $p, q \in \mathbin{⌂}^p \cap \mathbin{⌂}^{\mathsf{def}}$. $p \sim_s q$ implies $ax\ (\mathbin{⌂}^p) \vdash p = q$.*

As argued in Section 1, it is not straightforward to obtain an expansion law [46] for a process algebra in which action delays are governed by general distributions. For $\mathbin{⌂}$ an expansion law can be straightforwardly derived from the aforementioned axioms. This yields:

Theorem 35. *Let p and q be $\mathbin{⌂}$ terms such that $p = \{\!|C|\!\}\, p'$ and $q = \{\!|C'|\!\}\, q'$ with $p' = \sum_i C_i \mapsto a_i; p_i$ and $q' = \sum_j C'_j \mapsto b_j; q_j$. Suppose $p \,||_A\, q$ is α-definable and locally definable. The following equality can be proven in $ax\ (\mathbin{⌂}^p)$:*

$$p \,||_A\, q = \{\!|C \cup C'|\!\} \Big(\sum_{a_i \notin A} C_i \mapsto a_i; (p_i \,||_A\, q') + \sum_{b_j \notin A} C'_j \mapsto b_j; (p' \,||_A\, q_j) + \sum_{a_i = b_j \in A} (C_i \cup C'_j) \mapsto a_i; (p_i \,||_A\, q_j) \Big).$$

If $p[f]$ is α-definable and locally definable, the following equality also holds in $ax\ (\mathbin{⌂}^p)$.

$$p[f] = \{\!|C|\!\} \left(\textstyle\sum_i C_i \mapsto f(a_i); (p_i[f]) \right).$$

To illustrate equational reasoning in $\mathbin{⌂}$, we prove that the switch modelled using $\mathbin{⌂}$ in Example 4 is equal to the equation system that represents the stochastic automaton of the switch in Example 16.

Example 36. Recall that the switch is described by:

$$\begin{aligned} \mathit{Arrival} &= \{\!|x|\!\}\, \{x\} \mapsto \mathit{on}; \mathit{Arrival} \\ \mathit{SwitchOff} &= \mathit{on}; \mathit{SwitchOn} \\ \mathit{SwitchOn} &= \mathit{on}; \mathit{SwitchOn} + \{\!|y|\!\}\, \{y\} \mapsto \mathit{off}; \mathit{SwitchOff} \\ \mathit{System} &= \mathit{Arrival} \,||_{\mathit{on}}\, \mathit{SwitchOff} \end{aligned}$$

The following calculations proceed by using the axioms, the expansion law, and process variable folding and unfolding.

$$\begin{aligned} \mathit{System} &= \mathit{Arrival} \,||_{\mathit{on}}\, \mathit{SwitchOff} \\ &= (\{\!|x|\!\}\, \{x\} \mapsto \mathit{on}; \mathit{Arrival}) \,||_{\mathit{on}}\, \mathit{on}; \mathit{SwitchOn} \\ &= \{\!|x|\!\}\, \{x\} \mapsto \mathit{on}; (\mathit{Arrival} \,||_{\mathit{on}}\, \mathit{SwitchOn}) \end{aligned}$$

We then proceed by deriving:

$$\begin{aligned}
&Arrival \;||_{on}\; SwitchOn \\
&\quad = (\{\!|x|\!\}\,\{x\}\mapsto on;\; Arrival) \;||_{on}\; (on;\; SwitchOn + \{\!|y|\!\}\,\{y\}\mapsto off;\; SwitchOff\,) \\
&\quad = \{\!|x, y|\!\}\,(\{x\}\mapsto on;\; (Arrival \;||_{on}\; SwitchOn\,) + \{y\}\mapsto off;\; (\,\{x\}\mapsto on;\; Arrival \;||_{on}\; SwitchOff\,)) \\
&\quad = \{\!|x, y|\!\}\,(\{x\}\mapsto on;\; (Arrival \;||_{on}\; SwitchOn\,) + \{y\}\mapsto off;\; (Arrival_{\mathrm{noset}} \;||_{on}\; SwitchOff\,))
\end{aligned}$$

where:

$$\begin{aligned}
Arrival_{\mathrm{noset}} \;||_{on}\; SwitchOff &= \{x\}\mapsto on;\; Arrival \;||_{on}\; on;\; SwitchOn \\
&= \{x\}\mapsto on;\; (\;Arrival \;||_{on}\; SwitchOn\;)
\end{aligned}$$

Recall that in Example 16 the obtained equation system was as follows:

$$\begin{aligned}
X_{System} &= \{\!|x|\!\}\,\{x\}\mapsto on;\; X_{Arrival||_{on}\,SwitchOn} \\
X_{Arrival||_{on}\,SwitchOn} &= \{\!|x, y|\!\}\;(\{x\}\mapsto on;\; X_{Arrival||_{on}\,SwitchOn} + \{y\}\mapsto off;\; X_{\mathrm{noset}(Arrival)||_{on}\,SwitchOff}) \\
X_{\mathrm{noset}(Arrival)||_{on}\,SwitchOff} &= \{\!|\varnothing|\!\}\,\{x\}\mapsto on;\; X_{Arrival||_{on}\,SwitchOn}
\end{aligned}$$

Though, apparently, the calculation is about to finish, we are not yet in conditions to actually conclude that $System = X_{System}$ *(continued below)*.

In fact, both tuples $\langle System, Arrival \;||_{on}\; SwitchOn, \mathsf{noset}(Arrival) \;||_{on}\; SwitchOff\rangle$ and $\langle X_{System}, X_{Arrival||_{on}\,SwitchOn}, X_{\mathrm{noset}(Arrival)||_{on}\,SwitchOff}\rangle$ solve the same system of equations, but we did not yet provide any tool that allows to conclude that both solutions are the same. To characterise those models in which recursive specifications, i.e., systems of recursive equations, have (unique) solutions, process algebras provide the following principles [6,5]:

- The Recursive Specification Principle (RSP) is the assumption that every guarded recursive specification has at most one solution.
- The Restricted Recursive Definition Principle (RDP^-) is the assumption that every guarded recursive specification has a solution.

Notice that a model that satisfies both RSP and RDP^- guarantees that every guarded recursive specification has a unique solution. Every guarded recursive specification in $\hat{\mathcal{G}}$ has a solution. As a result, $\hat{\mathcal{G}}$ satisfies RDP^-. The proof that $\hat{\mathcal{G}}$ satisfies RSP is more tedious and is omitted here.

We just mention that it is a straightforward adaptation of the proof given for ACP [5]. As a consequence:

Theorem 37. *The algebra with carrier set* $\mathbb{P}^{\alpha}$, *signature* $sig(\mathbb{P}^{p})$, *and* $\sim_o$ *as equivalence relation satisfies RSP and RDP*$^{-}$.

Example 38 (*Continuation of Example 36*)**.** Because RSP and RDP^{-} hold in $\mathbb{P}$, guarded recursive specifications have unique solutions. Therefore:

$$\begin{aligned} X_{System} &= System \\ X_{Arrival||_{on}SwitchOn} &= Arrival \,||_{on}\, SwitchOn \\ X_{\mathsf{noset}(Arrival)||_{on}SwitchOff} &= \mathsf{noset}(Arrival) \,||_{on}\, SwitchOff \end{aligned}$$

which concludes our proof.

4.4. Summary of results

This section summarises our main results and places them in perspective. Let BC be a basic calculus (*à la* CCS) with signature $sig(\mathrm{BC}) = \{\mathbf{0}, a;\_, +\}$ and $ax(\mathrm{BC})$ containing the axioms:

A1 $p + q = q + p$, **A3** $a; p + a; p = a; p$,
A2 $(p + q) + r = p + (q + r)$, **A4** $p + \mathbf{0} = p$.

Despite that axiom **A3** is not the traditional CCS idempotency, it is not difficult to see that, for closed terms, this axiom system is equivalent to the traditional CCS axiom set. Fig. 4 summarises the relationship between the equational theories presented in this paper. The arrows have the following meaning: $ax(A) \longrightarrow ax(B)$ expresses that B is a *conservative extension* [5] of A, i.e., for all closed terms p, q in A it holds: $ax(A) \vdash p = q$ if and only if $ax(B) \vdash p = q$. $ax(A) \cdots\!\blacktriangleright ax(B)$ states

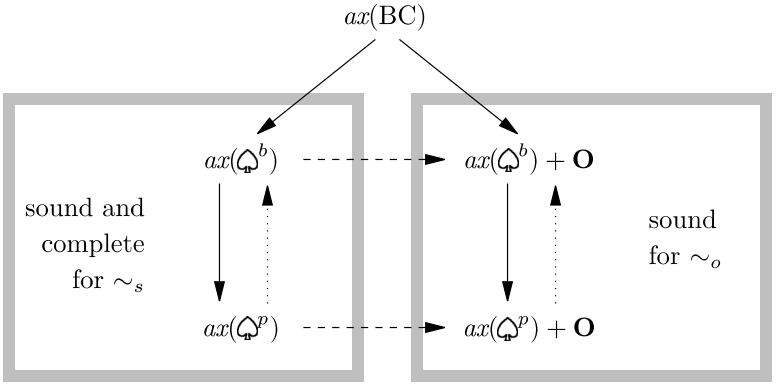


Fig. 4. Summary of $\mathbb{P}$ axiomatisations.

that for every closed term p in A there is a term q in B such that $ax\ (A) \vdash p = q$, i.e., operations in $sig(A) - sig(B)$ can be *eliminated* and replaced by those in $sig(B)$. Finally, $ax\ (A) \dashrightarrow ax\ (B)$ expresses that $sig(A) = sig(B)$ and $ax\ (A) \vdash p = q$ implies $ax\ (B) \vdash p = q$.

The results in Fig. 4 are justified as follows. From results on conservative extensions [26] it follows that $ax\ (\mathrm{BC}) \longrightarrow ax\ (\spadesuit^b)$; for details we refer to [19, Chapter 11]. Since $\sim_s$ and $\sim_o$ coincide for terms in the basic calculus BC, it immediately follows $ax\ (\mathrm{BC}) \longrightarrow ax\ (\spadesuit^b) + \mathbf{Open}$. $ax\ (\spadesuit^b) \longrightarrow ax\ (\spadesuit^p)$ can also be proven using the general results in [26]. This technique cannot be applied to prove $ax\ (\spadesuit^b) + \mathbf{Open} \longrightarrow ax\ (\spadesuit^p) + \mathbf{Open}$, since it requires that $ax\ (\spadesuit^b) + \mathbf{Open}$ is complete for $\sim_o$, which is not the case. It can, however, be straightforwardly proven using classical process algebra techniques (see, e.g. [5]). The result $ax\ (\spadesuit^p) \cdots\!\!\blacktriangleright ax\ (\spadesuit^b)$ follows directly from Theorem 33. Consequently, $ax\ (\spadesuit^p) + \mathbf{Open} \cdots\!\!\blacktriangleright ax\ (\spadesuit^b) + \mathbf{Open}$ also holds. Finally, $ax\ (\spadesuit^b) \dashrightarrow ax\ (\spadesuit^b) + \mathbf{Open}$ and $ax\ (\spadesuit^p) \dashrightarrow ax\ (\spadesuit^p) + \mathbf{Open}$ follow from the fact that $ax\ (\spadesuit^b) \subseteq ax\ (\spadesuit^b) + \mathbf{Open}$ and $ax\ (\spadesuit^p) \subseteq ax\ (\spadesuit^p) + \mathbf{Open}$.

5. Related work

Several authors have proposed different approaches to incorporate general distributions in a process algebraic framework. We discuss the main approaches and their differences with $\spadesuit$.

TIPP [30] is the earliest process algebra with general distributions. Its basic construct is the prefix a_F; p which corresponds to $\{\!|x|\!\}\ \{x\}\!\mapsto a$; p with $F_x = F$. The semantics is based on labelled transition systems in which transitions are decorated with a distribution function and, to keep track of the execution of parallel processes, a number (a start reference) that indicates how many times an action has not been chosen to be executed. This approach yields infinite semantic objects, even for simple regular processes. Similar techniques have been adopted by [2,45,49].

Harrison and Strulo [33,34,55] introduce a process algebra for discrete-event simulation. Three operations are considered: one that randomly assigns a value to a variable, another that starts a timer and waits for it to reach a value (possibly stored in a variable), and the occurrence of immediate actions. (Notice that in $\spadesuit$ the operator $\{\!|x|\!\}\ p$ starts a timer *and* randomly sets its expiration time, while $\{x\}\!\mapsto p$ only waits for a timer to reach its expiration value.) Their process algebra includes an urgent and a delay prefix, thus combining the closed and open system interpretation. The semantic model is similar to probabilistic transition systems where non-deterministic transitions are split into discrete transitions and timed transitions. Due to these explicit time transitions, semantic objects contain usually uncountably many states and transitions. Since timer-initialisation and termination are a single operation, an expansion law cannot be obtained.

A non-interleaving semantics using a stochastic extension of event structures has been proposed by Brinksma et al. [15]. Non-interleaving semantics seems to be more natural to deal with general distributions since activities that are causally independent (i.e., concurrent activities) are unordered, as opposed to interleaving semantics. Using techniques like McMillan prefixes, finite semantic objects can be obtained for recursive processes and analysed [52].

A general semi-Markovian process algebra (GSMPA) based on the Markovian process algebra EMPA has been discussed in [11,10]. GSMPA has a semantics in a semi-interleaving model that

allows for refinement of actions (the so-called ST-semantics). Although this calculus preserves finiteness of semantic objects, the semantics is rather complex. A nice characteristic of GSMPA is that it represents the GSMP model in a complete way. Recently, Bravetti and Gorrieri [13,10] adapted the previous work to consider—like in this paper—the separation between the beginning of a delay, the ending of it, and the execution of an action. This yields an elegant semantics for GSMPA, and exploits known results from ST-semantics to deal, for instance, with naming of clocks (name clashes). Apart from strong bisimulations, a weak equivalence relation (accompanied with a complete axiomatic characterisation) is defined and allows for the abstraction of internal immediate actions.

There are two main distinguishing aspects between ♤ and most other approaches toward general distributions. Whereas most approaches use infinite-state probabilistic transition systems as semantical model, ♤ takes a layered approach, by first providing a mapping onto stochastic automata. These automata are mostly (e.g., for regular processes) finite state, thus enabling formal reasoning. The role of stochastic automata is similar to that of timed automata, a well-known symbolic model for real-time systems. A similar approach has been advocated by Bravetti and Gorrieri [11,13,10] who exploit ST-semantics to handle general distributions. For dealing with action-prefixes of the form $a_F; p$, there is no need for considering α-conversion, but one can resort to standard techniques of ST-semantics to bind names of completion events to those of start events, e.g., using the static approach in [1]. This is illustrated in [12]. In the setting of ♤, however, start and completion events do not always occur as pairs, e.g., clocks that are set may never be used, or used multiple times. We therefore take a somewhat more general approach and consider renaming. Bravetti and D'Argenio recently provided a detailed account of the differences between the approach taken for ♤ and the one using ST-semantics [12]. A second issue is the presence of non-determinism: all approaches (with the notable exception of [13,10,33,34,55]) listed above take a purely probabilistic viewpoint, in which all choices are resolved probabilistically. Non-determinism plays a prominent role in ♤. We finally remark that if all clocks in ♤ are exponential, a calculus is obtained that strongly resembles Hermanns' language for describing interactive Markov chains [35].

The language ♤ is equally expressive as stochastic automata, and as GSMPs are a proper subset of stochastic automata, ♤ can be used as a high-level language for discrete-event simulation. A different approach to use process algebra for discrete-event simulation has been taken by Pooley [48] and Tofts and Birtwistle [56]. They use (traditional) process algebra as semantical model for practical discrete-event simulation languages, enabling e.g., deadlock analysis techniques to simulation models.

6. Concluding remarks

This paper introduced the stochastic process algebra ♤ whose semantics is defined in terms of stochastic automata, which, in turn, are interpreted as (infinite) probabilistic transition systems. The structured operational semantics of ♤ is rather concise and simple. In the accompanying paper [22], several notions of strong bisimulation have been defined on stochastic automata including probabilistic, symbolic, and structural bisimulation. The technical results of this paper are strongly based on these equivalence notions and include:

(*i*) the semantics of a term up to α-conversion is unique up to symbolic bisimulation;
(*ii*) stochastic automata and ⌂ are equally expressive up to structural bisimulation;
(*iii*) open probabilistic and structural bisimulation are congruences for ⌂;
(*iv*) a sound and complete axiomatisation for structural bisimulation;
(*v*) a sound axiomatisation for open probabilistic bisimulation.

Put in a nutshell, ⌂ can be considered as a language for the modular representation of stochastic automata that has a solid algebraic foundation.

Concerning the analysis of properties of ⌂ terms, the techniques applicable to stochastic automata do apply. This includes discrete-event simulation [25] for assessing quantitative properties and verification techniques such as model checking for untimed safety properties [25] as well as timed safety and liveness properties [20]. Recently, a light-weight modeling language [21] with accompanying tool support has been developed based on ⌂. This specification formalism enriches ⌂ with data types, control flow constructs (e.g., iteration and exception handling), non-deterministic timing and discrete probabilistic branching.

Acknowledgments

We thank Ed Brinksma for his collaboration in this research. The reviewers are gratefully acknowledged for their valuable comments that improved the presentation of the paper significantly. Part of this work was done while the first author was working for the STW/PROGRESS project TES-4999 "Verification of Hard and Softly Timed Systems (HaaST)" at the University of Twente.

Glossary

Probabilities

Ω:	Sample space;
$\mathcal{F}$:	σ-algebra;
μ:	Probability measure;
$Prob(\Omega)$:	Set of probability spaces in Ω;
$\mathcal{B}(\mathbb{R}^n)$:	Borel algebra on the nth real hyperspace;
$\mathcal{R}(F_1,\ldots,F_n)$:	Probability space on $\mathcal{B}(\mathbb{R}^n)$ with measure uniquely defined by $F_1,\ldots,F_n$;
$\mathcal{D}_v^s$:	Decoration (measurable) function;

Semantics

$\mathcal{A}$:	Set of action names;
$\mathcal{C}$:	Set of clock names;
$\mathbf{V}$:	Set of valuations;
$\mathcal{S}$:	Set of locations;
κ:	Clock setting function;
$\rightarrow$:	Edge on the definable semantics;
$\rightarrow_\alpha$:	Edge on the α-definable semantics;
$\mathsf{exp}_d(v,C)$:	Expiration predicate;
$\mathsf{mpr}_d(s,v)$:	Maximal progress predicate;

(p, v): Probabilistic state;
$[p, v]$: Non-deterministic state;
T: Probabilistic transition;
$\rightarrow$: Non-deterministic transition;
$\cong$: Isomorphism on stochastic automata;
$\sim_p$: Probabilistic bisimulation;
$\sim_c$: Closed p-bisimulation;
$\sim_o$: Open p-bisimulation;
$\sim_\&$: Symbolic bisimulation;
$\sim_s$: Structural bisimulation;

$\spadesuit$ *operators and terms*
$a; p$: Action prefix;
$C \mapsto p$: Triggering condition;
$\{\!|C'|\!\}\, p$: Clock setting;
$p + p$: Sum or (non-deterministic) choice;
$p \,||_A\, p$: Parallel composition or merge;
$p \,|\!|_A\, p$: Left merge;
$p \,|_A\, p$: Communication merge;
$p[f]$: (Action) renaming;
$\mathcal{V}$: Set of process variables;
$\mathsf{noset}(p)$: Auxiliary clock removal operation;
$\mathsf{fv}(p)$: Free clock variables in a process;
$[C'/C]$: Substitution;
$\{x/C\}$: Multiple substitution;
$\simeq_\alpha$: α-congruence;
$\mathrm{lcv}(p)$: Local conflict of clock names;
$\mathsf{ldef}(p)$: Locally definable process;
$\mathsf{def}(p)$: Definable process;
$\alpha\mathsf{def}(p)$: α-definable process;
$\mathsf{csf}(p)$: Clock set-free process;
$\mathsf{Lnk}(p)$: Linked clocks in a process;
$ax\ (A)$: Set of axioms for equational theory A;
$sig(A)$: Signature of equational theory A;
$\spadesuit$: Set of all $\spadesuit$ terms;
$\spadesuit^{\mathsf{def}}$: Set of definable $\spadesuit$ terms;
$\spadesuit^{\alpha}$: Set of α-definable $\spadesuit$ terms;
$\spadesuit^{\alpha-}$: Set of non-locally conflicting α-definable and $\spadesuit$ terms;
$\spadesuit^{b}$: Set of closed basic $\spadesuit$ terms;
$\spadesuit^{p}$: Set of closed $\spadesuit$ terms;

References

[1] L. Aceto, A static view of localities, Formal Aspects Comput. 6 (1994) 201–222.
[2] M. Ajmone Marsan, A. Bianco, L. Ciminiera, R. Sisto, A. Valenzano, A LOTOS extension for the performance analysis of distributed systems, IEEE/ACM Trans. Networking 2 (2) (1994) 151–165.

[3] R. Alur, D. Dill, A theory of timed automata, Theor. Comput. Sci. 126 (1994) 183–235.
[4] J.C.M. Baeten, C. Verhoef, A congruence theorem for structured operational semantics with predicates, in: E. Best (Ed.), Concurrency Theory (CONCUR), Lecture Notes in Computer Science, vol. 715, Springer, Berlin, 1993, pp. 477–492.
[5] J.C.M. Baeten, W.P. Weijland, Process Algebra, volume 18 of Cambridge Tracts in Theoretical Computer Science, Cambridge University Press, Cambridge, 1990.
[6] J.A. Bergstra, J.W. Klop, Verification of an alternating bit protocol by means of process algebra, in: W. Bibel, K.P. Jantke (Eds.), Math. Methods of Spec. and Synthesis of Software Systems, Akademie Verlag, 1986, pp. 9–23.
[7] M. Bernardo, Theory and application of extended Markovian process algebras. PhD thesis. Università di Bologna, Padova, Venezia, 1999.
[8] M. Bernardo, R. Gorrieri, A tutorial on EMPA: a theory of concurrent processes with nondeterminism, priorities, probabilities and time, Theor. Comput. Sci. 202 (1–2) (1998) 1–54.
[9] T. Bolognesi, E. Brinksma, Introduction to the ISO specification language LOTOS, Comput. Networks 14 (1987) 25–59.
[10] M. Bravetti, Specification and analysis of stochastic real-time systems. PhD thesis. Università di Bologna, Padova, Venezia, 2002.
[11] M. Bravetti, M. Bernardo, R. Gorrieri, Towards performance evaluation with general distributions in process algebra, in: D. Sangiorgi, R. de Simone (Eds.), Concurrency Theory (CONCUR), Lecture Notes in Computer Science, vol. 1466, Springer, Berlin, 1998, pp. 405–422.
[12] M. Bravetti, P.R. D'Argenio, Tutte le algebre insiemi: concepts, discussions and relations of stochastic process algebras with general distributions, in: C. Baier (Ed.), et al., Validation of Stochastic Systems, Lecture Notes in Computer Science, vol. 2925, Springer, Berlin, 2004, pp. 44–89.
[13] M. Bravetti, R. Gorrieri, The theory of interactive generalized semi-Markov processes, Theor. Comput. Sci. 286 (2002) 5–32.
[14] E. Brinksma, H. Hermanns, Process algebra and Markov chains, in: E. Brinksma, H. Hermanns, J.-P. Katoen (Eds.), Lectures on Formal Methods and Performance Analysis, Lecture Notes in Computer Science, vol. 2090, Springer, Berlin, 2001.
[15] E. Brinksma, J.-P. Katoen, R. Langerak, D. Latella, A stochastic causality-based process algebra, Comput. J. 38 (6) (1995) 552–565.
[16] P. Buchholz, Markovian process algebra: composition and equivalence, in: U. Herzog, M. Rettelbach (Eds.). Process Algebra and Performance Modelling (PAPM). Arbeitsbericht 29(9), University of Erlangen-Nürnberg, 1994, pp. 11–30.
[17] C.G. Cassandras, Discrete Event Systems. Modeling and Performance Analysis, Aksen Associates, Irwin, 1993.
[18] M.E. Crovella, Performance evaluation with heavy tail distributions (extended abstract), in: B.R. Haverkort, H.C. Bohnenkamp, C.U. Smith (Eds.), Computer Performance Evaluation: Modelling, Techniques and Tools, Lecture Notes in Computer Science, vol. 1786, Springer, Berlin, 2000, pp. 1–9.
[19] P.R. D'Argenio, Algebras and automata for timed and stochastic systems, PhD thesis, University of Twente, 1999.
[20] P.R. D'Argenio, A compositional translation of stochastic automata into timed automata, CTIT Tech. Rep. 00-08, University of Twente, 2000.
[21] P.R. D'Argenio, H. Hermanns, J.-P. Katoen, J. Klaren, MoDeST—a modelling and description language for stochastic timed systems, in: L. de Alfaro, S. Gilmore (Eds.), Process Algebra and Probabilistic Methods, Lecture Notes in Computer Science, vol. 2165, Springer, Berlin, 2001, pp. 87–104.
[22] P.R. D'Argenio, J.-P. Katoen, A theory of stochastic systems. Part I: Stochastic automata, Inf. Comput. 2005, 10.1016/j.ic.2005.07.001.
[23] P.R. D'Argenio, J.-P. Katoen, E. Brinksma, A stochastic automata model and its algebraic approach, in: E. Brinksma, A. Nymeyer (Eds.), Process Algebra and Performance Modeling (PAPM), CTIT Tech. Rep. 97-14, pp. 1–16, University of Twente, 1997.
[24] P.R. D'Argenio, J.-P. Katoen, E. Brinksma, An algebraic approach to the specification of stochastic systems (extended abstract), in: D. Gries, W.-P. de Roever (Eds.), IFIP Working Conf. on Programming Concepts and Methods (PROCOMET), IFIP Conf. Proceedings 125, Chapman & Hall, London, 1998, pp. 126–147.

[25] P.R. D'Argenio, J.-P. Katoen, E. Brinksma, Specification and analysis of soft real-time systems: quantity and quality, in: 20th IEEE Real-Time Systems Symposium (RTSS), IEEE CS Press, 1999, pp. 104–114.
[26] P.R. D'Argenio, C. Verhoef, A general conservative extension theorem in process algebras with inequalities, Theor. Comput. Sci. 177 (2) (1997) 351–380.
[27] W.J. Fokkink, The tyft/tyxt format reduces to tree rules, in: M. Hagiya, J.C. Mitchell (Eds.), Symposium on Theoretical Aspects of Computer Software (TACS), Lecture Notes in Computer Science, vol. 789, Springer, Berlin, 1994, pp. 440–453.
[28] W.J. Fokkink, Introduction to Process Algebra, Springer, 2000.
[29] P.W. Glynn, A GSMP formalism for discrete event simulation, Proc. IEEE 77 (1) (1989) 14–23.
[30] N. Götz, U. Herzog, M. Rettelbach, TIPP—introduction and application to protocol performance analysis, in: H. König (Ed.), Formale Beschreibungstechniken für verteilte Systeme, FOKUS series, Saur Publishers, 1993.
[31] J.F. Groote, F.W. Vaandrager, Structured operational semantics and bisimulation as a congruence, Inf. Comput. 100 (2) (1992) 202–260.
[32] P.G. Harrison, N.M. Patel, Performance Modelling of Communication Networks and Computer Architectures, Addison-Wesley, Reading, MA, 1992.
[33] P.G. Harrison, B. Strulo, Stochastic process algebra for discrete event simulation, in: F. Bacelli, A. Jean-Marie, I. Mitrani (Eds.), Quantitative Methods in Parallel Systems, Esprit Basic Research Series, Springer, Berlin, 1995, pp. 18–37.
[34] P.G. Harrison, B. Strulo, SPADES: Stochastic process algebra for discrete event simulation, J. Logic Comput. 10 (1) (2000) 3–42.
[35] H. Hermanns, in: Interactive Markov Chains—The Quest for Quantified Quality, Lecture Notes in Computer Science, vol. 2428, Springer, Berlin, 2002.
[36] H. Hermanns, U. Herzog, J.-P. Katoen, Process algebra for performance evaluation, Theor. Comput. Sci. 274 (1–2) (2002) 43–87.
[37] H. Hermanns, M. Rettelbach, Syntax, semantics, equivalences, and axioms for MTIPP, in: U. Herzog, M. Rettelbach (Eds.). Process Algebra and Performance Modelling (PAPM). Arbeitsbericht 29(9), University of Erlangen-Nürnberg, 1994, pp. 71–87.
[39] J. Hillston, The nature of synchronisation, in: U. Herzog, M. Rettelbach (Eds.). Process Algebra and Performance Modelling (PAPM). Arbeitsbericht 29(9), University of Erlangen-Nürnberg, 1994, pp. 51–70.
[40] J. Hillston, A Compositional Approach to Performance Modelling, Cambridge University Press, Cambridge, 1996.
[41] J. Hillston, M. Ribaudo, Stochastic process algebras: a new approach to performance modeling, in: K. Bagchi, J. Walrand, G. Zobrist (Eds.), Modeling and Simulation of Advanced Computer Systems, Gordon Breach, 1998.
[42] C.A.R. Hoare, Communicating Sequential Processes, Prentice-Hall, New Jersey, 1985.
[43] R. Jain, The Art of Computer Systems Performance Analysis, John Wiley & Sons, New York, 1991.
[45] N. López, M. Núnez, NMSPA: a non-Markovian model for stochastic processes, in: Distributed System Validation and Verification (DSVV), 2000. http://www.math.ntu.edu.tw/∼eric/dsvv_proc/.
[46] R. Milner, Communication and Concurrency, Prentice-Hall, New Jersey, 1989.
[47] G.D. Plotkin, A structural approach to operational semantics, Report DAIMI FN-19, Computer Science Department, Aarhus University, 1981.
[48] R.J. Pooley, Integrating behavioural and simulation modelling, in: Quantitative Evaluation of Computing and Communication Systems, Lecture Notes in Computer Science, vol. 977, Springer, Berlin, 1995, pp. 102–116.
[49] C. Priami, Stochastic π-calculus with general distributions, in: M. Ribaudo (Ed.), Process Algebra and Performance Modelling (PAPM), Università di Torino, 1996, pp. 41–57.
[50] A. Rensink, Bisimilarity of open terms, Inf. Comput. 156 (2000) 345–385.
[52] T. Ruys, R. Langerak, J.-P. Katoen, D. Latella, M. Massink, First passage time analysis of stochastic process algebra using partial orders, in: T. Margaria, W. Yi (Eds.), Tools and Algorithms for the Construction and Analysis of Systems (TACAS), Lecture Notes in Computer Science, vol. 2031, Springer, Berlin, 2001, pp. 220–235.
[53] A.N. Shiryaev, in: Probability, Graduate Texts in Mathematics, vol. 95, Springer, Berlin, 1996.
[54] W. Stewart, Introduction to the Numerical Solution of Markov Chains, Princeton University Press, New Jersey, 1994.

[55] B. Strulo, Process algebra for discrete event simulation, PhD thesis, Imperial College, 1993.
[56] C. Tofts, G.M. Birtwistle, Denotational semantics for a process-based simulation language, ACM Trans. Modeling Comput. Simulation 8 (3) (1998) 281–305.
[57] W. Whitt, Continuity of generalized semi-Markov processes, Math. Oper. Res. 5 (1980) 494–501.