Comparing dtsdPBC with other stochastic process algebras

Igor V. Tarasyuk

A.P. Ershov Institute of Informatics Systems,
Siberian Branch of the Russian Academy of Sciences,
Acad. Lavrentiev pr. 6, 630090, Novosibirsk, Russian Federation
itar@iis.nsk.su

Abstract. Petri box calculus (PBC) is a well-known algebra of parallel processes with a Petri net semantics. Discrete time stochastic and deterministic PBC (dtsdPBC) extends PBC with discrete time stochastic and deterministic delays. dtsdPBC has a step operational semantics via labeled probabilistic transition systems and a Petri net denotational semantics via dtsd-boxes, a subclass of labeled discrete time stochastic and deterministic Petri nets. To evaluate performance in dtsdPBC, the underlying semi-Markov chains (SMCs) and (reduced) discrete time Markov chains (DTMCs and RDTMCs) of the process expressions are analyzed. We determine the main positive features of dtsdPBC by comparing it with well-known or similar stochastic process algebras. We classify them by the time model (continuous of discrete) and concept (integrated or orthogonal), probability distribution of stochastic delays, deterministic (including immediate) (multi)actions and semantic parallelism. The detected strong points of dtsdPBC are discrete stochastic time, deterministic multiactions and step semantics. We also discuss the analytical solution, concurrency interpretation, application area and general advantages of dtsdPBC.

Keywords: stochastic process algebra, stochastic Petri net, Petri box calculus, discrete time, stochastic delay, deterministic delay, transition system, operational semantics, time aspect, probability distribution, determinism, parallelism, comparison, classification.

1 Introduction

Process calculi, like CSP [169], ACP [20] and CCS [217] are well-known formal models for specification of computing systems and analysis of their behaviour. In such process algebras (PAs), formulas describe processes, and verification of the functionality properties of their behaviour is accomplished at a syntactic level via equivalences, axioms and inference rules. In order to represent stochastic timing and analyze the performance properties, stochastic extensions of PAs were proposed, like MTIPP [163], PEPA [164,166] and EMPA [26]. Such stochastic process algebras (SPAs) specify actions which can occur (qualitative features) and associate with the actions the distribution parameters of their random delays (quantitative characteristics).

1.1 Petri box calculus (PBC)

Petri box calculus (PBC) [29,31,30,28] is a flexible and expressive process algebra developed as a tool for specification of the Petri nets (PNs) structure and their interrelations. Its goal was also to propose a compositional semantics for high level constructs of concurrent programming languages in terms of elementary PNs. Formulas of PBC are combined from multisets of elementary actions and their conjugates, called multiactions (*basic formulas*). The empty multiset of actions is interpreted as the silent multiaction specifying an invisible activity. The operational semantics of PBC is of step type, since its SOS rules have transitions with (multi)sets of activities, corresponding to simultaneous executions of activities (steps). A denotational semantics of PBC was proposed via a subclass of PNs with an interface and considered up to isomorphism, called Petri boxes. The extensions of PBC with a deterministic, a nondeterministic or a stochastic model of time exist.

1.2 Time extensions of PBC

A time extension of PBC with a nondeterministic time model, called time Petri box calculus (tPBC), was proposed in [183]. In tPBC, timing information is added by associating time intervals with instantaneous *actions*. tPBC has a step time operational semantics in terms of labeled transition systems. Its denotational semantics was defined in terms of a subclass of labeled time Petri nets (LtPNs), based on tPNs [216] and called time Petri boxes (ct-boxes).

Another time enrichment of PBC, called Timed Petri box calculus (TPBC), was defined in [205,206], it accommodates a deterministic model of time. In contrast to tPBC, multiactions of TPBC are not instantaneous, but have time durations. TPBC has a step timed operational semantics in terms of labeled transition systems. The denotational semantics of TPBC was defined in terms of a subclass of labeled Timed Petri nets (LTPNs), based on TPNs [254] and called Timed Petri boxes (T-boxes).

The third time extension of PBC, called arc time Petri box calculus (atPBC), was constructed in [226,227], and it implements a nondeterministic time. In atPBC, multiactions are associated with time delay intervals. atPBC possesses a step time operational semantics in terms of labeled transition systems. Its denotational semantics was defined on a subclass of labeled arc time Petri nets (atPNs), based of those from [39,152], where time restrictions are associated with the arcs, called arc time Petri boxes (at-boxes). tPBC, TPBC and atPBC, all adapt the discrete time approach, but TPBC has no immediate (multi)actions (those with zero delays).

1.3 Stochastic extensions of PBC

A stochastic extension of PBC, called stochastic Petri box calculus (sPBC), was proposed in [201,197,198]. In sPBC, multiactions have stochastic delays that follow (negative) exponential distribution. Each multiaction is equipped with

a rate that is a parameter of the corresponding exponential distribution. The (instantaneous) execution of a stochastic multiaction is possible only after the corresponding stochastic time delay. The calculus has an interleaving operational semantics defined via transition systems labeled with multiactions and their rates. Its denotational semantics was defined on a subclass of labeled continuous time stochastic PNs, based on CTSPNs [207,11] and called stochastic Petri boxes (s-boxes).

sPBC was enriched with immediate multiactions having zero delay in [199,200]. We call such an extension generalized sPBC (gsPBC). An interleaving operational semantics of gsPBC was constructed via transition systems labeled with stochastic or immediate multiactions together with their rates or probabilities. A denotational semantics of gsPBC was defined via a subclass of labeled generalized stochastic PNs, based on GSPNs [207,11,12] and called generalized stochastic Petri boxes (gs-boxes).

In [261,262,263,265], we presented a discrete time stochastic extension dtsPBC of the algebra PBC. In dtsPBC, the residence time in the process states is geometrically distributed. A step operational semantics of dtsPBC was constructed via labeled probabilistic transition systems. Its denotational semantics was defined in terms of a subclass of labeled discrete time stochastic PNs (LDTSPNs), based on DTSPNs [221,223] and called discrete time stochastic Petri boxes (dts-boxes).

In [270,271,272,273,274], a calculus dtsiPBC was proposed as an extension with immediate multiactions of dtsPBC. Immediate multiactions increase the specification capability: they can model logical conditions, probabilistic branching, instantaneous probabilistic choices and activities whose durations are negligible in comparison with those of others. They are also used to specify urgent activities and the ones that are not relevant for performance evaluation. The step operational semantics of dtsiPBC was constructed with the use of labeled probabilistic transition systems. Its denotational semantics was defined in terms of a subclass of labeled discrete time stochastic and immediate PNs (LDTSIPNs), called dtsi-boxes.

In [266,267,268,269], we defined dtsdPBC, an extension of dtsiPBC with deterministic multiactions. In dtsdPBC, besides the probabilities from the real-valued interval $(0; 1)$, applied to calculate discrete time delays of stochastic multiactions, also non-negative integers are used to specify fixed delays of deterministic multiactions (including zero delay, which is the case of immediate multiactions). To resolve conflicts among deterministic multiactions, they are additionally equipped with positive real-valued weights. As argued in [302,298,299], a combination of deterministic and stochastic delays fits well to model technical systems with constant (fixed) durations of the regular non-random activities and probabilistically distributed (stochastic) durations of the randomly occurring activities. dtsdPBC has a step operational semantics, defined via labeled probabilistic transition systems. The denotational semantics of dtsdPBC was defined in terms of a subclass of labeled discrete time stochastic and deterministic Petri nets (LDTSDPNs), called dtsd-boxes.

1.4 Our contributions

As a basis model, we take *discrete time stochastic and deterministic Petri box calculus* (dtsdPBC), presented in [266,267,268,269], featuring a step operational semantics. Here we do not consider the Petri net denotational semantics of the calculus, since it was extensively described in [267]. In that paper, a consistency of the operational and denotational semantics with respect to step stochastic bisimulation equivalence was proved. Hence, all the results established for the former can be readily transferred to the latter up to that equivalence. In order to evaluate performance in dtsdPBC, the underlying semi-Markov chains (SMCs) and (reduced) discrete time Markov chains (DTMCs and RDTMCs) of the process expressions are analyzed [268].

In the present paper, the enhanced related work overview is done, where strong points of dtsdPBC with respect to other SPAs are detected. In overall, we compare dtsdPBC with more than 90 existing SPAs and then classify them according to the time model and concept, parallelism in the (operational) semantics, existence of immediate or positively deterministic (waiting) (multi)actions and (distribution) types of the stochastic delays.

If to compare dtsdPBC with the classical SPAs MTIPP, PEPA and EMPA, the first main difference between them comes from PBC, since dtsdPBC is based on this calculus: all algebraic operations and a notion of multiaction are inherited from PBC. The second main difference is discrete probabilities of activities induced by the discrete time approach, whereas action rates are used in the standard SPAs with continuous time. As a consequence, dtsdPBC has a non-interleaving step operational semantics. This is in contrast to the classical SPAs, where concurrency is modeled by interleaving because of the continuous probability distributions of action delays and the race condition applied when several actions can be executed in a state. The third main difference is deterministic (particularly, immediate) multiactions. There are no even instantaneous activities in MTIPP and PEPA while immediate actions in EMPA can have different priority levels. In dtsdPBC, all immediate (zero deterministic) multiactions have the same (highest) priority, and all waiting (positive deterministic) multiactions have the same (medium) priority (by leaving the lowest priority to stochastic multiactions). The intention is to simplify the specification and analysis, since weights (assigned also to immediate actions in EMPA) are enough to denote preferences among deterministic multiactions and to produce the conformable probabilistic behaviours.

The salient point of dtsdPBC is a combination of deterministic multiactions, discrete stochastic time and step semantics in an SPA. In the extensive discussion, analytical solution, concurrency interpretation, application area and general advantages of dtsdPBC are explained.

Thus, the main contributions of the paper are as follows.

- Comparison of dtsdPBC with existing SPAs, according to the time aspects, semantic parallelism and delay types.
- Discussion about the analytical solution, concurrency interpretation, application area and advantages of dtsdPBC.

1.5 Structure of the paper

In Section 2, the syntax of algebra dtsdPBC is proposed. In Section 3, the operational semantics of the calculus in terms of labeled probabilistic transition systems is presented. The differences and similarities between dtsdPBC and other well-known or similar SPAs are considered in Section 4. The advantages of dtsdPBC are explained in Section 5. Section 6 summarizes the results obtained and outlines future research.

2 Syntax

In this section, we define the syntax: activities, operations and expressions.

2.1 Activities and operations

Multiset allows identical elements in a set.

Definition 1. *Let X be a set. A finite* multiset (bag) *M over X is a mapping $M:X\to\mathbb{N}$ with $|\{x\in X\,|\,M(x)>0\}|<\infty$, i.e. it has a finite number of elements.*

The *set of all finite multisets* over a set X is $\mathbb{N}^X_{fin}$. Let $M, M' \in \mathbb{N}^X_{fin}$. The *cardinality* of M is $|M| = \sum_{x\in X} M(x)$. We write $x \in M$ if $M(x) > 0$ and $M \subseteq M'$ if $\forall x \in X\ M(x) \leq M'(x)$. We define $(M + M')(x) = M(x) + M'(x)$ and $(M - M')(x) = \max\{0, M(x) - M'(x)\}$. When $\forall x \in X,\ M(x) \leq 1$, M is seen as a proper set $M \subseteq X$. The *set of all subsets (powerset)* of X is 2^X.

Let $Act=\{a,b,\ldots\}$ be the set of *elementary actions*. Then $\widehat{Act}=\{\hat{a},\hat{b},\ldots\}$ is the set of *conjugated actions (conjugates)* with $\hat{a}\neq a$ and $\hat{\hat{a}}=a$. Let $\mathcal{A}=Act\cup\widehat{Act}$ be the set of *all actions*, and $\mathcal{L}=\mathbb{N}^{\mathcal{A}}_{fin}$ be the set of *all multiactions*. Here $\emptyset\in\mathcal{L}$ specifies an internal move, i.e. the execution of a multiaction without visible actions. The *alphabet* of $\alpha\in\mathcal{L}$ is $\mathcal{A}(\alpha)=\{x\in\mathcal{A}\mid\alpha(x)>0\}$.

A *stochastic multiaction* is a pair (α,ρ), where $\alpha\in\mathcal{L}$ and $\rho\in(0;1)$ is the *probability* of the multiaction α. This probability is interpreted as that of independent execution of the stochastic multiaction at the next discrete time moment. Such probabilities are used to calculate those to execute (possibly empty) sets of stochastic multiactions after one time unit delay. The probability 1 is left for (implicitly assigned to) waiting multiactions, i.e. positively delayed deterministic multiactions (to be defined later), which have weights to resolve conflicts with other waiting multiactions. Let $\mathcal{SL}$ be the set of *all stochastic multiactions*.

A *deterministic multiaction* is a pair $(\alpha,\natural^\theta_l)$, where $\alpha\in\mathcal{L}$, $\theta\in\mathbb{N}$ is the non-negative integer-valued *(fixed) delay* and $l\in\mathbb{R}_{>0}=(0;\infty)$ is the positive real-valued *weight* of the multiaction α. This weight is interpreted as a measure of importance (urgency, interest) or a bonus reward associated with execution of the deterministic multiaction at the moment when the corresponding delay has expired. Such weights are used to calculate the probabilities to execute sets of deterministic multiactions after their delays. An *immediate multiaction* is a

deterministic multiaction with the delay 0 while a *waiting multiaction* is a deterministic multiaction with a positive delay. In case of no conflicts among waiting multiactions, whose remaining times to execute (RTEs) are equal to one time unit, they are executed with probability 1 at the next moment. Deterministic multiactions have a priority over stochastic ones while immediate multiactions have a priority over waiting ones. Different types of multiactions cannot participate together in some step (parallel execution). Let $\mathcal{DL}$ be the set of *all deterministic multiactions*, $\mathcal{IL}$ be the set of *all immediate multiactions* and $\mathcal{WL}$ be the set of *all waiting multiactions*. We have $\mathcal{DL} = \mathcal{IL} \cup \mathcal{WL}$.

The same multiaction $\alpha \in \mathcal{L}$ may have different probabilities, (fixed) delays and weights in the same specification. An *activity* is a stochastic or a deterministic multiaction. Let $\mathcal{SDL} = \mathcal{SL} \cup \mathcal{DL} = \mathcal{SL} \cup \mathcal{IL} \cup \mathcal{WL}$ be the set of *all activities*. The *alphabet* of an activity $(\alpha, \kappa) \in \mathcal{SDL}$ is $\mathcal{A}(\alpha, \kappa) = \mathcal{A}(\alpha)$. The *alphabet* of a multiset of activities $\Upsilon \in \mathbb{N}_{fin}^{\mathcal{SDL}}$ is $\mathcal{A}(\Upsilon) = \cup_{(\alpha,\kappa)\in\Upsilon} \mathcal{A}(\alpha)$.

Activities are combined into formulas (process expressions) by the operations of *sequence* ;, *choice* [], *parallelism* $\|$, *relabeling* $[f]$ of actions, *restriction* rs over a single action, *synchronization* sy on an action and its conjugate, and *iteration* $[**]$ with three arguments: initialization, body and termination.

Sequence (sequential composition) and choice (composition) have a standard interpretation, like in other PAs, but parallelism (parallel composition) does not include synchronization, unlike the corresponding operation in CCS.

Relabeling functions $f : \mathcal{A} \to \mathcal{A}$ are bijections preserving conjugates, i.e. $\forall x \in \mathcal{A}\ f(\hat{x}) = \widehat{f(x)}$. Relabeling is extended to multiactions: for $\alpha \in \mathcal{L}$ we define $f(\alpha) = \sum_{x\in\alpha} f(x) = \sum_{x\in\mathcal{A}} \alpha(x) f(x)$. Relabeling is extended to activities: for $(\alpha, \kappa) \in \mathcal{SDL}$ we define $f(\alpha, \kappa) = (f(\alpha), \kappa)$. Relabeling is extended to the multisets of activities: for $\Upsilon \in \mathbb{N}_{fin}^{\mathcal{SDL}}$ we define $f(\Upsilon) = \sum_{(\alpha,\kappa)\in\Upsilon} (f(\alpha), \kappa)$.

Restriction over an elementary action $a \in Act$ means that, for a given expression, any process behaviour containing a or its conjugate $\hat{a}$ is not allowed.

Let $\alpha, \beta \in \mathcal{L}$ be two multiactions such that for some elementary action $a \in Act$ we have $a \in \alpha$ and $\hat{a} \in \beta$, or $\hat{a} \in \alpha$ and $a \in \beta$. Then, synchronization of α and β by a is defined as $(\alpha \oplus_a \beta)(x) = \begin{cases} \alpha(x) + \beta(x) - 1, & x = a \text{ or } x = \hat{a}; \\ \alpha(x) + \beta(x), & \text{otherwise.} \end{cases}$
Activities are synchronized via their multiaction parts, i.e. the synchronization by a of two activities, whose multiaction parts α and β possess the above properties, results in the activity with the multiaction part $\alpha \oplus_a \beta$. We may synchronize activities of the same type only: either both stochastic multiactions or both deterministic ones *with the same delay*, since stochastic, waiting and immediate multiactions have different priorities, and diverse delays of waiting multiactions would contradict their joint timing. Note that the execution of immediate multiactions takes no time, unlike that of waiting or stochastic ones. Synchronization by a means that, for a given expression with a process behaviour containing two concurrent activities that can be synchronized by a, there exists also the behaviour that differs from the former only in that the two activities are replaced by the result of their synchronization.

In the iteration, the initialization subprocess is executed first, then the body is performed zero or more times, and finally, the termination is executed.

2.2 Process expressions

Static expressions specify the structure of processes, i.e. how activities are combined by operations to construct the composite process-algebraic formulas. As for the PN intuition, static expressions correspond to unmarked LDTSDPNs [266,267]. A marking is the allocation of tokens in the places of a PN. Markings are used to describe dynamic behaviour of PNs in terms of transition firings.

We assume that every waiting multiaction has a countdown timer associated, whose value is the time left till the moment when the waiting multiaction can be executed. Therefore, besides standard (unstamped) waiting multiactions $(\alpha, \natural_l^\theta) \in \mathcal{WL}$, a special case of the *stamped* waiting multiactions should be considered in the definition of static expressions. Each (time) stamped waiting multiaction $(\alpha, \natural_l^\theta)^\delta$ has an extra superscript $\delta \in \{1, \ldots, \theta\}$ that specifies a time stamp indicating the *latest* value of the timer associated with that multiaction. The standard waiting multiactions have no time stamps, to demonstrate irrelevance of the timer values for them (for example, their timers have not yet started or have already finished). The notion of the alphabet part for (the multisets of) stamped waiting multiactions is defined like that for (the multisets of) unstamped waiting multiactions.

For simplicity, we do not assign the timer value superscripts δ to immediate multiactions, a special case of deterministic multiactions $(\alpha, \natural_l^\theta)$ with the delay $\theta = 0$ in the form of $(\alpha, \natural_l^0)$, since their timer values always equal to 0.

Definition 2. *Let* $(\alpha, \kappa) \in \mathcal{SDL}$, $(\alpha, \natural_l^\theta) \in \mathcal{WL}$, $\delta \in \{1, \ldots, \theta\}$ *and* $a \in Act$. *A* static expression *of dtsdPBC is*

$$E ::= (\alpha, \kappa) \mid (\alpha, \natural_l^\theta)^\delta \mid E; E \mid E[]E \mid E \| E \mid E[f] \mid E \text{ rs } a \mid E \text{ sy } a \mid [E * E * E].$$

Let *StatExpr* denote the set of *all static expressions* of dtsdPBC.

To avoid technical difficulties with the iteration operator, we should not allow concurrency at the highest level of the second argument of iteration. This is not a severe restriction, since we can always prefix parallel expressions by an activity with the empty multiaction part.

Definition 3. *Let* $(\alpha, \kappa) \in \mathcal{SDL}$, $(\alpha, \natural_l^\theta) \in \mathcal{WL}$, $\delta \in \{1, \ldots, \theta\}$ *and* $a \in Act$. *A* regular static expression *of dtsdPBC is*

$$E ::= (\alpha, \kappa) \mid (\alpha, \natural_l^\theta)^\delta \mid E; E \mid E[]E \mid E \| E \mid E[f] \mid E \text{ rs } a \mid E \text{ sy } a \mid [E * D * E],$$
$$\text{where } D ::= (\alpha, \kappa) \mid (\alpha, \natural_l^\theta)^\delta \mid D; E \mid D[]D \mid D[f] \mid D \text{ rs } a \mid D \text{ sy } a \mid [D * D * E].$$

Let *RegStatExpr* denote the set of *all regular static expressions* of dtsdPBC.

Let E be a regular static expression. The *underlying timer-free regular static expression* $\downarrow E$ of E is obtained by removing all timer value superscripts.

The set of *all stochastic multiactions (from the syntax) of* E is $\mathcal{SL}(E) = \{(\alpha, \rho) \mid (\alpha, \rho)$ is a subexpression of $E\}$. The set of *all immediate multiactions*

(from the syntax) of E is $\mathcal{IL}(E) = \{(\alpha, \natural_l^0) \mid (\alpha, \natural_l^0)$ is a subexpression of $E\}$. The set of *all waiting multiactions (from the syntax) of* E is $\mathcal{WL}(E) = \{(\alpha, \natural_l^\theta) \mid (\alpha, \natural_l^\theta)$ or $(\alpha, \natural_l^\theta)^\delta$ is a subexpression of E for $\delta \in \{1, \ldots, \theta\}\}$. Thus, the set of *all deterministic multiactions (from the syntax) of* E is $\mathcal{DL}(E) = \mathcal{IL}(E) \cup \mathcal{WL}(E)$ and the set of *all activities (from the syntax) of* E is $\mathcal{SDL}(E) = \mathcal{SL}(E) \cup \mathcal{DL}(E) = \mathcal{SL}(E) \cup \mathcal{IL}(E) \cup \mathcal{WL}(E)$.

Dynamic expressions specify the states of processes, i.e. particular stages of the process behaviour. As for the Petri net intuition, dynamic expressions correspond to marked LDTSDPNs [266,267]. Dynamic expressions are obtained from static ones, by annotating them with upper or lower bars which specify the active components of the system at the current moment of time. The dynamic expression with upper bar (the overlined one) $\overline{E}$ denotes the *initial*, and that with lower bar (the underlined one) $\underline{E}$ denotes the *final* state of the process specified by a static expression E.

For every overlined stamped waiting multiaction $\overline{(\alpha, \natural_l^\theta)^\delta}$, the superscript $\delta \in \{1, \ldots, \theta\}$ specifies the *current* value of the *running* countdown timer associated with the waiting multiaction. That decreasing discrete timer is started with the *initial* value θ (the waiting multiaction delay) at the moment when the waiting multiaction becomes overlined. Then such a newly overlined stamped waiting multiaction $\overline{(\alpha, \natural_l^\theta)^\theta}$ is similar to the freshly overlined unstamped waiting multiaction $\overline{(\alpha, \natural_l^\theta)}$. Such similarity will be captured by the structural equivalence, defined later.

While the stamped waiting multiaction stays overlined with the process execution, the timer decrements by one discrete time unit with each global time tick until the timer value becomes 1. This means that one unit of time remains till execution of that multiaction (the remaining time to execute, RTE, equals one). Its execution should follow in the next moment with probability 1, in case there are no conflicting with it immediate multiactions or conflicting waiting multiactions whose RTEs equal to one, and it is not affected by restriction. An activity is affected by restriction, if it is within the scope of a restriction operation with the argument action, such that it or its conjugate is contained in the multiaction part of that activity.

Definition 4. *Let* $E \in StatExpr$, $a \in Act$. *A* dynamic expression *of dtsdPBC is*

$$G ::= \overline{E} \mid \underline{E} \mid G; E \mid E; G \mid G[]E \mid E[]G \mid G \| G \mid G[f] \mid G \ \mathsf{rs}\ a \mid G \ \mathsf{sy}\ a \mid [G * E * E] \mid [E * G * E] \mid [E * E * G].$$

Let $DynExpr$ denote the set of *all dynamic expressions* of dtsdPBC.

Let G be a dynamic expression. The *underlying static (line-free) expression* $\lfloor G \rfloor$ of G is obtained by removing from it all upper and lower bars.

Definition 5. *A dynamic expression* G *is* regular *if* $\lfloor G \rfloor$ *is regular.*

$RegDynExpr$ denotes the set of *all regular dynamic expressions* of dtsdPBC.

Let G be a regular dynamic expression. The *underlying timer-free regular dynamic expression* $\lfloor G$ of G is got by removing all timer value superscripts.

The set of *all stochastic (immediate or waiting, respectively) multiactions (from the syntax) of* G is defined as $\mathcal{SL}(G) = \mathcal{SL}(\lfloor G \rfloor)$ ($\mathcal{IL}(G) = \mathcal{IL}(\lfloor G \rfloor)$ or $\mathcal{WL}(G) = \mathcal{WL}(\lfloor G \rfloor)$, respectively). Thus, the set of *all deterministic multiactions (from the syntax) of* G is $\mathcal{DL}(G) = \mathcal{IL}(G) \cup \mathcal{WL}(G)$ and the set of *all activities (from the syntax) of* G is $\mathcal{SDL}(G) = \mathcal{SL}(G) \cup \mathcal{DL}(G) = \mathcal{SL}(G) \cup \mathcal{IL}(G) \cup \mathcal{WL}(G)$.

3 Operational semantics

In this section, we define the operational semantics via transition systems.

3.1 Inaction rules

The inaction rules for dynamic expressions describe their structural transformations in the form of $G \Rightarrow \widetilde{G}$ which do not change the states of the specified processes. The goal of those syntactic transformations is to obtain the well-structured resulting expressions called operative ones to which no inaction rules can be further applied. The application of an inaction rule to a dynamic expression does not lead to any discrete time tick or any transition firing in the corresponding LDTSDPN [266,267], hence, its current marking stays unchanged.

An application of every inaction rule does not require a delay, i.e. the dynamic expression transformation described by the rule is accomplished instantly.

In Table 1, we define inaction rules for regular dynamic expressions being overlined and underlined static ones, where $(\alpha, \natural_l^\theta) \in \mathcal{WL}$, $\delta \in \{1, \ldots, \theta\}$, $E, F, K \in RegStatExpr$ and $a \in Act$. The first inaction rule suggests that the timer value of each newly overlined waiting multiaction is set to its delay.

Table 1. Inaction rules for overlined and underlined regular static expressions

$\overline{(\alpha, \natural_l^\theta)} \Rightarrow \overline{(\alpha, \natural_l^\theta)^\theta}$	$\overline{E; F} \Rightarrow \overline{E}; F$	$\underline{E}; F \Rightarrow E; \overline{F}$
$E; \underline{F} \Rightarrow \underline{E; F}$	$\overline{E[]F} \Rightarrow \overline{E}[]F$	$\overline{E[]F} \Rightarrow E[]\overline{F}$
$\underline{E}[]F \Rightarrow \underline{E[]F}$	$E[]\underline{F} \Rightarrow \underline{E[]F}$	$\overline{E\|\|F} \Rightarrow \overline{E}\|\|\overline{F}$
$\underline{E}\|\|\underline{F} \Rightarrow \underline{E\|\|F}$	$\overline{E[f]} \Rightarrow \overline{E}[f]$	$\underline{E}[f] \Rightarrow \underline{E[f]}$
$\overline{E \text{ rs } a} \Rightarrow \overline{E} \text{ rs } a$	$\underline{E} \text{ rs } a \Rightarrow \underline{E \text{ rs } a}$	$\overline{E \text{ sy } a} \Rightarrow \overline{E} \text{ sy } a$
$\underline{E} \text{ sy } a \Rightarrow \underline{E \text{ sy } a}$	$\overline{[E * F * K]} \Rightarrow [\overline{E} * F * K]$	$[\underline{E} * F * K] \Rightarrow [E * \overline{F} * K]$
$[E * \underline{F} * K] \Rightarrow [E * \overline{F} * K]$	$[E * \underline{F} * K] \Rightarrow [E * F * \overline{K}]$	$[E * F * \underline{K}] \Rightarrow \underline{[E * F * K]}$

In Table 2, we introduce inaction rules for regular dynamic expressions in the arbitrary form, where $E, F \in RegStatExpr$, $G, H, \widetilde{G}, \widetilde{H} \in RegDynExpr$ and

$a \in Act$. For brevity, two distinct inaction rules with the same premises are sometimes collated, resulting in the inaction rules with double conclusion.

Table 2. Inaction rules for arbitrary regular dynamic expressions

$$\frac{G \Rightarrow \widetilde{G},\ \circ \in \{;, []\}}{G \circ E \Rightarrow \widetilde{G} \circ E,\ E \circ G \Rightarrow E \circ \widetilde{G}} \qquad \frac{G \Rightarrow \widetilde{G}}{G \| H \Rightarrow \widetilde{G} \| H,\ H \| G \Rightarrow H \| \widetilde{G}}$$

$$\frac{G \Rightarrow \widetilde{G}}{G[f] \Rightarrow \widetilde{G}[f]} \qquad \frac{G \Rightarrow \widetilde{G},\ \circ \in \{\mathsf{rs}, \mathsf{sy}\}}{G \circ a \Rightarrow \widetilde{G} \circ a} \qquad \frac{G \Rightarrow \widetilde{G}}{[G * E * F] \Rightarrow [\widetilde{G} * E * F]}$$

$$\frac{G \Rightarrow \widetilde{G}}{[E * G * F] \Rightarrow [E * \widetilde{G} * F]} \qquad \frac{G \Rightarrow \widetilde{G}}{[E * F * G] \Rightarrow [E * F * \widetilde{G}]}$$

Definition 6. *A regular dynamic expression G is* operative *if no inaction rule can be applied to it.*

Let $OpRegDynExpr$ denote the set of *all operative regular dynamic expressions* of dtsdPBC. Any dynamic expression can be always transformed into a (not necessarily unique) operative one by using the inaction rules.

We shall consider regular expressions only and omit the word "regular".

Definition 7. *The relation $\approx\ = (\Rightarrow \cup \Leftarrow)^*$ is a* structural equivalence *of dynamic expressions in dtsdPBC. Thus, two dynamic expressions G and G' are* structurally equivalent*, denoted by $G \approx G'$, if they can be reached from each other by applying the inaction rules in a forward or a backward direction.*

Let G be a dynamic expression. Then $[G]_{\approx} = \{H \mid G \approx H\}$ is the equivalence class of G with respect to the structural equivalence, called the (corresponding) *state*. Next, G is an *initial* dynamic expression, denoted by $init(G)$, if $\exists E \in RegStatExpr\ G \in [\overline{E}]_{\approx}$. Further, G is a *final* dynamic expression, denoted by $final(G)$, if $\exists E \in RegStatExpr\ G \in [\underline{E}]_{\approx}$.

Let G be a dynamic expression and $s = [G]_{\approx}$. The set of *all enabled stochastic multiactions of s* is $EnaSto(s) = \{(\alpha, \rho) \in \mathcal{SL} \mid \exists H \in s \cap OpRegDynExpr\ \overline{(\alpha, \rho)}$ is a subexpression of $H\}$. The set of *all enabled immediate multiactions of s* is $EnaImm(s) = \{(\alpha, \natural_l^0) \in \mathcal{IL} \mid \exists H \in s \cap OpRegDynExpr\ \overline{(\alpha, \natural_l^0)}$ is a subexpression of $H\}$. The set of *all enabled waiting multiactions of s* is $EnaWait(s) = \{(\alpha, \natural_l^\theta) \in \mathcal{WL} \mid \exists H \in s \cap OpRegDynExpr\ \overline{(\alpha, \natural_l^\theta)^\delta},\ \delta \in \{1, \ldots, \theta\},$ is a subexpression of $H\}$. The set of *all newly enabled waiting multiactions of s* is $EnaWaitNew(s) = \{(\alpha, \natural_l^\theta) \in \mathcal{WL} \mid \exists H \in s \cap OpRegDynExpr\ \overline{(\alpha, \natural_l^\theta)^\theta}$ is a subexpression of $H\}$.

The set of *all enabled deterministic multiactions of s* is $EnaDet(s) = EnaImm(s) \cup EnaWait(s)$ and the set of *all enabled activities of s* is $Ena(s) = EnaSto(s) \cup EnaDet(s) = EnaSto(s) \cup EnaImm(s) \cup EnaWait(s)$. Then

$Ena(s) = Ena([G]_{\approx})$ is an algebraic analogue of the set of all transitions enabled at the initial marking of the LDTSDPN [266,267] corresponding to G. The activities, resulted from synchronization, are not present in the syntax of the dynamic expressions. Their enabledness status can be recovered by observing that of the pair of synchronized activities from the syntax (they both should be enabled for enabling their synchronous product), even if they are affected by restriction after the synchronization.

Definition 8. *An operative dynamic expression G is* saturated (with the values of timers)*, if each enabled waiting multiaction of $[G]_{\approx}$, being superscribed with the value of its timer and possibly overlined, is the subexpression of G.*

Let *SaOpRegDynExpr* denote the set of *all saturated operative dynamic expressions* of dtsdPBC.

Proposition 1. *Any operative dynamic expression can be always transformed into the saturated one by a forward or a backward applying the inaction rules.*

Proof. See [266,267]. □

Thus, any dynamic expression can be transformed into a (not always unique) saturated operative one by (possibly reverse) applying the inaction rules.

Let G be a saturated operative dynamic expression. Then $\circlearrowleft G$ denotes the *timer decrement* operator $\circlearrowleft$, applied to G. The result is a saturated operative dynamic expression, obtained from G via decrementing by one all greater than 1 values of the timers associated with all (if any) stamped waiting multiactions from the syntax of G. Each such stamped waiting multiaction changes its timer value from $\delta \in \mathbb{N}_{\geq 1}$ in G to $\max\{1, \delta - 1\}$ in $\circlearrowleft G$. The timer decrement operator affects the (possibly overlined or underlined) stamped waiting multiactions being the subexpressions of G as: $(\alpha, \natural_l^{\theta})^{\delta}$ is replaced with $(\alpha, \natural_l^{\theta})^{\max\{1,\delta-1\}}$, and similarly for the overlined or underlined ones.

Note that when $\delta = 1$, we have $\max\{1, \delta - 1\} = \max\{1, 0\} = 1$, hence, the timer value $\delta = 1$ may remain unchanged for a stamped waiting multiaction that is not executed by some reason at the next time moment, but stays stamped. For example, that stamped waiting multiaction may be affected by restriction. If the timer values cannot be decremented with a time tick for all stamped waiting multiactions (if any) from G then $\circlearrowleft G = G$ and we obtain so-called *empty loop* transition, defined later.

The timer decrement operator keeps stamping of the waiting multiactions, since it may only decrease their timer values, and the stamped waiting multiactions stay stamped (with their timer values, possibly decremented by one).

3.2 Action and empty move rules

The action rules are applied when some activities are executed. With these rules we capture the prioritization among different types of multiactions. We also have the empty move rule, used to capture a delay of one discrete time unit

when no immediate or waiting multiactions are executable. In this case, the empty multiset of activities is executed. The action and empty move rules will be used later to determine all multisets of activities which can be executed from the structural equivalence class of every dynamic expression (i.e. from the state of the corresponding process). This information together with that about probabilities or delays and weights of the activities to be executed from the current process state will be used to calculate the probabilities of such executions.

The action rules with stochastic (immediate or waiting, respectively) multiactions describe dynamic expression transformations in the form of $G \stackrel{\Gamma}{\to} \widetilde{G}$ ($G \stackrel{I}{\to} \widetilde{G}$ or $G \stackrel{W}{\to} \widetilde{G}$, respectively) due to execution of non-empty multisets Γ of stochastic (I of immediate or W of waiting, respectively) multiactions. The rules represent possible state changes of the specified processes when some non-empty multisets of stochastic (immediate or waiting, respectively) multiactions are executed. The application of an action rule with stochastic (immediate or waiting, respectively) multiactions to a dynamic expression leads in the corresponding LDTSDPN [266,267] to a discrete time tick at which some stochastic or waiting transitions fire (or to the instantaneous firing of some immediate transitions) and possible change of the current marking. The current marking stays unchanged only if there is a self-loop produced by the iterative execution of a non-empty multiset, which must be one-element, since we allow no concurrency at the highest level of the second argument of iteration.

The empty move rule (applicable only when no immediate or waiting multiactions can be executed from the current state) describes dynamic expression transformations in the form of $G \stackrel{\emptyset}{\to} \circlearrowleft G$, called the *empty moves*, due to execution of the empty multiset of activities at a discrete time tick. When no timer values are decremented within G with the empty multiset execution at the next moment (for example, if G contains no stamped waiting multiactions), we have $\circlearrowleft G = G$. In such a case, the empty move from G is in the form of $G \stackrel{\emptyset}{\to} G$, called the *empty loop*. The application of the empty move rule to a dynamic expression leads to a discrete time tick in the corresponding LDTSDPN [266,267] at which no transitions fire and the current marking is not changed, but the timer values of the waiting transitions enabled at the marking (if any) are decremented by one. This is a new rule that has no prototype among inaction rules of PBC, since it represents a time delay.

Thus, an application of every action rule with stochastic or waiting multiactions or the empty move rule requires one discrete time unit delay, i.e. the execution of a (possibly empty) multiset of stochastic or (non-empty) multiset of waiting multiactions leading to the dynamic expression transformation described by the rule is accomplished instantly after one time unit. An application of every action rule with immediate multiactions does not take any time, i.e. the execution of a (non-empty) multiset of immediate multiactions is accomplished instantly at the current moment.

The expressions of dtsdPBC can contain identical activities. To avoid technical difficulties, such as calculation of the probabilities for multiple transitions,

we can enumerate coinciding activities from left to right in the syntax of expressions. The new activities, resulted from synchronization, will be annotated with concatenation of numberings of the activities they come from, hence, the numbering should have a tree structure to reflect the effect of multiple synchronizations. We now define the numbering which encodes a binary tree with the leaves labeled by natural numbers.

Definition 9. *The* numbering *of expressions is* $\iota ::= n \mid (\iota)(\iota)$*, where* $n \in \mathbb{N}$.

Let Num denote the set of *all numberings* of expressions.

The new activities resulting from synchronizations in different orders should be considered up to permutation of their numbering. In this way, we shall recognize different instances of the same activity. If we compare the contents of different numberings, i.e. the sets of natural numbers in them, we shall identify the mentioned instances. The *content* of a numbering $\iota \in Num$ is

$$Cont(\iota) = \begin{cases} \{\iota\}, & \iota \in \mathbb{N}; \\ Cont(\iota_1) \cup Cont(\iota_2), & \iota = (\iota_1)(\iota_2). \end{cases}$$

After the enumeration, the multisets of activities from the expressions become proper sets. We suppose that the identical activities are enumerated when needed to avoid ambiguity. This enumeration is considered to be implicit.

Definition 10. *Let* $G \in OpRegDynExpr$*. We define* $Can(G)$*, the* set of all non-empty multisets of activities which can be potentially executed from G*. Let* $(\alpha, \kappa) \in \mathcal{SDL}$*,* $E, F \in RegStatExpr$*,* $H \in OpRegDynExpr$ *and* $a \in Act$*.*

1. *If* $final(G)$ *then* $Can(G) = \emptyset$*.*
2. *If* $G = \overline{(\alpha, \kappa)^{\delta}}$ *and* $\kappa = \natural_l^{\theta}$*,* $\theta \in \mathbb{N}_{\geq 2}$*,* $l \in \mathbb{R}_{>0}$*,* $\delta \in \{2, \ldots, \theta\}$*, then* $Can(G) = \emptyset$*.*
3. *If* $G = \overline{(\alpha, \kappa)}$ *and* $\kappa \in (0; 1)$ *or* $\kappa = \natural_l^0$*,* $l \in \mathbb{R}_{>0}$*, then* $Can(G) = \{\{(\alpha, \kappa)\}\}$*.*
4. *If* $G = \overline{(\alpha, \kappa)^1}$ *and* $\kappa = \natural_l^{\theta}$*,* $\theta \in \mathbb{N}_{\geq 1}$*,* $l \in \mathbb{R}_{>0}$*, then* $Can(G) = \{\{(\alpha, \kappa)\}\}$*.*
5. *If* $\Upsilon \in Can(G)$ *then* $\Upsilon \in Can(G \circ E)$*,* $\Upsilon \in Can(E \circ G)$ $(\circ \in \{;, []\})$*,* $\Upsilon \in Can(G \| H)$*,* $\Upsilon \in Can(H \| G)$*,* $f(\Upsilon) \in Can(G[f])$*,* $\Upsilon \in Can(G \textsf{ rs } a)$ *(when* $a, \hat{a} \notin \mathcal{A}(\Upsilon)$*),* $\Upsilon \in Can(G \textsf{ sy } a)$*,* $\Upsilon \in Can([G * E * F])$*,* $\Upsilon \in Can([E * G * F])$*,* $\Upsilon \in Can([E * F * G])$*.*
6. *If* $\Upsilon \in Can(G)$ *and* $\Xi \in Can(H)$ *then* $\Upsilon + \Xi \in Can(G \| H)$*.*
7. *If* $\Upsilon \in Can(G \textsf{ sy } a)$ *and* $(\alpha, \kappa), (\beta, \lambda) \in \Upsilon$ *are different,* $a \in \alpha$*,* $\hat{a} \in \beta$*, then*
   (a) $\Upsilon - \{(\alpha, \kappa), (\beta, \lambda)\} + \{(\alpha \oplus_a \beta, \kappa \cdot \lambda)\} \in Can(G \textsf{ sy } a)$ *if* $\kappa, \lambda \in (0; 1)$*;*
   (b) $\Upsilon - \{(\alpha, \kappa), (\beta, \lambda)\} + \{(\alpha \oplus_a \beta, \natural_{l+m}^{\theta})\} \in Can(G \textsf{ sy } a)$ *if* $\kappa = \natural_l^{\theta}$*,* $\lambda = \natural_m^{\theta}$*,* $\theta \in \mathbb{N}$*,* $l, m \in \mathbb{R}_{>0}$*.*
   When we synchronize a multiset of activities in different orders, we get several activities with the same multiaction and probability or delay and weight parts, but different numberings with the same content. Then we only consider a single resulting activity.

If $\Upsilon \in Can(G)$ then by definition of $Can(G)$, $\forall \Xi \subseteq \Upsilon$, $\Xi \neq \emptyset$, we get $\Xi \in Can(G)$.

Let $G \in OpRegDynExpr$ and $Can(G) \neq \emptyset$. Obviously, if there are only stochastic (immediate or waiting, respectively) multiactions in the multisets

from $Can(G)$ then these stochastic (immediate or waiting, respectively) multiactions can be executed from G. Otherwise, besides stochastic ones, there are also deterministic (immediate and/or waiting) multiactions in the multisets from $Can(G)$. By the note above, there are non-empty multisets of deterministic multiactions in $Can(G)$ as well, i.e. $\exists \Upsilon \in Can(G)\ \Upsilon \in \mathbb{N}_{fin}^{\mathcal{DL}} \setminus \{\emptyset\}$. In this case, no stochastic multiactions can be executed from G, even if $Can(G)$ contains non-empty multisets of stochastic multiactions, since deterministic multiactions have a priority over stochastic ones, and should be executed first. Further, if there are no stochastic, but both waiting and immediate multiactions in the multisets from $Can(G)$, then, analogously, no waiting multiactions can be executed from G, since immediate multiactions have a priority over waiting ones (besides that over stochastic ones).

When there are only waiting and, possibly, stochastic multiactions in the multisets from $Can(G)$ then only waiting ones can be executed from G. Then just *maximal* non-empty multisets of waiting multiactions can be executed from G, since all non-conflicting waiting multiactions cannot wait and they should occur at the next time moment with probability 1.

Definition 11. *Let $G \in OpRegDynExpr$. The* set of all non-empty multisets of activities which can be executed from G *is*

$$Now(G)=\begin{cases} Can(G)\cap\mathbb{N}_{fin}^{\mathcal{IL}}, & Can(G)\cap\mathbb{N}_{fin}^{\mathcal{IL}}\neq\emptyset;\\ \{W\in Can(G)\cap\mathbb{N}_{fin}^{\mathcal{WL}} \mid & (Can(G)\cap\mathbb{N}_{fin}^{\mathcal{IL}}=\emptyset)\wedge\\ \forall V\in Can(G)\cap\mathbb{N}_{fin}^{\mathcal{WL}}\ W\subseteq V\Rightarrow V=W\}, & (Can(G)\cap\mathbb{N}_{fin}^{\mathcal{WL}}\neq\emptyset);\\ Can(G), & \textit{otherwise.}\end{cases}$$

Let $G \in OpRegDynExpr$. The expression G is *s-tangible (stochastically tangible)*, denoted by $stang(G)$, if $Now(G) \subseteq \mathbb{N}_{fin}^{\mathcal{SL}} \setminus \{\emptyset\}$. In particular, we have $stang(G)$, if $Now(G) = \emptyset$. The expression G is *w-tangible (waitingly tangible)*, denoted by $wtang(G)$, if $\emptyset \neq Now(G) \subseteq \mathbb{N}_{fin}^{\mathcal{WL}} \setminus \{\emptyset\}$. The expression G is *tangible*, denoted by $tang(G)$, if $stang(G)$ or $wtang(G)$, i.e. $Now(G) \subseteq (\mathbb{N}_{fin}^{\mathcal{SL}} \cup \mathbb{N}_{fin}^{\mathcal{WL}}) \setminus \{\emptyset\}$. Again, we particularly have $tang(G)$, if $Now(G) = \emptyset$. Otherwise, the expression G is *vanishing*, denoted by $vanish(G)$, and in this case $\emptyset \neq Now(G) \subseteq \mathbb{N}_{fin}^{\mathcal{IL}} \setminus \{\emptyset\}$. Note that the operative dynamic expressions from $[G]_{\approx}$ may have different types.

Let $G \in RegDynExpr$. We write $stang([G]_{\approx})$, if $\forall H \in [G]_{\approx} \cap OpRegDynExpr\ stang(H)$. We write $wtang([G]_{\approx})$, if $\exists H \in [G]_{\approx} \cap OpRegDynExpr\ wtang(H)$ and $\forall H' \in [G]_{\approx} \cap OpRegDynExpr\ tang(H')$. We write $tang([G]_{\approx})$, if $stang([G]_{\approx})$ or $wtang([G]_{\approx})$. Otherwise, we write $vanish([G]_{\approx})$, and in this case $\exists H \in [G]_{\approx} \cap OpRegDynExpr\ vanish(H)$.

In Table 3, we define the action and empty move rules, where $(\alpha, \rho), (\beta, \chi) \in \mathcal{SL},\ (\alpha, \natural_l^0), (\beta, \natural_m^0) \in \mathcal{IL},\ (\alpha, \natural_l^\theta), (\beta, \natural_m^\theta) \in \mathcal{WL},\ E, F \in RegStatExpr,\ G, H \in SatOpRegDynExpr,\ \widetilde{G}, \widetilde{H} \in RegDynExpr,\ a \in Act,\ \Gamma, \Delta \in \mathbb{N}_{fin}^{\mathcal{SL}} \setminus \{\emptyset\},\ \Gamma' \in \mathbb{N}_{fin}^{\mathcal{SL}},\ I, J \in \mathbb{N}_{fin}^{\mathcal{IL}} \setminus \{\emptyset\},\ I' \in \mathbb{N}_{fin}^{\mathcal{IL}},\ V, W \in \mathbb{N}_{fin}^{\mathcal{WL}} \setminus \{\emptyset\},\ V' \in \mathbb{N}_{fin}^{\mathcal{WL}}$ and $\Upsilon \in \mathbb{N}_{fin}^{\mathcal{SDL}} \setminus \{\emptyset\}$. We denote $\Upsilon_a = \{(\alpha, \kappa) \in \Upsilon \mid (a \in \alpha) \vee (\hat{a} \in \alpha)\}$.

We use the following abbreviations in the names of the rules: "**E**" for "**E**mpty move", "**B**" for "**B**asis case", "**S**" for "**S**equence", "**C**" for "**C**hoice", "**P**" for

"**P**arallel", "**L**" for "re**L**abeling", "**R**" for "**R**estriction", "**I**" for "**I**teraton" and "**Sy**" for "**Sy**nchronization". The first rule in the table is the empty move rule **E**. The other rules are the action rules, describing transformations of dynamic expressions, which are built using particular algebraic operations. If we cannot merge the rules with stochastic, immediate ans waiting multiactions in one rule for some operation then we get the coupled action rules. In such cases, the names of the action rules with stochastic multiactions have a suffix '**s**', those with immediate multiactions have a suffix '**i**', and those with waiting multiactions have a suffix '**w**'. For explanation of the rules in Table 3, see [266,267].

Table 3. Action and empty move rules

$$\mathbf{E}\ \frac{stang([G]_{\approx})}{G \xrightarrow{\emptyset} \circlearrowleft G} \qquad \mathbf{Bs}\ \overline{(\alpha,\rho)} \xrightarrow{\{(\alpha,\rho)\}} \underline{(\alpha,\rho)} \qquad \mathbf{Bi}\ \overline{(\alpha,\natural_l^0)} \xrightarrow{\{(\alpha,\natural_l^0)\}} \underline{(\alpha,\natural_l^0)} \qquad \mathbf{Bw}\ \overline{(\alpha,\natural_l^\theta)^1} \xrightarrow{\{(\alpha,\natural_l^\theta)\}} \underline{(\alpha,\natural_l^\theta)}$$

$$\mathbf{S}\ \frac{G \xrightarrow{\Upsilon} \widetilde{G}}{G;E \xrightarrow{\Upsilon} \widetilde{G};E,\ E;G \xrightarrow{\Upsilon} E;\widetilde{G}} \qquad \mathbf{Cs}\ \frac{G \xrightarrow{\Gamma} \widetilde{G},\ \neg init(G) \vee (init(G) \wedge stang([\overline{E}]_{\approx}))}{G[]E \xrightarrow{\Gamma} \widetilde{G}[]\downarrow E,\ E[]G \xrightarrow{\Gamma} \downarrow E[]\widetilde{G}}$$

$$\mathbf{Ci}\ \frac{G \xrightarrow{I} \widetilde{G}}{G[]E \xrightarrow{I} \widetilde{G}[]\downarrow E,\ E[]G \xrightarrow{I} \downarrow E[]\widetilde{G}} \qquad \mathbf{Cw}\ \frac{G \xrightarrow{V} \widetilde{G},\ \neg init(G) \vee (init(G) \wedge tang([\overline{E}]_{\approx}))}{G[]E \xrightarrow{V} \widetilde{G}[]\downarrow E,\ E[]G \xrightarrow{V} \downarrow E[]\widetilde{G}}$$

$$\mathbf{P1s}\ \frac{G \xrightarrow{\Gamma} \widetilde{G},\ stang([H]_{\approx})}{G\|H \xrightarrow{\Gamma} \widetilde{G}\| \circlearrowleft H,\ H\|G \xrightarrow{\Gamma} \circlearrowleft H\|\widetilde{G}} \qquad \mathbf{P1i}\ \frac{G \xrightarrow{I} \widetilde{G}}{G\|H \xrightarrow{I} \widetilde{G}\|H,\ H\|G \xrightarrow{I} H\|\widetilde{G}}$$

$$\mathbf{P1w}\ \frac{G \xrightarrow{V} \widetilde{G},\ stang([H]_{\approx})}{G\|H \xrightarrow{V} \widetilde{G}\| \circlearrowleft H,\ H\|G \xrightarrow{V} \circlearrowleft H\|\widetilde{G}} \qquad \mathbf{P2s}\ \frac{G \xrightarrow{\Gamma} \widetilde{G},\ H \xrightarrow{\Delta} \widetilde{H}}{G\|H \xrightarrow{\Gamma+\Delta} \widetilde{G}\|\widetilde{H}} \qquad \mathbf{P2i}\ \frac{G \xrightarrow{I} \widetilde{G},\ H \xrightarrow{J} \widetilde{H}}{G\|H \xrightarrow{I+J} \widetilde{G}\|\widetilde{H}}$$

$$\mathbf{P2w}\ \frac{G \xrightarrow{V} \widetilde{G},\ H \xrightarrow{W} \widetilde{H}}{G\|H \xrightarrow{V+W} \widetilde{G}\|\widetilde{H}} \qquad \mathbf{L}\ \frac{G \xrightarrow{\Upsilon} \widetilde{G}}{G[f] \xrightarrow{f(\Upsilon)} \widetilde{G}[f]} \qquad \mathbf{R}\ \frac{G \xrightarrow{\Upsilon} \widetilde{G}}{G\ \mathsf{rs}\ a \xrightarrow{\Upsilon-\Upsilon_a} \widetilde{G}\ \mathsf{rs}\ a}$$

$$\mathbf{I1}\ \frac{G \xrightarrow{\Upsilon} \widetilde{G}}{[G*E*F] \xrightarrow{\Upsilon} [\widetilde{G}*E*F]} \qquad \mathbf{I2s}\ \frac{G \xrightarrow{\Gamma} \widetilde{G},\ \neg init(G) \vee (init(G) \wedge stang([\overline{F}]_{\approx}))}{[E*G*F] \xrightarrow{\Gamma} [E*\widetilde{G}*\downarrow F],\ [E*F*G] \xrightarrow{\Gamma} [E*\downarrow F*\widetilde{G}]}$$

$$\mathbf{I2i}\ \frac{G \xrightarrow{I} \widetilde{G}}{[E*G*F] \xrightarrow{I} [E*\widetilde{G}*\downarrow F],\ [E*F*G] \xrightarrow{I} [E*\downarrow F*\widetilde{G}]}$$

$$\mathbf{I2w}\ \frac{G \xrightarrow{V} \widetilde{G},\ \neg init(G) \vee (init(G) \wedge tang([\overline{F}]_{\approx}))}{[E*G*F] \xrightarrow{V} [E*\widetilde{G}*\downarrow F],\ [E*F*G] \xrightarrow{V} [E*\downarrow F*\widetilde{G}]}$$

$$\mathbf{Sy1}\ \frac{G \xrightarrow{\Upsilon} \widetilde{G}}{G\ \mathsf{sy}\ a \xrightarrow{\Upsilon} \widetilde{G}\ \mathsf{sy}\ a} \qquad \mathbf{Sy2s}\ \frac{G\ \mathsf{sy}\ a \xrightarrow{\Gamma'+\{(\alpha,\rho)\}+\{(\beta,\chi)\}} \widetilde{G}\ \mathsf{sy}\ a,\ a \in \alpha,\ \hat{a} \in \beta}{G\ \mathsf{sy}\ a \xrightarrow{\Gamma'+\{(\alpha\oplus_a\beta,\rho\cdot\chi)\}} \widetilde{G}\ \mathsf{sy}\ a}$$

$$\mathbf{Sy2i}\ \frac{G\ \mathsf{sy}\ a \xrightarrow{I'+\{(\alpha,\natural_l^0)\}+\{(\beta,\natural_m^0)\}} \widetilde{G}\ \mathsf{sy}\ a,\ a \in \alpha,\ \hat{a} \in \beta}{G\ \mathsf{sy}\ a \xrightarrow{I'+\{(\alpha\oplus_a\beta,\natural_{l+m}^0)\}} \widetilde{G}\ \mathsf{sy}\ a}$$

$$\mathbf{Sy2w}\ \frac{G\ \mathsf{sy}\ a \xrightarrow{V'+\{(\alpha,\natural_l^\theta)\}+\{(\beta,\natural_m^\theta)\}} \widetilde{G}\ \mathsf{sy}\ a,\ a \in \alpha,\ \hat{a} \in \beta}{G\ \mathsf{sy}\ a \xrightarrow{V'+\{(\alpha\oplus_a\beta,\natural_{l+m}^\theta)\}} \widetilde{G}\ \mathsf{sy}\ a}$$

Notice that the timers of all waiting multiactions that lose their enabledness when a state change occurs become inactive (turned off) and their values become irrelevant while the timers of all those preserving their enabledness continue running with their stored values. Hence, we adapt the *enabling memory* policy [208,1,11,12] when the process states are changed and the enabledness of deterministic multiactions is possibly modified (immediate multiactions may be seen as those with the timers displaying a single value 0, so we do not need to store their values). Then the timer values of waiting multiactions are taken as the enabling memory variables.

Like in [183], we are interested in the dynamic expressions, inferred by applying the inaction rules (also in the reverse direction) and action rules from the overlined static expressions, such that no stamped (superscribed with the timer values) waiting multiaction is a subexpression of them. The reason is to ensure that time proceeds uniformly and only enabled waiting multiactions are stamped. We call such dynamic expressions reachable, by analogy with the reachable states of LDTSDPNs [266,267].

Definition 12. *A dynamic expression G is* reachable*, if there exists a static expression E without timer value superscripts, such that $\overline{E} \approx G$ or $\overline{E} \approx G_0 \stackrel{\Upsilon_1}{\rightarrow} H_1 \approx G_1 \stackrel{\Upsilon_2}{\rightarrow} \ldots \stackrel{\Upsilon_n}{\rightarrow} H_n \approx G$ for some $\Upsilon_1, \ldots, \Upsilon_n \in \mathbb{N}_{fin}^{\mathcal{SDL}}$.*

We now consider the enabledness of the stamped waiting multiactions.

Proposition 2. *Let G be a reachable dynamic expression. Then only waiting multiactions from $EnaWait([G]_{\approx})$ are stamped in G.*

Proof. See [266,267]. □

3.3 Transition systems

We now construct labeled probabilistic transition systems associated with dynamic expressions. The transition systems are used to define the operational semantics of dynamic expressions.

Let G be a dynamic expression and $s = [G]_{\approx}$. The set of *all multisets of activities executable in* s is defined as $Exec(s) = \{\Upsilon \mid \exists H \in s \; \exists \widetilde{H} \; H \stackrel{\Upsilon}{\rightarrow} \widetilde{H}\}$. Here $H \stackrel{\Upsilon}{\rightarrow} \widetilde{H}$ is an inference by the rules from Table 3. It can be proved by induction on the structure of expressions that $\Upsilon \in Exec(s) \setminus \{\emptyset\}$ implies $\exists H \in s \; \Upsilon \in Now(H)$. The reverse statement does not hold, since the preconditions in the action rules disable executions of the activities with the lower-priority types from every $H \in s$, see [266,267].

The state s is *s-tangible (stochastically tangible)*, denoted by $stang(s)$, if $Exec(s) \subseteq \mathbb{N}_{fin}^{\mathcal{SL}}$. For an s-tangible state s we always have $\emptyset \in Exec(s)$ by rule **E**, hence, we may have $Exec(s) = \{\emptyset\}$. The state s is *w-tangible (waitingly tangible)*, denoted by $wtang(s)$, if $Exec(s) \subseteq \mathbb{N}_{fin}^{\mathcal{WL}} \setminus \{\emptyset\}$. The state s is *tangible*, denoted by $tang(s)$, if $stang(s)$ or $wtang(s)$, i.e. $Exec(s) \subseteq \mathbb{N}_{fin}^{\mathcal{SL}} \cup \mathbb{N}_{fin}^{\mathcal{WL}}$. Again, for a tangible state s we may have $\emptyset \in Exec(s)$ and $Exec(s) = \{\emptyset\}$. Otherwise, the state s is *vanishing*, denoted by $vanish(s)$, and in this case $Exec(s) \subseteq \mathbb{N}_{fin}^{\mathcal{IL}} \setminus \{\emptyset\}$.

Definition 13. *The* derivation set *of a dynamic expression G, denoted by $DR(G)$, is the minimal set such that*

- $[G]_{\approx} \in DR(G)$;
- *if* $[H]_{\approx} \in DR(G)$ *and* $\exists \Upsilon\ H \xrightarrow{\Upsilon} \widetilde{H}$ *then* $[\widetilde{H}]_{\approx} \in DR(G)$.

The set of *all s-tangible states from* $DR(G)$ is denoted by $DR_{ST}(G)$, and the set of *all w-tangible states from* $DR(G)$ is denoted by $DR_{WT}(G)$. The set of *all tangible states from* $DR(G)$ is denoted by $DR_T(G) = DR_{ST}(G) \cup DR_{WT}(G)$. The set of *all vanishing states from* $DR(G)$ is denoted by $DR_V(G)$. Then $DR(G) = DR_T(G) \cup DR_V(G) = DR_{ST}(G) \cup DR_{WT}(G) \cup DR_V(G)$.

Let now G be a dynamic expression and $s, \tilde{s} \in DR(G)$.

Let $\Upsilon \in Exec(s) \setminus \{\emptyset\}$. The *probability that the multiset of stochastic multiactions Υ is ready for execution in s* or the *weight of the multiset of deterministic multiactions Υ which is ready for execution in s* is

$$PF(\Upsilon, s) = \begin{cases} \displaystyle\prod_{(\alpha,\rho)\in\Upsilon} \rho \cdot \prod_{\{(\beta,\chi)\}\in Exec(s)|(\beta,\chi)\notin\Upsilon\}} (1-\chi), & s \in DR_{ST}(G); \\ \displaystyle\sum_{(\alpha,\natural_l^\theta)\in\Upsilon} l, & s \in DR_{WT}(G) \cup DR_V(G). \end{cases}$$

In the case $\Upsilon = \emptyset$ and $s \in DR_{ST}(G)$ we define

$$PF(\emptyset, s) = \begin{cases} \displaystyle\prod_{\{(\beta,\chi)\}\in Exec(s)} (1-\chi), & Exec(s) \neq \{\emptyset\}; \\ 1, & Exec(s) = \{\emptyset\}. \end{cases}$$

Let $\Upsilon \in Exec(s)$. Besides Υ, other multisets of activities may be ready for execution in s, hence, a normalization is needed to calculate the execution probability. The *probability to execute the multiset of activities Υ in s* is

$$PT(\Upsilon, s) = \frac{PF(\Upsilon, s)}{\sum_{\Xi\in Exec(s)} PF(\Xi, s)}.$$

The *probability to move from s to $\tilde{s}$ by executing any multiset of activities* is

$$PM(s, \tilde{s}) = \sum_{\{\Upsilon|\exists H\in s\ \exists \widetilde{H}\in\tilde{s}\ H\xrightarrow{\Upsilon}\widetilde{H}\}} PT(\Upsilon, s).$$

Definition 14. *Let G be a dynamic expression. The* (labeled probabilistic) transition system *of G is a quadruple $TS(G) = (S_G, L_G, \mathcal{T}_G, s_G)$, where*

- *the set of* states *is* $S_G = DR(G)$;
- *the set of* labels *is* $L_G = \mathbb{N}_{fin}^{\mathcal{SPL}} \times (0; 1]$;
- *the set of* transitions *is* $\mathcal{T}_G = \{(s, (\Upsilon, PT(\Upsilon, s)), \tilde{s}) \mid s, \tilde{s} \in DR(G), \exists H \in s\ \exists \widetilde{H} \in \tilde{s}\ H \xrightarrow{\Upsilon} \widetilde{H}\}$;
- *the* initial state *is* $s_G = [G]_{\approx}$.

The transition system $TS(G)$ associated with a dynamic expression G describes all the steps (parallel executions) that occur at discrete time moments with some (one-step) probability and consist of multisets of activities. Every step consisting of stochastic (waiting, respectively) multiactions or the empty step (consisting of the empty multiset of activities) occurs instantly after one discrete time unit delay. Each step consisting of immediate multiactions occurs instantly without any delay. The step can change the current state to a different one. The states are the structural equivalence classes of dynamic expressions obtained by application of action rules starting from the expressions belonging to $[G]_{\approx}$. A transition $(s,(\Upsilon,\mathcal{P}),\tilde{s}) \in \mathcal{T}_G$ will be written as $s \stackrel{\Upsilon}{\rightarrow}_{\mathcal{P}} \tilde{s}$. It is interpreted as: the probability to change from state s to $\tilde{s}$ as a result of executing Υ is $\mathcal{P}$.

From every s-tangible state the empty multiset of activities can always be executed by rule **E**. Hence, for s-tangible states, Υ may be the empty multiset, and its execution only decrements by one the timer values (if any) of the current state. Then we have a transition $s \stackrel{\emptyset}{\rightarrow}_{\mathcal{P}} \circlearrowleft s$ from an s-tangible state s to the tangible state $\circlearrowleft s = [\circlearrowleft H]_{\approx}$ for $H \in s \cap SatOpRegDynExpr$. Since structurally equivalent saturated operative dynamic expressions remain so after decreasing by one their timers, $\circlearrowleft s$ is unique for each s and the definition is correct. Thus, $\circlearrowleft s$ corresponds to applying the empty move rule to an arbitrary saturated operative dynamic expression from s, followed by taking the structural equivalence class of the result. We have to keep track of such executions, called the *empty moves*, since they affect the timers and have non-zero probabilities. This follows from the definition of $PF(\emptyset, s)$ and the fact that the probabilities of stochastic multiactions belong to the interval $(0;1)$. When it holds $\circlearrowleft H = H$ for $H \in s \cap SatOpRegDynExpr$, we obtain $\circlearrowleft s = s$. Then the empty move from s is in the form of $s \stackrel{\emptyset}{\rightarrow}_{\mathcal{P}} s$, called the *empty loop*. For w-tangible and vanishing states Υ cannot be the empty multiset, since we must execute some immediate (waiting) multiactions from them at the current (next) moment.

The step probabilities belong to the interval $(0;1]$, being 1 when the only transition from an s-tangible state s is the empty move one $s \stackrel{\emptyset}{\rightarrow}_1 \circlearrowleft s$, or if there is a single transition from a w-tangible or a vanishing state. We write $s \stackrel{\Upsilon}{\rightarrow} \tilde{s}$ if $\exists \mathcal{P}\ s \stackrel{\Upsilon}{\rightarrow}_{\mathcal{P}} \tilde{s}$ and $s \rightarrow \tilde{s}$ if $\exists \Upsilon\ s \stackrel{\Upsilon}{\rightarrow} \tilde{s}$.

Isomorphism is a coincidence of systems up to renaming of their components.

Definition 15. *Let for dynamic expressions G, G', $TS(G)=(S_G, L_G, \mathcal{T}_G, s_G)$, $TS(G') = (S_{G'}, L_{G'}, \mathcal{T}_{G'}, s_{G'})$. A mapping $\beta : S_G \rightarrow S_{G'}$ is an* isomorphism *between $TS(G)$ and $TS(G')$, denoted by $\beta : TS(G) \simeq TS(G')$, if*

1. *β is a bijection such that $\beta(s_G) = s_{G'}$;*
2. *$\forall s, \tilde{s} \in S_G\ \forall \Upsilon\ s \stackrel{\Upsilon}{\rightarrow}_{\mathcal{P}} \tilde{s} \ \Leftrightarrow\ \beta(s) \stackrel{\Upsilon}{\rightarrow}_{\mathcal{P}} \beta(\tilde{s})$.*

Two transition systems $TS(G)$ and $TS(G')$ are isomorphic*, denoted by $TS(G) \simeq TS(G')$, if $\exists \beta : TS(G) \simeq TS(G')$.*

Definition 16. *Two dynamic expressions G and G' are* equivalent with respect to transition systems*, denoted by $G =_{ts} G'$, if $TS(G) \simeq TS(G')$.*

4 Comparative study

In this section, we consider in detail differences and similarities between dtsdPBC and other well-known or similar SPAs for the purpose of subsequent determining the specific advantages of dtsdPBC.

4.1 Continuous time and interleaving semantics

Let us compare dtsdPBC with classical SPAs: Markovian TImed Processes and Performability (Performance and dependability) evaluation (MTIPP) [163], Performance Evaluation Process Algebra (PEPA) [164,166] and Extended Markovian Process Algebra (EMPA) [26].

In MTIPP, every activity is a pair consisting of the action name (including the symbol τ for the *internal*, invisible action) and the parameter of exponential distribution of the action delay (the *rate*). The operations are *prefix*, *choice*, *parallel* composition including *synchronization* on the specified action set and *recursion*. It is possible to specify processes by recursive equations. The interleaving semantics is defined on the basis of Markovian (i.e. extended with the specification of rates) labeled transition systems. Note that we have the interleaving behaviour here because the exponential PDF is a continuous one, and a simultaneous execution of any two activities has zero probability according to the properties of continuous distributions. CTMCs can be derived from the mentioned transition systems to analyze performance.

In PEPA, activities are the pairs consisting of action types (including the *unknown*, unimportant type τ) and activity rates. The rate is either the parameter of exponential distribution of the activity duration or it is *unspecified*, denoted by $\top$. An activity with unspecified rate is *passive* by its action type. The set of operations includes *prefix*, *choice*, *cooperation*, *hiding* and constants whose meaning is given by the defining equations including the *recursive* ones. The cooperation is accomplished on the set of action types (the cooperation set) on which the components must *synchronize* or cooperate. If the cooperation set is empty, the cooperation operator turns into the *parallel* combinator. The semantics is interleaving, it is defined via the extension of labeled transition systems with a possibility to specify activity rates. Based on the transition systems, the continuous time Markov processes (CTMPs) are generated which are used for performance evaluation with the help of the embedded continuous time Markov chains (ECTMCs).

In EMPA, each action is a pair consisting of its type and rate. Actions can be *external* or *internal* (denoted by τ) according to types. There are three kinds of actions according to rates: *timed* ones with exponentially distributed durations (essentially, the actions from MTIPP and PEPA), *immediate* ones with priorities and weights (the actions analogous to immediate transitions of GSPNs) and *passive* ones (similar to passive actions of PEPA). Timed actions specify activities that are relevant for performance analysis. Immediate actions model logical events and the activities that are irrelevant from the performance viewpoint or much faster than others. Passive actions model activities waiting for

the synchronization with timed or immediate ones, and express nondeterministic choice. The set of operators consist of *prefix*, functional *abstraction*, functional *relabeling*, *alternative* composition and *parallel* composition ones. Parallel composition includes *synchronization* on the set of action types like in TCSP [169]. The syntax also includes *recursive* definitions given by means of constants. The semantics is interleaving and based on the labeled transition systems enriched with the information about action rates. For the exponentially timed kernel of the algebra (the sublanguage including only exponentially timed and passive actions), it is possible to construct CTMCs from the transition systems of the process terms to analyze performance.

In dtsdPBC, every activity is a pair consisting of the multiaction (not just an action, as in the classical SPAs) as a first element. The second element is either the probability (not the rate, as in the classical SPAs) to execute the multiaction independently (the activity is called a stochastic multiaction in this case) or a combined specification of the (fixed) delay and weight expressing how important is the execution of this multiaction (the activity is called a deterministic multiaction in this case). Immediate (zero delay deterministic) multiactions in dtsdPBC are similar to immediate actions in EMPA, but all the immediate multiactions in dtsdPBC have the same (implicit) priority 2. The purpose is to execute them always before waiting (positive delay deterministic) multiactions with the same (implicit) priority 1, and stochastic multiactions with the same (implicit) priority 0. The immediate actions in EMPA can have different priority levels. Associating the same priority with all immediate (or waiting) multiactions in dtsdPBC results in the simplified specification and analysis, and such a decision is also appropriate to the calculus. The reason is that, as mentioned in [155], weights (assigned also to immediate actions in EMPA) are enough to denote preferences among immediate multiactions (designating their advantages or prescribing sub-priorities to them) and to produce the conformable probabilistic behaviours when one has to make a choice among several immediate multiactions executable in some state. There are no deterministic actions in MTIPP and PEPA. Immediate actions are only available in immediate PEPA (iPEPA) [158], where they are analogous to immediate multiactions in dtsdPBC, and in a variant of TIPP [142] discussed while constructing the calculus Probabilistic Markovian TIPP (PM-TIPP) in [256,257], but there immediate activities are used just to specify probabilistic branching and they cannot be synchronized.

dtsdPBC has the sequence operation, in contrast to the prefix one in the classical SPAs. One can combine arbitrary expressions with the sequence operator, i.e. it is more flexible than the prefix one, where the first argument should be a single activity. The choice operation in dtsdPBC is analogous to that in MTIPP and PEPA, as well as to the alternative composition in EMPA, in the sense that the choice is probabilistic, but a discrete probability function is used in dtsdPBC, unlike continuous ones in the classical calculi. Concurrency and synchronization in dtsdPBC are different operations (this feature is inherited from PBC), unlike the situation in the classical SPAs where parallel composition (combinator) has a synchronization capability. Relabeling in dtsdPBC is

analogous to that in EMPA, but it is additionally extended to conjugated actions. The restriction operation in dtsdPBC differs from hiding in PEPA and functional abstraction in EMPA, where the hidden actions are labeled with a symbol of "silent" action τ. In dtsdPBC, restriction by an action means that, for a given expression, any process behaviour containing the action or its conjugate is not allowed. The synchronization on an elementary action in dtsdPBC collects all the pairs consisting of this elementary action and its conjugate which are contained in the multiactions from the synchronized activities. The operation produces new activities such that the first element of every resulting activity is the union of the multiactions from which all the mentioned pairs of conjugated actions are removed. The second element is either the product of the probabilities of the synchronized stochastic multiactions or a specification of the joint delay and the sum of the weights of the synchronized deterministic multiactions with the same delay. This differs from the way synchronization is applied in the classical SPAs where it is accomplished over identical action names, and every resulting activity consists of the same action name and the rate calculated via some expression (including sums, minimums and products) on the rates of the initial activities, such as the apparent rate in PEPA. dtsdPBC has no recursion or recursive definitions, but it has the iteration operation to specify infinite looping behaviour with the explicitly defined start and termination.

dtsdPBC has a discrete time semantics, and residence time in the tangible states is geometrically distributed, unlike the classical SPAs with continuous time semantics and exponentially distributed activity delays. As a consequence, the semantics of dtsdPBC is the step one, in contrast to the interleaving semantics of the classical SPAs. The performance is investigated via the underlying SMCs and (reduced) DTMCs extracted from the labeled probabilistic transition systems associated with expressions of dtsdPBC. In the classical SPAs, CTMCs are usually used for performance evaluation. In [139], a denotational semantics of PEPA has been proposed via PEPA nets that are high-level CTSPNs with coloured tokens (coloured CTSPNs), from which the underlying CTMCs can be retrieved. In [25,21], a denotational semantics of EMPA based on GSPNs has been defined, from which one can also extract the underlying SMCs and CTMCs (when both immediate and timed transitions are present) or DTMCs (but when there are only immediate transitions). dtsdPBC has a denotational semantics in terms of LDTSIPNs from which the underlying SMCs and (reduced) DTMCs can be derived.

Consider *other SPAs with continuous time and interleaving semantics*. Such SPAs without immediate (and without positive deterministic) actions belong to the (general) classification group of MTIPP and PEPA. Such SPAs with immediate (and without positive deterministic) actions are (generally) classified as belonging to the group of EMPA.

Continuous time interleaving SPAs without immediate actions.

Stochastic Process Algebra (PA_S) and Generalized Stochastic Process Algebra (PA_{GS}) [179] extend LOTOS [39,38] with exponential and generally distributed continuous delays, respectively. PA_S has operations of inaction, prefix

with (both) the rate (of an exponential delay) and action, choice, parallel composition (with synchronization by a set of actions), relabeling (with a function) and hiding (of a set of actions). In PA_{GS}, rate and action prefix is replaced with PDF (of the general delay) and action prefix. The remaining operations of PA_S are supplemented in PA_{GS} by successful termination, enabling and disrupt. PA_S and PA_{GS} have the operational semantics on labeled transition systems and denotational semantics on stochastic event structures. Immediate actions can be easily added to the two SPAs. Continuous phase type [225,260,171,280,187,173] delays can be defined in PA_{GS}.

PEPA with phase type distributions ($PEPA^{\infty}_{ph}$) [118] extends PEPA to specify and analyze particular queues types with potentially infinite number of clients. The activities (with visible actions or internal one) of the $PEPA^{\infty}_{ph}$ components have phase type distributed durations. The $PEPA^{\infty}_{ph}$ operators are: (action and duration or passive symbol) prefix, action choice, probabilistic choice, synchronization (by the actions set), hiding (of the actions set) and constant (for recursive definition). The operational semantics of the $PEPA^{\infty}_{ph}$ components is defined on labeled transition systems (multi-graphs). The stationary probabilities for the processes of a $PEPA^{\infty}_{ph}$ fragment are calculated with the matrix-geometric method, to overcome the state explosion problem.

Generalized (General) Process Algebra (GPA) [67] implements generalized cost operations from the semi-ring structures. GPA demonstrates a novel approach to process algebras (PAs) with measurable transitions that permits to construct different classes of PAs, such as untimed, probabilistic and stochastic ones. The mathematical structure that generally represents the transition costs operations in such PAs is a semi-ring. The GPA actions can be visible or invisible. The GPA operations are: terminal agent, (action and cost) prefix, choice, parallel composition (with synchronization by a set of actions), hiding (of a set of actions) and (recursive) definition. Operational semantics of GPA is based on multi-labeled transition systems, where each transition is labeled by (possibly invisible) action and cost (of the transition execution).

Markov Chains (MC) and Markov Action-labeled Chains (MAC) [160] are the SPAs constructed on the basis of CTMCs. In MC, the processes describe (unlabeled) CTMCs as compositions of the transition rates by the operations of (finite) sum of the prefixed (with the rates) processes, recursion (over variables) and (simple) parallel composition. In MAC (also called pure Markovian process algebra), the processes describe labeled (with visible and invisible actions) CTMCs as compositions of the pairs of actions and the transition rates by the operations of (finite) sum of the prefixed (with such pairs) processes, parallel composition (with synchronization by a set of actions), renaming and recursion (over variables).

Stochastic Probes (SP) [9] specify over SPAs the performance requirements to software systems (beginning and end of a measurement by the model developer). The types of measures are: steady-state, transient and passage-time. The SP specifications are based on the regular expressions syntax describing the behaviour that a software model must demonstrate before starting or stopping the

performance measurement. Stochastic probes are themselves transformed into SPA components before a software model is explored with the process composition. SP has operators of sequence, choice, zero-or-one, iteration, range, positive closure and (standard) closure.

BioNetGen [35,119,120] is a rule-based language allowing one to construct a computational model of dynamics for the biochemical process of cellular signal transduction. The language can respect completely and exactly the specified enzymatic activities, potential modifications and interactions in signalling molecules. Binding and enzymatic biomolecular reactions are described by the rate-assigned reaction rules for transforming reactants into products. BioNetGen provides a graphical representation for the signal transduction networks in biology.

Grouped PEPA (GPEPA) [167,157,146,122] stems from the PEPA performance analysis technique for the large-scaled systems with many replicated components. The technique is based on the ODE systems, instead of the traditional CTMCs. GPEPA is a PEPA conservative extension, to which the fluid-flow analysis method is applied for approximating the mean number of the component types. The method takes as continuous the discrete state space of a process and transforms the discrete model into a coupled ODE system. The GPEPA operations over component groups (purely concurrent groups of standard PEPA components) are cooperation (over a set of synchronized actions), hiding and labeling. Operational semantics of GPEPA is used to construct population CTMCs, being the aggregated CTMCs whose states represent the sets of population members.

Stochastic Kernel language for agents interaction and mobility (StoKlaim) [231,230,229] extends programming and modeling language Klaim [228,32] by adding exponential action delays to describe random phenomena. The StoKlaim operations are: null process, prefixing (by action and its rate), choice, parallel composition and process instantiation. StoKlaim has operational semantics based on the rate (that extend the labeled) transition systems and transition-labeled CTMCs. The underlying stochastic process of StoKlaim is CTMC, for which the transient or stationary probabilities are calculated.

Stochastic Pi-Machine (SPiM) [240,243,241,238,292,233] is a graphical calculus for Sπ [246,247,248,240,188,27], aiming to specify biological processes. SPiM is reduction equivalent to Sπ, hence, they have the same expressive power. Such a graphical representation permits to detect cycles and to animate interactions of the system components for the dynamics visualization, as well as to serve as simulator of Sπ for visual modeling and simulation of biological systems by non-specialists. SPiM is a syntactic subset of Sπ, where choice of actions is allowed only on the highest level of definitions. Stochastic behaviour is embedded into the system by assigning channels with interaction rates and delays with extinction rates, both being the parameters of exponential distribution.

κ-calculus (κ) [100,101,102,188,184] is a formal proteins language, whose biological interaction rules on the set of agents have rates. The rules prevent combinatory explosion when describing the dynamics with ODEs, have intuitive graphical representation based on biological knowledge, and become a natural part of building, changing and discussing the model. κ specifies well biological signal and

control processes, being massively distributed systems. It formalizes directly and transparently molecular agents and their interactions in signalling networks.

Stochastic BioAmbients (SBioA) [61,239,188,27] provides calculus of Biological Ambients BioAmbients (BioA) [255,214,148,188,27] with a stochastic operational semantics to respect quantitative information. BioAmbients was intended to specify, simulate and analyze biological entities. The SBioA semantics is based on the stochastic simulation algorithm that calculates the real rates (parameters of exponential distribution governing the delays) of transitions. The semantics represents an influence of chemical and physical parameters (such as molecules concentration) to dynamics of living matter and constructs stochastic transition systems, from which CTMCs are extracted. The stationary probability distributions of those CTMCs are calculated, aiming to explore behaviour of biological systems in their steady state with the reward techniques for computing performance measures.

Markovian Process Calculus (MPC) [22] describes simple Markov (with stochastic delays governed by exponential distribution) processes, constructed with the operators of null term, Markovian action prefix (with an exponentially timed action, a pair of the action name and the rate of its exponential delay), alternative composition and process constant (specified by the equation with a recursion possibility, i.e. by potentially recursive specification). The operational semantics of MPC is defined on labeled (multi)transition systems.

$PEPA + \Pi$ [133] extends PEPA, aiming to model biological systems, by applying mass action law and bounded capacity law cooperations. In $PEPA{+}\Pi$, cooperation operator of $PEPA$ is supplemented by the cooperation set with mass action kinetics (in addition to the standard one with bounded capacity kinetics). A special notation is also proposed for parallel composition of large numbers of independent (non-cooperating) identical processes. The relationship is established between two semantics of $PEPA + \Pi$: that in terms of CTMCs (with large state space) and that on coupled ODE systems (to handle massive quantities of processes).

Stochastic Bigraphs (SBG) [185] calculus offers a stochastic semantics for Bigraphical Reactive Systems (BRSs), a unifying framework for designing models of concurrent and mobile systems. Such reactive systems are described by rewriting rules with an initial bigraph, to which the rules are applied. Bigraphs are the algebraic terms, represented by special graphs that represent communication of agents and their spatial configuration, so that some nodes can contain others. SBG provides BRSs with a uniform stochastic interpretation, where abstract rules of biomolecular reactions have positive rates assigned, used to calculate reaction rates. Stochastic transition systems, obtained from the SBG semantics, are taken to derive CTMCs that are analyzed with simulation.

Stochastic Păun- (P-) Systems (SPS) [234] is a class of computational models for cell biology and membrane computing. The main ingredients of a P-system are membrane structure (that delimits compartments), multisets of objects and biochemical reaction rules. The rules are endowed with the rates reflecting propensity of the corresponding reactions and can handle both objects

and membranes. The known types of P-systems are cellular, tissular and neural ones. The analysis of P-systems consists in applying symbolic probabilistic model checking.

Chemical Ground Form (CGF) [72,73,296,141] is a calculus for modeling biochemical reactions, a modification for biological systems of Sπ without communication. The actions in CGF have associated stochastic rates (positive real numbers, the exponential distribution parameters). Invisible action expresses unary reaction while complementary identically named visible actions specify two reactants in a binary reaction with the same name. The operators of CGF are: (successful) termination, prefix and parallel composition. The probabilistic semantics of CGF is based on DTMCs, its stochastic discrete-state semantics is constructed on CTMCs and its continuous-state semantics is defined on ODEs. The abstract probabilistic semantics of CGF is built by extracting labeled interval DTMCs from abstract labeled transition systems, based on abstract multisets, with intervals of integers used instead of single multiplicities. CGF corresponds to basic chemistry.

Chemical Parametric Form (CPF) [73] extends CGF with parametrization, communication and reuse, being more general subset of Sπ. The CPF stochastic processes can be converted to chemical reactions (interrelated with CGF). The mapping of CPF to chemistry results in the parametric and compositional indirect (two-step) mapping of CPF to ODEs that is easier to define and understand than a direct (one-step) mapping. That indirect mapping can be interpreted as the ODE semantics of the CPF processes.

Biochemical Ground Form (BGF) [77,296] extends CGF with the capabilities of complexation (joining) and splitting molecules, through association and dissociation. Calculus BGF adds to the syntax of CGF two pairs of complementary actions, intended to specify association and dissociation. The operators of BGF additionally include trailing of the association histories. The discrete-state semantics of BGF (like that of CGF) is based on CTMCs, extracted from labeled transition graphs. Differently from CGF, calculus BGF is Turing powerful and corresponds to biochemistry.

Stochastic Calculus of Communicating Systems (StoCCS) [182,232] is a CCS stochastic extension being a fragment of Sπ. The StoCCS operators are: null process, prefixing (by action label and its rate), stochastic choice and parallel composition. Labeled state-to Function Transition Systems (FuTSs) are used to unify definitions of the (S)PAs semantics with a goal to compare the calculi. Based on FuTSs, two stochastic enhancements of CCS with binary synchronization are proposed: $StoCCS_{AA}$ with active input and output actions of the channel (along which the synchronization signal is transmitted from the input to output action) and $StoCCS_{AP}$ with passive input and active output actions of the channel.

Stochastic Calculus of Looping Sequences (SCLS) [13,16] extends stochastically Calculus of Looping Sequences (CLS) [17]. SCLS is a quantitative term rewriting formalism for describing evolution of the microbiological systems (such as cellular pathways) while taking into account the activities speed, represented

by stochastic rates (the exponential distribution parameters). SCLS has operators of sequencing, looping, containment and parallel composition. The looping operator connects the ends of sequence, resulting in the circular (looping) sequence that can specify membrane. CTMCs are extracted from the semantics of the SCLS systems, with a goal of simulating and verifying their properties with stochastic model checking.

Language for Biochemical Systems (LBS) [235,236] combines modeling (with rewrite rules) and modularity. LBS is based on Calculus of Biochemical Systems (CBS), intended for modular specification of metabolic, signalling and regulatory networks, as reactions between modified complexes that occur concurrently in the hierarchy of compartments, with possible interactions and transport across compartments. LBS has the species expressions, parametrized modules with subtypes, nondeterminism, as well as nested declarations of species and compartments. Formal specification of the language is given by abstract syntax and general semantics, being parametric on the structure of the target semantic objects: PNs, coloured PNs (CPNs), ODEs and CTMCs.

Biochemical Performance Evaluation Process Algebra (Bio-PEPA) [87,88,86,89,129,148,125,130,126,147,212,213,91,211,259,127,168,27] is constructed to model and analyze biological networks. For that, PEPA is extended with stoichiometry (quantitative interrelations of reactants in biochemical reactions), the species roles in reactions and functional rates for different kinetic rules types of the reaction dynamics. The processes in Bio-PEPA are seen as species rather than molecules, like in Sπ. The Bio-PEPA operators are: prefix combinator (with the pair of action type and its stoichiometry coefficient, in the role of reactant, product, activator, inhibitor or generic modifier), choice, constant, cooperation (by the activities set) and concentration level. The operational semantics of Bio-PEPA is defined on stochastic labeled transition systems, based on the discrete concentration levels. Bio-PEPA maintains several analysis methods: Stochastic Simulation Algorithm (SSA) [137,138], numerical solution for the steady-state analysis of the CTMC (with discrete concentration levels) underlying the model semantics, translation into the equivalent deterministic model of ODEs, as well as stochastic model checking.

Context-dependent Bioambient Calculus (CoBiC) [52] is a stochastic extension with functional rates of Biological Ambients calculus, BioAmbients (BioA). The rates in CoBiC are calculated by respecting as the volume of ambients (such as cells), as the whole context (surrounding environment) with the concentration and pressure (context-dependent rates). To model transport of molecules in and out of membranes, CoBiC has both the notion of membrane or compartment (to separate inside from outside) and the internal or external compartment concentration functions. The channel and ambient names in CoBiC are connected with the operations of inactive process, local sum (standard choice of the processes, prefixed with the ambient capabilities, including exponential delays), restriction (of a name), recursion (to model infinite behaviour), ambient (named compartment with a process, and basic or minimal volume associated) and parallel composition. The operational semantics of CoBiC is defined by reduction rules. The

rates of basic actions are the functions depending on the context of the executing processes (the global configuration of the system). Those functions are evaluated to positive real numbers in each state, so that from the labeled transition system a (time-homogenous) CTMC can be derived, used to simulate the model.

Spatial Calculus of Looping Sequences (that we call SpCLS) [14,15] is a spatial extension of Calculus of Looping Sequences (CLS) that observes the position and taken space of biological elements with time passage in a continuous two- or three-dimensional space. The movement of elements in the space can be exactly described, and they can interact when constraints on their positions are satisfied. Both deterministic and stochastic movements of the elements can be specified. Like in SCLS, rewrite rules for reactions in SpCLS are endowed with kinetic parameters defining their stochastic propensity rates. The reaction rates are the parameters of exponential distribution that models the expected duration of a reaction with a specific combination of reactants.

Typed Stochastic Calculus of Looping Sequences (TSCLS) [116,34] is an extension of SCLS with the types of the elements that speed up or slow down reactions, such as positive of negative catalyzers. The operational semantics of TSCLS that respects the types of the species is applied to derive the stochastic evolution of a system, where the activities speeds can be modified by catalyzers. The types offer an abstraction that can represent the interactions of elements without exact specification of their positions. The rewrite rules have the rates that permit the evolutions of the rules follow different probability distributions, what is useful for high-level simulation. The typed stochastic semantics generates the transition systems, producing the CTMCs that are applied in the simulation procedure.

Stochastic Calculus of Wrapped Compartments (SCWC) [93,94,258] naturally describes a wide class of biological systems via direct representation of membranes and compartments. SCWC is a variant of SCLS without sequencing operator and with multisets (instead of ordered sequences) of atomic elements, to specify membranes. SCWC is intended to simplify the development of automatic analysis tools while preserving the SCLS expressiveness. Every reaction rule in SCWC has an assigned rate function of the context. SCWC has a stochastic operational semantics, from which a CTMC is extracted to verify the system properties. To identify kinetic parameters of biological systems in SCWC, an effective stochastic simulator is applied (instead of standard ODE-based methods), thus extending the class of investigated systems.

Stochastic Calculus of Communicating Systems (that we call stCCS) [75,76] is a stochastic extension of CCS without replication [217]. Each action (label, transition) is associated with the exponential distribution rate, which is the same for the paired action. The synchronization of each action and its paired one results in an internal action with the rate defined by the mass action law. The operations of stCCS are: empty (null) process, prefix (guard), parallel composition and choice (summation). The structural operational semantics of stCCS is defined via measure theory and assigns to each process a set of measures over the processes space. The measures encode the rates of the transitions from a

process to a measurable set of processes. The stochastic behaviour is derived using continuous time Markov processes (CTMPs).

Stochastic strand algebra (that we call stSA) [74] is a formal language with a simple relational semantics and compositional descriptions, where each component maps directly to DNA structures. stSA is designed for DNA computing (such as DNA strand displacement) by specifying DNA strands and gates, as well as their interactions. The atomic elements of stSA are signals and (null and curried) gates (from signals to signals). The stochastic rates (positive reals) are assigned to gates. The stSA operators are: persistency of (null or curried) gates and parallel (concurrent) composition. The semantics of stSA is given by labeled transition graphs (LTGs), from which CTMCs are derived. The translation from stSA to CTSPNs maps signals to marked places and gates to transitions with the associated rates. Since CTSPNs can be represented as finite stochastic chemical systems (SCSs), with each transition corresponding to a chemical reaction, and SCSs can be translated to stSA, CTSPNs are equivalent to stSA.

Markovian Agent Spatial Stochastic Process Algebra (MASSPA) [145] formally describes behaviour of Markovian Agent Models (MAMs), a spatial stochastic modeling framework. A Markovian agent in a MAM is a simple sequential component that can have local transitions (with exponential rates), possibly sends messages and can have message-induced transitions. The MASSPA operations are: (exponentially rated) prefix, choice, (Poisson distributed number of) message sending, (probabilistic) message reception, constant, null process and parallel (with the message exchange). The underlying CTMC of a lumped process is approximated with special techniques. The CTMC describes the density evolution of an agent type at current moment for a given location. The ODE-based analysis of higher moments (such as variation) is proposed in the performance evaluation of discrete spatial stochastic models. Stochastic simulation is used to verify the ODE-based approximation of mean and standard deviations for counting the model components.

Process Algebra with Hooks (PAH) [106,107,108] is intended to model biological systems at multiple levels of detail (scales). The processes of PAH describe different scales, such as biochemistry, cells and tissue. The operators of PAH include the deadlock process, agent definition, sequential execution and nondeterministic choice. In addition, two symmetric operators of synchronization (on the set of actions) are used to compose processes within one level of detail (horizontal cooperation) and between the levels (vertical cooperation). Stochastic semantics of PAH is based on functional rates of reactions. Continuous time and exponential delays are applied.

Stochastic Brane Calculus (that we call SBC) [10,27] is a stochastic extension of Brane Calculus (BC) [71,148,27]. The membranes are collections of actions while the systems consist of nested membranes. The systems are built with the operations of empty system, parallel composition and nesting (within a membrane). The semantics of each SBC process is intended to be a measure of the stochastic distribution of its derivations (outcomes). The processes form a measurable space, and each process has an action-indexed family of measures on this

space. The stochastic semantics of the Brane systems defines them as continuous time Markov processes (CTMPs) over the measurable space generated by terms up-to syntactic congruence. The compositional and syntax-driven structural operational (SOS) representation of this stochastic semantics is provided.

Fluid Process Algebra (FPA) [281,282] is a subalgebra of GPEPA being a conservative extension of PEPA with fluid semantics, intended to simplify solving the systems of coupled ODEs. FPA has the expressive power of GPEPA without hiding operator. The FPA operations over the PEPA-like model components, specified using operations of (action and rate) prefix, choice, recursive definition with constant, and cooperation, are: cooperation (over a set of synchronized actions) and labeling. The labels are used to distinguish the representative components, which are replicated. A fluid atom is an occurrence of some labeled component in a process. FPA has a fluid semantics based on the underlying systems of ODEs that are used for the analysis. Each ODE system approximates the evolution in time of the processes population representing a local state.

Simple Stochastic Process Algebra (SSPA) [290] describes CTMCs with a product-form solution, implying that their stationary distributions are effectively solvable. The proofs of important properties for SSPA are simpler than for labeled Markov Automata (LMAs) that have a direct relation with CTMCs, but do not permit to use the inductive structure of the language. SSPA preserves semantics of the cooperation operator of LMAs, what is important for correctness of the product-form solution. The operators of SSPA are the empty process, identifier, choice (from a set of processes, prefixed with actions and rates), closure (replacing variable by a real-valued rate in each pair of an action and a variable) and interaction (among many processes, by a set of actions).

Calculus of Chemical Systems (that we call CChS) [245] is proposed for modular description of chemical reaction systems and modeling with rules in systems biology. CChS is based on CCS, but with communication replaced by chemical reactions. The operations of (quantitative version of) CChS include rule (with a positive real-valued rate), parallel composition, the empty process, local definition and process identifier. Different compositional semantics of (quantitative) CChS are given, based on quantitative PNs (CTSPNs), ODEs and stochastic transition matrices. Complete axiomatizations and normal forms are presented for all the semantics.

Fluid Extended Process Algebra (FEPA) [283,284,174] explores the models, specified with large systems of ordinary differential equations (ODEs). The sequential process components, called fluid atoms, can have a multiplicity (the number of copies in the model specification). There are two variants of synchronization: with the minimum of the rates of the synchronized processes (to model computer systems, as in PEPA or with their product (to represent chemical reactions and biological networks with the rule of large numbers, as in Bio-PEPA. The FEPA processes are described by the ODE systems with the derivatives of the population functions that define the multiplicities (numbers of replicas in a population) of fluid atoms by one variable (time). The typical FEPA multiplicity values are rather large and interpreted as non-negative real (instead of

natural) numbers, defined by the population functions of time, whose values can be found for every particular moment. The FEPA expressiveness is restricted to the processes being a parallel composition (with the embedded synchronization by the cooperation actions) of the fluid atoms denoting a large number of copies of simple sequential components, specified with the operations of (action and rate) prefix, choice and recursive definition with constants. The FEPA fluid atoms are considered uniformly, without dividing into "discrete" atoms with small multiplicities and "continuous" ones with large multiplicities.

Probabilistic Programming Process Algebra (ProPPA) [134,135,168] is an extension of Bio-PEPA. ProPPA permits uncertain description of models and application of the machine learning techniques, aiming to include observational information in the modeling. The semantics of ProPPA is defined on probabilistic constraint Markov chains (PCMCs), an extension of constraint Markov chains (CMCs). CMCs generalize DTMCs so that the state change probabilities become not fixed, but satisfy some constraints or belong to a set of acceptable values. Markov decision processes (MDPs) or uncertain Markov chains (UMCs) are used to simulate CMCs. Analogously, PCMCs generalize CTMCs, but they associate a probability distribution with the constraint satisfaction set of values. The stochastic relation defines the rate of the transition from one complete system to another. The rate is generalized in ProPPA to a distribution over possible rates.

Collective Adaptive Resource-sharing Markovian Agents (CARMA) [46,196,132] is used to specify and analyze collective adaptive systems. CARMA has linguistic constructs for modeling and programming systems that work in openended and unpredictable environments. A model is a collective of components, each expressing a set of attributes. To model dynamic aggregations (ensembles), CARMA has communication primitives based on predicates (over the expressed attributes), to select the communication participants. There are multicast- and unicast communications. The CARMA operations are: empty process, component destroy, action prefix, choice, parallel composition, predicate guarding and recursive definition. The operational semantics is defined on labeled state-to Function Transition Systems (FuTSs) [232], from which the action-labeled CTMCs are derived.

Cox and Convenience Calculus (CCC) [252,33] is constructed to generate of and manipulate continuous acyclic phase type (APH) distributions for compositional representation of process delays. The delays correspond to the completion times of activities. Basic delays are described by exponential distributions. Complex delays are obtained by composing basic delays with stochastic operations on continuous probability distributions: summation (convolution), minimum and maximum. CCC generates representations of APH distributions in Cox forms. The stochastic operations have the respective ones among CCC operators: rate (of exponential delay), disabling (race between two exponential distributions), sequential composition (corresponds to convolution), choice (corresponds to minimum) and parallel composition (corresponds to maximum). The operational semantics maps the CCC expressions onto Markov (decorated) transition systems and then interprets them as absorbing CTMCs.

Modelling in Ecology with Location Attributes (MELA) [291] is designed to model ecological systems while respecting location in space and influence of environment. MELA is a high-level language for formal description of the ecological concurrent systems of agents that can evolve simultaneously and interact. It specifies population models with single or multiple species. Its actions have rates being the exponential distribution parameters. The MELA operators are: no-influence action, influence action, probabilistic effect of action, choice, constant, null component and parallel. Operational semantics of MELA is defined on the labeled transition systems with qualitative and quantitative information about actions. MELA supports stochastic simulation and direct analysis of the underlying CTMCs, as well as numerical solution of the fluid approximation with the ODE systems.

Network of Broadcasting Agents (NBA) [45] is a fragment of CARMA without attributes. NBA supports both unicast and broadcast communication to model quantitative aspects of the systems of broadcasting processes. Within the agents, actions have rates (the exponential distribution parameters), broadcast messages have probabilities while unicast messages have weights, used to calculate probabilities. Stochastic operational semantics of NBA is defined on labeled state-to Function Transition Systems (FuTSs) [232]. Fluid approximation theorem is proved for the NBA population semantics, based on population CTMCs.

Continuous time interleaving SPAs with immediate actions.

Markovian Process Algebra of M. Bernardo, L. Donatiello and R. Gorrieri (MPA) [24] is used to model functionality and performance. The MPA action is a pair of its type and rate. According to type, the actions can be external (observable) and internal (invisible). According to rate, the actions can be passive (with zero rate) and active (with a positive rate). The active actions can be timed (with a finite positive rate) and immediate (with an infinite positive rate). The immediate actions have priorities ans weights. The operators of MPA include null term, prefix (with an action), functional abstraction (hiding, by the actions types set), temporal restriction (by the passive actions types set), relabeling (with a function), alternative composition (choice), parallel composition (with the synchronization set), constant (for recursive definition). The operational semantics of MPA is based on labeled transitions systems, from which homogenous CTMCs are extracted, being a performance evaluation formalism (stochastic model, Markovian semantics). The net semantics (distributed model) of MPA is defined on GSPNs that clearly represent parallelism and causality.

Markovian Process Algebra of P. Buchholz (the extended version that we call MPA-B) [63,64] is based on CCS and CSP. Every action in MPA-B has a basic transition rate. Activities are the pairs from an action and the value parameter, expressing the action speed or its invocations number. There is also a distinguished invisible action. For each activity, action is executed after exponential delay with the rate equal to the product of the activity value parameter and basic transition rate of the action. This enables compositional analysis with parallel composition of process expressions. MPA-B has the operators of termination, prefix, choice, parallel composition (with synchronization on the actions

set), hiding and recursion. The operational semantics of MPA-B is defined on finite multi-labeled transition systems (MLTSs), whose transitions are labeled by the pairs from an action and the value parameter. The MLTSs, constructed with the rules of structural operational semantics, are used to extract the underlying CTMCs. The extensions of MPA-B with immediate activities and non-exponential activities durations (such as phase type distributed) are proposed.

Probabilistic Markovian TImed Processes and Performability evaluation (PM-TIPP) [256,257] is an extension of the Markovian variant of TIPP by *probabilistic branching*. It can be specified by weighted immediate activities, also describing management actions, to test for resources availability. Instead, PM-TIPP is MTIPP with an additional *probabilistic choice* operator. PM-TIPP has hiding and relabeling operators as well. In PM-TIPP, Markovian transitions (with exponential delay) are merged at the semantic level with their direct followers, corresponding to the probabilistic choice. The operational semantics of PM-TIPP is based on labeled transition systems with two transition relations denoting Markovian transitions (labeled with actions, rates and additional words) and probabilistic transitions (labeled with probabilities and additional words).

Spectral Expansion TImed Processes and Performability evaluation (SE-TIPP) [220] is a modification of TIPP enabling solution with the spectral expansion method (SE). SE-TIPP can model processes with infinite state space, resulting in a significant modeling power increase. The systems with infinite number of states are modeled intuitively while the specification scheme for such systems gives a compact representation of infinite Markov processes, to be solved with SE. In SE-TIPP, actions have the associated rates (parameters of exponential distribution) The operations of SE-TIPP are: (successful) termination, variable, prefix (with an activity), choice, parallelism (with the synchronized actions set), hiding and recursion. The underlying Markov processes may be two-dimensional: finite in one direction and infinite in the other. The model description is transformed into a matrices set, used in the SE solution procedure, followed by the performance measures calculation.

Stochastic Process Algebra for Discrete Event Simulation of P.G. Harrison and B. Strulo (that we call SPADES-HS) [153] extends Timed CCS [294,295] to formally describe discrete event simulation. SPADES-HS specifies time progress and probabilistic choice (discrete or continuous): selecting from a countable processes number or taking a random waiting time. The SPA describes infinite (as a rule) semantic objects, has immediate and delayed prefixing, can specify separately random timer starts, timer completions and current activities. Time delays may be non-exponential, resulting in more generality. SPADES-HS has visible actions, their conjugates and the (self-conjugate) invisible action. The operators of SPADES-HS are: deadlocked (terminated) process, time prefix with fixed delay, (impatient) prefix with action, patient prefix with action, nondeterministic choice, probabilistic choice, time prefix with random delay (with the density), parallel, relabeling (with the renaming function), restriction (on the actions set) and recursion. The operational semantics is defined on labeled transitions systems with the labeled (with actions), probabilistic and (time) evolution transitions.

Stochastic Timed Calculus (STC) [162] extends CCS with stochastic time under the maximal progress assumption. The (visible and a special invisible) immediate actions of STC are separated from delays, governed by an exponential distribution with the parameters called rates. The STC operations are: delay (with a rate) prefix, action prefix, choice and recursion (by variables). The operational semantics of STC is based on labeled transition systems with the action (action labels) and timed (rate labels) transition relations.

Stochastic Process Algebra for Discrete Event Simulation of P.R. D'Argenio, J.-P. Katoen and E. Brinksma (SPADES) [6,7,3,160,5] is a non-Markovian stochastic calculus. The actions in SPADES are separated from continuous time generally distributed stochastic delays. The semantics of SPADES is based on stochastic automata (SAs) [4] that can be executed using discrete event simulation. The semantics of SAs themselves is defined via probabilistic (labeled) transition systems with general distributions (discrete, continuous and singular). The operations of SPADES include stop (inaction) process, (action) prefixing, triggering condition, choice, clock setting, parallel composition (with the synchronization set of actions), left merge (with set of actions), communication merge (with set of actions), renaming (with the function) and process instantiation (for recursive definition).

Stochastic Process Algebra ($\mathcal{SPA}$) [37] is an extension of TCSP with anonymous (unnamed) timed actions, to model continuous time stochastic delays between visible actions. All named (visible and invisible) actions are immediate (of zero duration) and have no delays assigned. Timed actions are specified by positive real numbers being the exponential distribution parameters of delays (rates) of those (Markovian) actions. Functional and temporal behaviour is treated separately. The $\mathcal{SPA}$ operators are: empty (stop) process, prefixing with immediate named actions, prefixing with timed anonymous actions, choice, parallel composition (with the synchronization set of actions), restriction (on the actions set) and recursion. Synchronization may occur only between immediate named actions and termed timeless [165]. The structural operational semantics of $\mathcal{SPA}$ is based on the rules for timed anonymous (Markovian) actions and for immediate named actions. It is defined on labeled transition systems with two transition relations: implementation of unnamed time delay or instantaneous execution of a named action. The compositional structure of the $\mathcal{SPA}$ specifications is used for a novel solution of the underlying stochastic process by reformulation of the underlying CTMC as semi-Markov processes.

Non-Markovian Stochastic Process Algebra (NMSPA) [193] permits general (not only exponential) probability distributions of delays, to increase the expressive power. Some practically important distributions types are used, such as uniform, discrete and Poisson. This fact allows one to specify passive, urgent and immediate actions. Besides visible actions, NMSPA has the (urgent) invisible action. The operators of NMSPA include deadlocked process (STOP), choice of the fastest action (with the random variable of delay) from those prefixing processes, parallel composition (with the synchronization set), restriction (on

the actions set), renaming (with the function) and recursion. The operational semantics of NMSPA is based on labeled transition systems.

Stochastic Basic Language Of Temporal Ordering Specification (that we call SB-LOTOS) [161] is an extension of Basic (data-absent) LOTOS with the continuous phase type distributed delays. The actions can be visible ones, the invisible (internal, unobservable) one and the successful termination one (denoted by δ). The SB-LOTOS operators are: inaction (stop), successful termination (exit), action prefix, rate (of the exponential delay) prefix, choice, sequential composition (enabling), disabling, parallel composition (by a set of actions), hiding (of a set of actions), relabeling (with a function), process instantiation (for recursive definitions) and elapse (describing continuous phase type delay via its absorbing CTMC using the actions start, delay and break). The SB-LOTOS operational semantics is defined on the generalization of Interactive Markov Chains (IMC) [159,160], with the transitions on the time being continuously phase type distributed.

Biochemical Stochastic π-calculus (BioSpi) [251,148] extends the name-passing SPA Sπ with the goal of investigating biomolecular systems and interactions. BioSpi describes the structure and dynamics of biochemical networks. The actions have the rates assigned (parameters of the exponential distribution of delays), corresponding to the basal reaction rates. The channel requests (send or/and receive, or withdraw) may have the infinite rate, i.e. be instantaneous. BioSpi inherits all operations of Sπ, but the prefixing actions are replaced by the pairs of actions and rates. Reduction semantics of BioSpi respects the time and probability of biochemical reactions. The quantitative analysis is based on the stochastic discrete simulation with support of mobility.

Immediate Markov Action-labeled Chains (IMAC) [160] enriches MAC with immediate actions that are executed without any delay. The operations of MAC are supplemented with immediate (action) prefix, such that the prefixing immediate action is executed instantly before evolving into the prefixed process. The operational semantics of IMAC is based on labeled transition systems with two transition relations, describing executions of durational actions (with rates) and those of immediate actions, respectively. The choice between durational actions is probabilistic while that between immediate actions is nondeterministic.

Interactive Markov Chains (IMC) and their Interactive Markov Language (IML) [159,160,23] is a compositional continuous time behavioural model. Immediate actions in IMC are added to MC, i.e. time transitions (with rates) and action (interactive, immediate) transitions are separated. The processes describe interactive Markov chains (IMCs) with visible nd invisible actions as compositions of the actions and transition rates by the operations of (finite) sum of the prefixed (with actions or rates) processes, parallel composition (with the synchronization set of actions), renaming and recursion (over variables). The operational semantics of IMC is based on the union of labeled transition systems and CTMCs. The semantic transitions are Markovian, with the rates-defined exponentially distributed delays, or interactive, corresponding to instant execution of (possibly invisible) actions.

Interactive Generalized Semi-Markov Processes (IGSMP) [59] is a calculus with interleaving semantics for (visible and invisible) actions and ST- (Start-Termination-) semantics for (non-Markovian) delays. The actions are all immediate and separated from delays, specified as a pair of distribution function (of the duration probability) and weight. IGSMP specifies the probabilistic timed delays with general continuous distributions and synchronized actions with zero duration, as well as probabilistic (under preselection policy), nondeterministic and prioritized choice. The operators are: empty process, delay prefix, action prefix, choice, hiding (the set of actions turned into invisible ones), relabeling (with the function), parallel (with the synchronization set of actions), recursion (over variables). The operational semantics of IGSMP constructs generalized semi-Markov processes (GSMPs) being the probabilistic systems with generally distributed time, which are extended with the action transitions describing interactions among the system components. The concurrent execution of delays is expressed by a variant of ST-semantics, based on dynamic names. For performance evaluation, GSMPs are extracted from IGSMPs and analyzed with mathematical and simulative methods to obtain the performance measures.

Value Passing Stochastic Process Algebra (VPSPA) [194,195] extends NMSPA with the value passing feature. VPSPA permits generally distributed delays. The properties of the VPSPA specifications are studied by translating them into the programs of concurrent functional programming language Eden, where parallel processes are executed and their quantitative properties are investigated. To analyze the specified generally delayed systems, simulation is used instead of model checking. The performance of the system implementation is simulated, in order to obtain the real estimates of its theoretical performance. The communication actions are input (message receiving) and output (message transmitting) ones. There is also the invisible action (urgent, immediate). There are data transmission channels with the values. Stochastic actions describe delays specified by random variables with generally distributed (continuous time) delays. Choice has no probability assigned while parallelism has the associated set of indexed starts and terminations of delays (like in the ST-semantics of IGSMP [59]). Further operators are termination, delay prefix, channel receiving, channel transmission, conditional, hiding and recursive definition. The operational semantics of VPSPA is based on labeled transition systems.

Semi-Markov PEPA (SM-PEPA) [53,54,55,8] extends PEPA with the action delays distributions that ensure the underlying stochastic model to be an SMC. In SM-PEPA, actions are associated with symbolic priorities and parameters of general delays. The parameters are the rates of exponential delays or the pairs of a weight and a generally distributed delay, defined by Laplace transform. The SM-PEPA operators are: prefix, cooperation, hiding and constant. At each priority level, only one type of actions is allowed: Markovian or semi-Markovian. The semi-Markov synchronization is specified by the user-defined functions of the combined weight and delay. Operational semantics of SM-PEPA is constructed with the rules for Markovian and semi-Markovian actions. For the model

analysis, the SM-PEPA specifications are automatically transformed into semi-Markov SPNs.

Stochastic Beta-binders (that we call SBB) [104,148,27] is a stochastic version of Beta-binders [250,214,148,27] with typed interaction sites for accurate description of biological entities. In SBB, quantitative measures on biological phenomena are studied. The quantitative parameters are extracted from typed interaction sites, resulting in the affinity concept. The SPA has exponential or zero action delays. The quantitative information is given by the action rates (exponential distribution parameters) that represent stochastic behaviour and define reaction speeds. The operators are: inactive process, (input, output, hiding, unhiding and exposing) prefixing (with the pair of action and rate), parallelism, static binding and multiple instances of prefixed process. The operational (stochastic reductional) semantics is based on labeled transition systems. The underlying stochastic process is CTMC.

MOdeling and DEscription language for Stochastic Timed systems (MODEST) [36,149,69,154] is a formalism for modular description of reactive systems behaviour respecting functional and nonfunctional aspects (timing or service quality) of systems in a single specification. The actions are separated from (random) delays, there are simple and structured data types, structuring mechanisms (like parallel composition and abstraction), means to control the assignments granularity, exception handling, nondeterministic and random branching, timing. There are patient and impatient actions, exception names, the unhandled error action, the break action and the unobservable (silent) action. The operations include stop (no activity), abort (unhandled error), break (action with no restriction), act (action with no restriction), condition (when), urgency (of the first activity), process instantiation, call by value, choice, sequential composition, loop, relabeling, alphabet extension, exception handling, probabilistic prefix and parallel composition (with the multiway synchronization set). The operational semantics is defined on stochastic timed automata, a union of timed and stochastic automata [4], interpreted over (infinite) timed probabilistic transition systems (with immediate action transitions to discrete probability distributions over successor states, and timed transitions with positive real-valued delays). MODEST describes a wide spectrum of models: labeled transition systems, timed and hybrid automata (and probabilistic variants of them), stochastic processes like (discrete and continuous time, generalized semi-) Markov chains and (discrete and continuous time) Markov decision processes, Markov and stochastic (timed and hybrid) automata.

Stochastic Concurrent Constraint Programming (sCCP) [41,47,42,43,48,49,50,51] is a stochastic extension of CCP [244]. sCCP is proposed for modeling and analysis of biological systems. In sCCP, communication is asynchronous, species are described by variables, reactions are represented by constraints on the variables and rates are specified by functions. The sCCP operations are: tell prefix (with rate), ask prefix (with rate), choice, empty process, (recursive) procedure call (with rate), hiding and parallel composition. Two operational semantics of sCCP are defined: the continuous time one (standard) and

discrete time one (with the rates interpreted as weights to calculate probabilities; we call the latter SPA dsCCP), resulting in CTMCs and DTMCs, respectively, as the performance analysis model. The traditional semantics (the standard one on CTMCs and differential one on ODEs with fluid flow approximation) are supplemented with hybrid semantics on (Non-)Deterministic Hybrid Automata (DHA, NDA) and Simple Stochastic Hybrid Automata (SSHA). The analysis consists in stochastic simulation or in translations into the stochastic verification programming system, ODEs and hybrid systems.

Nano-κ-calculus (nanoκ) [96,97,191], based on κ [100,101,102,188,184], is intended for modeling, analysis and prediction of the molecular devices properties. Biochemical systems are modeled in nanoκ by defining their reaction sets. The semantics of nanoκ is based on reaction rules, where reactions (creations, destructions or exchanges) have finite or infinite rates. The stochastic model of nanoκ is based on the stochastic transition system with finite (for Markovian transitions) and infinite (for invisible, silent, interactive transitions) rates, thus resulting in the Markovian and transient states, respectively. From that transition system (with only silent actions), interactive Markov chain (IMC) [159,160] is extracted that can be downgraded to CTMC, if all invisible interactive transitions are partitioned into the confluent directed acyclic graphs of finite depth. The nanoκ implementation into SPiM takes molecules as processes and derives the overall (stochastic) behavior by communication rules.

Stochastic Pi-Calculus for Concurrent Objects (SPiCO) [186,214] is a modeling and simulation language for systems biology, based on Sπ. SPiCO supports high-level modeling by using the multi-profile concurrent objects with static inheritance that correctly represent interacting molecules. The SPiCO operators are: empty process, parallel composition, channel creation, sum, application, pattern input (receive), tuple output (send) and (recursive) definition. The SPiCO stochastic semantics is defined on CTMCs. The transitions can be timed (exponential delays with finite rates) or immediate (zero delay with infinite rate and probabilities). To construct CTMC, immediate transitions (corresponding to instantaneous reactions) are eliminated and their probability effects are respected. SPiCO is encoded back into BioSpi while preserving the semantics.

Stochastic π-calculus with polyadic synchronization (Sπ@) [286,287,288] is an enrichment of the language Sπ. The calculus Sπ@ is a stochastic extension of the process algebra π@ [289]. In Sπ@, finite and infinite rates are allowed, hence, there exist immediate actions (reaction types), taken as possessing higher priority (from two possible, defined by types of the rates) than standard ones. The operators of Sπ@ are: null process, guarded (action prefixed) choice, parallel, guarded (action prefixed) replication and scope restriction (of a name). The language Sπ@ flexibly models multiple compartments with dynamic structure and provides enhanced biological faithfulness. The biological systems specified in Sπ@ are used in the extension of Stochastic Simulation Algorithm (SSA) [137,138] that handles multiple compartments with varying volumes.

Attributed π-calculus ($\pi(\mathcal{L})$) [176] extends π-calculus [218,219] with attributed processes and attribute dependent synchronization, for application in sys-

tems biology. $\pi(\mathcal{L})$ is parametrized with the language $\mathcal{L}$ defining the attribute values and expresses polyadic synchronization with different compartment organizations. The $\pi(\mathcal{L})$ operators are: defined process, parallel composition, channel creation, summation of (receiver of sender prefixed) alternative choices, empty solution and parametric process definition. The nondeterministic (small step reduction) and stochastic (CTMCs extraction) semantics are proposed, with the rates possibly dependent on the attribute values. Finite and infinite (immediate transition) rates are allowed in the stochastic semantics, for which a simulation algorithm is developed.

EXtended Stochastic Probes (XSP) [8,90] is an enrichment of SP [9] for the state-aware performance analysis with the queries combining instantaneous observations of the model states and finite sequences of the model activities observations. The queries are realized in XSP by composing the observers with the described by an SPA model that has a discrete time representation via CTMC. The communicating local probes have immediate actions for instantaneous communication among components of the probes and transferral the states information without affecting the behaviour and without perturbing the performance analysis. The XSP novelty is a combination of the state and activity specifications with local and global observations. The XSP activity probes operators are: (activity) observation, sequence, choice, labeling, (upper bounded) iteration, range (lower and upper bounded) iteration, one-or-more, zero-or-more, zero-or-one, resetting and bracketing.

Phase Type Processes (PTP) [293] allow for probabilistic and nondeterministic choices, as well as continuous phase type (generalizing exponential) and zero delays. The (visible or invisible) action transitions, used to react on the external stimuli, are separated from the phase type transitions. The PTP semantics is constructed via the path probabilities with respect to schedulers resolving the nondeterministic choices in the timed process history. Parallel composition is studied in the context of the partial memoryless property. A mapping from PTP to a subclass of the single phase processes with exponentially distributed delays is defined.

BlenX [111,112,113,249,110,27] is a language used with Beta-binders calculus as a basis for scaled structure for modeling, simulation and analysis of biological systems. With that goal, a programming system Beta Workbench (BWB), based on BlenX, is described that simplifies development of the bio-systems models at different abstraction levels, can simulate their dynamic behaviour, check and ask the simulation results. BWB has three tools that jointly use the compiler and runtime environment of BlenX: stochastic simulator, CTMCs generator and reactions generator. The actions in BlenX have the rates (parameters of the exponential distribution of delay) or executed without delay. The operators over the BlenX processes are: deadlocked process, parallel (logical *and*), (guarded) choice (logical *or*), conditional (if-then), (guarded) replication and (action sequence) prefixing.

DNA Strand Displacement language (DSD) [242,189,190,188] is intended for designing, modeling and simulation of the DNA circuits that make computati-

ons via strand displacement. The examples of applying that computational mechanism are the digital logic circuits and catalytic signal amplification circuits, functioning as efficient molecular detectors. The DSD syntax describes molecules, their segments and three types of sequences: concatenation, left and right overhangings. DSD can model slow reactions with finite rates and fast reactions that occur instantly or much faster than others. After exploring all trajectories (interleavings of reactions, reduction paths) of the specified system, CTMCs are generated, used to analyze quantitative properties of its behaviour.

Markovian Calculus of Communicating Systems (mCCS) [114,115] is a Markovian extension of CCS with the interpretation on Markov automata (MAs) that describe systems behaviour via nondeterministic, probabilistic and timed events. Markov labeled transition systems are extracted from MAs to study their behaviour. Analogously to [159,117], external actions are taken as immediate, time progresses when no internal activity is possible, and timed actions only demonstrate Markovian behaviuor. The mCCS operations are: (successful) termination, indefinite (imprecise) and definite rate prefixing, insistent prefixing with an action, choice, parallelism, process constant (associated with definition) and probabilistic choice (with a finite index set).

Stochastic HYPE (that we call SHYPE) [44,127] is a HYPE [131,128] stochastic extension, designed for the fine-grained modeling stochastic hybrid systems. Each flow or influence affecting a variable is modeled separately and the general system's behaviour is obtained by composing these elements. A flow is an influence that continuously modifies a variable and has the strength and form that are changed by events. The continuous behaviour of a system is governed by the ODEs sets and is altered by discrete events. The discrete behaviour of a system is defined either by urgent actions, executed as soon as an activation condition is satisfied, or by non-urgent actions, which can wait some (non-zero) time before execution. In SHYPE, non-urgent actions are associated with probability distributions of their delays and thus become stochastic actions. The events are divided into instantaneous and stochastic and combined by the operations of prefix (with/without influence), choice, parallel (with/without synchronization) and constant (for recursive definitions). The operational semantics of SHYPE is defined via labeled multitransition systems. The stochastic hybrid semantics of SHYPE maps those transition systems into (or directly constructs from the syntactic model) Transition-Driven Stochastic Hybrid Automata, a subset of Piecewise Deterministic Markov Processes.

Markov Automata Process Algebra (MAPA) [275,276,143,144] is proposed for effective compositional specification, generation and modeling of Markov automata (MAs), whose events can be nondeterministic or happen probabilistically, or have exponential timed delay. The operations of MAPA include process instantiation (allowing recursion), conditional, nondeterministic choice, (possibly infinite) nondeterministic choice over data type, probabilistic choice over data type and rate (of the exponential delay) prefixing. For the modular construction of large systems, the top-level operations can be added: parallelism, encapsula-

tion (analogous to restriction), hiding and renaming. The operational semantics of MAPA is defined in terms of MAs.

Immediate PEPA (iPEPA) [158] adds to PEPA immediate actions with weights. The weights are transformed into probabilities while each parallel composition application and the resulted probabilities are recalculated at all composition levels. Immediate actions supplement standard timed actions having rates and are used to represent communication between the measurement processes, intended for specification of stationary and transient passage time measures. The iPEPA operations are: standard and immediate (high-priority) prefix, choice, constant and cooperation (on the action types from a set). After removing vanishing states (in which only immediate actions are executed) from the transition systems of the well-behaved (without immediate cycles and with deterministic initial behaviour) iPEPA components, the derived transition systems (with only timed transitions, accomplished by timed actions) are obtained, translated into CTMCs for performance analysis.

Immediate GPEPA (iGPEPA) [158] is an enrichment of GPEPA with immediate actions having weights. The actions in iGPEPA are timed (a pair of timed action type and rate) or immediate (a pair of immediate action type and weight). A component of iGPEPA is a component group with the group labeling or a cooperative composition of the iGPEPA components. The component group is an f-(fluid-) component or an unsynchronized parallel composition of f-components. The iGPEPA operations over component groups (purely concurrent groups of standard components of iPEPA [158]) are cooperation (over a set of synchronized actions) and labeling. The iGPEPA models have the associated systems of coupled first-order ODEs, used to calculate stationary and transient fluid passage times. The vanishing states (that enable immediate actions) are eliminated at the level of f-components when two regularity conditions are satisfied: absence of immediate cycles and deterministic initial behaviour. The operational semantics of iGPEPA is defined on the underlying CTMCs.

PHASE [84,85,83] is designed to model non-Markovian systems by implementing phase type distributed action delays. PHASE has sequential, choice and parallel operators. The elementary process is a phase type transition (a pair of action and infinitesimal generator matrix of its phase type delay). The parallel processes are synchronization by actions. The PHASE operational semantics represents phase type distributions through their generating CTMCs. It defines Markovian transitions (expressing exponentially distributed delays) and action transitions (corresponding to the instantaneous executions of actions). PHASE advantageously models and more accurately analyzes performance of non-Markovian systems with phase type distributions. PHASE is applied in the general analysis method for such systems, to obtain the processes translated into a probabilistic model checker for studying quantitative properties of the Markovian approximations.

Process Algebra for LOcated MArkovian agents (PALOMA) [121,122] describes the systems of populations consisting from the agents distributed over space, where the relative positions of agents influence their interaction, and com-

prises Markovian Multi-class, Multi-message Markovian Agent Models (M^2MAM). PALOMA can construct formal models of large collective adaptive systems, with the agents distributed over the named locations. Each action in PALOMA is either spontaneous (durational with a rate of the exponential occurrence time when it can emit a broadcast or unicast message of the same type) or induced (immediate with a probability to receive a broadcast or unicast message of the same type). The PALOMA operators over agents (parameterized by locations) are: spontaneous (with broadcast, unicast or without any emission) action prefixing, induced (by broadcast or unicast) action prefixing, choice and parallel composition. PALOMA has a discrete operational semantics, based on the labeled transition systems with delay and probabilistic transitions, from which semi-Markov chains (SMCs) can be extracted. The calculus also has a differential, population, operational semantics, from which population CTMC (pCTMCs) can be obtained, used while deriving ODEs for the mean-field model.

Stochastic Hybrid Communicating Sequential Processes (SHCSP) [237] extends Hybrid Communicating Sequential Processes (HCSP) calculus [297] with probability and stochasticity. In SHCSP, nondeterministic choice is replaced by probabilistic one and ODEs are generalized by stochastic differential equations (SDEs) that describe stochastic continuous evolution, including Brownian motion. In addition to the HCSP operations of null process, assignment, receiving or sending value along channel, sequential composition, alternative statement and repetition, the SHCSP operations include probabilistic choice and SDE-governed evolution, can specify preemption, weights, communication and concurrency, aiming to construct stochastic hybrid processes in a modular way.

Continuous time interleaving SPAs with positive deterministic actions.

Bio-PEPA with delays (Bio-PEPAd) [70,127] is an enrichment of Bio-PEPA, by adding non-Markovian action delays. The syntax of Bio-PEPAd is inherited from Bio-PEPA and endowed with the action delays functions. Bio-PEPAd has a Start-Termination- (ST-) operational semantics, where the beginning and end of each action execution are taken as separate events, defined by different action delays: exponentially distributed and positively deterministically timed, respectively. Distinguishing the starts and completions of actions is similar to the idea of ST-semantics for GSMPA [58,56]. The processes of Bio-PEPAd are translated into generalized semi-Markov processes (GSMPs) [59,4], used in the Delay Stochastic Simulation Algorithm (DSSA) and in Delay Differential Equations (DDEs) extending the deterministic formalism of ODEs to model biological systems with delays.

Table 4 (specifically) classifies the continuous time interleaving SPAs surveyed above (including MTIPP, PEPA and EMPA) and in Section 1 (sPBC and gsPBC) according to whether the time delays are associated with (multi)actions (integrated or orthogonal time [92,168]), the presence of (positive) deterministic or (only) immediate (multi)actions, and the type of stochastic delays (exponentially or phase type, or generally distributed). The names of SPAs with the SPN-based denotational semantics are printed in bold font.

Table 4. Classification of the continuous time interleaving stochastic process algebras

Time	Determin. (multi)act.	Exponential delays	Phase type delays	General delays
Integ-rated	Non-exist	MTIPP, **PEPA**, PA_S, **sPBC**, MAC, SP, BioNetGen, GPEPA, StoKlaim, SPiM, κ, SBioA, MPC, $PEPA{+}\Pi$, SBG, SPS, CGF, CPF, BGF, StoCCS, SCLS, **LBS**, Bio-PEPA, CoBiC, SpCLS, TSCLS, SCWC, stCCS, **stSA**, PAH, SBC, FPA, SSPA, **CChS**, FEPA, ProPPA, CARMA, MELA, NBA	$PEPA^{\infty}_{ph}$	PA_{GS}, GPA
	Immediate	MPA, MPA-B, PM-TIPP, SE-TIPP, **EMPA**, BioSpi, IMAC, SBB, nanoκ, SPiCO, Sπ@, $\pi(\mathcal{L})$, XSP, BlenX, **gsPBC**, DSD, SHYPE, iPEPA, iGPEPA, PALOMA	PHASE	NMSPA, **SM-PEPA**
	Positive	Bio-PEPAd	—	—
Ortho-gonal	Non-exist	MC, MASSPA	CCC	—
	Immediate	STC, $\mathcal{SPA}$, IMC, IML, sCCP, mCCS, MAPA	SB-LOTOS, PTP	SPADES-HS, SPADES, IGSMP, VPSPA, MODEST, SHCSP

4.2 Continuous time and non-interleaving semantics

Only a few non-interleaving SPAs were considered among non-Markovian ones [180,57]. The semantics of all Markovian calculi is interleaving and their action delays have exponential distribution, which is the only continuous probability distribution with memoryless (Markovian) property.

In [60], Generalized Stochastic Process Algebra (GSPA) was introduced. It has a true-concurrent denotational semantics in terms of generalized stochastic event structures (GSESs) with non-Markovian stochastic delays of events. In that paper, no operational semantics or performance evaluation methods for GSPA were presented. In [181], generalized semi-Markov processes (GSMPs) [59,4] were extracted from GSESs to analyze performance.

In [247,248,188], Generalized Stochastic π-calculus (that we call GSπ) with general continuous distributions of activity delays was defined. It has a proved operational semantics with transitions labeled by encodings of their deduction trees. No well-established underlying performance model for this version of GSπ was described.

In [58,56], Generalized Semi-Markovian Process Algebra (GSMPA) was developed with an ST-operational semantics and non-Markovian action delays. The performance analysis in GSMPA is accomplished via GSMPs.

Again, the first fundamental difference between dtsdPBC and the calculi GSPA, GSπ and GSMPA is that dtsdPBC is based on PBC, whereas GSPA is

an extension of simple Process Algebra (PA) from [60], GSπ extends π-calculus [218,219] and GSMPA is an enrichment of EMPA. Therefore, both GSPA and GSMPA have *prefixing*, *choice* (*alternative* composition), *parallel* composition, *renaming* (*relabeling*) and *hiding* (*abstraction*) operations, but only GSMPA has *constants*. Unlike dtsdPBC, GSPA has neither iteration or recursion, GSMPA allows only *recursive* definitions, whereas GSπ additionally has operations to specify *mobility*. Note also that GSPA, GSπ and GSMPA do not specify even instantaneous events or activities while dtsdPBC has deterministic multiactions.

The second significant difference is that geometrically distributed or zero delays are associated with process states in dtsdPBC, unlike generally distributed delays assigned to events in GSPA or to activities in GSπ and GSMPA. As a consequence, dtsdPBC has a discrete time operational semantics allowing for concurrent execution of activities in steps. GSPA has no operational semantics while GSπ and GSMPA have continuous time ones. In continuous time semantics, concurrency is simulated by interleaving, since simultaneous occurrence of any two events has zero probability according to the properties of continuous probability distributions. Therefore, interleaving transitions are often annotated with an additional information to keep concurrency data. The transition labels in the operational semantics of GSπ encode the action causality information and allow one to derive the enabling relations and the firing distributions of concurrent transitions from the transition sequences. At the same time, abstracting from stochastic delays leads to the classical early interleaving semantics of π-calculus [218,219]. The ST-operational semantics of GSMPA is based on decorated transition systems governed by transition rules with rather complex preconditions. There are two types of transitions: the choice (action beginning) and the termination (action ending) ones. The choice transitions are labeled by weights of single actions chosen for execution while the termination transitions have no labels. Only single actions can begin, but several actions can end in parallel. Thus, the choice transitions happen just sequentially while the termination transitions can happen simultaneously. As a result, the decorated interleaving / step transition systems are obtained. dtsdPBC has an SPN-based denotational semantics. In comparison with event structures, PNs are more expressive and visually tractable formalism, capable of finitely specifying an infinite behaviour. Recursion in GSPA produces infinite GSESs while dtsdPBC has iteration operation with a finite SPN semantics. Identification of infinite GSESs that can be finitely represented in GSPA was left for a future research.

4.3 Discrete time

In [1], a class of compositional DTSPNs with generally distributed discrete time transition delays was proposed, called dts-nets. The denotational semantics of a stochastic extension (that we call stochastic ACP or sACP) of a subset of Algebra of Communicating Processes (ACP) [20] can be constructed via dts-nets. There are two types of transitions in dts-nets: immediate (timeless) ones, with zero delays, and time ones, whose delays are random variables having general

discrete distributions. The top-down synthesis of dts-nets consists in the substitution of their transitions by blocks (dts-subnets) corresponding to the sequence, choice, parallelism and iteration operators. It was explained how to calculate the throughput time of dts-nets using the service time (defined as holding time or delay) of their transitions. For this, the notions of service distribution for the transitions and throughput distribution for the building blocks were defined. Since the throughput time of the parallelism block was calculated as the maximal service time for its two constituting transitions, the analogue of the step semantics was implemented.

In [203,204], an SPA called Theory of Communicating Processes with discrete stochastic time (TCP^{dst}) was introduced, later in [202] called Theory of Communicating Processes with discrete real and stochastic time (TCP^{drst}). It has discrete real time (deterministic) delays (including zero delays) and discrete stochastic time delays. The algebra generalizes real time processes to discrete stochastic time ones by applying real time properties to stochastic time and imposing race condition to real time semantics. TCP^{dst} has an interleaving operational semantics in terms of stochastic transition systems. The performance is analyzed via discrete time probabilistic reward graphs which are essentially the reward transition systems with probabilistic states having finite number of outgoing probabilistic transitions and timed states having a single outgoing timed transition. The mentioned graphs can be transformed by unfolding or geometrization into discrete time Markov reward chains (DTMRCs) appropriate for transient or stationary analysis.

The first difference between dtsdPBC and the algebras sACP and TCP^{dst} is that dtsdPBC is based on PBC, but sACP and TCP^{dst} are the extensions of ACP [20]. sACP has taken from ACP only *sequence*, *choice*, *parallelism* and *iteration* operations, whereas dtsdPBC has additionally relabeling, restriction and synchronization ones, inherited from PBC. In TCP^{dst}, besides standard action *prefixing*, *alternative* composition, *parallel* composition, *encapsulation* (similar to *restriction*) and *recursive* variables, there are also *timed delay prefixing*, *dependent delays scope* and the *maximal time progress* operators, which are new both for ACP and dtsdPBC.

The second difference is that dtsdPBC, sACP and TCP^{dst} have zero delays, however, discrete time delays in dtsdPBC are zeros or geometrically distributed (being 1 or ∞ as special cases) and associated with process states. The zero delays are possible just in vanishing states while geometrically distributed delays are possible only in tangible states. For each s-tangible (w-tangible) state, the parameter of geometric distribution governing the delay in the state is completely determined by the probabilities (weights) of all stochastic (waiting) multiactions executable from it. In sACP and TCP^{dst}, delays are generally distributed, but they are assigned to transitions in sACP and separated from actions (excepting zero delays) in TCP^{dst}. A special attention is given to zero delays in sACP and deterministic delays in TCP^{dst}. In sACP, immediate (timeless) transitions with zero delays serve as source and sink transitions of the dts-subnets corresponding to the choice, parallelism and iteration operators. In TCP^{dst}, zero delays of

actions are specified by undelayable action prefixes while positive deterministic delays of processes are specified with timed delay prefixes. Neither formal syntax nor operational semantics for sACP are defined and it is not explained how to derive Markov chains from the algebraic expressions or the corresponding dts-nets to analyze performance. It is not stated explicitly, which type of semantics (interleaving or step) is accommodated in sACP. In spite of the discrete time approach, operational semantics of TCP^{dst} is still interleaving, unlike that of dtsdPBC. In addition, no denotational semantics was defined for TCP^{dst}.

Consider *other SPAs with discrete time and interleaving semantics.*

Discrete time interleaving SPAs without immediate actions.

Weighted Synchronous Calculus of Communicating Systems (WSCCS) [277,278,215] is an extension of SCCS [217] with weights. WSCCS is a calculus of probabilistic processes, where probabilities are not directly assigned to the choice operation. Instead, weights are interpreted as the probabilistic specifications using the relative frequency concept and corresponding equality criterium. There exist special weights expressing priorities. The weights and actions in WSCCS are separated. The actions in WSCCS form an Abelian group with the identity action and inverse of each action. The WSCCS operators are: empty (null) process, (action) prefix, weighted choice (with a finite number of weights), (synchronous) parallel composition, permit (only actions in a set), prioritized parts (taking only), relabeling and recursion (or recursive definition). The discrete model of time is applied in WSCCS and processes are executed at time ticks. Either weighted choice or named action execution occur at each tick, resulting in the interleaving semantics and *stratified* model [140]. WSCCS is also used to calculate upper bounds on the performance of mutually affected systems, in which action delays are specified symbolically as random values with general discrete phase type distributions.

Discrete time variant (that we call dsCCP) of stochastic Concurrent Constraint Programming (sCCP) was proposed in [41]. The calculus sCCP is constructed to model and analyze biological systems. In sCCP, communication is asynchronous, species are described by variables, reactions are seen as constraints on the variables and rates are defined using functions. The operations of sCCP include ask prefix (with rate), tell prefix (with rate), choice, empty process, procedure call (with rate), hiding and parallel composition. The analysis consists in stochastic simulation, as well as in translation into a probabilistic verification tool, ODEs and hybrid systems. A discrete time version (with rates interpreted as weights, used to calculate probabilities) of the sCCP operational semantic results in the algebra dsCCP, with DTMCs being the performance analysis model.

Discrete time interleaving SPAs with immediate actions.

Interactive Probabilistic Chains (IPC) [95,156] calculus unifies in a single probabilistic discrete time model the capabilities of compositional modeling, functional verification and performance analysis (through translation into DTMCs) for industrial systems and networks on chips. The operators of IPC are termination, sequential composition, probabilistic choice (with a set of probabilities), nondeterministic choice, parallel composition (with synchronization set), hiding

(of actions set), process call and (possibly recursive) process definition. Being a discrete time analogue of the algebra IMC [159,160], IPC has an interleaving operational semantics on the unification of labeled transition systems and DTMCs. The transitions in that semantics are either probabilistic (that occur with particular probabilities during exactly one discrete time tick) or interactive (corresponding to the instantaneous execution of some actions, possibly invisible). The performance model of IPC is DTMCs, obtained from interactive probabilistic chains using schedulers to resolve a nondeterministic choice by replacing it with a probabilistic choice.

The three SPAs are rather specific: unlike standard approach, weights in WSCCS, rates (weights) in dsCCP and probabilities in IPC are not associated with actions. In dsCCP, probabilities are calculated using rates (weights) that are assigned to operations. In IPC, actions are executed instantaneously while probabilistic choices take one unit time. In the common SPAs with the *integrated time* concept, the time parameters are combined with actions into pairs called activities. In dsCCP and IPC, the *orthogonal time* concept is applied, where time progress is separated from actions, assumed to be immediate and to specify logical progress [92,168].

Table 5 summarizes the SPAs comparison above and that from Section 1 (the calculi sPBC, gsPBC, dtsPBC and dtsiPBC), by (generally) classifying the SPAs according to the concept of time, the presence of (positive or arbitrary) deterministic or (only) immediate (multi)actions, and the operational semantics type. The names of SPAs with the denotational semantics based on SPNs are printed in bold font. The underlying stochastic process (if defined) for each presented SPA is specified in parentheses near its name.

Table 5. Classification of stochastic process algebras

Time	Deterministic (multi)actions	Interleaving semantics	Non-interleaving semantics
Continuous	Non-exist	MTIPP (CTMC), **PEPA** (CTMP), **sPBC** (CTMC)	GSPA (GSMP), GSπ, GSMPA (GSMP)
	Immediate	**EMPA** (SMC, CTMC), **gsPBC** (SMC)	—
	Positive	Bio-PEPAd (GSMP)	—
Discrete	Non-exist	WSCCS (DTMC), dsCCP (DTMC)	**dtsPBC** (DTMC)
	Immediate	IPC (DTMC)	**dtsiPBC** (SMC, DTMC)
	Arbitrary	TCP^{dst} (DTMRC)	**sACP**, **dtsdPBC** (SMC, DTMC)

5 Discussion

Let us now discuss which advantages has dtsdPBC in comparison with the SPAs described in Section 4.

5.1 Analytical solution

An important aspect is the analytical tractability of the underlying stochastic process, used for performance evaluation in SPAs. The underlying CTMCs in MTIPP and PEPA, as well as SMCs in EMPA, are treated analytically, but these continuous time SPAs have interleaving semantics. GSPA, GSπ and GSMPA are the continuous time models, for which a non-interleaving semantics is constructed, but for the underlying GSMPs in GSPA and GSMPA, only simulation and numerical methods are applied, whereas no performance model for GSπ is defined. sACP and TCP^{dst} are the discrete time models with the associated analytical methods for the throughput calculation in sACP or for the performance evaluation based on the underlying DTMRCs in TCP^{dst}, but both models have interleaving semantics. dtsdPBC is a discrete time model with a non-interleaving semantics, where analytical methods are applied to the underlying SMCs. Hence, if an interleaving model is appropriate as a framework for the analytical solution towards performance evaluation then one has a choice between the continuous time SPAs MTIPP, PEPA, EMPA and the discrete time ones sACP, TCP^{dst}. Otherwise, if one needs a non-interleaving model with the associated analytical methods for performance evaluation and the discrete time approach is feasible then dtsdPBC is the right choice.

The existence of an analytical solution also permits to interpret quantitative values (rates, probabilities, weights etc.) from the system specifications as parameters, which can be adjusted to optimize the system performance, like in dtsPBC, dtsiPBC and dtsdPBC. The DTMCs whose transition probabilities are parameters were introduced in [103]. The parameters can also be adjusted in parametric probabilistic transition systems (PTSs) [192], i.e. in the DTMCs whose transition probabilities may be real-valued parameters. Parametric CTMCs with the transition rates treated as parameters were investigated in [151]. Parametric probabilistic timed automata (PTAs) were defined in [78]. Parametric DTMCs with the transition probabilities being polynomials over real-valued parameters were investigated in [150]. In [175], a new method of computing the reachability probabilities was proposed for parametric DTMCs whose state change probabilities are the fractions of polynomials over the set of parameters. The parameter value synthesis problem was studied in [109] for parametric interval DTMCs (IDTMCs), in which the parameters are the borders of the transition probability intervals. In [253], a new parameter synthesis technique called lifting was proposed for parametric models: stochastic games (SGs), Markov decision processes (MDPs) and DTMCs. Parametric verification for concurrent systems modeled by parametric versions of timed automata (TAs), interval (DT)MCs (IDTMCs), PNs and logic Action-Restricted CTL (ARCTL) was surveyed in [2]. For parametric verification with logic PCTL in [98], uncertain MDPs (UMDPs)

were applied whose parameters may be either controlled (as in the standard parametric MDPs) or uncontrolled (being random values with the probability distributions), aiming to specify uncertainty of the transition probabilities and reward functions. In [178], the parameter synthesis problem was investigated and the algorithms of its solution were proposed for two Markovian models: parametric DTMCs and MDPs, being the subclasses of parametric SGs.

On the other hand, no parameters in formulas of SPAs were considered in the literature so far. In dtsdPBC we can easily construct examples with more parameters than we did in our case study. The performance indices will be then interpreted as functions of several variables. The advantage of our approach is that, unlike of the method from [192] and other works, we should not impose to the parameters any special conditions needed to guarantee that the real values, interpreted as the transition probabilities, always lie in the interval $[0;1]$. To be convinced of this fact, just remember that, as we have demonstrated, the positive probability functions PF, PT, PM define probability distributions, hence, they always return values belonging to $(0;1]$ for any probability parameters from $(0;1)$ and weight parameters from $\mathbb{R}_{>0}$. In addition, the transition constraints (their probabilities, rates and guards), calculated using the parameters, in our case should not always be polynomials over variables-parameters, as often required in the mentioned papers, but they may also be fractions of polynomials, like in our case study.

5.2 Concurrency interpretation

One can see that the stochastic process calculi proposed in the literature are based on interleaving, as a rule, and parallelism is simulated by synchronous or asynchronous execution. As a semantic domain, the interleaving formalism of transition systems is often used. However, to properly support intuition of the behaviour of concurrent and distributed systems, their semantics should treat parallelism as a primitive concept that cannot be reduced to nondeterminism. Moreover, in interleaving semantics, some important properties of these systems cannot be expressed, such as simultaneous occurrence of concurrent transitions [105] or local deadlock in the spatially distributed processes [224]. Therefore, investigation of stochastic extensions for more expressive and powerful algebraic calculi is an important issue. The development of step or "true concurrency" (such that parallelism is considered as a causal independence) SPAs is an interesting and nontrivial problem, which has attracted special attention last years. Nevertheless, not so many formal stochastic models of parallel systems were defined whose underlying stochastic processes are based on DTMCs. As mentioned in [123], such models are more difficult to analyze, since several events can occur simultaneously in discrete time systems (the models have a step semantics) and the probability of a set of events cannot be easily related to the probability of the single ones. Thus, parallel executions of actions are often not considered also in the discrete time SPAs, such as TCP^{dst}, whose underlying stochastic process is DTMCs with rewards (DTMRCs). As observed in [170], even for stochastic models with generally distributed delays, the concurrency degree restrictions

were imposed to simplify their analysis techniques. In particular, the enabling restriction requires that no two generally distributed transitions are enabled in any reachable marking. Hence, their activity periods do not intersect and no two such transitions can fire simultaneously. This results in interleaving semantics of the model.

Stochastic models with discrete time and step semantics have the following important advantage over those having just an interleaving semantics. The underlying Markov chains of parallel stochastically timed processes have the additional transitions corresponding to the simultaneous execution of concurrent (i.e. non-synchronized) activities. The transitions of that kind allow one to bypass a lot of intermediate states, which otherwise should be visited when interleaving semantics is accommodated. When step semantics is used, the intermediate states can also be visited with some probability (this is an advantage, since some alternative system's behaviour may start from these states), but this probability is not greater than the corresponding one in case of interleaving semantics. While in interleaving semantics, only the empty or singleton (multi)sets of activities can be executed, in step semantics, generally, the (multi)sets of activities with more than one element can be executed as well. Hence, in step semantics, there are more variants of execution from each state than in the interleaving case and the executions probabilities, whose sum should be equal to 1, are distributed among more possibilities. Therefore, the systems with parallel stochastic processes usually have smaller average run-through. In case the underlying Markov chains of the processes are ergodic, they will generally take less discrete time units to stabilize the behaviour, since their TPMs will be usually denser because of additional non-zero elements outside the main diagonal. Hence, both the first passage-time performance indices based on the transient probabilities and the steady-state performance indices based on the stationary probabilities can be potentially computed quicker, resulting in mostly faster quantitative analysis of the systems. On the other hand, step semantics, induced by simultaneous firing several transitions at each step, is natural for Petri nets and allows one to exploit full power of the model. Therefore, it is important to respect the probabilities of parallel executions of activities in discrete time SPAs, especially in those with a Petri net denotational semantics.

The speed (rate) of converging the transient PMF for a DTMC to its stationary PMF was studied in [187] (the quantitative estimate via the TPM's second eigenvalue, by the absolute value descendance) and in [124] (the equivalent qualitative conditions in terms of geometric ergodicity, i.e. exponentially fast approaching the stationary distribution with time progress).

5.3 Application area

From the application viewpoint, one considers what kind of systems are more appropriate to be modeled and analyzed within SPAs. MTIPP and PEPA are well-suited for the interleaving continuous time systems such that the activity rates or the average sojourn time in the states are known in advance and exponential distribution approximates well the activity delay distributions, whereas

EMPA can be used to model the mentioned systems with the activity delays of different duration order or the extended systems, in which purely probabilistic choices or urgent activities must be implemented. GSPA and GSMPA fit well for modeling the continuous time systems with a capability to keep the activity causality information, and with the known activity delay distributions, which cannot be approximated accurately by exponential distribution, while GSπ can additionally model mobility in such systems. TCP^{dst} is a good choice for interleaving discrete time systems with deterministic (fixed) and generalized stochastic delays, whereas sACP is capable to model non-interleaving systems as well, but it offers not enough performance analysis methods.

dtsdPBC is consistent for the step discrete time systems such that the independent execution probabilities of activities are known and geometrical distribution approximates well the state residence time distributions. These include Dirac distribution of the positive deterministic sojourn time, which is then splitted into one time units and allocated with the consecutive process states. In addition, dtsdPBC can model the mentioned systems featuring very scattered activity delays, or even more complex systems with instantaneous probabilistic choice or urgency. Hence, dtsdPBC can be taken as a non-interleaving discrete time counterpart of TCP^{dst}.

5.4 Advantages of our approach

Table 6 contains a classification of the (labeled) SPNs classes mentioned in this paper, according to the model of time (continuous or discrete) and presence of (besides stochastic) immediate or deterministic (i.e. immediate and waiting) transitions. We consider (labeled) CTSPNs [222,207,65,11,19,12], GSPNs [79,81,207,82,208,66,11,19,12], WDTSPNs [68], DTSPNs [221,223,261,262,263,264,265], spTPNs [62], DTSIPNs [270,271,272,273,274], DTDSPNs [302,298,299] and DTSDPNs [266,267,268,269]. We also consider a continuous time model of deterministic stochastic Petri nets (DSPNs) [209,210] with stochastic (exponential) and deterministic transitions. In the parentheses near the SPNs classes, the names of the SPAs discussed here are written whose denotational semantics is based on the respective types of SPNs. For example, denotational semantics of PEPA is constructed using (labeled) CTSPNs while that of dtsiPBC is defined via dtsi-boxes, a subclass of LDTSIPNs. The names of the SPNs and SPAs, defined by us, are printed in bold font. In the table, all the SPNs with continuous time have interleaving semantics whereas those with discrete time have non-interleaving (step) semantics.

Thus, the main advantages of dtsdPBC are the flexible multiaction labels, stochastic and deterministic multiactions, powerful operations, as well as a step operational and a Petri net denotational semantics allowing for concurrent execution of activities (transitions), together with an ability for analytical and parametric performance evaluation. The uniqueness of our approach consists in applying a parallel semantics for the process expressions and preserving the concurrency level in the extracted performance models (SMC, DTMC and RDTMC) through their state changes corresponding to the simultaneous executions.

Table 6. Classification of stochastic Petri nets

Time	Stochastic transitions	Stochastic and immediate transitions	Stochastic and deterministic transitions
Continuous	(L)CTSPNs (PEPA, sPBC)	(L)GSPNs (EMPA, gsPBC)	DSPNs (—)
Discrete	**(L)WDTSPNs** (—), **(L)DTSPNs** (**dtsPBC**)	spTPNs (—), **(L)DTSIPNs** (**dtsiPBC**)	DTDSPNs (—), **(L)DTSDPNs** (**dtsdPBC**)

6 Conclusion

In this paper, we have considered dtsdPBC [266,267,268,269], an extension with discrete stochastic and deterministic time of Petri box calculus (PBC) [29,31,30,28]. Stochastic process algebra dtsdPBC has a parallel step operational semantics, based on labeled probabilistic transition systems, and a Petri net denotational semantics in terms of dtsd-boxes, a special subclass of LDTSDPNs [266,267]. The underlying semi-Markov chains (SMCs) and (reduced) discrete time Markov chains (DTMCs and RDTMCs) of the process expressions are analyzed in dtsdPBC to evaluate performance [268]. We have determined the advantages of dtsdPBC by comparing it with more than 90 other SPAs, most of which appeared to adapt continuous time, interleaving semantics and exponential delays. We have discussed the SPAs approaches to the analytical solution, concurrency interpretation and application area.

The advantage of our framework is twofold. First, one can specify in it concurrent composition and synchronization of (multi)actions, what is not possible in classical Markov chains. As argued in [279], (stochastic) PNs represent the systems structure more concisely and can be an intermediate formalism for their more intuitive translation into Markov chains. Second, algebraic formulas represent processes in a more compact way than PNs and allow one to apply syntactic transformations and comparisons. Process algebras are compositional by definition and their operations naturally correspond to operators of programming languages. Hence, it is much easier to construct a complex model in the algebraic setting than in PNs. The complexity of PNs generated for practical models in the literature demonstrates that it is not straightforward to construct such PNs directly from the system specifications.

dtsdPBC is well suited for the discrete time applications, whose discrete states change with a global time tick, such as business processes, neural and transportation networks, computer and communication systems, timed web services [285], as well as for those, in which the distributed architecture or the concurrency level should be preserved while modeling and analysis, such as genetic regulatory and cellular signalling networks (featuring maximal parallelism) in biology [100,101,102,40,18] (remember that we have additional transitions due to concurrent executions in step semantics). In [136], biological networks were

jointly modeled by (standard, qualitative) PNs, CTSPNs and continuous PNs (CPNs), to demonstrate their complementarity that makes necessary adding deterministic time to stochastic models, as well as combining stochastic and continuous (deterministic) aspects into one model (such as stochastic rates of reactions and continuous amounts of species).

dtsdPBC can also model and analyze parallel systems with fixed durations of the typical activities (loading, processing, transfer, repair, low-level events, message delivery) and stochastic durations of the randomly occurring activities (arrival, departure, failure, packet loss, message collision), including industrial, manufacturing, queueing, computing and network systems.

Future work consists in constructing a congruence relation for dtsdPBC, i.e. the equivalence that withstands application of all operations of the algebra. A possible candidate is a stronger version of the equivalence with respect to transition systems, with two extra transitions skip and redo, like in sPBC [198]. Moreover, recursion operation could be added to dtsdPBC to increase specification power of the algebra.

We also plan to extend dtsdPBC with discrete phase type multiaction delays that are described by arbitrary finite absorbing DTMCs and include geometric and non-Markovian (like deterministic) delays as special cases. Discrete phase type probability distributions approximate with any precision general discrete distributions over positive integers and are closed under minimum (alternative composition, conflict), maximum (parallel composition, parallelism), finite convolution (sequential composition, precedence), finite weighted and infinite geometric summations [225,260,171,280,187,173].

Some known SPNs with phase type transition delays are: SPNs with phase-type distributed transition times (PTDTT-SPNs) [99] and phased delay PNs (PDPNs) [177] (the both classes with discrete and continuous time), as well as defective discrete phase SPNs (DDP-SPNs) [80], discrete deterministic and stochastic PNs (DDSPNs) [300,301] and non-Markovian SPNs (NMSPNs) [172] (the three classes with discrete time). Only NMSPNs have a non-interleaving transition firing semantics, but it is complex and technical.

Some existing SPAs with phase type action delays are: a modification of PA_{GS} [179], $PEPA^{\infty}_{ph}$ [118], SB-LOTOS [161], PTP [293], PHASE [84,85,83] and CCC [252,33] (the six SPAs with continuous time), as well as a variant of WSCCS [278] (with discrete time). All those SPAs have only interleaving operational and no SPN-based denotational semantics. Those interleaving phase SPAs are rather specialized or theoretically-oriented and hardly applicable in practice or with restricted specification capabilities. In detail, PA_{GS} is rather theoretical, $PEPA^{\infty}_{ph}$ describes very special subclasses of non-Markovian systems, SB-LOTOS separates actions and delays, WSCCS has technically complex and non-sufficiently intuitive syntax and semantics, PTP has cooperating processes that cannot be synchronized by the shared activities, PHASE offers just a few operators, whereas CCC does not have actions, synchronization and recursion. Unlike PA_{GS}, $PEPA^{\infty}_{ph}$, SB-LOTOS, PTP, PHASE and CCC with

continuous time, WSCCS adapts a discrete time model, but the semantics of WSCCS is still interleaving.

Thus, it is actual to construct a discrete time SPA with phase type delays and non-interleaving semantics: operational one (on the labeled transition systems with parallel executions of activities) and denotational one (on the SPNs with phase delays and parallel firings of transitions).

References

1. W.M.P. van der Aalst, K.M. van Hee, H.A. Reijers, *Analysis of discrete-time stochastic Petri nets*, Statistica Neerlandica, **54**:2 (2000), 237–255. Zbl 0994.68091
2. É. André, M. Knapik, D. Lime, W. Penczek, L. Petrucci, *Parametric verification: an introduction*, Lecture Notes in Computer Science, **11790** (2019), 64–100.
3. P.R. D'Argenio, *Algebras and automata for timed and stochastic systems*, Ph.D. thesis, CTIT PhD-Thesis Series, **99-25**, Department of Computer Science, University of Twente, Enschede, The Netherlands, 1999.
4. P.R. D'Argenio, J.-P. Katoen, *A theory of stochastic systems. Part I: Stochastic automata*, Information and Computation, **203**:1 (2005), 1–38.
5. P.R. D'Argenio, J.-P. Katoen, *A theory of stochastic systems. Part II: Process algebra*, Information and Computation, **203**:1 (2005), 39–74.
6. P.R. D'Argenio, J.-P. Katoen, E. Brinksma, *An algebraic approach to the specification of stochastic systems (extended abstract)*, Proc. IFIP TC2/WG2.2, 2.3 Int. Conf. on Programming Concepts and Methods (PROCOMET) 1998 (D. Gries, W.-P. de Roever, eds.), Shelter Island, New York, USA, 126–147, Chapman & Hall, London, UK, 1998.
7. P.R. D'Argenio, J.-P. Katoen, E. Brinksma, *A compositional approach to generalised semi-Markov processes*, Proc. 4^{th} Int. Workshop on Discrete Event Systems (WODES) 1998 (A. Guia, R. Smedinga, M.P. Spathopoulos, eds.), Cagliary, Italy, 391–397, IEE Publisher, London, UK, 1998.
8. A. Argent-Katwala, J.T. Bradley, A. Clark, S. Gilmore, *Location-aware quality of service measurements for service-level agreements*, Lecture Notes in Computer Science, **4912** (2008), 222–239.
9. A. Argent-Katwala, J.T. Bradley, N.J. Dingle, *Expressing performance requirements using regular expressions to specify stochastic probes over process algebra models*, Proc. 4^{th} Int. Workshop on Software and Performance (WOSP) 2004, Redwood Shores, California, USA, 49–58, ACM Press, 2004.
10. G. Bacci, M. Miculan, *Measurable stochastics for Brane Calculus*, Theoretical Computer Science, **431** (2012), 117–136.
11. G. Balbo, *Introduction to stochastic Petri nets*, Lecture Notes in Computer Science, **2090** (2001), 84–155. Zbl 0990.68092
12. G. Balbo, *Introduction to generalized stochastic Petri nets*, Lecture Notes in Computer Science, **4486** (2007), 83–131. Zbl 1323.68400
13. R. Barbuti, G. Caravagna, A. Maggiolo-Schettini, P. Milazzo, G. Pardini, *The calculus of looping sequences*, Lecture Notes in Computer Science, **5016** (2008), 387–423.
14. R. Barbuti, A. Maggiolo-Schettini, P. Milazzo, G. Pardini, *Spatial calculus of looping sequences*, Proc. 2^{nd} Workshop From Biology to Concurrency and Back (FBTC) 2008 (N. Cannata, E. Merelli, I. Ulidowski, eds.), Reykjavik, Iceland, Electronic Notes in Theoretical Computer Science, **229**:1 (2009), 21–39.

15. R. Barbuti, A. Maggiolo-Schettini, P. Milazzo, G. Pardini, *Spatial calculus of looping sequences*, Theoretical Computer Science, **412**:43 (2011), 5976–6001.
16. R. Barbuti, A. Maggiolo-Schettini, P. Milazzo, P. Tiberi, P. Troina, *Stochastic calculus of looping sequences for the modelling and simulation of cellular pathways*, Lecture Notes in Computer Science, **5121** (2008), 86–113.
17. R. Barbuti, A. Maggiolo-Schettini, P. Milazzo, A. Troina, *A calculus of looping sequences for modelling microbiological systems*, Fundamenta Informaticae, **72**:1–3 (2006), 21–35. Zbl 1101.92021
18. E. Bartocci, P. Lió, *Computational modeling, formal analysis, and tools for systems biology*, PLoS Computational Biology, **12**:1 (2016), e1004591.
19. F. Bause, P.S. Kritzinger, *Stochastic Petri nets: an introduction to the theory*, Friedrich Vieweg and Sohn, Braunschweig / Wiesbaden, Germany, 2002. Zbl 1013.60065
20. J.A. Bergstra, J.W. Klop, *Algebra of communicating processes with abstraction*, Theoretical Computer Science, **37** (1985), 77–121.
21. M. Bernardo, *Theory and application of extended Markovian process algebra*, Ph.D. thesis, University of Bologna, Italy, 1999.
22. M. Bernardo, S. Botta, *A survey of modal logics characterizing behavioural equivalences for non-deterministic and stochastic systems*, Mathematical Structures in Computer Science, **18**:1 (2008), 29–55. MR2459612
23. M. Bernardo, F. Corradini, L. Tesei, *Timed process calculi with deterministic or stochastic delays: commuting between durational and durationless actions*, Theoretical Computer Science (2016), **629**, 2–39.
24. M. Bernardo, L. Donatiello, R. Gorrieri, *Modeling and analyzing concurrent systems with MPA*, Proc. 2^{nd} Int. Workshop on Process Algebra and Performance Modelling (PAPM) 1994 (U. Herzog, M. Rettelbach, eds.), Regensberg / Erlangen, Germany, Arbeitsberichte des IMMD, **27**:4 (1994), 71–88, Universität Erlangen-Nürnberg, Germany.
25. M. Bernardo, L. Donatiello, R. Gorrieri, *A formal approach to the integration of performance aspects in the modeling and analysis of concurrent systems*, Information and Computatison, **144**:2 (1998), 83–154.
26. M. Bernardo, R. Gorrieri, *A tutorial on EMPA: a theory of concurrent processes with nondeterminism, priorities, probabilities and time*, Theoretical Computer Science, **202**:1–2 (1998), 1–54.
27. A. Bernini, L. Brodo, P. Degano, M. Falaschi, D. Hermith, *Process calculi for biological processes*, Natural Computing, **17** (2018), 345–373.
28. E. Best, R. Devillers, *Petri net primer: a compendium on the core model, analysis, and synthesis*, Computer Science Foundations and Applied Logic Series (CSFAL), Springer International Publishing / Birkhäuser, 2024.
29. E. Best, R. Devillers, J.G. Hall, *The box calculus: a new causal algebra with multi-label communication*, Lecture Notes in Computer Science, **609** (1992), 21–69.
30. E. Best, R. Devillers, M. Koutny, *Petri net algebra*, EATCS Monographs on Theoretical Computer Science, Springer, 2001.
31. E. Best, M. Koutny, *A refined view of the box algebra*, Lecture Notes in Computer Science, **935** (1995), 1–20.
32. L. Bettini, V. Bono, R. De Nicola, G.L. Ferrari, D. Gorla, M. Loreti, E. Moggi, R. Pugliese, E. Tuosto, B. Venneri, *The Klaim project: theory and practice*, Lecture Notes in Computer Science, **2874** (2003), 88–150.

33. A. Bies, H. Hermanns, M.A. Köhl, A. Schmidt, *Matching distributions under structural constraints*, Lecture Notes in Computer Science, **14287** (2023), 221–237.
34. L. Bioglio, M. Dezani-Ciancaglini, P. Giannini, A. Troina, *Typed stochastic semantics for the calculus of looping sequences*, Theoretical Computer Science, **431** (2012), 165–180.
35. M.L. Blinov, J.R. Faeder, B. Goldstein, W.S. Hlavacek, *BioNetGen: software for rule-based modeling of signal transduction based on the interactions of molecular domains*, Bioinformatics, **20**:17 (2004), 3289–3291.
36. H.C. Bohnenkamp, P.R. D'Argenio, H. Hermanns, J.-P. Katoen, *MODEST: a compositional modeling formalism for hard and softly timed systems*, IEEE Transactions on Software Engineering, **32**:10 (2006), 812–830.
37. H.C. Bohnenkamp, B.R. Haverkort, *Semi-numerical solution of stochastic process algebra models*, Lecture Notes in Computer Science, **1601** (1999), 228–243.
38. T. Bolognesi, F. Lucidi, *LOTOS-like process algebras with urgent or timed interactions*, Proc. IFIP TC6/WG6.1 4^{th} Int. Conf. on Formal Description Techniques for Distributed Systems and Communication Protocols: Formal Description Techniques, IV (FORTE) 1991 (K.R. Parker, G.A. Rose, eds.), Sydney, Australia, IFIP Transactions C: Communication Systems (1992), 249–264, Elsevier Science Publishers (North-Holland), Amsterdam, The Netherlands, 1992.
39. T. Bolognesi, F. Lucidi, S. Trigila, *From timed Petri nets to timed LOTOS*, Proc. IFIP WG6.1 10^{th} Int. Symposium on Protocol Specification, Testing and Verification (PSTV) 1990 (L. Logrippo, R.L. Probert, H. Ural, eds.), Ottawa, Canada, 395–408, Elsevier Science Publishers (North-Holland), Amsterdam, The Netherlands 1990.
40. N. Bonzanni, K.A. Feenstra, W. Fokkink, E. Krepska, *What can formal methods bring to systems biology?* Lecture Notes in Computer Science, **5850** (2009), 16–22.
41. L. Bortolussi, *Stochastic concurrent constraint programming*, Proc. 4^{th} Int. Workshop on Quantitative Aspects of Programming Languages (QAPL) 2006 (A. Di Pierro, H. Wiklicky, eds.), Vienna, Austria, Electronic Notes in Theoretical Computer Science, **164**:3 (2006), 65–80.
42. L. Bortolussi, *Constraint-based approaches to stochastic dynamics of biological systems*, Ph.D. thesis, Ph.D. Thesis Series, **CS2007/1**, University of Udine, Italy, 2007.
43. L. Bortolussi, S. Fonda, A. Policriti, *Constraint-based simulation of biological systems described by Molecular Interaction Maps*, Proc. Int. Conf. on Bioinformatics and Biomedicine (BIBM) 2007 (X. Hu, I. Mandoiu, Z. Obradovic, J. Xia, eds.), Fremont, CA, USA, 288–293, IEEE Computer Society Press, 2007.
44. L. Bortolussi, V. Galpin, J. Hillston, *HYPE with stochastic events*, Proc. 9^{th} Int. Workshop on Quantitative Aspects of Programming Languages and Systems (QAPL) 2011 (M. Massink, G. Norman, eds.), Saarbrücken, Germany, Electronic Proceedings in Theoretical Computer Science, **57** (2011), 120–133.
45. L. Bortolussi, J. Hillston, M. Loreti, *Fluid approximation of broadcasting systems*, Theoretical Computer Science, **816** (2020), 221–248.
46. L. Bortolussi, R. De Nicola, V. Galpin, S. Gilmore, J. Hillston, D. Latella, M. Loreti, M. Massink, *CARMA: Collective Adaptive Resource-sharing Markovian Agents*, Proc. 13^{th} Workshop on Quantitative Aspects of Programming Languages and Systems (QAPL) 2015 (N. Bertrand, M. Tribastone, eds.), London, UK, Electronic Proceedings in Theoretical Computer Science, **194** (2015), 16–31.

47. L. Bortolussi, A. Policriti, *Modeling biological systems in concurrent constraint programming*, Proc. 2^{nd} Int. Workshop on Constraint Based Methods in Bioinformatics (WCB) 2006 (A. Dovier, A. Dal Palù, S. Will, eds.), Nantes, France, 6–29, 2006.
48. L. Bortolussi, A. Policriti, *Stochastic concurrent constraint programming and differential equations*, Proc. 5^{th} Workshop on Quantitative Aspects of Programming Languages (QAPL) 2007 (A. Aldini, F. van Breugel, eds.), Braga, Portugal, Electronic Notes in Theoretical Computer Science, **190**:3 (2007), 27–42.
49. L. Bortolussi, A. Policriti, *Modeling biological systems in stochastic concurrent constraint programming*, Constraints, **13**:1–2 (2008), 66–90.
50. L. Bortolussi, A. Policriti, *Hybrid systems and biology: continuous and discrete modeling for systems biology*, Lecture Notes in Computer Science, **5016** (2008), 424–448.
51. L. Bortolussi, A. Policriti, *Hybrid dynamics of stochastic programs*, Theoretical Computer Science, **411** (2010), 2052–2077.
52. L. Bortolussi, M.G. Vigliotti, *CoBiC: context-dependent bioambient calculus*, Proc. 7^{th} Workshop on Quantitative Aspects of Programming Languages (QAPL) 2009 (C. Baier, A. di Pierro, eds.), York, UK, Electronic Notes in Theoretical Computer Science, **253**:3 (2009), 187–201.
53. J.T. Bradley, *Semi-Markov PEPA: a contradiction in terms?* Proc. 2^{nd} Workshop on Process Algebras and Stochastically Timed Activities (PASTA) 2003 (S.T. Gilmore, ed.), 1–6, LFCS, Edinburgh, UK, 2003.
54. J.T. Bradley, *Semi-Markov PEPA: compositional modelling and analysis with generally distributed actions*, Proc. 20^{th} Annual UK Performance Engineering Workshop (UKPEW) 2004 (I. Awan, ed.), 266–275, University of Bradford, UK, 2004.
55. J.T. Bradley, *Semi-Markov PEPA: modelling with generally distributed actions*, International Journal of Simulation: Systems, Science and Technology, **6**:3–4 (2005), 43–51.
56. M. Bravetti, *Specification and analysis of stochastic real-time systems*, Ph.D. thesis, University of Bologna, Italy, 2002.
57. M. Bravetti, P.R. D'Argenio, *Tutte le algebre insieme: concepts, discussions and relations of stochastic process algebras with general distributions*, Lecture Notes in Computer Science, **2925** (2004), 44–88.
58. M. Bravetti, M. Bernardo, R. Gorrieri, *Towards performance evaluation with general distributions in process algebras*, Lecture Notes in Computer Science, **1466** (1998), 405–422. MR1683349
59. M. Bravetti, R. Gorrieri, *The theory of interactive generalized semi-Markov processes*, Theoretical Computer Science, **282**:1 (2002), 5–32.
60. E. Brinksma, J.-P. Katoen, R. Langerak, D. Latella, *A stochastic causality-based process algebra*, The Computer Journal, **38**:7 (1995), 552–565.
61. L. Brodo, P. Degano, C. Priami, *A stochastic semantics for BioAmbients*, Lecture Notes in Computer Science, **4671** (2007), 22–34.
62. G. Bucci, L. Sassoli, E. Vicario, *Correctness verification and performance analysis of real-time systems using stochastic preemptive time Petri nets.* IEEE Transactions on Software Engineering, **31**:11 (2005), 913–927.
63. P. Buchholz, *On a Markovian process algebra*, Forschungsbericht, **500** (1994), Fachbereich Informatik, Technische Universität Dortmund, Germany.
64. P. Buchholz, *Markovian process algebra: composition and equivalence*, Proc. 2^{nd} Int. Workshop on Process Algebras and Performance Modelling (PAPM) 1994 (U.

Herzog, M. Rettelbach, eds.), Regensberg / Erlangen, Germany, Arbeitsberichte des IMMD, **27**:4 (1994), 11–30.
65. P. Buchholz, *A notion of equivalence for stochastic Petri nets*, Lecture Notes in Computer Science, **935** (1995), 161–180. MR1461026
66. P. Buchholz, *Iterative decomposition and aggregation of labeled GSPNs*, Lecture Notes in Computer Science, **1420** (1998), 226–245.
67. P. Buchholz, P. Kemper, *Quantifying the dynamic behavior of process algebras*, Lecture Notes in Computer Science, **2165** (2001), 184–199.
68. P. Buchholz, I.V. Tarasyuk, *Net and algebraic approaches to probabilistic modeling*, Joint Novosibirsk Computing Center and Institute of Informatics Systems Bulletin, Series Computer Science, **15** (2001), 31–64. Zbl 1004.68112
69. Y. Butkova, A. Hartmanns, H. Hermanns, *A Modest approach to modelling and checking Markov automata*, Lecture Notes in Computer Science, **11785** (2019), 52–69.
70. G. Caravagna, J. Hillston, *Bio-PEPAd: a non-Markovian extension of Bio-PEPA*, Theoretical Computer Science, **419** (2012), 26–49.
71. L. Cardelli, *Brane calculi*, Lecture Notes in Computer Science, **3082** (2005), 257–278.
72. L. Cardelli, *On process rate semantics*, Theoretical Computer Science, **391**:3 (2008), 190–215.
73. L. Cardelli, *From processes to ODEs by chemistry*, Proc. 5^{th} IFIP Int. Conf. on Theoretical Computer Science (TCS) 2008, IFIP 20^{th} World Computer Congress (WCC) 2008, TC1, Foundations of Computer Science (G. Ausiello, J. Karhumäki, G. Mauri, L. Ong, eds.), Milan, Italy, IFIP International Federation for Information Processing (IFIPAICT), **273** (2008), 261–281.
74. L. Cardelli, *Strand algebras for DNA computing*, Natural Computing, **10** (2011), 407–428.
75. L. Cardelli, R. Mardare, *The measurable space of stochastic processes*, Proc. 7^{th} Int. Conf. on the Quantitative Evaluation of Systems (QEST) 2010, Williamsburg, VA, USA, 171–180, IEEE Computer Society Press, 2010.
76. L. Cardelli, R. Mardare, *The measurable space of stochastic processes*, Fundamenta Informaticae, **131**:3–4 (2014), 351–371.
77. L. Cardelli, G. Zavattaro, *On the computational power of biochemistry*, Lecture Notes in Computer Science, **5147** (2008), 65–80.
78. N. Chamseddine, M. Duflot, L. Fribourg, C. Picaronny, J. Sproston, *Computing expected absorption times for parametric determinate probabilistic timed automata*, Proc. 5^{th} International Conf. on Quantitative Evaluation of Systems (QEST) 2008, St. Malo, France, 254–263, IEEE Computer Society Press, 2008.
79. G. Chiola, *A software package for the analysis of generalized stochastic Petri net models*, Proc. 1^{st} Int. Workshop on Timed Petri Nets 1985, Turin, Italy, IEEE Computer Society Press, 1985.
80. G. Ciardo, *Discrete-time Markovian stochastic Petri nets*, Computations with Markov Chains: Proc. 2^{nd} Int. Workshop on the Numerical Solution of Markov Chains (NSMC) 1995 (W.J. Stewart, ed.), Raleigh, NC, USA, 339–358, Kluwer Academic Publishers, Boston, MA, USA, 1995. Zbl 0862.60079
81. G. Ciardo, J.K. Muppala, K.S. Trivedi, *SPNP: stochastic Petri net package*, Proc. 3^{rd} Int. Workshop on Petri Nets and Performance Models (PNPM) 1989, Kyoto, Japan, 142–151, IEEE Computer Society Press, 1989.
82. G. Ciardo, J.K. Muppala, K.S. Trivedi, *On the solution of GSPN reward models*, Performance Evaluation, **12**:4 (1991), 237–253. Zbl 0754.60097

83. G. Ciobanu, *Analyzing non-Markovian systems by using a stochastic process calculus and a probabilistic model checker*, Mathematics, **11** (2023), Article 302.
84. G. Ciobanu, A.S. Rotaru, *PHASE: a stochastic formalism for phase-type distributions*, Lecture Notes in Computer Science, **8829** (2014), 91–106.
85. G. Ciobanu, A.S. Rotaru, *Phase-type approximations for non-Markovian systems: a case study*, Lecture Notes in Computer Science, **8938** (2015), 323–334.
86. F. Ciocchetta, M.L. Guerriero, *Modelling biological compartments in Bio-PEPA*, Proc. 2^{nd} Int. Meeting on Membrane Computing and Biologically Inspired Process Calculi (MeCBIC) 2008 (G. Ciobanu, ed.), Iasi, Romania, Electronic Notes in Theoretical Computer Science, **227** (2009), 77–95.
87. F. Ciocchetta, J. Hillston, *Bio-PEPA: an extension of the process algebra PEPA for biochemical networks*, Proc. 1^{st} Workshop From Biology To Concurrency and back (FBTC) 2007 (N. Cannata, E. Merelli, eds.), Lisbon, Portugal, Electronic Notes in Theoretical Computer Science, **194**:3 (2008), 103–117.
88. F. Ciocchetta, J. Hillston, *Process algebras in systems biology*, Lecture Notes in Computer Science, **5016** (2008), 265–312.
89. F. Ciocchetta, J. Hillston, *Bio-PEPA: a framework for the modelling and analysis of biochemical networks*, Theoretical Computer Science, **410**:33–34 (2009), 3065–3084.
90. G. Clark, S. Gilmore, *State-aware performance analysis with eXtended Stochastic Probes*, Lecture Notes in Computer Science, **5261** (2008), 125–140.
91. G. Clark, S. Gilmore, M.L. Guerriero, J. Hillston, *Conservation of mass analysis for Bio-PEPA*, Proc. 6^{th} Int. Workshop on Practical Applications of Stochastic Modelling (PASM) 2012 and 11^{th} Int. Workshop on Parallel and Distributed Methods in Verification (PDMC) 2012 (J. Bradley, K. Heljanko, W. Knottenbelt, N. Thomas, eds.), London, UK, Electronic Notes in Theoretical Computer Science, **296** (2013), 107–126.
92. G. Clark, S. Gilmore, J. Hillston, M. Tribastone, *Stochastic process algebras*, Lecture Notes in Computer Science, **4486** (2007), 132–179.
93. M. Coppo, F. Damiani, M. Drocco, E. Grassi, A. Troina, *Stochastic calculus of wrapped compatnents*, Proc. 8^{th} Int. Workshop on Quantitative Aspects of Programming Languages (QAPL) 2010 (A. Di Pierro, G. Norman, eds.), Electronic Proceedings in Theoretical Computer Science, **28** (2010), 82–98.
94. M. Coppo, F. Damiani, M. Drocco, E. Grassi, M. Guether, A. Troina, *Modelling ammonium transporters in Arbuscular Mycorrhiza symbiosis*, Lecture Notes in Computer Science, **6575** (2011), 85–109.
95. N. Coste, H. Hermanns, E. Lantreibecq, W. Serwe, *Towards performance prediction of compositional models in industrial GALS designs*, Lecture Notes in Computer Science, **5643** (2009), 204–218.
96. A. Credi, M. Garavelli, C. Laneve, S. Pradalier, S. Silvi, G. Zavattaro, *Modelization and simulation of nano devices in nanoκ calculus*, Lecture Notes in Computer Science, **4695** (2007), 168–183.
97. A. Credi, M. Garavelli, C. Laneve, S. Pradalier, S. Silvi, G. Zavattaro, *nanoκ: a calculus for the modeling and simulation of nano devices*, Theoretical Computer Science, **408**:1 (2008), 17–30.
98. M. Cubuktepe, N. Jansen, S. Junges, J.-P. Katoen, U. Topcu, *Scenario-based verification of uncertain MDPs*, Lecture Notes in Computer Science, **12078** (2020), 287–305.
99. A. Cumani, *ESP — a package for the evaluation of stochastic Petri nets with phase-type distributed transition times*, Proc. 1^{st} Int. Workshop on Timed Petri Nets 1985, Turin, Italy, 144–151, IEEE Computer Society Press, 1985.

100. V. Danos, J. Feret, W. Fontana, R. Harmer, J. Krivine, *Rule-based modelling of cellular signalling*, Lecture Notes in Computer Science, **4703** (2007), 17–41.
101. V. Danos, J. Feret, W. Fontana, J. Krivine, *Scalable simulation of cellular signaling networks*, Lecture Notes in Computer Science, **4807** (2007), 139–157.
102. V. Danos, J. Feret, W. Fontana, J. Krivine, *Abstract interpretation of cellular signalling networks*, Lecture Notes in Computer Science, **4905** 2008, 83–97.
103. C. Daws, *Symbolic and parametric model checking of discrete-time Markov chains*, Lecture Notes in Computer Science, **3407** (2005), 280–294.
104. P. Degano, D. Prandi, C. Priami, P. Quaglia, *Beta-binders for biological quantitative experiments*, Proc. 4^{th} Int. Workshop on Quantitative Aspects of Programming Languages (QAPL) 2006 (A. Di Pierro, H. Wiklicky, eds.), Vienna, Austria, Electronic Notes in Theoretical Computer Science, **164**:3 (2006), 101–117.
105. P. Degano, C. Priami, *Non-interleaving semantics for mobile processes*, Theoretical Computer Science, **216**:1–2 (1999), 237–270.
106. A. Degasperi, *Multi-scale modelling of biological systems in process algebra*, Ph.D. thesis, School of Computing Science, College of Science and Engineering, University of Glasgow, UK, 2011.
107. A. Degasperi, M. Calder, *Multi-scale modelling of biological systems in process algebra with multi-way synchronisation*, Proc. 9^{th} Int. Conf. on Computational Methods in Systems Biology (CMSB) 2011, Paris, France, 195–208, ACM Press, 2011.
108. A. Degasperi, M. Calder, *A process algebra framework for multi-scale modelling of biological systems*, Theoretical Computer Science, **488** (2013), 15–45.
109. B. Delahaye, D. Lime, L. Petrucci, *Parameter synthesis for parametric interval Markov chains*, Lecture Notes in Computer Science, **9583** (2016), 372–390.
110. L. Dematté, R. Larcher, A. Palmisano, C. Priami, A. Romanel, *Programming biology in BlenX*, Systems Biology for Signaling Networks (S. Choi, ed.), Chapter **31**, 777–820, Systems Biology Series, Volume **1**, Springer, New York, NY, USA, 2010.
111. L. Dematté, C. Priami, A. Romanel, *Modelling and simulation of biological processes in BlenX*, ACM SIGMETRICS Performance Evaluation Review, **35**:4 (2008), 32–39.
112. L. Dematté, C. Priami, A. Romanel, *The BetaWorkbench: a computational tool to study the dynamics of biological systems*, Briefings In Bioinformatics, **9**:5 (2008), 437–449.
113. L. Dematté, C. Priami, A. Romanel, *The BlenX language: a tutorial*, Lecture Notes in Computer Science, **5016** (2008), 313–365.
114. Y. Deng, M.C.B. Hennessy, *On the semantics of Markov automata*, Lecture Notes in Computer Science, **6756** (2011), 307–318.
115. Y. Deng, M.C.B. Hennessy, *On the semantics of Markov automata*, Information and Computation, **222** (2013), 139–168.
116. M. Dezani-Ciancaglini, P. Giannini, A. Troina, *A type system for a stochastic CLS*, Proc. 3^{rd} Workshop on Membrane Computing and Biologically Inspired Process Calculi (MeCBIC) 2009 (G. Ciobanu, ed.), Bologna, Italy, Electronic Proceedings in Theoretical Computer Science, **11** (2009), 91–105.
117. Ch. Eisentraut, H. Hermanns, L. Zhang, *On probabilistic automata in continuous time*, Proc. 25^{th} Annual IEEE Symposium on Logic in Computer Science (LICS) 2010, Edinburgh, UK, 342–351, IEEE Computer Society Press, 2010.

118. A. El-Rayes, M. Kwiatkowska, G. Norman, *Solving infinite stochastic process algebra models through matrix-geometric methods*, Proc. 7^{th} Int. Workshop on Process Algebra and Performance Modelling (PAPM) 1999 (J. Hillston, M. Silva, eds.), Zaragoza, Spain, 41–62, Prensas Universitarias de Zaragoza, Spain, 1999.
119. J.R. Faeder, M.L. Blinov, B. Goldstein, W.S. Hlavacek, *Combinatorial complexity and dynamical restriction of network flows in signal transduction*, Systems Biology, **2**:1 (2005), 5–15.
120. J.R. Faeder, M.L. Blinov, W.S. Hlavacek, *Graphical rule-based representation of signal-transduction networks*, Proc. ACM Symposium on Applied Computing (SAC) 2005, Santa Fe, New Mexico, USA, 133–140, ACM Press, 2005.
121. Ch. Feng, J. Hillston, *PALOMA: a process algebra for located Markovian agents*, Lecture Notes in Computer Science, **8657** (2014), 265–280.
122. Ch. Feng, J. Hillston, *Speed-up of stochastic simulation of PCTMC models by statistical model reduction*, Lecture Notes in Computer Science, **9272** (2015), 291–305.
123. J.M. Fourneau, *Collaboration of discrete-time Markov chains: tensor and product form*, Performance Evaluation, **67** (2010), 779–796.
124. M.A. Gallegos-Herrada, D. Ledvinka, J.S. Rosenthal, *Equivalences of geometric ergodicity of Markov chains*, Journal of Theoretical Probability (2023).
125. V. Galpin, *Equivalences for a biological process algebra*, Theoretical Computer Science, **412**:43 (2011), 6058–6082.
126. V. Galpin, *Modelling trafficking of proteins within the mammalian cell using Bio-PEPA*, Lecture Notes in Computer Science, **7605** (2012), 374–377.
127. V. Galpin, *Hybrid semantics for Bio-PEPA*, Information and Computation, **236** (2014), 122–145.
128. V. Galpin, L. Bortolussi, J. Hillston, *HYPE: a process algebra for compositional flows and emergent behaviour*, Lecture Notes in Computer Science, **5710** (2009), 305–320.
129. V. Galpin, J. Hillston, *Equivalence and discretisation in Bio-PEPA*, Lecture Notes in Computer Science, **5688** (2009), 189–204.
130. V. Galpin, J. Hillston, *A semantic equivalence for Bio-PEPA based on discretisation of continuous values*, Theoretical Computer Science, **412**:21 (2011), 2142–2161.
131. V. Galpin, J. Hillston, L. Bortolussi, *HYPE applied to the modelling of hybrid biological systems*, Proc. 24^{th} Conf. on the Mathematical Foundations of Programming Semantics (MFPS XXIV) 2008 (A. Bauer, M. Mislove, eds.), Philadelphia, PA, USA, Electronic Notes in Theoretical Computer Science, **218** (2008), 33–51.
132. V. Galpin, N. Zoń, P. Wilsdorf, S. Gilmore, *Mesoscopic modelling of pedestrian movement using Carma and its tools*, ACM Transactions on Modeling and Computer Simulation (TOMACS), **28**:2 (2018), Article 11, 11:2–11:26.
133. N. Geisweiller, J. Hillston, M. Stenico, *Relating continuous and discrete PEPA models of signalling pathways*, Theoretical Computer Science, **404**:1–2 (2008), 97–111.
134. A. Georgoulas, J. Hillston, D. Milios, G. Sanguinetti, *Probabilistic programming process algebra*, Lecture Notes in Computer Science, **8657** (2014), 249–264.
135. A. Georgoulas, J. Hillston, G. Sanguinetti, *ProPPA: probabilistic programming for stochastic dynamical systems*, ACM Transactions on Modeling and Computer Simulation (TOMACS), **28**:1 (2018), Article 3, 3:1–3:23.

136. D. Gilbert, M. Heiner, S. Lehrack, *A unifying framework for modelling and analysing biochemical pathways using Petri nets*, Lecture Notes in Computer Science, **4695** (2007), 200–216.
137. D.T. Gillespie, *Exact stochastic simulation of coupled chemical reactions*, Journal of Physical Chemistry, **81**:25 (1977), 2340–2361.
138. D.T. Gillespie, *Approximate accelerated stochastic simulation of chemically reacting systems*, Journal of Chemical Physics, **115**:4 (2001), 1716–1733.
139. S. Gilmore, J. Hillston, L. Kloul, M. Ribaudo, *PEPA nets: a structured performance modelling formalism*, Performance Evaluation, **54**:2 (2003), 79–104.
140. R.J. van Glabbeek, S.A. Smolka, B. Steffen, *Reactive, generative, and stratified models of probabilistic processes*, Information and Computation, **121**:1 (1995), 59–80. Zbl 0832.68042
141. R. Gori, F. Levi, *An analysis for proving probabilistic termination of biological systems*, Theoretical Computer Science, **471** (2013), 27–73.
142. N. Götz, U. Herzog, M. Rettelbach, *Multiprocessor and distributed system design: the integration of functional specification and performance analysis using stochastic process algebras*, Lecture Notes in Computer Science, **729** (1993), 121–146.
143. D. Guck, H. Hatefi, H. Hermanns, J.-P. Katoen, M. Timmer, *Modelling, reduction and analysis of Markov automata*, Lecture Notes in Computer Science, **8054** (2013), 55–71.
144. D. Guck, H. Hatefi, H. Hermanns, J.-P. Katoen, M. Timmer, *Analysis of timed and long-run objectives for Markov automata*, Logical Methods in Computer Science, **10**:3:17 (2014), 1–29.
145. M.C. Guenther, J.T. Bradley, *Higher moment analysis of a spatial stochastic process algebra*, Lecture Notes in Computer Science, **6977** (2011), 87–101.
146. M.C. Guenther, A. Stefanek, J.T. Bradley, *Moment closures for performance models with highly non-linear rates*, Lecture Notes in Computer Science, **7587** (2013), 32–47.
147. M.L. Guerriero, A. Pokhilko, A.P. Fernández, K.J. Halliday, A.J. Millar, J. Hillston, *Stochastic properties of the plant circadian clock*, Journal of the Royal Society Interface, **9**:69 (2012), 744–756.
148. M.L. Guerriero, D. Prandi, C. Priami, P. Quaglia, *Process calculi abstractions for biology*, Algorithmic Bioprocesses, Chapter **3.3**, 463–486 (A. Condon, D. Harel, J.N. Kok, A. Salomaa, E. Winfree, eds.), Natural Computing Series (NCS), Springer, Berlin, Heidelberg, Germany, 2009.
149. E.M. Hahn, A. Hartmanns, H. Hermanns, J.-P. Katoen, *A compositional modelling and analysis framework for stochastic hybrid systems*, Formal Methods in System Design, **43**:2 (2013), 191–232.
150. E.M. Hahn, H. Hermanns, L. Zhang, *Probabilistic reachability for parametric Markov models*, International Journal on Software Tools for Technology Transfer, **13**:1 (2011), 3–19.
151. T. Han, J.-P. Katoen, A. Mereacre, *Approximate parameter synthesis for probabilistic time-bounded reachability*, Proc. 29^{th} IEEE Real-Time Systems Symposium (RTSS) 2008, New York, USA, 173–182, IEEE Computer Society Press, 2008.
152. H.M. Hanish, *Analysis of place/transition nets with timed-arcs and its application to batch process control*, Lecture Notes in Computer Science, **691** (1993), 282–299.

153. P.G. Harrison, B. Strulo, *Stochastic process algebra for discrete event simulation*, Quantitative Methods in Parallel Systems, Esprit Basic Research Series, 18–37 (F. Bacelli, A. Jean-Marie, I. Mitrani, eds.), Springer, Berlin, Germany, 1995.
154. A. Hartmanns, *An overview of Modest models and tools for real stochastic timed systems*, Proc. 5^{th} Workshop on Models for Formal Analysis of Real Systems (MARS) 2022 (C. Dubslaff, B. Luttik, eds.), Electronic Proceedings in Theoretical Computer Science, **355** (2022), 1–12.
155. V. Hashemi, H. Hermanns, L. Song, *Reward-bounded reachability probability for uncertain weighted MDPs*, Lecture Notes in Computer Science, **9583** (2016), 351–371.
156. H. Hatefi, H. Hermanns, *Improving time bounded reachability computations in interactive Markov chains*, Lecture Notes in Computer Science, **8161** (2013), 250–266.
157. R.A. Hayden, J.T. Bradley, *A fluid analysis framework for a Markovian process algebra*, Theoretical Computer Science, **411** (2010), 2260–2297.
158. R.A. Hayden, J.T. Bradley, A. Clark, *Performance specification and evaluation with unified stochastic probes and fluid analysis*, IEEE Transactions on Software Engineering, **39**:1 (2013), 97–118.
159. H. Hermanns, *Interactive Markov chains: the quest for quantified quality*, Lecture Notes in Computer Science, **2428** (2002), 1–217. Zbl 1012.68142
160. H. Hermanns, U. Herzog, J.-P. Katoen, *Process algebra for performance evaluation*, Theoretical Computer Science, **274**:1–2 (2002), 43–87.
161. H. Hermanns, J.-P. Katoen, *Automated compositional Markov chain generation for a plain-old telephone system*, Science of Computer Programming, **36**:1 (2000), 97–127.
162. H. Hermanns, M. Lohrey, *Observational congruence in stochastic timed calculus with maximal progress*, Technischer Bericht, **IMMD VII-7/97** (1997), Institut für Matematische Maschinen und Datenverarbeitung (IMMD), Friedrich-Alexander Universität Erlangen-Nürnberg (FAU), Erlangen, Germany.
163. H. Hermanns, M. Rettelbach, *Syntax, semantics, equivalences and axioms for MTIPP*, Proc. 2^{nd} Int. Workshop on Process Algebras and Performance Modelling (PAPM) 1994 (U. Herzog, M. Rettelbach, eds.), Regensberg / Erlangen, Germany, Arbeitsberichte des IMMD, **27**:4 (1994), 71–88.
164. J. Hillston, *A compositional approach to performance modelling*, Ph.D. thesis, Department of Computer Science, University of Edinburgh, UK, 1994.
165. J. Hillston, *The nature of synchronisation*, Proc. 2^{nd} Int. Workshop on Process Algebra and Performance Modelling (PAPM) 1994 (U. Herzog, M. Rettelbach, eds.), Regensberg / Erlangen, Germany, Arbeitsberichte des IMMD, **27**:4 (1994), 51–70.
166. J. Hillston, *A compositional approach to performance modelling*, Cambridge University Press, Cambridge, UK, 1996. Zbl 1080.68003
167. J. Hillston, *Fluid flow approximation of PEPA models*, Proc. 2^{nd} Int. Conf. on the Quantitative Evaluation of Systems (QEST) 2005, Turin, Italy, 33–43, IEEE Computer Society Press, 2005.
168. J. Hillston, *Stochastic process algebras and their Markovian semantics*, ACM SIGLOG News, **5**:2 (2018), 20–35.
169. C.A.R. Hoare, *Communicating sequential processes*, Prentice-Hall, London, UK, 1985.
170. A. Horváth, M. Paolieri, L. Ridi, E. Vicario, *Transient analysis of non-Markovian models using stochastic state classes*, Performance Evaluation, **69**:7–8 (2012), 315–335.

171. A. Horváth, M. Paolieri, E. Vicario, *Approximating distributions and transient probabilities by matrix exponential distributions and functions*, Quantitative Assessments of Distributed Systems: Methodologies and Techniques (D. Bruneo, S. Distefano, eds.), Chapter **5**, 107–127, Performability Engineering Series, Scrivener Publishing LLC, Beverly, MA, USA, 2015.
172. A. Horváth, A. Puliafito, M. Scarpa, M. Telek, *Analysis and evaluation of non-Markovian stochastic Petri nets*, Lecture Notes in Computer Science, **1786** (2000), 171–187. Zbl 0967.68114
173. G. Horváth, E. Vicario, *Construction of phase type distributions by Bernstein exponentials*, Lecture Notes in Computer Science, **14231** (2023), 201–215.
174. G. Iacobelli, M. Tribastone, A. Vandin, *Differential bisimulation for a Markovian process algebra*, Lecture Notes in Computer Science, **9234** (2015), 293–306. Zbl 06482743
175. N. Jansen, F. Corzilius, M. Volk, R. Wimmer, E. Ábráham, J.-P. Katoen, B. Becker,
Accelerating parametric probabilistic verification, Lecture Notes in Computer Science, **8657** (2014), 404–420.
176. M. John, C. Lhoussaine, J. Niehren, A.M. Uhrmacher, *The attributed pi calculus*, Lecture Notes in Computer Science, **5307** (2008), 83–102.
177. R.L. Jones, G. Ciardo, *On phased delay stochastic Petri nets*, Proc. 9^{th} Int. Workshop on Petri Nets and Performance Models (PNPM) 2001, Aachen, Germany, 165–174, IEEE Computer Society Press, 2001.
178. S. Junges, E. Ábrahám, Ch. Hensel, N. Jansen, J.-P. Katoen, T. Quatmann, M. Volk, *Parameter synthesis for Markov models: covering the parameter space*, Formal Methods in System Design (2024).
179. J.-P. Katoen, *Quantitative and qualitative extensions of event structures*, Ph.D. thesis, CTIT Ph.D.-thesis series, **96-09**, Centre for Telematics and Information Technology, University of Twente, Enschede, The Netherlands, 1996.
180. J.-P. Katoen, P.R. D'Argenio, *General distributions in process algebra*, Lecture Notes in Computer Science, **2090** (2001), 375–429.
181. J.-P. Katoen, E. Brinksma, D. Latella, R. Langerak, *Stochastic simulation of event structures*, Proc. 4^{th} Int. Workshop on Process Algebra and Performance Modelling (PAPM) 1996 (M. Ribaudo, ed.), Torino, Italy, 21–40, CLUT Press, 1996.
182. B. Klin, V. Sassone, *Structural operational semantics for stochastic process calculi*, Lecture Notes in Computer Science, **4962** (2008), 428–442.
183. M. Koutny, *A compositional model of time Petri nets*, Lecture Notes in Computer Science, **1825** (2000), 303–322.
184. J. Krivine, *Systems biology*, ACM SIGLOG News, **4**:3 (2017), 43–61.
185. J. Krivine, R. Milner, A. Troina, *Stochastic bigraphs*, Proc. 24^{th} Conf. on the Mathematical Foundations of Programming Semantics (MFPS XXIV) 2008 (A. Bauer, M. Mislove, eds.), Philadelphia, PA, USA, Electronic Notes in Theoretical Computer Science, **218** (2008), 73–96.
186. C. Kuttler, C. Lhoussaine, J. Niehren, *A stochastic Pi calculus for concurrent objects*, Lecture Notes in Computer Science, **4545** (2007), 232–246.
187. L. Lakatos, L. Szeidl, M. Telek, *Introduction to queueing systems with telecommunication applications*, Springer Nature, Cham, Switzerland, 2019.
188. M.R. Lakin, L. Paulevé, A. Phillips, *Stochastic simulation of multiple process calculi for biology*, Theoretical Computer Science, **431** (2012), 181–206.

189. M.R. Lakin, A. Phillips, *Modelling, simulating and verifying Turing-powerful strand displacement systems*, Lecture Notes in Computer Science, **6937** (2011), 130–144.
190. M.R. Lakin, S. Youssef, L. Cardelli, A. Phillips, *Abstractions for DNA circuit design*, Journal of the Royal Society Interface, **9**:68 (2012), 470–486.
191. C. Laneve, S. Pradalier, G. Zavattaro, *From biochemistry to stochastic processes*, Proc. 7^{th} Workshop on Quantitative Aspects of Programming Languages (QAPL) 2009 (C. Baier, A. di Pierro, eds.), York, UK, Electronic Notes in Theoretical Computer Science, **253**:3 (2009), 167–185.
192. R. Lanotte, A. Maggiolo-Schettini, A. Troina, *Parametric probabilistic transition systems for system design and analysis*, Formal Aspects of Computing, **19**:1 (2007), 93–109.
193. B.N. López, G.M. Núñez, *NMSPA: a non-Markovian model for stochastic processes*, Proc. Int. ICDCS Workshop on Distributed System Validation and Verification (DSVV) 2000 (T.-H. Lai, ed.), Taipei, Taiwan, China, 33–40, 2000.
194. B.N. López, G.M. Núñez, F. Rubio, *Stochastic process algebras meet Eden*, Lecture Notes in Computer Science, **2335** (2002), 29–48.
195. B.N. López, G.M. Núñez, F. Rubio, *An integrated framework for the performance analysis of asynchronous communicating stochastic processes*, Formal Aspects of Computing, **16**:3 (2004), 238–262.
196. M. Loreti, J. Hillston, *Modelling and analysis of collective adaptive systems with CARMA and its tools*, Lecture Notes in Computer Science, **9700** (2016), 83–119.
197. H. Macià, V. Valero, D.C. Cazorla, F. Cuartero, *Introducing the iteration in sPBC*, Lecture Notes in Computer Science, **3235** (2004), 292–308. Zbl 1110.68420
198. H. Macià, V. Valero, F. Cuartero, D. de Frutos, *A congruence relation for sPBC*, Formal Methods in System Design, **32**:2 (2008), 85–128. Zbl 1138.68040
199. H. Macià, V. Valero, F. Cuartero, M.C. Ruiz, *sPBC: a Markovian extension of Petri box calculus with immediate multiactions*, Fundamenta Informaticae, **87**:3–4 (2008), 367–406. Zbl 1154.68092
200. H. Macià, V. Valero, F. Cuartero, M.C. Ruiz, I.V. Tarasyuk, *Modelling a video conference system with sPBC*, Applied Mathematics and Information Sciences **10**:2 (2016), 475–493.
201. H. Macià, V. Valero, D. de Frutos, *sPBC: a Markovian extension of finite Petri box calculus*, Proc. 9^{th} IEEE Int. Workshop on Petri Nets and Performance Models (PNPM) 2001, Aachen, Germany, 207–216, IEEE Computer Society Press, 2001.
202. J. Markovski, P.R. D'Argenio, J.C.M. Baeten, E.P. de Vink, *Reconciling real and stochastic time: the need for probabilistic refinement*, Formal Aspects of Computing, **24**:4–6 (2012), 497–518. MR2947264
203. J. Markovski, E.P. de Vink, *Extending timed process algebra with discrete stochastic time*, Lecture Notes of Computer Science, **5140** (2008), 268–283. Zbl 1170.68542
204. J. Markovski, E.P. de Vink, *Performance evaluation of distributed systems based on a discrete real- and stochastic-time process algebra*, Fundamenta Informaticae, **95**:1 (2009), 157–186. MR2590801
205. O. Marroquín, D. de Frutos, *TPBC: timed Petri box calculus*, Technical Report, Departamento de Sistemas Infofmáticos y Programación, Universidad Complutense de Madrid, Spain, 2000 (in Spanish).
206. O. Marroquín, D. de Frutos, *Extending the Petri box calculus with time*, Lecture Notes in Computer Science, **2075** (2001), 303–322. Zbl 0986.68082
207. M.A. Marsan, *Stochastic Petri nets: an elementary introduction*, Lecture Notes in Computer Science, **424** (1990), 1–29.

208. M.A. Marsan, G. Balbo, G. Conte, S. Donatelli, G. Franceschinis, *Modelling with generalised stochastic Petri nets*, Wiley Series in Parallel Computing, John Wiley and Sons, 1995. Zbl 0843.68080
209. M.A. Marsan, G. Chiola, *On Petri nets with deterministic and exponentially distributed firing times*, Lecture Notes in Computer Science, **266** (1987), 132–145.
210. M.A. Marsan, G. Chiola, A. Fumagalli, *Improving the efficiency of the analysis of DSPN models*, Lecture Notes in Computer Science, **424** (1990), 30–50.
211. M. Massink, M. Brambilla, D. Latella, M. Dorigo, M. Birattari, *On the use of Bio-PEPA for modelling and analysing collective behaviours in swarm robotics*, Swarm Intelligence, **7** (2013), 201–228.
212. M. Massink, D. Latella, *Fluid analysis of foraging ants*, Lecture Notes in Computer Science, **7274** (2012), 152–165.
213. M. Massink, D. Latella, A. Bracciali, M.D. Harrison, J. Hillston, *Scalable context-dependent analysis of emergency egress models*, Formal Aspects of Computing, **24**:2 (2012), 267–302.
214. C. Maus, M. John, M. Röhl, A.M. Uhrmacher, *Hierarchical modeling for computational biology*, Lecture Notes in Computer Science, **5016** (2008), 81–124.
215. Ch. McCaig, R. Norman, C. Shankland, *Process algebra models of population dynamics*, Lecture Notes in Computer Science, **5147** (2008), 139–155.
216. Ph.M. Merlin, D.J. Farber, *Recoverability of communication protocols: implications of a theoretical study*, IEEE Transactions on Communications, **24**:9 (1976), 1036–1043. Zbl 0362.68096
217. R.A.J. Milner, *Communication and concurrency*, Prentice-Hall, Upper Saddle River, NJ, USA, 1989. Zbl 0683.68008
218. R.A.J. Milner, J.G. Parrow, D.J. Walker, *A calculus of mobile processes, I*, Information and Computation, **100**:1 (1992), 1–40.
219. R.A.J. Milner, J.G. Parrow, D.J. Walker, *A calculus of mobile processes, II*, Information and Computation, **100**:1 (1992), 41–77.
220. I. Mitrani, A. Ost, M. Rettelbach, *TIPP and the spectral expansion method*, Quantitative Methods in Parallel Systems, Esprit Basic Research Series, 99–113 (F. Bacelli, A. Jean-Marie, I. Mitrani, eds.), Springer, Berlin, Germany, 1995.
221. M.K. Molloy, *On the integration of the throughput and delay measures in distributed processing models*, Ph.D. thesis, Report, **CSD-810-921** (1981), University of California, Los Angeles, CA, USA.
222. M.K. Molloy, *Performance analysis using stochastic Petri nets*, IEEE Transactions on Computing, **31**:9 (1982), 913–917.
223. M.K. Molloy, *Discrete time stochastic Petri nets*, IEEE Transactions on Software Engineering, **11**:4 (1985), 417–423. Zbl 0558.68053
224. U. Montanari, M. Pistore, D. Yankelevich, *Efficient minimization up to location equivalence*, Lecture Notes in Computer Science, **1058** (1996), 265–279.
225. M.F. Neuts, *Matrix-geometric solutions in stochastic models: an algorithmic approach*, Johns Hopkins Series in the Mathematical Sciences, **2**, Johns Hopkins University Press, Baltimore, MD, USA, 1981.
226. A. Niaouris, *An algebra of Petri nets with arc-based time restrictions*, Lecture Notes in Computer Science, **3407** (2005), 447–462. Zbl 1109.68076
227. A. Niaouris, M. Koutny, *An algebra of timed-arc Petri nets*, Technical Report, **CS-TR-895** (2005), School of Computer Science, University of Newcastle upon Tyne, UK.
228. R. De Nicola, G.L. Ferrari, R. Pugliese, *Klaim: a kernel language for agents interaction and mobility*, IEEE Transactions on Software Engineering, **24**:5 (1998), 315–330.

229. R. De Nicola, D. Latella, M. Loreti, M. Massink, *Quantitative analysis of distributed systems in StoKlaim: a tutorial*, Quantitative Assessments of Distributed Systems: Methodologies and Techniques (D. Bruneo, S. Distefano, eds.), Chapter **2**, 27–55, Performability Engineering Series, Scrivener Publishing LLC, Beverly, MA, USA, 2015.
230. R. De Nicola, J.-P. Katoen, D. Latella, M. Loreti, M. Massink, *Model checking mobile stochastic logic*, Theoretical Computer Science, **382**:1 (2007), 42–70.
231. R. De Nicola, J.-P. Katoen, D. Latella, M. Massink, *StoKlaim: a stochastic extension of Klaim*, Technical Report, **2006-TR-01** (2006), Consiglio Nazionale delle Ricerche, Istituto di Scienza e Tecnologie dell'Informazione 'A. Faedo', Italy.
232. R. De Nicola, D. Latella, M. Loreti, M. Massink, *A uniform definition of stochastic process calculi*, ACM Computing Surveys, **46**:1 (2013), Article 5.
233. L. Paulevé, S. Youssef, M.R. Lakin, A. Phillips, *A generic abstract machine for stochastic process calculi*, Proc. 8^{th} Int. Conf. on Computational Methods in Systems Biology (CMSB) 2010, Trento, Italy, 43–54, ACM Press, 2010.
234. Gh. Păun, F.J. Romero-Campero, *Membrane computing as a modeling framework. Cellular systems case studies*, Lecture Notes in Computer Science, **5016** (2008), 168–214.
235. M.R. Pedersen, G.D. Plotkin, *A language for biochemical systems*, Lecture Notes in Computer Science, **5307** (2008), 63–82.
236. M.R. Pedersen, G.D. Plotkin, *A language for biochemical systems: design and formal specifications*, Lecture Notes in Computer Science, **5945** (2010), 77–145.
237. Y. Peng, Sh. Wang, N. Zhan, L. Zhang, *Extending hybrid CSP with probability and stochasticity*, Lecture Notes in Computer Science, **9409** (2015), 87–102.
238. A. Phillips, *A visual process calculus for biology*, Symbolic Systems Biology: Theory and Methods (M.S. Iyengar, ed.), Chapter **5**, Jones and Bartlett Publishers, 2009.
239. A. Phillips, *An abstract machine for the stochastic bioambient calculus*, Proc. 2^{nd} Int. Meeting on Membrane Computing and Biologically Inspired Process Calculi (MeCBIC) 2008 (G. Ciobanu, ed.), Iasi, Romania, Electronic Notes in Theoretical Computer Science, **227** (2009), 143–159.
240. A. Phillips, L. Cardelli, *A correct abstract machine for the stochastic pi-calculus*, Proc. Workshop on Concurrent Models in Molecular Biology (BioConcur) 2004, Electronic Notes in Theoretical Computer Science, 2004.
241. A. Phillips, L. Cardelli, *Efficient, correct simulation of biological processes in the stochastic pi-calculus*, Lecture Notes in Computer Science, **4695** (2007), 184–199.
242. A. Phillips, L. Cardelli, *A programming language for composable DNA circuits*, Journal of the Royal Society Interface, **6**:Suppl4 (2009), S419–S436.
243. A. Phillips, L. Cardelli, G. Castagna, *A graphical representation for biological processes in the stochastic pi-calculus*, Lecture Notes in Computer Science, **4230** (2006), 123–152.
244. L.F. Pino, F. Boncho, F.D. Valencia, *A behavioural congruence for concurrent constraint programming with nondeterministic choice*, Lecture Notes in Computer Science, **8687** (2014), 351–368.
245. G.D. Plotkin, *A calculus of chemical systems*, Lecture Notes in Computer Science, **8000** (2013), 445–465.
246. C. Priami, *Stochastic π-calculus*, The Computer Journal, **38**:7 (1995), 578–589.
247. C. Priami, *Stochastic π-calculus with general distributions*, Proc. 4^{th} Int. Workshop on Process Algebra and Performance Modelling (PAPM) 1996 (M. Ribaudo, ed.), Torino, Italy, 41–57, CLUT Press, 1996.

248. C. Priami, *Language-based performance prediction for distributed and mobile systems*, Information and Computation, **175**:2 (2002), 119–145. MR1911524
249. C. Priami, P. Ballarini, P. Quaglia, *BlenX4Bio — BlenX for biologists*, Lecture Notes in Computer Science, **5688** (2009), 26–51.
250. C. Priami, P. Quaglia, *Beta-binders for biological interactions*, Lecture Notes in Computer Science, **3082** (2005), 20–33.
251. C. Priami, A. Regev, W. Silverman, E. Shapiro, *Application of a stochastic namepassing calculus to representation and simulation of molecular processes*, Information Processing Letters, **80**:1 (2001), 25–31.
252. R. Pulungan, H. Hermanns, *A construction and minimization service for continuous probability distributions*, International Journal on Software Tools for Technology Transfer, **17**:1 (2015), 77–90.
253. T. Quatmann, Ch. Dehnert, N. Jansen, S. Junges, J.-P. Katoen, *Parameter synthesis for Markov models: faster than ever*, Lecture Notes in Computer Science, **9938** (2016), 50–67.
254. Ch. Ramchandani, *Performance evaluation of asynchronous concurrent systems by timed Petri nets*, Ph.D. thesis, Department of Electrical Engineering, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA, 1973.
255. A. Regev, E.M. Panina, W. Silverman, L. Cardelli, E. Shapiro, *BioAmbients: an abstraction for biological compartments*, Theoretical Computer Science, **325**:1 (2004), 141–167.
256. M. Rettelbach, *Immediate transitions in stochastic process algebras — theory and application*, Ph.D. thesis, Universität Erlangen-Nürnberg, Germany, 1995 (in German).
257. M. Rettelbach, *Probabilistic branching in Markovian process algebras*, The Computer Journal, **38**:7 (1995), 590–599.
258. E. Sciacca, S. Spinella, C. Calcagno, F. Damiani, M. Coppo, *Parameter identification and assessment of nutrient transporters in AM symbiosis through stochastic simulations*, Proc. 3*rd* Int. Workshop on Interactions Between Computer Science and Biology (CS2Bio) 2012 (P. Giannini, E. de Vink, eds.), Stockholm, Sweden, Electronic Notes in Theoretical Computer Science, **293** (2013), 83–96.
259. E. Scott, A. Hoyle, C. Shankland, *PEPA'd oysters: converting dynamic energy budget models to Bio-PEPA, illustrated by a Pacific oyster case study*, Proc. 6^{th} Int. Workshop on Practical Applications of Stochastic Modelling (PASM) 2012 and 11^{th} Int. Workshop on Parallel and Distributed Methods in Verification (PDMC) 2012 (J. Bradley, K. Heljanko, W. Knottenbelt, N. Thomas, eds.), London, UK, Electronic Notes in Theoretical Computer Science, **296** (2013), 211–228.
260. W.J. Stewart, *Probability, Markov chains, queues, and simulation. The mathematical basis of performance modeling*, Princeton University Press, Princeton, NJ, USA, 2009.
261. I.V. Tarasyuk, *Discrete time stochastic Petri box calculus*, Berichte aus dem Department für Informatik, **3/05**, Carl von Ossietzky Universität Oldenburg, Germany, 2005.
262. I.V. Tarasyuk, *Iteration in discrete time stochastic Petri box calculus*, Bulletin of the Novosibirsk Computing Center, Series Computer Science, IIS Special Issue, **24** (2006), 129–148. Zbl 1249.68132
263. I.V. Tarasyuk, *Stochastic Petri box calculus with discrete time*, Fundamenta Informaticae, **76**:1–2 (2007), 189–218.

264. I.V. Tarasyuk, *Equivalences for behavioural analysis of concurrent and distributed computing systems*, Geo Academic Publisher, Novosibirsk, Russia, 2007 (in Russian).
265. I.V. Tarasyuk, *Equivalence relations for modular performance evaluation in dtsPBC*, Mathematical Structures in Computer Science, **24**:1 (2014), e240103.
266. I.V. Tarasyuk, *Discrete time stochastic and deterministic Petri box calculus*, arXiv: **1905.00456**, Computing Research Repository, Cornell University Library, Ithaca, NY, USA, 2019.
267. I.V. Tarasyuk, *Discrete time stochastic and deterministic Petri box calculus dtsdPBC*, Siberian Electronic Mathematical Reports, **17** (2020), 1598–1679. Zbl 1448.68352
268. I.V. Tarasyuk, *Performance evaluation in stochastic process algebra dtsdPBC*, Siberian Electronic Mathematical Reports, **18**:2 (2021), 1105–1145. Zbl 1482.68156
269. I.V. Tarasyuk, *Performance preserving equivalence for stochastic process algebra dtsdPBC*, Siberian Electronic Mathematical Reports, **20**:2 (2023), 646–699.
270. I.V. Tarasyuk, H. Macià, V. Valero, *Discrete time stochastic Petri box calculus with immediate multiactions*, Technical Report, **DIAB-10-03-1**, Department of Computer Systems, High School of Computer Science Engineering, University of Castilla - La Mancha, Albacete, Spain, 2010.
271. I.V. Tarasyuk, H. Macià, V. Valero, *Discrete time stochastic Petri box calculus with immediate multiactions dtsiPBC*, Proc. 6^{th} Int. Workshop on Practical Applications of Stochastic Modelling (PASM) 2012 and 11^{th} Int. Workshop on Parallel and Distributed Methods in Verification (PDMC) 2012 (J. Bradley, K. Heljanko, W. Knottenbelt, N. Thomas, eds.), London, UK, Electronic Notes in Theoretical Computer Science, **296** (2013), 229–252.
272. I.V. Tarasyuk, H. Macià, V. Valero, *Performance analysis of concurrent systems in algebra dtsiPBC*, Programming and Computer Software, **40**:5 (2014), 229–249. Zbl 1339.68033
273. I.V. Tarasyuk, H. Macià, V. Valero, *Stochastic process reduction for performance evaluation in dtsiPBC*, Siberian Electronic Mathematical Reports, **12** (2015), 513–551. Zbl 1346.60118
274. I.V. Tarasyuk, H. Macià, V. Valero, *Stochastic equivalence for performance analysis of concurrent systems in dtsiPBC*, Siberian Electronic Mathematical Reports, **15** (2018), 1743–1812. Zbl 1414.60062
275. M. Timmer, J.-P. Katoen, J. van de Pol, M.I.A. Stoelinga, *Efficient modelling and generation of Markov automata*, Lecture Notes in Computer Science, **7454** (2012), 364–379. Zbl 1364.68295
276. M. Timmer, J. van de Pol, M.I.A. Stoelinga, *Confluence reduction for Markov automata*, Lecture Notes in Computer Science, **8053** (2013), 243–257.
277. C. Tofts, *Processes with probabilities, priority and time*, Formal Aspects of Computing, **6**:5 (1994), 536–564.
278. C. Tofts, *Symbolic approaches to probability distributions in process algebra*, Formal Aspects of Computing, **12**:5 (2000), 392–415.
279. K.S. Trivedi, *Probability and statistics with reliability, queuing, and computer science applications*, John Wiley and Sons, Hoboken, NJ, USA, 2016.
280. K.S. Trivedi, A. Bobbio, *Reliability and availability engineering: modeling, ıanalysist and applications*, Cambridge University Press, Cambridge, UK, 2017.
281. M. Tschaikowski, M. Tribastone, *Exact fluid lumpability for Markovian process algebra*, Lecture Notes in Computer Science, **7454** (2012), 380–394. Zbl 1364.68297

282. M. Tschaikowski, M. Tribastone, *Exact fluid lumpability in Markovian process algebra*, Theoretical Computer Science, **538** (2014), 140–166. Zbl 1359.68228
283. M. Tschaikowski, M. Tribastone, *Extended differential aggregations in process algebra for performance and biology*, Proc. 12^{th} Int. Workshop on Quantitative Aspects of Programming Languages and Systems (QAPL) 2014, Grenoble, France, Electronic Proceedings in Theoretical Computer Science, **154** (2014), 34–47.
284. M. Tschaikowski, M. Tribastone, *A unified framework for differential aggregations in Markovian process algebra*, Journal of Logical and Algebraic Methods in Programming, **84** (2015), 238–258. Zbl 1319.68151
285. V. Valero, M.E. Cambronero, *Using unified modelling language to model the publish/subscribe paradigm in the context of timed Web services with distributed resources*, Mathematical and Computer Modelling of Dynamical Systems, **23**:6 (2017), 570–594.
286. C. Versari, N. Busi, *Stochastic simulation of biological systems with dynamical compartment structure*, Lecture Notes in Computer Science, **4695** (2007), 80–95.
287. C. Versari, N. Busi, *Efficient stochastic simulation of biological systems with multiple variable volumes*, Proc. 1^{st} Workshop From Biology To Concurrency and back (FBTC) 2007 (N. Cannata, E. Merelli, eds.), Lisbon, Portugal, Electronic Notes in Theoretical Computer Science, **194**:3 (2008), 165–180.
288. C. Versari, N. Busi, *Stochastic biological modelling in the presence of multiple compartments*, Theoretical Computer Science, **410** (2009), 3039–3064.
289. C. Versari, R. Gorrieri, *π@: a π-based process calculus for the implementation of compartmentalised bio-inspired calculi*, Lecture Notes in Computer Science, **5016** (2008), 449–506.
290. M.G. Vigliotti, *Operational semantics for product-form solution*, Lecture Notes in Computer Science, **7587** (2013), 16–31.
291. L.L. Vissat, J. Hillston, G. Marion, M.J. Smith, *MELA: modelling in ecology with location attributes*, Proc. 14^{th} Int. Workshop on Quantitative Aspects of Programming Languages and Systems (QAPL) 2016 (M. Tribastone, H. Wiklicky, eds.), Eindhoven, The Netherlands, Electronic Proceedings in Theoretical Computer Science, **227** (2016), 82–97.
292. D.Y.Q. Wang, L. Cardelli, A. Phillips, N. Piterman, J. Fisher, *Computational modeling of the EGFR network elucidates control mechanisms regulating signal dynamics*, BMC Systems Biology, **3** (2009), Article 118.
293. V. Wolf, *Equivalences on phase type processes*, Ph.D. thesis, University of Mannheim, Mannheim, Germany, 2008.
294. W. Yi, *Real-time behaviour of asynchronous agents*, Lecture Notes in Computer Science, **458** (1990), 502–520.
295. W. Yi, *CCS + time = an interleaving model for real time systems*, Lecture Notes in Computer Science, **510** (1991), 217–228.
296. G. Zavattaro, *A gentle introduction to Stochastic (Poly)Automata Collectives and the (Bio)Chemical Ground Form*, Lecture Notes in Computer Science, **5016** (2008), 507–523.
297. Ch. Zhou, J. Wang, A.P. Ravn, *A formal description of hybrid systems*, Lecture Notes in Computer Science, **1066** (1996), 511–530.
298. R. Zijal, *Discrete time deterministic and stochastic Petri nets*, Proc. Int. Workshop on Quality of Communication-Based Systems 1994, Technical University of Berlin, Germany, 123–136, Kluwer Academic Publishers, 1995. Zbl 0817.68111
299. R. Zijal, *Analysis of discrete time deterministic and stochastic Petri nets*, Ph.D. thesis, Technical University of Berlin, Germany, 1997.

300. R. Zijal, G. Ciardo, *Discrete deterministic and stochastic Petri nets*, ICASE Report, **96-72**, Institute for Computer Applications in Science and Engineering (ICASE), NASA, Langley Research Centre, Hampton, VA, USA, 1996.
301. R. Zijal, G. Ciardo, G. Hommel, *Discrete deterministic and stochastic Petri nets*, Proc. 9^{th} ITG/GI Professional Meeting on Measuring, Modeling and Evaluation of Computer and Communication Systems (MMB) 1997 (K. Irmscher, Ch. Mittasch, K. Richter, eds.), Freiberg, Germany, 103–117, VDE-Verlag, Berlin, Germany, 1997.
302. R. Zijal, R. German, *A new approach to discrete time stochastic Petri nets*, Proc. 11^{th} Int. Conf. on Analysis and Optimization of Systems, Discrete Event Systems (DES) 1994 (G. Cohen, J.-P. Quadrat, eds.), Sophia-Antipolis, France, Lecture Notes in Control and Information Sciences, **199** (1994), 198–204.