Discrete time stochastic Petri box calculus with immediate multiactions

Igor V. Tarasyuk*, Hermenegilda Macià and Valentín Valero†

Abstract

Petri box calculus (PBC) is a flexible and expressive process algebra defined by E. Best, R. Devillers and others in 1992. In this paper, we present the extension of discrete time stochastic Petri box calculus (dtsPBC) presented by I.V. Tarasyuk in 2007 with immediate multiactions, which is a discrete time analog of stochastic Petri box calculus (sPBC) with immediate multiactions proposed by H. Macià, V. Valero and others in 2008 within a continuous time domain. The step operational semantics of the calculus is constructed via labeled probabilistic transition systems. The denotational semantics is defined on the basis of a subclass of labeled discrete time stochastic Petri nets with immediate transitions. A consistency of both semantics is demonstrated. In order to evaluate performance, the corresponding stochastic process is analyzed, which is the same for both semantics. A case study of the shared memory system is presented as an example of specification, modeling, behaviour analysis and performance evaluation for parallel systems.

Keywords: stochastic Petri net, stochastic process algebra, Petri box calculus, discrete time, immediate multiaction, transition system, operational semantics, immediate transition, dtsi-box, denotational semantics, Markov chain, performance evaluation, shared memory system.

1 Introduction

Algebraic process calculi like CSP [10], ACP [6] and CCS [20] are a well-known formal model for the specification of computing systems and analysis of their behaviour. In such process algebras (PAs), systems and processes are specified by formulas, and verification of their properties is accomplished at a syntactic level via equivalences, axioms and inference rules. In the last decades, stochastic extensions of PAs were proposed such as MTIPP [11], PPA [22], PEPA [9] and EMPA [5]. Stochastic process algebras (SPAs) do not just specify actions which can happen as usual process algebras (qualitative features), but they associate some quantitative parameters with actions (quantitative characteristics).

Petri box calculus (PBC) [3, 4] is a flexible and expressive process algebra developed as a tool for specification of Petri nets 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 Petri nets. PBC has a step operational semantics in terms of labeled transition systems. Its denotational semantics was proposed in terms of a subclass of Petri nets (PNs) equipped with interface and considered up to isomorphism, called Petri boxes. Last years, several extensions of PBC with a deterministic or a stochastic model of time were presented.

A deterministic time model is considered both in time Petri box calculus (tPBC) [12] and in timed Petri box calculus (TPBC) [19]. In tPBC each action has a time interval associated (earliest and lasted firing time, respectively), and an interleaving operational semantics is defined. The denotational semantics is then defined in terms of a subclass of labeled time Petri nets (LtPNs), called time Petri boxes (ct-boxes). In contrast to tPBC, multiactions of TPBC are not instantaneous, but have time durations. For the latter model a step operational semantics is also considered, and a denotational semantics, using a subclass of labeled timed Petri nets (LTPNs), called timed Petri boxes (T-boxes).

A stochastic extension of PBC, called stochastic Petri box calculus (sPBC), was proposed in [16]. Only a finite part of PBC was initially used for the stochastic enrichment, i.e., in its former version sPBC has neither refinement nor recursion nor iteration operations. The calculus has an interleaving operational semantics in terms of labeled transition systems. Its denotational semantics was defined in terms of a subclass of labeled continuous time stochastic PNs (LCTSPNs), called s-boxes. In [13], the iteration was added to sPBC. The resulting calculus was enriched further with immediate multiactions in [15].

*The author was supported in part by Deutsche Forschungsgemeinschaft (DFG), grant 436 RUS 113/1002/01, and Russian Foundation for Basic Research (RFBR), grant 09-01-91334.

†The authors were supported by Spanish government (co-financed by FEDER founds) with the project "Modeling and analyzing composite Web services using formal methods", with reference TIN2009-14312-C02-02.

In [25], a discrete time stochastic extension dtsPBC of finite PBC was presented. A step operational semantics of dtsPBC was constructed via labeled probabilistic transition systems. Its denotational semantics was defined based on a subclass of labeled discrete time stochastic PNs (LDTSPNs), called dts-boxes. A variety of probabilistic equivalences were proposed and their interrelations were studied. In [24, 26], the iteration operator was added to dtsPBC.

In this paper, we present dtsPBC with iteration extended with immediate multiactions, called *discrete time stochastic and immediate Petri box calculus* (dtsiPBC), which is a discrete time analog of sPBC, which has iteration and immediate multiactions within the context of a continuous time domain. The step operational semantics is constructed with the use of labeled probabilistic transition systems. The denotational semantics is defined on the basis of a subclass of labeled discrete time stochastic and immediate PNs (LDTSPNs with immediate transitions, LDTSIPNs), called dtsi-boxes. A consistency of both semantics is demonstrated. The corresponding stochastic process, which is a semi-Markov chain, is constructed and investigated, with the purpose of performance evaluation, which is the same for both semantics. At the end, we present a case study of the shared memory system explaining how to specify, model and analyze performance of concurrent systems within the calculus.

The paper is organized as follows. In Section 2, the syntax of the extended calculus dtsiPBC is presented. In Section 3, we construct the operational semantics of the algebra in terms of labeled probabilistic transition systems. In Section 4, we propose the denotational semantics based on a subclass of LDTSIPNs. Next, in Section 5, the corresponding stochastic process is defined and analyzed, and in Section 6, a shared memory system is presented as a case study. Finally, Section 7 summarizes the results obtained and outlines research perspectives in this area.

2 Syntax

In this section, we propose the syntax of dtsiPBC. First, we recall a definition of multiset that is an extension of the set notion by allowing several identical elements.

Definition 2.1 *Let X be a set. A finite* multiset (bag) *M over X is a mapping $M : X \to \mathbb{N}$ such that $|\{x \in X \mid M(x) > 0\}| < \infty$, i.e., it can contain a finite number of elements only.*

We denote the *set of all finite multisets* over X by $\mathbb{N}_f^X$. The *cardinality* of a multiset M is defined as $|M| = \sum_{x \in X} M(x)$. 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 a proper set such that $M \subseteq X$. The *set of all subsets* of X is denoted by 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)* such that $a \neq \hat{a}$ and $\hat{\hat{a}} = a$. Let $\mathcal{A} = Act \cup \widehat{Act}$ be the set of *all actions*, and $\mathcal{L} = \mathbb{N}_f^{\mathcal{A}}$ be the set of *all multiactions*. Note that $\emptyset \in \mathcal{L}$, this corresponds to an internal activity, i.e., the execution of a multiaction that contains no visible action names. The *alphabet* of $\alpha \in \mathcal{L}$ is defined as $\mathcal{A}(\alpha) = \{x \in \mathcal{A} \mid \alpha(x) > 0\}$.

A *stochastic multiaction* is a pair (α, ρ), where $\alpha \in \mathcal{L}$ and $\rho \in (0; 1)$ is the *conditional probability* of the multiaction α. The multiaction probabilities are used to calculate the probabilities of state changes (steps) at discrete time moments. The probabilities of stochastic multiactions are required not to be equal to 1, since this value is left for immediate multiactions which will be defined later. On the other hand, there is no sense to allow zero probabilities of multiactions, since they would never be performed in this case. Let $\mathcal{SL}$ be the set of *all stochastic multiactions*.

An *immediate multiaction* is a pair (α, l), where $\alpha \in \mathcal{L}$ and $l \in \mathbb{N} \setminus \{0\}$ is the non-zero *weight* of the multiaction α. Immediate multiactions have a priority over stochastic ones. This means that in a state where both kinds of multiactions can happen, immediate multiactions always happen before stochastic ones. Stochastic and immediate multiactions cannot be executed together in some concurrent step, i.e., the steps consisting only of immediate multiactions or those including only stochastic multiactions are allowed. Let $\mathcal{IL}$ be the set of *all immediate multiactions*.

Let us note that the same multiaction $\alpha \in \mathcal{L}$ may have different probabilities and weights in the same specification. It is easy to differentiate between probabilities and weights, hence, between stochastic and immediate multiactions, since the probabilities of stochastic multiactions belong to the interval $(0; 1)$, and the weights of immediate multiactions are non-zero (positive) natural numbers from $\mathbb{N} \setminus \{0\} = \{1, 2, \ldots\}$. Let $\mathcal{SIL} = \mathcal{SL} \cup \mathcal{IL}$ be the set of *all activities*. The *alphabet* of $(\alpha, \kappa) \in \mathcal{SIL}$ is defined as $\mathcal{A}(\alpha, \kappa) = \mathcal{A}(\alpha)$. For $(\alpha, \kappa) \in \mathcal{SIL}$, we define its *multiaction part* as $\mathcal{L}(\alpha, \kappa) = \alpha$ and its *probability* or *weight part* as $\Omega(\alpha, \kappa) = \kappa$.

Activities are combined into formulas by the following operations: *sequential execution* ;, *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.

Sequential execution and choice have the standard interpretation like in other process algebras, but parallelism does not include synchronization unlike the corresponding operation in CCS [20].

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 in the usual way: for $\alpha \in \mathcal{L}$ we define $f(\alpha) = \sum_{x \in \alpha} f(x)$.

Restriction over an action a 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 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 $\alpha \oplus_a \beta = \gamma$, where

$$\gamma(x) = \begin{cases} \alpha(x) + \beta(x) - 1, & x = a \text{ or } x = \hat{a}; \\ \alpha(x) + \beta(x), & \text{otherwise.} \end{cases}$$

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

Static expressions specify the structure of processes. As we shall see, the expressions correspond to unmarked LDTSIPNs (note that LDTSIPNs are marked by definition).

Definition 2.2 *Let $(\alpha, \kappa) \in \mathcal{SIL}$ and $a \in Act$. A* static expression *of dtsiPBC is defined as*

$$E ::= (\alpha, \kappa) \mid E; E \mid E[]E \mid E \| E \mid E[f] \mid E \ \mathsf{rs}\ a \mid E \ \mathsf{sy}\ a \mid [E * E * E].$$

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

To make the grammar above unambiguous, one can add parentheses in the productions with binary operations: $(E; E)$, $(E[]E)$, $(E \| E)$. However, here and further we prefer the PBC approach and add them to resolve ambiguities only.

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

Definition 2.3 *Let $(\alpha, \kappa) \in \mathcal{SIL}$ and $a \in Act$. A* regular static expression *of dtsiPBC is defined as*

$$\begin{array}{l} E ::= (\alpha, \kappa) \mid E; E \mid E[]E \mid E \| E \mid E[f] \mid E \ \mathsf{rs}\ a \mid E \ \mathsf{sy}\ a \mid [E * D * E], \\ \textit{where } D ::= (\alpha, \kappa) \mid D; E \mid D[]D \mid D[f] \mid D \ \mathsf{rs}\ a \mid D \ \mathsf{sy}\ a \mid [D * D * E]. \end{array}$$

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

Dynamic expressions specify the states of processes. 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. As we shall see, dynamic expressions correspond to LDTSIPNs (which are marked by default). 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. The *underlying static expression* of a dynamic one is obtained by removing all upper and lower bars from it.

Definition 2.4 *Let $a \in Act$ and $E \in StatExpr$. A* dynamic expression *of dtsiPBC is defined as*

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

Let *DynExpr* denote the set of *all dynamic expressions* of dtsiPBC.

Note that if the underlying static expression of a dynamic one is not regular, the corresponding LDTSIPN can be non-safe (though, it is 2-bounded in the worst case, see [4]). A dynamic expression is *regular* if its underlying static expression is regular. Let *RegDynExpr* denote the set of *all regular dynamic expressions* of dtsiPBC.

In the following, we shall consider regular static and dynamic expressions only, hence, we shall omit the word "regular".

3 Operational semantics

In this section, we define the step operational semantics in terms of labeled transition systems.

3.1 Inaction rules

Inaction rules describe expression transformations due to the execution of the empty multiset of activities. The rules will be used later to define the *empty loop* transitions which reflect a non-zero probability to stay in the current state at the next time moment, which is an essential feature of discrete time stochastic processes. As we shall see, for every empty loop transition, its net analog in an LDTSIPN also does not change the current marking corresponding to the initial dynamic expression from the applied inaction rule.

First, in Table 1, we define inaction rules for the dynamic expressions in the form of overlined and underlined static ones. Let $E, F, K \in RegStatExpr$ and $a \in Act$.

Table 1: Inaction rules for overlined and underlined static expressions

$\overline{E;F} \stackrel{\emptyset}{\to} \overline{E};F$	$\underline{E};F \stackrel{\emptyset}{\to} E;\overline{F}$	$E;\underline{F} \stackrel{\emptyset}{\to} \underline{E;F}$	$\overline{E[]F} \stackrel{\emptyset}{\to} \overline{E}[]F$
$\overline{E[]F} \stackrel{\emptyset}{\to} E[]\overline{F}$	$\underline{E}[]F \stackrel{\emptyset}{\to} \underline{E[]F}$	$E[]\underline{F} \stackrel{\emptyset}{\to} \underline{E[]F}$	$\overline{E\|\|F} \stackrel{\emptyset}{\to} \overline{E}\|\|\overline{F}$
$\underline{E}\|\|\underline{F} \stackrel{\emptyset}{\to} \underline{E\|\|F}$	$\overline{E[f]} \stackrel{\emptyset}{\to} \overline{E}[f]$	$\underline{E}[f] \stackrel{\emptyset}{\to} \underline{E[f]}$	$\overline{E \text{ rs } a} \stackrel{\emptyset}{\to} \overline{E} \text{ rs } a$
$\underline{E} \text{ rs } a \stackrel{\emptyset}{\to} \underline{E \text{ rs } a}$	$\overline{E \text{ sy } a} \stackrel{\emptyset}{\to} \overline{E} \text{ sy } a$	$\underline{E} \text{ sy } a \stackrel{\emptyset}{\to} \underline{E \text{ sy } a}$	$\overline{[E*F*K]} \stackrel{\emptyset}{\to} [\overline{E}*F*K]$
$[\underline{E}*F*K] \stackrel{\emptyset}{\to} [E*\overline{F}*K]$	$[E*\underline{F}*K] \stackrel{\emptyset}{\to} [E*\overline{F}*K]$	$[E*\underline{F}*K] \stackrel{\emptyset}{\to} [E*F*\overline{K}]$	$[E*F*\underline{K}] \stackrel{\emptyset}{\to} \underline{[E*F*K]}$

Second, in Table 2, we propose inaction rules for the dynamic expressions in the arbitrary form. Let $E, F \in RegStatExpr$, $G, H, \widetilde{G}, \widetilde{H} \in RegDynExpr$ and $a \in Act$.

Table 2: Inaction rules for arbitrary dynamic expressions

$G \stackrel{\emptyset}{\to} G$	$\dfrac{G \stackrel{\emptyset}{\to} \widetilde{G},\ \circ \in \{;,[]\}}{G \circ E \stackrel{\emptyset}{\to} \widetilde{G} \circ E}$	$\dfrac{G \stackrel{\emptyset}{\to} \widetilde{G},\ \circ \in \{;,[]\}}{E \circ G \stackrel{\emptyset}{\to} E \circ \widetilde{G}}$	$\dfrac{G \stackrel{\emptyset}{\to} \widetilde{G}}{G\|\|H \stackrel{\emptyset}{\to} \widetilde{G}\|\|H}$	$\dfrac{H \stackrel{\emptyset}{\to} \widetilde{H}}{G\|\|H \stackrel{\emptyset}{\to} G\|\|\widetilde{H}}$
$\dfrac{G \stackrel{\emptyset}{\to} \widetilde{G}}{G[f] \stackrel{\emptyset}{\to} \widetilde{G}[f]}$	$\dfrac{G \stackrel{\emptyset}{\to} \widetilde{G},\ \circ \in \{\text{rs}, \text{sy}\}}{G \circ a \stackrel{\emptyset}{\to} \widetilde{G} \circ a}$	$\dfrac{G \stackrel{\emptyset}{\to} \widetilde{G}}{[G*E*F] \stackrel{\emptyset}{\to} [\widetilde{G}*E*F]}$	$\dfrac{G \stackrel{\emptyset}{\to} \widetilde{G}}{[E*G*F] \stackrel{\emptyset}{\to} [E*\widetilde{G}*F]}$	$\dfrac{G \stackrel{\emptyset}{\to} \widetilde{G}}{[E*F*G] \stackrel{\emptyset}{\to} [E*F*\widetilde{G}]}$

A regular dynamic expression G is *operative* if no inaction rule can be applied to it, with the exception of $G \stackrel{\emptyset}{\to} G$.

Let $OpRegDynExpr$ denote the set of *all operative regular dynamic expressions* of dtsiPBC.

Note that any dynamic expression can be always transformed into a (not necessarily unique) operative one by using the inaction rules.

Definition 3.1 *Let $\approx\ = (\stackrel{\emptyset}{\to} \cup \stackrel{\emptyset}{\leftarrow})^*$ be the structural equivalence of dynamic expressions in dtsiPBC. 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 forward or backward direction.*

Note that the rule $G \stackrel{\emptyset}{\to} G$ was intentionally included in the set of inaction rules to define later the empty loop transitions. This is a new rule that has no prototype among inaction rules of PBC. However, in the context of discrete time stochastic transitions we need to capture the possibility to stay at the same state such as the singleton structural equivalence class of the dynamic expression $\overline{(\alpha, \rho)}$ to which no different structurally equivalent ones exist.

3.2 Action rules

Action rules describe expression transformations due to the execution of non-empty multisets of activities. The rules will be used later to define transitions representing the state changes when some non-empty multisets of activities are executed. As we shall see, for every such transition, its net analog in the LDTSIPN changes the current marking corresponding to the initial dynamic expression from the applied action rule, unless there is a self-loop produced by the iterative execution of a non-empty multiset (which should be additionally the one-element one, i.e., the single activity, since we do not allow concurrency in the highest level of the second argument of iteration).

Note that expressions of dtsiPBC can contain identical activities. To avoid technical difficulties, such as the proper calculation of the state change probabilities for multiple transitions, we can always enumerate coinciding

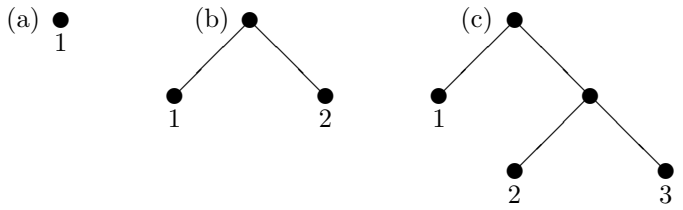

Figure 1: The binary trees encoded with the numberings 1, (1)(2) and (1)((2)(3))

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

Definition 3.2 *Let* $\iota \in I\!N$. *The* numbering *of expressions is defined as*

$$\iota ::= \ \iota \mid (\iota)(\iota).$$

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

Example 3.1 *The numbering* 1 *encodes the binary tree depicted in Figure 1(a) with the root labeled by* 1. *The numbering* (1)(2) *corresponds to the binary tree depicted in Figure 1(b) without internal nodes and with two leaves labeled by* 1 *and* 2. *The numbering* (1)((2)(3)) *represents the binary tree depicted in Figure 1(c) with one internal node, which is the root for the subtree* (2)(3)*, and three leaves labeled by* 1, 2 *and* 3.

The new activities resulting from synchronizations in different orders should be considered up to the permutation of their numbering. In this way, we shall recognize the 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 be able to identify the mentioned instances.

The *content* of a numbering $\iota \in Num$ is

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

After the enumeration, the multisets of activities from the expressions will be then the proper sets. In the following, we suppose that the identical activities are enumerated when it is needed to avoid ambiguity. In this case, $\mathcal{SL}$, $\mathcal{IL}$ and $\mathcal{SIL}$ are supposed to contain all the enumerated elements as well. This enumeration is considered to be implicit.

Let X be some set. We denote the cartesian product $X \times X$ by X^2. Let $\mathcal{E} \subseteq X^2$ be an equivalence relation on X. Then the *equivalence class* (with respect to $\mathcal{E}$) of an element $x \in X$ is defined by $[x]_{\mathcal{E}} = \{y \in X \mid (x, y) \in \mathcal{E}\}$. The equivalence $\mathcal{E}$ partitions X into the *set of equivalence classes* $X/_{\mathcal{E}} = \{[x]_{\mathcal{E}} \mid x \in X\}$.

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. G is an *initial* dynamic expression, denoted by $init(G)$, if $\exists E \in RegStatExpr\ G \in [\overline{E}]_{\approx}$. G is a *final* dynamic expression, denoted by $final(G)$, if $\exists E \in RegStatExpr\ G \in [\underline{E}]_{\approx}$.

Let $\Upsilon \in 2^{\mathcal{SIL}}$. Relabeling is extended to the sets of activities as follows: $f(\Upsilon) = \{(f(\alpha), \kappa) \mid (\alpha, \kappa) \in \Upsilon\}$. The *alphabet* of Υ is defined as $\mathcal{A}(\Upsilon) = \cup_{(\alpha,\kappa)\in\Upsilon}\mathcal{A}(\alpha)$.

Let $G \in OpRegDynExpr$. We now define the *set of all sets of non-conflicting activities which can be executed from* G, denoted by $Can(G)$. Let $(\alpha, \kappa) \in \mathcal{SIL},\ E, F \in RegStatExpr,\ G, H \in OpRegDynExpr$ and $a \in Act$.

1. If $final(G)$ then $Can(G) = \emptyset$.
2. If $G = \overline{(\alpha, \kappa)}$ then $Can(G) = \{\{(\alpha, \kappa)\}\}$.
3. If $\Upsilon \in Can(G)$ then $\Upsilon \in Can(G \circ E),\ \Upsilon \in Can(E \circ G)$ $(\circ \in \{;, [], \|\}),\ f(\Upsilon) \in Can(G[f])$, $\Upsilon \in Can(G\ \mathsf{rs}\ a)$ (when $a, \hat{a} \notin \mathcal{A}(\Upsilon)$), $\Upsilon \in Can(G\ \mathsf{sy}\ a),\ \Upsilon \in Can([G * E * F]),\ \Upsilon \in Can([E * G * F])$, $\Upsilon \in Can([E * F * G])$.
4. If $\Upsilon \in Can(G)$ and $\Xi \in Can(H)$ then $\Upsilon + \Xi \in Can(G \| H)$.
5. If $\Upsilon \in Can(G\ \mathsf{sy}\ a)$ and $(\alpha, \kappa), (\beta, \lambda) \in \Upsilon$ are different activities such that $a \in \alpha,\ \hat{a} \in \beta$ then

(a) $(\Upsilon + \{(\alpha \oplus_a \beta, \kappa \cdot \lambda)\}) \setminus \{(\alpha,\kappa),(\beta,\lambda)\} \in Can(G \text{ sy } a)$, if $\kappa, \lambda \in (0;1)$;

(b) $(\Upsilon + \{(\alpha \oplus_a \beta, \kappa + \lambda)\}) \setminus \{(\alpha,\kappa),(\beta,\lambda)\} \in Can(G \text{ sy } a)$, if $\kappa, \lambda \in I\!N \setminus \{0\}$.

When we synchronize the same set of activities in different orders, we obtain several resulting activities with the same multiaction and probability or weight parts, but with different numberings having the same content. Then we only consider a single one of the resulting activities to avoid introducing redundant ones.

For example, the synchronization of stochastic multiactions $(\alpha,\rho)_1$ and $(\beta,\chi)_2$ in different orders generates the activities $(\alpha \oplus_a \beta, \rho \cdot \chi)_{(1)(2)}$ and $(\beta \oplus_a \alpha, \chi \cdot \rho)_{(2)(1)}$. Similarly, the synchronization of immediate multiactions $(\alpha,l)_1$ and $(\beta,m)_2$ in different orders generates the activities $(\alpha \oplus_a \beta, l+m)_{(1)(2)}$ and $(\beta \oplus_a \alpha, m+l)_{(2)(1)}$. Since $Cont((1)(2)) = \{1,2\} = Cont((2)(1))$, in both cases, only the first activity (or, symmetrically, the second one) resulting from synchronization will appear in a set from $Can(G \text{ sy } a)$.

Note that if $\Upsilon \in Can(G)$ then by definition of $Can(G)$ $\forall \Xi \subseteq \Upsilon,\ \Xi \neq \emptyset$ we have $\Xi \in Can(G)$.

The expression $G \in OpRegDynExpr$ is *tangible*, denoted by $tang(G)$, if $Can(G)$ contains only sets of stochastic multiactions (possibly including the empty set), i.e., $\forall \Upsilon \in Can(G)\ \Upsilon \in 2^{\mathcal{SL}}$. Otherwise, G is *vanishing*, denoted by $vanish(G)$, meaning that there are immediate multiactions in the sets from $Can(G)$, hence, according to the note above, there are non-empty sets of immediate multiactions in $Can(G)$ as well, i.e., $\exists \Upsilon \in Can(G)\ \Upsilon \in 2^{\mathcal{IL}} \setminus \{\emptyset\}$. Obviously, immediate multiactions are only executable from vanishing operative dynamic expressions. Stochastic multiactions are only executable from tangible ones, since no stochastic multiactions can be executed from a vanishing operative dynamic expression G, even if $Can(G)$ contains sets of stochastic multiactions. The reason is that immediate multiactions have a priority over stochastic ones, and should be executed first.

Now, in Table 3, we define the action rules which describe expression transformations due to the execution of non-empty sets of activities. Let $(\alpha,\rho),(\beta,\chi) \in \mathcal{SL}$, $(\alpha,l),(\beta,m) \in \mathcal{IL}$ and $(\alpha,\kappa) \in \mathcal{SIL}$. Further, let $E,F \in RegStatExpr$, $G,H \in OpRegDynExpr$, $\widetilde{G},\widetilde{H} \in RegDynExpr$ and $a \in Act$. Moreover, let $\Gamma,\Delta \in 2^{\mathcal{SL}} \setminus \{\emptyset\}$, $I,J \in 2^{\mathcal{IL}} \setminus \{\emptyset\}$ and $\Upsilon \in 2^{\mathcal{SIL}} \setminus \{\emptyset\}$.

Table 3: Action rules

B $\dfrac{}{\overline{(\alpha,\kappa)} \overset{\{(\alpha,\kappa)\}}{\longrightarrow} \underline{(\alpha,\kappa)}}$

S $\dfrac{G \overset{\Upsilon}{\to} \widetilde{G}}{G;E \overset{\Upsilon}{\to} \widetilde{G};E \quad E;G \overset{\Upsilon}{\to} E;\widetilde{G}}$

C $\dfrac{G \overset{\Gamma}{\to} \widetilde{G},\ \neg init(G) \vee (init(G) \wedge tang(\overline{E}))}{G[]E \overset{\Gamma}{\to} \widetilde{G}[]E \quad E[]G \overset{\Gamma}{\to} E[]\widetilde{G}}$

Ci $\dfrac{G \overset{I}{\to} \widetilde{G}}{G[]E \overset{I}{\to} \widetilde{G}[]E \quad E[]G \overset{I}{\to} E[]\widetilde{G}}$

P1 $\dfrac{G \overset{\Gamma}{\to} \widetilde{G},\ tang(H)}{G \| H \overset{\Gamma}{\to} \widetilde{G} \| H \quad H \| G \overset{\Gamma}{\to} H \| \widetilde{G}}$

P1i $\dfrac{G \overset{I}{\to} \widetilde{G}}{G \| H \overset{I}{\to} \widetilde{G} \| H \quad H \| G \overset{I}{\to} H \| \widetilde{G}}$

P2 $\dfrac{G \overset{\Gamma}{\to} \widetilde{G},\ H \overset{\Delta}{\to} \widetilde{H},\ tang(G) \wedge tang(H)}{G \| H \overset{\Gamma+\Delta}{\longrightarrow} \widetilde{G} \| \widetilde{H}}$

P2i $\dfrac{G \overset{I}{\to} \widetilde{G},\ H \overset{J}{\to} \widetilde{H}}{G \| H \overset{I+J}{\longrightarrow} \widetilde{G} \| \widetilde{H}}$

L $\dfrac{G \overset{\Upsilon}{\to} \widetilde{G}}{G[f] \overset{f(\Upsilon)}{\longrightarrow} \widetilde{G}[f]}$

Rs $\dfrac{G \overset{\Upsilon}{\to} \widetilde{G},\ a,\hat{a} \notin \mathcal{A}(\Upsilon)}{G \text{ rs } a \overset{\Upsilon}{\to} \widetilde{G} \text{ rs } a}$

I1 $\dfrac{G \overset{\Upsilon}{\to} \widetilde{G}}{[G*E*F] \overset{\Upsilon}{\to} [\widetilde{G}*E*F]}$

I2 $\dfrac{G \overset{\Gamma}{\to} \widetilde{G},\ \neg init(G) \vee (init(G) \wedge tang(\overline{F}))}{[E*G*F] \overset{\Gamma}{\to} [E*\widetilde{G}*F]}$

I2i $\dfrac{G \overset{I}{\to} \widetilde{G}}{[E*G*F] \overset{I}{\to} [E*\widetilde{G}*F]}$

I3 $\dfrac{G \overset{\Gamma}{\to} \widetilde{G},\ \neg init(G) \vee (init(G) \wedge tang(\overline{F}))}{[E*F*G] \overset{\Gamma}{\to} [E*F*\widetilde{G}]}$

I3i $\dfrac{G \overset{I}{\to} \widetilde{G}}{[E*F*G] \overset{I}{\to} [E*F*\widetilde{G}]}$

Sy1 $\dfrac{G \overset{\Upsilon}{\to} \widetilde{G}}{G \text{ sy } a \overset{\Upsilon}{\to} \widetilde{G} \text{ sy } a}$

Sy2 $\dfrac{G \text{ sy } a \overset{\Gamma+\{(\alpha,\rho)\}+\{(\beta,\chi)\}}{\longrightarrow} \widetilde{G} \text{ sy } a,\ a \in \alpha,\ \hat{a} \in \beta,\ tang(G \text{ sy } a)}{G \text{ sy } a \overset{\Gamma+\{(\alpha \oplus_a \beta, \rho \cdot \chi)\}}{\longrightarrow} \widetilde{G} \text{ sy } a}$

Sy2i $\dfrac{G \text{ sy } a \overset{I+\{(\alpha,l)\}+\{(\beta,m)\}}{\longrightarrow} \widetilde{G} \text{ sy } a,\ a \in \alpha,\ \hat{a} \in \beta}{G \text{ sy } a \overset{I+\{(\alpha \oplus_a \beta, l+m)\}}{\longrightarrow} \widetilde{G} \text{ sy } a}$

Note that in the rule **Sy2** we multiply the probabilities of the synchronized stochastic multiactions, since this corresponds to the probability of the events intersection. In the rule **Sy2i** we sum the weights of the synchronized immediate multiactions to express that their synchronized execution has more importance than that of every single one. The reason for the summation of the weights is that the execution of immediate multiactions takes no time, hence, we prefer to execute in a step as much synchronized immediate multiactions as possible to get more significant progress in computation.

Observe also that we do not have self-synchronization, i.e., the synchronization of an activity with itself, since all the (enumerated) activities executed together are different.

3.3 Transition systems

Now we intend to construct labeled probabilistic transition systems associated with dynamic expressions. The transition systems are used to define the operational semantics of dynamic expressions.

Definition 3.3 *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* $[\widetilde{H}]_{\approx} \in DR(G)$.

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

The set of *all the sets of activities executable in* s is defined as $Exec(s) = \{\Upsilon \mid \exists H \in s\ \exists \widetilde{H}\ H \xrightarrow{\Upsilon} \widetilde{H}\}$. The state s is *tangible*, if $Exec(s) \subseteq 2^{\mathcal{SL}}$. For tangible states we may have $Exec(s) = \emptyset$. Otherwise, the state s is *vanishing*, and in this case $Exec(s) \subseteq 2^{\mathcal{IL}} \setminus \{\emptyset\}$. The set of *all tangible states from* $DR(G)$ is denoted by $DR_T(G)$, and the set of *all vanishing states from* $DR(G)$ is denoted by $DR_V(G)$. Obviously, $DR(G) = DR_T(G) \cup DR_V(G)$.

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

$$PF(\Upsilon, s) = \begin{cases} \prod_{(\alpha,\rho)\in\Upsilon} \rho \cdot \prod_{\{(\beta,\chi)\}\in Exec(s)|(\beta,\chi)\notin\Upsilon\}} (1-\chi), & s \in DR_T(G); \\ \sum_{(\alpha,l)\in\Upsilon} l, & s \in DR_V(G). \end{cases}$$

In the case $\Upsilon = \emptyset$ we define

$$PF(\emptyset, s) = \begin{cases} \prod_{\{(\beta,\chi)\}\in Exec(s)} (1-\chi), & s \in DR_T(G) \wedge Exec(s) \neq \emptyset; \\ 1, & s \in DR_T(G) \wedge Exec(s) = \emptyset; \\ 0, & s \in DR_V(G). \end{cases}$$

Thus, if $s \in DR_T(G)$ and $Exec(s) \neq \emptyset$, then $PF(\Upsilon, s)$ could be interpreted as a *joint* probability of independent events. Each such an event is interpreted as readiness or not readiness for execution of a particular stochastic multiaction from Υ. The multiplication in the definition is used because it reflects the probability of the event intersection. When only the empty set of activities can be executed in s, i.e., $Exec(s) = \emptyset$, we take $PF(\emptyset, s) = 1$, since we stay in s in this case. Note that for $s \in DR_T(G)$ we have $PF(\emptyset, s) \in (0; 1]$, hence, we can stay in s at the next time moment with a certain positive probability.

If $s \in DR_V(G)$ then $PF(\Upsilon, s)$ could be interpreted as the *overall* weight of the immediate multiactions from Υ, i.e., the sum of all their weights. The summation here is used to express that concurrent execution of the immediate multiactions has more importance than that of every single one. Since the execution of immediate multiactions takes no time, we prefer to execute in a step as much parallel immediate multiactions as possible to get more significant progress in computation. Note that this reasoning is the same as that used to define the probability of synchronized immediate multiactions in the rule **Sy2i**. When immediate multiactions can be executed from s, we take $PF(\emptyset, s) = 0$ to indicate that in this case we cannot stay in s at the next time moment, since we must execute some immediate multiaction.

Note that the definition of $PF(\Upsilon, s)$ (as well as the definitions of other probability functions which we shall present) is based on the enumeration of activities which is considered implicit.

Let $\Upsilon \in Exec(s)$. The *probability to execute the set of activities* Υ *in* s is

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

Thus, $PT(\Upsilon, s)$ is the probability of the set of stochastic multiactions or the weight of the set of immediate multiactions Υ which is ready for execution in s *normalized* by the corresponding probability or weight of *any* set executable in s. The denominator of the fraction above is a sum since it reflects the probability of the events union.

If s is tangible, then $PT(\emptyset, s) \in (0; 1]$, hence, there is a non-zero probability to stay in the state in the next time moment, and the residence time in s is at least 1 discrete time unit. If the state s is vanishing, then $PT(\emptyset, s) = 0$, hence, there is zero probability to stay in the state in the next time moment, and the residence time in s is 0.

Note that the sum of outgoing probabilities for the expressions belonging to the derivations of G is equal to 1. More formally, $\forall s \in DR(G)\ \sum_{\Upsilon\in Exec(s)} PT(\Upsilon, s) = 1$. This, obviously, follows from the definition of $PT(\Upsilon, s)$, and guarantees that $PT(\Upsilon, s)$ always defines a probability distribution.

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

$$PM(s,\tilde{s}) = \sum_{\{\Upsilon \mid \exists H \in s \ \exists \widetilde{H} \in \tilde{s} \ H \stackrel{\Upsilon}{\to} \widetilde{H}\}} PT(\Upsilon, s).$$

Since $PM(s,\tilde{s})$ is the probability for *any* set of activities (including the empty one) to change s to $\tilde{s}$, we use summation in the definition. Note that $\forall s \in DR(G)\ \sum_{\{\tilde{s} \mid \exists H \in s \ \exists \widetilde{H} \in \tilde{s} \ \exists \Upsilon \ H \stackrel{\Upsilon}{\to} \widetilde{H}\}} PM(s,\tilde{s}) = \sum_{\{\tilde{s} \mid \exists H \in s \ \exists \widetilde{H} \in \tilde{s} \ \exists \Upsilon \ H \stackrel{\Upsilon}{\to} \widetilde{H}\}} \sum_{\{\Upsilon \mid \exists H \in s \ \exists \widetilde{H} \in \tilde{s} \ H \stackrel{\Upsilon}{\to} \widetilde{H}\}} PT(\Upsilon, s) = \sum_{\Upsilon \in Exec(s)} PT(\Upsilon, s) = 1.$

Definition 3.4 *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 $L_G \subseteq 2^{\mathcal{SIL}} \times (0;1]$;*
- *the set of* transitions *is $\mathcal{T}_G = \{(s, (\Upsilon, PT(\Upsilon, s)), \tilde{s}) \mid s \in DR(G),\ \exists H \in s\ \exists \widetilde{H} \in \tilde{s}\ H \stackrel{\Upsilon}{\to} \widetilde{H}\}$;*
- *the* initial state *is $s_G = [G]_{\approx}$.*

The definition of $TS(G)$ is correct, i.e., for every state the sum of the probabilities of all the transitions starting from it is 1. This is guaranteed by the note after the definition of $PT(\Upsilon, s)$. Thus, we have defined the *generative* model of probabilistic processes, according to the classification from [8]. The reason is that the sum of the probabilities of the transitions with all possible labels should be equal to 1, not only of those with the same labels (up to enumeration of activities they include) as in the *reactive* models, and we do not have the nested choice as in the *stratified* models.

The transition system $TS(G)$ associated with a dynamic expression G describes all the steps that happen at moments of discrete time with some (one-step) probability and consist of sets of activities. Every step consisting of stochastic multiactions or the empty step (i.e., that consisting of the empty set of activities) happens instantaneously after one discrete time unit delay. Each step consisting of immediate multiactions happens instantaneously without any delay. The step can change the current state to another 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 $s \stackrel{\Upsilon}{\to}_{\mathcal{P}} \tilde{s}$. It is interpreted as follows: the probability to change the state s to $\tilde{s}$ in the result of executing Υ is $\mathcal{P}$.

Note that Υ can be the empty set (for tangible states), and its execution does not change the current state (i.e., the equivalence class), since we have a loop transition $s \stackrel{\emptyset}{\to}_{\mathcal{P}} s$ from a state s to itself. This corresponds to the application of inaction rules to the expressions from the equivalence class. We have to keep track of such executions, called *empty loops*, because they have nonzero probabilities. This follows from the definition of $PF(\emptyset, s)$ and the fact that multiaction probabilities cannot be equal to 1 as they belong to the interval $(0;1)$. But the step probabilities belong to the interval $(0;1]$, being 1 in the case when we cannot leave a state s and there exists the only transition from it which is the empty loop one $s \stackrel{\emptyset}{\to}_1 s$.

We write $s \stackrel{\Upsilon}{\to} \tilde{s}$ if $\exists \mathcal{P} > 0\ s \stackrel{\Upsilon}{\to}_{\mathcal{P}} \tilde{s}$ and $s \to \tilde{s}$ if $\exists \Upsilon\ s \stackrel{\Upsilon}{\to} \tilde{s}$. In addition, we write $s \to_{\mathcal{P}} \tilde{s}$ if $s \to \tilde{s}$ and $\mathcal{P} = PM(s, \tilde{s})$.

Isomorphism is a coincidence of systems up to renaming of their components or states. Let $\simeq$ denote isomorphism between transition systems that relates their initial states.

Transition systems of static expressions can be defined as well. For $E \in RegStatExpr$, let $TS(E) = TS(\overline{E})$.

Definition 3.5 *Two dynamic expressions G and G' are* equivalent with respect to transition systems*, denoted by $G =_{ts} G'$, if $TS(G) \simeq TS(G')$.*

Example 3.2 *Consider the expression* $\mathsf{Stop} = (\{g\}, \frac{1}{2})\ \mathsf{rs}\ g$ *specifying the special process that is only able to perform empty loops with probability* 1 *and never terminates. We could actually use any arbitrary action from $\mathcal{A}$ and any conditional probability belonging to the interval* $(0;1)$ *in the definition of* Stop*. Note that* Stop *is analogous to the one used in the examples of [14]. The latter is a continuous time stochastic analogue of the* stop *process proposed in [4].* Stop *is a discrete time stochastic analogue of the* stop*. Let*

$$E = [(\{a\}, \rho) * ((\{b\}, \chi); ((((\{c\}, l); (\{d\}, \theta))[]((\{e\}, m); (\{f\}, \phi)))) * \mathsf{Stop}].$$

$DR(\overline{E})$ consists of the equivalence classes

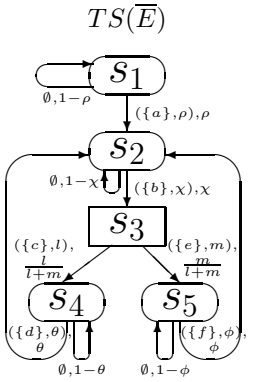

Figure 2: The transition system of $\overline{E}$ for $E = [(\{a\},\rho) * ((\{b\},\chi); ((((\{c\},l); (\{d\},\theta))[]((\{e\},m); (\{f\},\phi)))) * \mathsf{Stop}]$

$$
\begin{aligned}
s_1 &= [[\overline{(\{a\},\rho)} * (\underline{(\{b\},\chi)}; ((((\{c\},l); (\{d\},\theta))[]((\{e\},m); (\{f\},\phi)))) * \mathsf{Stop}]]_{\approx},\\
s_2 &= [[(\{a\},\rho) * \overline{(\{b\},\chi)}; \underline{((((\{c\},l); (\{d\},\theta))[]((\{e\},m); (\{f\},\phi))))} * \mathsf{Stop}]]_{\approx},\\
s_3 &= [[(\{a\},\rho) * ((\{b\},\chi); \overline{((((\{c\},l); \underline{(\{d\},\theta))}[]((\{e\},m); (\{f\},\phi))))} * \mathsf{Stop}]]_{\approx},\\
s_4 &= [[(\{a\},\rho) * ((\{b\},\chi); ((((\{c\},l); \overline{(\{d\},\theta))}[]((\{e\},m); \underline{(\{f\},\phi))))} * \mathsf{Stop}]]_{\approx},\\
s_5 &= [[(\{a\},\rho) * ((\{b\},\chi); ((((\{c\},l); (\{d\},\theta))[]((\{e\},m); \overline{(\{f\},\phi)}))) * \mathsf{Stop}]]_{\approx}.
\end{aligned}
$$

We have $DR_T(\overline{E}) = \{s_1, s_2, s_4, s_5\}$ *and* $DR_V(\overline{E}) = \{s_3\}$.

In Figure 2 the transition system $TS(\overline{E})$ *is presented. The tangible states are depicted in ovals and the vanishing ones are depicted in boxes. For simplicity of the graphical representation, the singleton sets of activities are written without braces.*

4 Denotational semantics

In this section, we construct the denotational semantics in terms of a subclass of labeled discrete time stochastic and immediate Petri nets (LDTSIPNs), called discrete time stochastic and immediate Petri boxes (dtsi-boxes).

4.1 Labeled DTSIPNs

Let us introduce a class of labeled discrete time stochastic and immediate Petri nets. First, we present a formal definition of LDTSIPNs.

Definition 4.1 *A* labeled discrete time stochastic and immediate Petri net (LDTSIPN) *is a tuple* $N = (P_N, T_N, W_N, \Omega_N, L_N, M_N)$, *where*

- P_N *and* $T_N = Ts_N \cup Ti_N$ *are finite sets of* places *and* stochastic and immediate transitions, *respectively, such that* $P_N \cup T_N \neq \emptyset$, $P_N \cap T_N = \emptyset$ *and* $Ts_N \cap Ti_N = \emptyset$*;*
- $W_N : (P_N \times T_N) \cup (T_N \times P_N) \to I\!N$ *is a function providing the* weights of arcs *between places and transitions;*
- $\Omega_N : T_N \to (0;1) \cup (I\!N \setminus \{0\})$ *is the* transition probability and weight *function associating stochastic transitions with probabilities and immediate ones with weights;*
- $L_N : T_N \to \mathcal{L}$ *is the* transition labeling *function assigning multiactions to transitions;*
- $M_N \in I\!N_f^{P_N}$ *is the* initial marking.

A graphical representation of LDTSIPNs is like that for standard labeled Petri nets, but with probabilities or weights written near the corresponding transitions. Square boxes of normal thickness depict stochastic transitions, and those with thick borders represent immediate transitions. In the case the probabilities or the weights are not given in the picture, they are considered to be of no importance in the corresponding examples, such as those used to describe stationary behaviour. The weights of arcs are depicted with them. The names of

places and transitions are depicted near them when needed. If the names are omitted but used, it is supposed that the places and transitions are numbered from left to right and from top to down.

Now we consider the semantics of LDTSIPNs.

Let N be an LDTSIPN and $t \in T_N$, $U \in I\!N_f^{T_N}$. The *precondition* ${}^\bullet t$ and the *postcondition* $t^\bullet$ of t are the multisets of places defined as $({}^\bullet t)(p) = W_N(p,t)$ and $(t^\bullet)(p) = W_N(t,p)$. The *precondition* ${}^\bullet U$ and the *postcondition* $U^\bullet$ of U are the multisets of places defined as ${}^\bullet U = \sum_{t\in U} {}^\bullet t$ and $U^\bullet = \sum_{t\in U} t^\bullet$.

Let N be an LDTSIPN and $M, \widetilde{M} \in I\!N_f^{P_N}$.

Immediate transitions have a priority over stochastic ones, thus, immediate transitions always fire first, if they can. A transition $t \in T_N$ is enabled in M if ${}^\bullet t \subseteq M$ and one of the following holds:

1. $t \in Ti_N$ or
2. $\forall u \in T_N\ {}^\bullet u \subseteq M \ \Rightarrow\ u \in Ts_N$.

In other words, a transition is enabled in a marking if it has enough tokens in its input places (i.e., in the places from its precondition) and it is either immediate or stochastic one, and in the latter case there exist no immediate transitions with enough tokens in their input places. Let $Ena(M)$ be the set of *all enabled transitions in* M. By definition, it follows that $Ena(M) \subseteq Ti_N$ or $Ena(M) \subseteq Ts_N$. A set of transitions $U \subseteq Ena(M)$ is enabled in a marking M if ${}^\bullet U \subseteq M$. Firings of transitions are atomic operations, and transitions may fire concurrently in steps. We assume that all transitions participating in a step should differ, hence, only the sets (not multisets) of transitions may fire. Thus, we do not allow self-concurrency, i.e., firing of transitions concurrently to themselves.

The marking M is *tangible*, denoted by $tang(M)$, if $Ena(M) \subseteq Ts_N$ or $Ena(M) = \emptyset$. Otherwise, the marking M is *vanishing*, denoted by $vanish(M)$, and in this case $Ena(M) \subseteq Ti_N$ and $Ena(M) \neq \emptyset$. If $tang(M)$ then a stochastic transition $t \in Ena(M)$ fires with probability $\Omega_N(t)$ when no other stochastic transitions conflicting with it are enabled.

Let $U \subseteq Ena(M)$, $U \neq \emptyset$ and ${}^\bullet U \subseteq M$. The *probability of the set of stochastic transitions* or the *weight of the set of immediate transitions* U *which is ready for firing in* M is

$$PF(U,M) = \begin{cases} \prod_{t\in U} \Omega_N(t) \cdot \prod_{u\in Ena(M)\setminus U}(1-\Omega_N(u)), & tang(M); \\ \sum_{t\in U} \Omega_N(t), & vanish(M). \end{cases}$$

In the case $U = \emptyset$ we define

$$PF(\emptyset,M) = \begin{cases} \prod_{u\in Ena(M)}(1-\Omega_N(u)), & tang(M) \wedge Ena(M) \neq \emptyset; \\ 1, & tang(M) \wedge Ena(M) = \emptyset; \\ 0, & vanish(M). \end{cases}$$

Thus, if $tang(M)$ and $Ena(M) \neq \emptyset$, then $PF(U,M)$ could be interpreted as a *joint* probability of independent events. Each such an event is interpreted as readiness or not readiness for firing of a particular transition from U. The multiplication in the definition is used because it reflects the probability of the events intersection. When no transitions are enabled in M, i.e., $Ena(M) = \emptyset$, we take $PF(\emptyset,M) = 1$, since we stay in M in this case. Note that for $tang(M)$ we have $PF(\emptyset,M) \in (0;1]$, hence, we can stay in M at the next time moment with a certain positive probability.

If $vanish(M)$ then $PF(U,M)$ could be interpreted as the *overall* weight of the immediate transitions from U, i.e., the sum of all their weights. When immediate transitions can be fired in M, we define $PF(\emptyset,M) = 0$ to indicate that in this case we cannot stay in M at the next time moment, since we must execute the immediate transitions.

Let $U \subseteq Ena(M)$, $U \neq \emptyset$ and ${}^\bullet U \subseteq M$. The concurrent firing of the transitions from U changes the marking M to $\widetilde{M} = M - {}^\bullet U + U^\bullet$, denoted by $M \xrightarrow{U}_{\mathcal{P}} \widetilde{M}$, where $\mathcal{P} = PT(U,M)$ is the *probability that the set of transitions* U *fires in* M defined as

$$PT(U,M) = \frac{PF(U,M)}{\sum_{\{V|{}^\bullet V\subseteq M\}} PF(V,M)}.$$

In the case $U = \emptyset$ we have $M = \widetilde{M}$ and

$$PT(\emptyset,M) = \frac{PF(\emptyset,M)}{\sum_{\{V|{}^\bullet V\subseteq M\}} PF(V,M)}.$$

Thus, $PT(U,M)$ is the probability of the set of stochastic transitions or the weight of the set of immediate transitions U which is ready for firing in M *normalized* by the corresponding probability for *any* set enabled in M. The denominator of the fraction above is a sum since it reflects the probability of the events union.

If $tang(M)$ then $PT(\emptyset, M) \in (0;1]$, hence, there is a non-zero probability to stay in the marking in the next time moment, and the residence time in M is at least 1 discrete time unit. If $vanish(M)$ then $PT(\emptyset, M) = 0$, hence, there is zero probability to stay in the marking in the next time moment, and the residence time in M is 0.

Note that for all markings of an LDTSIPN N, the sum of outgoing probabilities is equal to 1. More formally, $\forall M \in I\!N_f^{P_N}\ PT(\emptyset, M) + \sum_{\{U|\bullet U \subseteq M\}} PT(U, M) = 1$. This obviously follows from the definition of $PT(U, M)$ and guarantees that it defines a probability distribution.

The *probability to move from M to $\widetilde{M}$ by firing any set of transitions* is

$$PM(M, \widetilde{M}) = \sum_{\{U|M \stackrel{U}{\rightarrow} \widetilde{M}\}} PT(U, M).$$

Since $PM(M, \widetilde{M})$ is the probability for *any* (including the empty one) transition set to change marking M to $\widetilde{M}$, we use summation in the definition. Note that $\forall M \in RS(N)\ \sum_{\{\widetilde{M}|M \rightarrow \widetilde{M}\}} PM(M, \widetilde{M}) = \sum_{\{\widetilde{M}|M \rightarrow \widetilde{M}\}} \sum_{\{U|M \stackrel{U}{\rightarrow} \widetilde{M}\}} PT(U, M) = \sum_{\{U|\bullet U \subseteq M\}} PT(U, M) = 1$.

We write $M \stackrel{U}{\rightarrow} \widetilde{M}$ if $\exists \mathcal{P} > 0\ M \stackrel{U}{\rightarrow}_{\mathcal{P}} \widetilde{M}$ and $M \rightarrow \widetilde{M}$ if $\exists U\ M \stackrel{U}{\rightarrow} \widetilde{M}$. In addition, we write $M \rightarrow_{\mathcal{P}} \widetilde{M}$, if $M \rightarrow \widetilde{M}$, where $\mathcal{P} = PM(M, \widetilde{M})$.

Definition 4.2 *Let N be an LDTSIPN.*

- *The* reachability set *of N, denoted by $RS(N)$, is the minimal set of markings such that*
  - *$M_N \in RS(N)$;*
  - *if $M \in RS(N)$ and $M \rightarrow \widetilde{M}$ then $\widetilde{M} \in RS(N)$.*
- *The* reachability graph *of N, denoted by $RG(N)$, is a directed labeled graph with the set of nodes $RS(N)$ and the arcs labeled with $(U, \mathcal{P})$ between nodes M and $\widetilde{M}$ iff $M \stackrel{U}{\rightarrow}_{\mathcal{P}} \widetilde{M}$.*

The set of *all tangible markings from* $RS(N)$ is denoted by $RS_T(N)$, and the set of *all vanishing markings from* $RS(N)$ is denoted by $RS_V(N)$. Obviously, $RS(N) = RS_T(N) \cup RS_V(N)$.

4.2 Algebra of dtsi-boxes

Now we introduce discrete time stochastic and immediate Petri boxes and the algebraic operations to define a net representation of dtsiPBC expressions.

Definition 4.3 *A* discrete time stochastic and immediate Petri box (dtsi-box) *is a tuple* $N = (P_N, T_N, W_N, \Lambda_N)$*, where*

- *P_N and T_N are finite sets of* places *and* transitions*, respectively, such that $P_N \cup T_N \neq \emptyset$ and $P_N \cap T_N = \emptyset$;*
- *$W_N : (P_N \times T_N) \cup (T_N \times P_N) \rightarrow I\!N$ is a function providing the* weights of arcs *between places and transitions and vice versa;*
- *Λ_N is the* place and transition labeling *function such that*
  - *$\Lambda_N|_{P_N} : P_N \rightarrow \{\mathsf{e}, \mathsf{i}, \mathsf{x}\}$ (it specifies* entry, internal *and* exit *places, respectively);*
  - *$\Lambda_N|_{T_N} : T_N \rightarrow \{\varrho \mid \varrho \subseteq 2^{\mathcal{SIL}} \times \mathcal{SIL}\}$ (it associates transitions with the* relabeling relations *on activities).*

Moreover, $\forall t \in T_N\ {}^\bullet t \neq \emptyset \neq t^\bullet$. In addition, for the set of entry *places of N, defined as $^\circ N = \{p \in P_N \mid \Lambda_N(p) = \mathsf{e}\}$, and for the set of* exit *places of N, defined as $N^\circ = \{p \in P_N \mid \Lambda_N(p) = \mathsf{x}\}$, the following condition holds: $^\circ N \neq \emptyset \neq N^\circ$, $^\bullet(^\circ N) = \emptyset = (N^\circ)^\bullet$.*

A dtsi-box is *plain* if $\forall t \in T_N\ \Lambda_N(t) \in \mathcal{SIL}$, i.e., $\Lambda_N(t)$ is the constant relabeling that will be defined later. A *marked plain dtsi-box* is a pair (N, M_N), where N is a plain dtsi-box and $M_N \in I\!N_f^{P_N}$ is the *initial marking*. We shall use the following notation: $\overline{N} = (N, {}^\circ N)$ and $\underline{N} = (N, N^\circ)$. Note that a marked plain dtsi-box $(P_N, T_N, W_N, \Lambda_N, M_N)$ could be interpreted as the LDTSIPN $(P_N, T_N, W_N, \Omega_N, L_N, M_N)$, where functions Ω_N and L_N are defined as follows: $\forall t \in T_N\ \Omega_N(t) = \Omega(\Lambda_N(t))$ and $L_N(t) = \mathcal{L}(\Lambda_N(t))$. The behaviour of marked dtsi-boxes follows from the firing rule of LDTSIPNs. A plain dtsi-box N is *n-bounded* ($n \in I\!N$) if $\overline{N}$ is so, i.e., $\forall M \in RS(\overline{N})\ \forall p \in P_N\ M(p) \leq n$, and it is *safe* if it is 1-bounded. A plain dtsi-box N is *clean* if

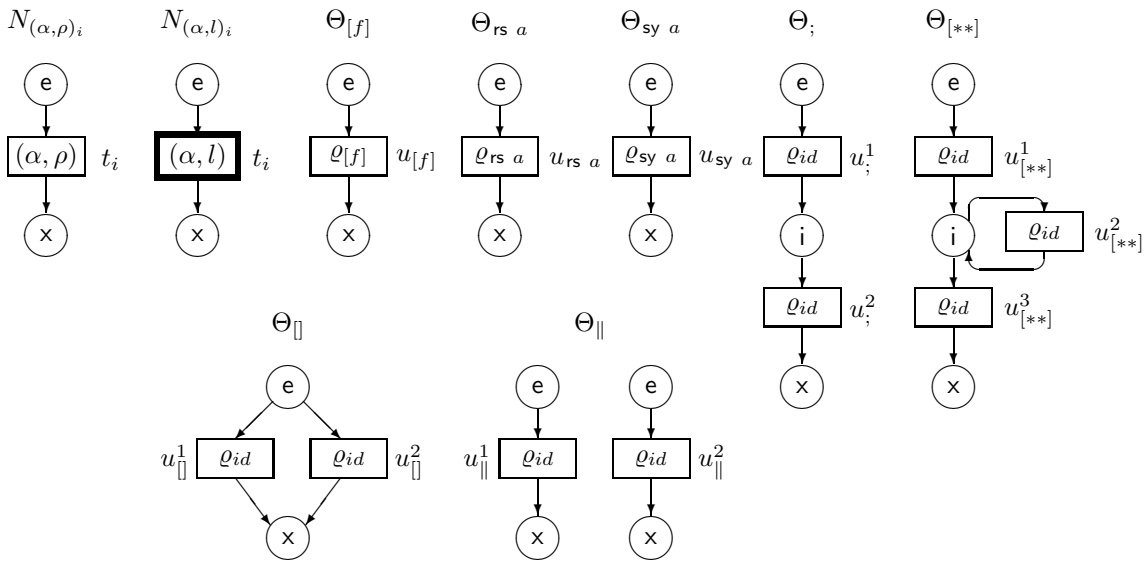

Figure 3: The plain and operator dtsi-boxes

$\forall M \in RS(\overline{N})\ {}^\circ N \subseteq M \ \Rightarrow\ M = {}^\circ N$ and $N^\circ \subseteq M \ \Rightarrow\ M = N^\circ$, i.e., if there are tokens in all its entry (exit) places then no other places have tokens.

To define the semantic function that associates a plain dtsi-box with every static expression of dtsiPBC, we introduce the *enumeration* function $Enu : T_N \to Num$, which associates the numberings with transitions of a plain dtsi-box N in accordance with those of activities. In the case of synchronization, the function associates with the resulting new transition the concatenation of the parenthesized numberings of the transitions it comes from.

The structure of the plain dtsi-box corresponding to a static expression is constructed like in PBC, see [4], i.e., we use simultaneous refinement and relabeling meta-operator (net refinement) in addition to the *operator dtsi-boxes* corresponding to the algebraic operations of dtsiPBC and featuring transformational transition relabelings. Thus, as we shall see in Theorem 4.1, the resulting plain dtsi-boxes are safe and clean. In the definition of the denotational semantics, we shall apply standard constructions used for PBC. Let Θ denotes *operator box* and u denotes *transition name* from PBC setting.

The relabeling relations $\varrho \subseteq 2^{\mathcal{SIL}} \times \mathcal{SIL}$ are defined as follows:

- $\varrho_{id} = \{(\{(\alpha,\kappa)\},(\alpha,\kappa)) \mid (\alpha,\kappa) \in \mathcal{SIL}\}$ is the *identity relabeling* keeping the interface as it is;
- $\varrho_{(\alpha,\kappa)} = \{(\emptyset,(\alpha,\kappa))\}$ is the *constant relabeling* that can be identified with $(\alpha,\kappa) \in \mathcal{SIL}$ itself;
- $\varrho_{[f]} = \{(\{(\alpha,\kappa)\},(f(\alpha),\kappa)) \mid (\alpha,\kappa) \in \mathcal{SIL}\}$;
- $\varrho_{\mathsf{rs}\ a} = \{(\{(\alpha,\kappa)\},(\alpha,\kappa)) \mid (\alpha,\kappa) \in \mathcal{SIL},\ a,\hat{a} \notin \alpha\}$;
- $\varrho_{\mathsf{sy}\ a}$ is the least relabeling relation containing in ϱ_{id} such that if $(\Upsilon,(\alpha,\kappa)),(\Xi,(\beta,\lambda)) \in \varrho_{\mathsf{sy}\ a}$ and $a \in \alpha,\ \hat{a} \in \beta$ then
  - $(\Upsilon + \Xi,(\alpha \oplus_a \beta, \kappa \cdot \lambda)) \in \varrho_{\mathsf{sy}\ a}$, if $\kappa,\lambda \in (0;1)$;
  - $(\Upsilon + \Xi,(\alpha \oplus_a \beta, \kappa + \lambda)) \in \varrho_{\mathsf{sy}\ a}$, if $\kappa,\lambda \in I\!N \setminus \{0\}$.

The plain and operator dtsi-boxes are presented in Figure 3. Note that the label i of internal places is usually omitted.

In the case of the synchronization, a decision that we must take is the selection of the operator box that we shall use for the iteration, since we have two proposals in plain PBC for that purpose (see [4]). One of them provides us with a safe version (with six transitions in the operator box), but there is also a simpler version, which has only three transitions in the operator box. In general, in PBC, with the latter version we may generate 2-bounded nets, which only occurs when a parallel behavior appears at the highest level of the body of the iteration. Nevertheless, in our case, and due to the syntactical restriction introduced for regular terms, this particular case cannot occur, so that the net obtained will be always safe.

Now we define the enumeration function Enu for every operator of dtsiPBC. Let $Box_{dtsi}(E) = (P_E, T_E, W_E, \Lambda_E)$ be the plain dtsi-box corresponding to a static expression E, and Enu_E be the enumeration function for T_E. We shall use the analogous notation for static expressions F and K.

- $Box_{dtsi}(E \circ F) = \Theta_\circ(Box_{dtsi}(E), Box_{dtsi}(F)),\ \circ \in \{;, [], \|\}$. Since we do not introduce new transitions, we preserve the initial numbering:

$$Enu(t) = \begin{cases} Enu_E(t), & t \in T_E; \\ Enu_F(t), & t \in T_F. \end{cases}$$

- $Box_{dtsi}(E[f]) = \Theta_{[f]}(Box_{dtsi}(E))$. Since we only replace the labels of some multiactions by a bijection, we preserve the initial numbering:

$$Enu(t) = Enu_E(t),\ t \in T_E.$$

- $Box_{dtsi}(E\ \mathsf{rs}\ a) = \Theta_{\mathsf{rs}\ a}(Box_{dtsi}(E))$. Since we remove all transitions labeled with multiactions containing a or $\hat{a}$, this does not change the numbering of the remaining transitions:

$$Enu(t) = Enu_E(t),\ t \in T_E,\ a, \hat{a} \notin \mathcal{L}(\Lambda_E(t)).$$

- $Box_{dtsi}(E\ \mathsf{sy}\ a) = \Theta_{\mathsf{sy}\ a}(Box_{dtsi}(E))$. Note that $\forall v, w \in T_E$ such that $\Lambda_E(v) = (\alpha, \kappa),\ \Lambda_E(w) = (\beta, \lambda)$ and $a \in \alpha,\ \hat{a} \in \beta$, the new transition t resulting from synchronization of v and w has the label $\Lambda(t) = (\alpha \oplus_a \beta, \kappa \cdot \lambda)$, if t is stochastic transition, or $\Lambda(t) = (\alpha \oplus_a \beta, \kappa + \lambda)$, if t is immediate one, and the numbering $Enu(t) = (Enu_E(v))(Enu_E(w))$.

  Thus, the enumeration function is defined as

$$Enu(t) = \begin{cases} Enu_E(t), & t \in T_E; \\ (Enu_E(v))(Enu_E(w)), & t \text{ results from synchronization of } v \text{ and } w. \end{cases}$$

  According to the definition of $\varrho_{\mathsf{sy}\ a}$, the synchronization is only possible when all the transitions in the set are stochastic or when all of them are immediate. If we synchronize the same set of transitions in different orders, we obtain several resulting transitions with the same label and probability or weight, but with the different numberings having the same content. Then, we only consider a single one from the resulting transitions in the plain dtsi-box to avoid introducing redundant transitions.

  For example, if the transitions t and u are generated by synchronizing v and w in different orders, we have $\Lambda(t) = (\alpha \oplus_a \beta, \kappa \cdot \lambda) = \Lambda(u)$ for stochastic transitions or $\Lambda(t) = (\alpha \oplus_a \beta, \kappa + \lambda) = \Lambda(u)$ for immediate ones, but $Enu(t) = (Enu_E(v))(Enu_E(w))$, whereas $Enu(u) = (Enu_E(w))(Enu_E(v))$ with $Cont(Enu(t)) = Cont(Enu(v)) \cup Cont(Enu(w)) = Cont(Enu(u))$. Then only one transition t (or, symmetrically, u) will appear in $Box_{dtsi}(E\ \mathsf{sy}\ a)$.

- $Box_{dtsi}([E * F * K]) = \Theta_{[**]}(Box_{dtsi}(E), Box_{dtsi}(F), Box_{dtsi}(K))$. Since we do not introduce new transitions, we preserve the initial numbering:

$$Enu(t) = \begin{cases} Enu_E(t), & t \in T_E; \\ Enu_F(t), & t \in T_F; \\ Enu_K(t), & t \in T_K. \end{cases}$$

Now we can formally define the denotational semantics as a homomorphism.

Definition 4.4 *Let $(\alpha, \kappa) \in \mathcal{SIL}$, $a \in Act$ and $E, F, K \in RegStatExpr$. The* denotational semantics *of dtsiPBC is a mapping Box_{dtsi} from RegStatExpr into the domain of plain dtsi-boxes defined as follows:*

1. $Box_{dtsi}((\alpha, \kappa)_i) = N_{(\alpha,\kappa)_i}$;
2. $Box_{dtsi}(E \circ F) = \Theta_\circ(Box_{dtsi}(E), Box_{dtsi}(F)),\ \circ \in \{;, [], \|\}$;
3. $Box_{dtsi}(E[f]) = \Theta_{[f]}(Box_{dtsi}(E))$;
4. $Box_{dtsi}(E \circ a) = \Theta_{\circ a}(Box_{dtsi}(E)),\ \circ \in \{\mathsf{rs}, \mathsf{sy}\}$;
5. $Box_{dtsi}([E * F * K]) = \Theta_{[**]}(Box_{dtsi}(E), Box_{dtsi}(F), Box_{dtsi}(K))$.

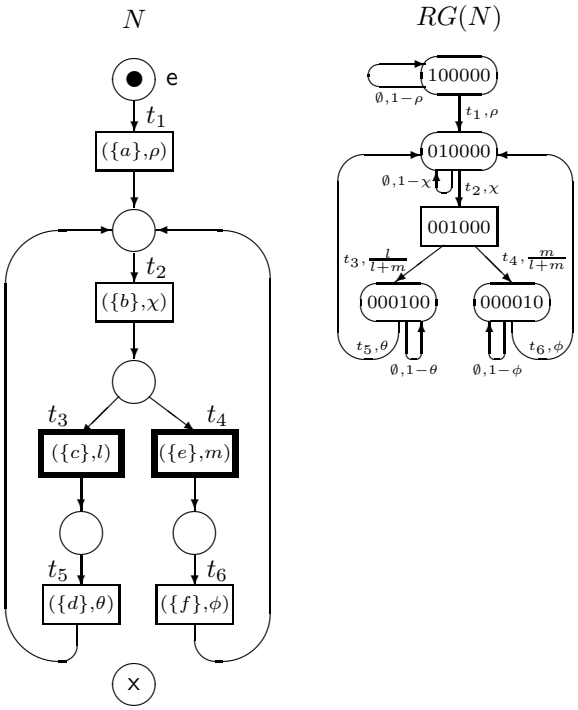

Figure 4: The marked dtsi-box $N = Box_{dtsi}(\overline{E})$ for $E = [(\{a\}, \rho) * ((\{b\}, \chi); ((((\{c\}, l); (\{d\}, \theta))[]((\{e\}, m); (\{f\}, \phi)))) * \mathsf{Stop}]$ and its reachability graph

The dtsi-boxes of dynamic expressions can be defined as well. For $E \in RegStatExpr$, let $Box_{dtsi}(\overline{E}) = \overline{Box_{dtsi}(E)}$ and $Box_{dtsi}(\underline{E}) = \underline{Box_{dtsi}(E)}$.

Observe that this definition is compositional in the sense that for any arbitrary dynamic expression, we may decompose it in some inner dynamic and static expressions, for which we may apply the definition, thus obtaining the corresponding plain dtsi-boxes, which can be joined according to the term structure (definition of Box_{dtsi}), the resulting plain box being marked in the places that were marked in the argument nets.

Theorem 4.1 *For any static expression E $Box_{dtsi}(\overline{E})$ is safe and clean.*

Proof. The structure of the net is obtained as in PBC, combining both refinement and relabeling. Consequently, the dtsi-boxes thus obtained will be safe and clean. □

Let $\simeq$ denote isomorphism between transition systems and reachability graphs that relates their initial states. Note that the names of transitions of the dtsi-box corresponding to a static expression could be identified with the enumerated activities of the latter.

Theorem 4.2 *For any static expression E*

$$TS(\overline{E}) \simeq RG(Box_{dtsi}(\overline{E})).$$

Proof. As for the qualitative (functional) behaviour, we have the same isomorphism as in PBC.

The quantitative behaviour is the same by the following reasons. First, the activities of an expression have the probability or weight parts coinciding with the probabilities or weights of the transitions belonging to the corresponding dtsi-box. Second, we use analogous probability or weight functions to construct the corresponding transition systems and reachability graphs. □

Example 4.1 *Let E be from Example 3.2. In Figure 4 the marked dtsi-box $N = Box_{dtsi}(\overline{E})$ and its reachability graph $RG(N)$ are presented. It is easy to see that $TS(\overline{E})$ and $RG(N)$ are isomorphic*

5 Performance evaluation

In this section we demonstrate how Markov chains corresponding to the expressions and dtsi-boxes can be constructed and then used for performance evaluation.

For a dynamic expression G, a discrete random variable is associated with every tangible state from $DR(G)$. The variable captures a residence time in the state. One can interpret staying in a state in the next discrete time moment as a failure and leaving it as a success of some trial series. It is easy to see that the random variables are geometrically distributed, since the probability to stay in a tangible state s for $k-1$ time moments and leave it at moment $k \geq 1$ is $PM(s,s)^{k-1}(1-PM(s,s))$ (the residence time is k in this case). The mean value formula for geometrical distribution allows us to calculate the average sojourn time in a tangible state s as $\frac{1}{1-PM(s,s)}$. Obviously, the average sojourn time in a vanishing state is zero. Thus, the *average sojourn time in the state* s is

$$SJ(s) = \begin{cases} \frac{1}{1-PM(s,s)}, & s \in DR_T(G); \\ 0, & s \in DR_V(G). \end{cases}$$

The *average sojourn time vector* of G, denoted by SJ, is that with the elements $SJ(s),\ s \in DR(G)$.

To evaluate performance of the process specified by a dynamic expression G, we should investigate the stochastic process associated with it. The process is the underlying semi-Markov chain (SMC) [23], denoted by $SMC(G)$, which can be analyzed by extracting from it the embedded (absorbing) discrete time Markov chain (EDTMC) corresponding to G, denoted by $EDTMC(G)$. The construction of the latter is similar to that applied in the context of generalized stochastic Petri nets (GSPNs) in [17, 1, 2], and also in the framework of discrete time deterministic and stochastic Petri nets (DTDSPNs) in [27]. $EDTMC(G)$ describes only the state changes of $SMC(G)$ while ignoring any time characteristics.

Thus, to construct the EDTMC, we should abstract from all time aspects of behaviour, i.e., from the sojourn time in the states. The sojourn time in every state of the EDTMC is deterministic and it equals to one discrete time unit. Let G be a dynamic expression. A transition system $TS(G)$ can have self-loops going from a state to itself which have a non-zero probability. Obviously, the current state remains unchanged in this case.

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

The *probability to stay in* s *due to* k $(k \geq 1)$ *self-loops* is

$$(PM(s,s))^k.$$

Let $s \to \tilde{s}$ and $s \neq \tilde{s}$. The *probability to move from* s *to* $\tilde{s}$ *by executing any set of activities after possible self-loops* is

$$PM^*(s,\tilde{s}) = PM(s,\tilde{s}) \sum_{k=0}^{\infty} (PM(s,s))^k = \frac{PM(s,\tilde{s})}{1-PM(s,s)}.$$

The value $k=0$ in the summation above corresponds to the case when no self-loops occur. Note that if $s \in DR_T(G)$ then $PM^*(s,\tilde{s}) = SJ(s)PM(s,\tilde{s})$.

Notice that after abstraction from the probabilities of transitions which do not change the states, the remaining transition probabilities are normalized. In order to calculate transition probabilities $PT(\Upsilon, s)$, we had to normalize $PF(\Upsilon, s)$. Then, to obtain transition probabilities of the state-changing steps $PM^*(s,\tilde{s})$, we now have to normalize $PM(s,\tilde{s})$. Thus, we have a two-stage normalization as a result.

Note that $PM^*(s,\tilde{s}) \leq 1$, i.e., it is really a probability, since for $s \neq \tilde{s}$ we have $PM(s,s) + PM(s,\tilde{s}) \leq \sum_{\{\bar{s} | s \to \bar{s}\}} PM(s,\bar{s}) = 1$, hence, $PM(s,\tilde{s}) \leq 1 - PM(s,s)$. Moreover, $PM^*(s,\tilde{s})$ defines a probability distribution, since $\forall s \in DR(G)$ such that s is not a terminal state, i.e., there exist transitions from it, we have $\sum_{\{\tilde{s} | s \to \tilde{s} \wedge s \neq \tilde{s}\}} PM^*(s,\tilde{s}) = \frac{1}{1-PM(s,s)} \sum_{\{\tilde{s} | s \to \tilde{s} \wedge s \neq \tilde{s}\}} PM(s,\tilde{s}) = \frac{1}{1-PM(s,s)}(1-PM(s,s)) = 1$.

We decided to consider self-loops followed only by a state-changing step just for convenience. Alternatively, we could take a state-changing step succeeded by self-loops or a state-changing step preceded and succeeded by self-loops. In all these three cases our sequence begins or/and ends with the loops which do not change states. At the same time, the overall probabilities of the evolutions can differ, since self-loops have positive probabilities. To avoid inconsistency of definitions and too complex description, we consider sequences ending with a state-changing step. It resembles in some sense a construction of branching bisimulation [7] if to take self-loops instead of silent transitions.

Definition 5.1 *Let G be a dynamic expression. The* embedded (absorbing) discrete time Markov chain (EDTMC) *of G, denoted by $EDTMC(G)$, has the state space $DR(G)$ and the transitions $s \twoheadrightarrow_{\mathcal{P}} \tilde{s}$, if $s \to \tilde{s}$ and $s \neq \tilde{s}$, where $\mathcal{P} = PM^*(s,\tilde{s})$.*

EDTMCs of static expressions can be defined as well. For $E \in RegStatExpr$, let $EDTMC(E) = EDTMC(\overline{E})$.

Let G be a dynamic expression. The elements $\mathcal{P}^*_{ij}$ $(1 \le i,j \le n = |DR(G)|)$ of (one-step) transition probability matrix (TPM) $\mathbf{P}^*$ for $EDTMC(G)$ are defined as

$$\mathcal{P}^*_{ij} = \begin{cases} PM^*(s_i,s_j), & s_i \to s_j \wedge s_i \neq s_j; \\ 0, & \text{otherwise.} \end{cases}$$

The transient (k-step, $k \in \mathbb{N}$) probability mass function (PMF) $\psi^*[k] = (\psi^*[k](s_1),\ldots,\psi^*[k](s_n))$ for $EDTMC(G)$ is the solution of the equation system

$$\psi^*[k] = \psi^*[0](\mathbf{P}^*)^k,$$

where $\psi^*[0] = (\psi^*[0](s_1),\ldots,\psi^*[0](s_n))$ is the initial PMF defined as

$$\psi^*[0](s_i) = \begin{cases} 1, & s_i = [G]_{\approx}; \\ 0, & \text{otherwise.} \end{cases}$$

Note also that $\psi^*[k+1] = \psi^*[k]\,\mathbf{P}^*$ $(k \in \mathbb{N})$.

The steady-state PMF $\psi^* = (\psi^*(s_1),\ldots,\psi^*(s_n))$ for $EDTMC(G)$ is the solution of the equation system

$$\begin{cases} \psi^*(\mathbf{P}^* - \mathbf{E}) = \mathbf{0} \\ \psi^*\mathbf{1}^T = 1 \end{cases},$$

where $\mathbf{E}$ is the unitary matrix of dimension n and $\mathbf{0}$ is a vector with n values 0, $\mathbf{1}$ is that with n values 1.

When $EDTMC(G)$ has the single steady state, we have $\psi^* = \lim_{k\to\infty}\psi^*[k]$.

The steady-state PMF for the underlying semi-Markov chain $SMC(G)$ is calculated via multiplication of every $\psi^*(s_i)$ $(1 \le i \le n)$ by the average sojourn time $SJ(s_i)$ in the state s_i, after which we normalize the resulting values. Remember that for a vanishing state $s \in DR_V(G)$ we have $SJ(s) = 0$.

Thus, the steady-state PMF $\varphi = (\varphi(s_1),\ldots,\varphi(s_n))$ for $SMC(G)$ is

$$\varphi(s_i) = \begin{cases} \frac{\psi^*(s_i)SJ(s_i)}{\sum_{j=1}^{n}\psi^*(s_j)SJ(s_j)}, & s_i \in DR_T(G); \\ 0, & s_i \in DR_V(G). \end{cases}$$

Example 5.1 *Let E be from Example 3.2. In Figure 5 the underlying SMC $SMC(\overline{E})$ is presented. Average sojourn time in the states of the underlying SMC is written next to them in bold font.*

The average sojourn time vector is

$$SJ = \left(\frac{1}{\rho}, \frac{1}{\chi}, 0, \frac{1}{\theta}, \frac{1}{\phi},\right).$$

The TPM for $EDTMC(\overline{E})$ is

$$\mathbf{P}^* = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & \frac{l}{l+m} & \frac{m}{l+m} \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix}.$$

The steady-state PMF for $EDTMC(\overline{E})$ is

$$\psi^* = \left(0, \frac{1}{3}, \frac{1}{3}, \frac{l}{3(l+m)}, \frac{m}{3(l+m)}\right).$$

The steady-state PMF ψ^* weighted by SJ is

$$\left(0, \frac{1}{3\chi}, 0, \frac{l}{3\theta(l+m)}, \frac{m}{3\phi(l+m)}\right).$$

It remains to normalize the steady-state weighted PMF dividing it by the sum of its components

$$\psi^* SJ^T = \frac{\theta\phi(l+m) + \chi(\phi l + \theta m)}{3\chi\theta\phi(l+m)}.$$

Thus, the steady-state PMF for $SMC(\overline{E})$ is

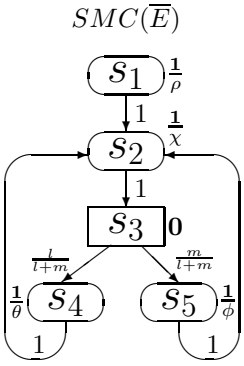

Figure 5: The underlying SMC of $\overline{E}$ for $E = [(\{a\}, \rho) * (((\{b\}, \chi); ((((\{c\}, l); (\{d\}, \theta))[](\{e\}, m); (\{f\}, \phi)))) * \mathsf{Stop}]$

$$\varphi = \frac{1}{\theta\phi(l+m) + \chi(\phi l + \theta m)} \, (0, \theta\phi(l+m), 0, \chi\phi l, \chi\theta m) \, .$$

In the case $l = m$ and $\theta = \phi$ we have

$$\varphi = \frac{1}{2(\chi+\theta)} (0, 2\theta, 0, \chi, \chi).$$

Let G be a dynamic expression and $s, \tilde{s} \in DR(G)$. The following standard *performance indices (measures)* can be calculated based on the steady-state PMF for $SMC(G)$, see, in particular, [21].

- The *average recurrence (return) time in the state s* (i.e., the number of discrete time units or steps required for this) is $\frac{1}{\varphi(s)}$.
- The *fraction of residence time in the state s* is $\varphi(s)$.
- The *relative fraction of residence time in the state s with respect to that in the state $\tilde{s}$* is $\frac{\varphi(s)}{\varphi(\tilde{s})}$.
- The *fraction of residence time in the set of states $S \subseteq DR(G)$* or the *probability of the event determined by a condition that is true for all states from S* is $\sum_{s \in S} \varphi(s)$.
- The *rate of leaving the state s* is $\frac{\varphi(s)}{SJ(s)}$.
- The *steady-state probability to perform a step with an activity (α, κ)* is $\sum_{s \in DR(G)} \varphi(s) \sum_{\{\Upsilon \mid (\alpha,\kappa) \in \Upsilon\}} PT(\Upsilon, s)$.
- The *probability of the event determined by a reward function r on the states* is $\sum_{s \in DR(G)} \varphi(s) r(s)$.

Let $N = (P_N, T_N, W_N, \Omega_N, L_N, M_N)$ be a LDTSIPN and $M, \widetilde{M} \in I\!N_f^{P_N}$. Then the average sojourn time $SJ(M)$, the probabilities $PM^*(M, \widetilde{M})$, the transition relation $M \twoheadrightarrow_{\mathcal{P}} \widetilde{M}$, the *EDTMC* $EDTMC(N)$, the *underlying SMC* $SMC(N)$ and the steady-state PMF for it are defined like the corresponding notions for dynamic expressions.

As we mentioned earlier, every marked plain dtsi-box could be interpreted as the LDTSIPN. Therefore, we can evaluate performance with the LDTSIPNs corresponding to dtsi-boxes and then transfer the results to the latter.

Let $\simeq$ denote isomorphism between SMCs that relates their initial states.

Proposition 5.1 *For any static expression E*

$$SMC(\overline{E}) \simeq SMC(Box_{dtsi}(\overline{E})).$$

Proof. By Theorem 4.2 and definitions of underlying SMCs for dynamic expressions and LDTSIPNs taking into account the following. First, for the associated SMCs, the average sojourn time in the states is the same since it is defined via the analogous probability functions. Second, the transition probabilities of the associated SMCs are the sums of those belonging to transition systems or reachability graphs. □

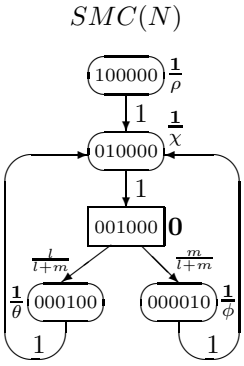

Figure 6: The underlying SMC of $N = Box_{dtsi}(\overline{E})$ for $E = [(\{a\}, \rho) * ((\{b\}, \chi); ((((\{c\}, l); (\{d\}, \theta))[]((\{e\}, m); (\{f\}, \phi)))) * \mathsf{Stop}]$

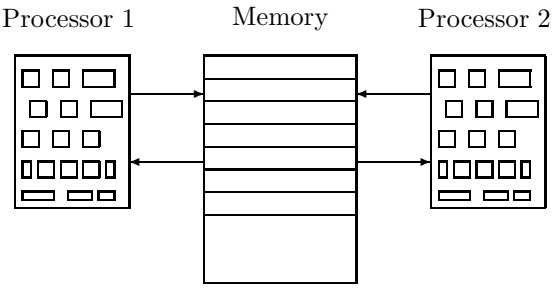

Figure 7: The diagram of the shared memory system

Example 5.2 *Let E be from Example 3.2. In Figure 6 the underlying SMC $SMC(N)$ is presented. It is easy to see that $SMC(\overline{E})$ and $SMC(N)$ are isomorphic. Thus, the steady-state PMF for $SMC(N)$ is the same as for $SMC(\overline{E})$.*

6 Shared memory system

In this section with a case study of the shared memory system we demonstrate how steady-state distribution can be used for performance evaluation. The example also illustrates the method of performance analysis simplification described above.

Consider a model of two processors accessing a common shared memory described in [18, 1, 2] in the continuous time setting on GSPNs. We shall analyze this shared memory system in the discrete time stochastic setting of dtsPBC where concurrent execution of activities is possible. The model performs as follows. After activation of the system (turning the computer on), two processors are active, and the common memory is available. Each processor can request an access to the memory after which it waits for the instantaneous decision. When the decision is made in favour of a processor, it starts an acquisition of the memory and another processor should wait until the former one ends its memory operations, and the system returns to the state with both active processors and the available common memory. The diagram of the system is depicted in Figure 7.

Let us explain the meaning of actions from syntax of the dtsPBC expressions which will specify the system modules. The action a corresponds to the system activation. The actions r_i $(1 \leq i \leq 2)$ represent the common memory request of processor i. The instantaneous action d_i corresponds to the decision on the memory allocation in favour of the processor i. The action m_i represents the common memory access of processor i. The other actions are used for communication purpose only via synchronization, and we abstract from them later using restriction.

The static expression of the first processor is

$$E_1 = [(\{x_1\}, \frac{1}{2}) * ((\{r_1\}, \frac{1}{2}); (\{d_1, y_1\}, 1); (\{m_1, z_1\}, \frac{1}{2})) * \mathsf{Stop}].$$

The static expression of the second processor is

$$E_2 = [(\{x_2\}, \frac{1}{2}) * ((\{r_2\}, \frac{1}{2}); (\{d_2, y_2\}, 1); (\{m_2, z_2\}, \frac{1}{2})) * \mathsf{Stop}].$$

The static expression of the shared memory is

$$E_3 = [(\{a, \widehat{x_1}, \widehat{x_2}\}, \frac{1}{2}) * ((((\{\widehat{y_1}\}, 1); (\{\widehat{z_1}\}, \frac{1}{2}))[]((\{\widehat{y_2}\}, 1); (\{\widehat{z_2}\}, \frac{1}{2}))) * \mathsf{Stop}].$$

The static expression of the shared memory system with two processors is

$$E = (E_1 \| E_2 \| E_3)\ \mathsf{sy}\ x_1\ \mathsf{sy}\ x_2\ \mathsf{sy}\ y_1\ \mathsf{sy}\ y_2\ \mathsf{sy}\ z_1\ \mathsf{sy}\ z_2\ \mathsf{rs}\ x_1\ \mathsf{rs}\ x_2\ \mathsf{rs}\ y_1\ \mathsf{rs}\ y_2\ \mathsf{rs}\ z_1\ \mathsf{rs}\ z_2.$$

Let us illustrate an effect of synchronization. In the result of synchronization of immediate multiactions $(\{d_i, y_i\}, 1)$ and $(\{\widehat{y_i}\}, 1)$ we obtain $(\{d_i\}, 2)$ $(1 \le i \le 2)$. The synchronization of stochastic multiactions $(\{m_i, z_i\}, \frac{1}{2})$ and $(\{\widehat{z_i}\}, \frac{1}{2})$ produces $(\{m_i\}, \frac{1}{4})$ $(1 \le i \le 2)$. The result of synchronization of $(\{a, \widehat{x_1}, \widehat{x_2}\}, \frac{1}{2})$ with $(\{x_1\}, \frac{1}{2})$ is $(\{a, \widehat{x_2}\}, \frac{1}{4})$, and that of synchronization of $(\{a, \widehat{x_1}, \widehat{x_2}\}, \frac{1}{2})$ with $(\{x_2\}, \frac{1}{2})$ is $(\{a, \widehat{x_1}\}, \frac{1}{4})$. After applying synchronization to $(\{a, \widehat{x_2}\}, \frac{1}{4})$ and $(\{x_2\}, \frac{1}{2})$, as well as to $(\{a, \widehat{x_1}\}, \frac{1}{4})$ and $(\{x_1\}, \frac{1}{2})$ we obtain the same activity $(\{a\}, \frac{1}{8})$.

$DR(\overline{E})$ consists of the equivalence classes

$s_1 = [([\overline{(\{x_1\}, \frac{1}{2})} * ((\{r_1\}, \frac{1}{2}); (\{d_1, y_1\}, 1); (\{m_1, z_1\}, \frac{1}{2})) * \mathsf{Stop}] \| [\overline{(\{x_2\}, \frac{1}{2})} * ((\{r_2\}, \frac{1}{2}); (\{d_2, y_2\}, 1); (\{m_2, z_2\}, \frac{1}{2})) * \mathsf{Stop}] \| [\overline{(\{a, \widehat{x_1}, \widehat{x_2}\}, \frac{1}{2})} * ((((\{\widehat{y_1}\}, 1); (\{\widehat{z_1}\}, \frac{1}{2}))[]((\{\widehat{y_2}\}, 1); (\{\widehat{z_2}\}, \frac{1}{2}))) * \mathsf{Stop}])\ \mathsf{sy}\ x_1\ \mathsf{sy}\ x_2\ \mathsf{sy}\ y_1\ \mathsf{sy}\ y_2\ \mathsf{sy}\ z_1\ \mathsf{sy}\ z_2\ \mathsf{rs}\ x_1\ \mathsf{rs}\ x_2\ \mathsf{rs}\ y_1\ \mathsf{rs}\ y_2\ \mathsf{rs}\ z_1\ \mathsf{rs}\ z_2]_{\approx}$,

$s_2 = [([\{x_1\}, \frac{1}{2}) * \overline{((\{r_1\}, \frac{1}{2}); (\{d_1, y_1\}, 1); (\{m_1, z_1\}, \frac{1}{2}))} * \mathsf{Stop}] \| [(\{x_2\}, \frac{1}{2}) * \overline{((\{r_2\}, \frac{1}{2}); (\{d_2, y_2\}, 1); (\{m_2, z_2\}, \frac{1}{2}))} * \mathsf{Stop}] \| [(\{a, \widehat{x_1}, \widehat{x_2}\}, \frac{1}{2}) * ((((\{\widehat{y_1}\}, 1); (\{\widehat{z_1}\}, \frac{1}{2}))[]((\{\widehat{y_2}\}, 1); (\{\widehat{z_2}\}, \frac{1}{2}))) * \mathsf{Stop}])\ \mathsf{sy}\ x_1\ \mathsf{sy}\ x_2\ \mathsf{sy}\ y_1\ \mathsf{sy}\ y_2\ \mathsf{sy}\ z_1\ \mathsf{sy}\ z_2\ \mathsf{rs}\ x_1\ \mathsf{rs}\ x_2\ \mathsf{rs}\ y_1\ \mathsf{rs}\ y_2\ \mathsf{rs}\ z_1\ \mathsf{rs}\ z_2]_{\approx}$,

$s_3 = [([\{x_1\}, \frac{1}{2}) * ((\{r_1\}, \frac{1}{2}); \overline{(\{d_1, y_1\}, 1)}; (\{m_1, z_1\}, \frac{1}{2})) * \mathsf{Stop}] \| [(\{x_2\}, \frac{1}{2}) * \overline{((\{r_2\}, \frac{1}{2}); (\{d_2, y_2\}, 1); (\{m_2, z_2\}, \frac{1}{2}))} * \mathsf{Stop}] \| [(\{a, \widehat{x_1}, \widehat{x_2}\}, \frac{1}{2}) * (\overline{(\{\widehat{y_1}\}, 1)}; (\{\widehat{z_1}\}, \frac{1}{2}))[]((\{\widehat{y_2}\}, 1); (\{\widehat{z_2}\}, \frac{1}{2}))) * \mathsf{Stop}])\ \mathsf{sy}\ x_1\ \mathsf{sy}\ x_2\ \mathsf{sy}\ y_1\ \mathsf{sy}\ y_2\ \mathsf{sy}\ z_1\ \mathsf{sy}\ z_2\ \mathsf{rs}\ x_1\ \mathsf{rs}\ x_2\ \mathsf{rs}\ y_1\ \mathsf{rs}\ y_2\ \mathsf{rs}\ z_1\ \mathsf{rs}\ z_2]_{\approx}$,

$s_4 = [([\{x_1\}, \frac{1}{2}) * \overline{((\{r_1\}, \frac{1}{2}); (\{d_1, y_1\}, 1); (\{m_1, z_1\}, \frac{1}{2}))} * \mathsf{Stop}] \| [(\{x_2\}, \frac{1}{2}) * ((\{r_2\}, \frac{1}{2}); \overline{(\{d_2, y_2\}, 1)}; (\{m_2, z_2\}, \frac{1}{2})) * \mathsf{Stop}] \| [(\{a, \widehat{x_1}, \widehat{x_2}\}, \frac{1}{2}) * ((((\{\widehat{y_1}\}, 1); (\{\widehat{z_1}\}, \frac{1}{2}))[](\overline{(\{\widehat{y_2}\}, 1)}; (\{\widehat{z_2}\}, \frac{1}{2}))) * \mathsf{Stop}])\ \mathsf{sy}\ x_1\ \mathsf{sy}\ x_2\ \mathsf{sy}\ y_1\ \mathsf{sy}\ y_2\ \mathsf{sy}\ z_1\ \mathsf{sy}\ z_2\ \mathsf{rs}\ x_1\ \mathsf{rs}\ x_2\ \mathsf{rs}\ y_1\ \mathsf{rs}\ y_2\ \mathsf{rs}\ z_1\ \mathsf{rs}\ z_2]_{\approx}$,

$s_5 = [([\{x_1\}, \frac{1}{2}) * ((\{r_1\}, \frac{1}{2}); (\{d_1, y_1\}, 1); \overline{(\{m_1, z_1\}, \frac{1}{2})}) * \mathsf{Stop}] \| [(\{x_2\}, \frac{1}{2}) * \overline{((\{r_2\}, \frac{1}{2}); (\{d_2, y_2\}, 1); (\{m_2, z_2\}, \frac{1}{2}))} * \mathsf{Stop}] \| [(\{a, \widehat{x_1}, \widehat{x_2}\}, \frac{1}{2}) * ((((\{\widehat{y_1}\}, 1); \overline{(\{\widehat{z_1}\}, \frac{1}{2})})[]((\{\widehat{y_2}\}, 1); (\{\widehat{z_2}\}, \frac{1}{2}))) * \mathsf{Stop}])\ \mathsf{sy}\ x_1\ \mathsf{sy}\ x_2\ \mathsf{sy}\ y_1\ \mathsf{sy}\ y_2\ \mathsf{sy}\ z_1\ \mathsf{sy}\ z_2\ \mathsf{rs}\ x_1\ \mathsf{rs}\ x_2\ \mathsf{rs}\ y_1\ \mathsf{rs}\ y_2\ \mathsf{rs}\ z_1\ \mathsf{rs}\ z_2]_{\approx}$,

$s_6 = [([\{x_1\}, \frac{1}{2}) * ((\{r_1\}, \frac{1}{2}); \overline{(\{d_1, y_1\}, 1)}; (\{m_1, z_1\}, \frac{1}{2})) * \mathsf{Stop}] \| [(\{x_2\}, \frac{1}{2}) * ((\{r_2\}, \frac{1}{2}); \overline{(\{d_2, y_2\}, 1)}; (\{m_2, z_2\}, \frac{1}{2})) * \mathsf{Stop}] \| [(\{a, \widehat{x_1}, \widehat{x_2}\}, \frac{1}{2}) * (\overline{(\{\widehat{y_1}\}, 1)}; (\{\widehat{z_1}\}, \frac{1}{2}))[](\overline{(\{\widehat{y_2}\}, 1)}; (\{\widehat{z_2}\}, \frac{1}{2}))) * \mathsf{Stop}])\ \mathsf{sy}\ x_1\ \mathsf{sy}\ x_2\ \mathsf{sy}\ y_1\ \mathsf{sy}\ y_2\ \mathsf{sy}\ z_1\ \mathsf{sy}\ z_2\ \mathsf{rs}\ x_1\ \mathsf{rs}\ x_2\ \mathsf{rs}\ y_1\ \mathsf{rs}\ y_2\ \mathsf{rs}\ z_1\ \mathsf{rs}\ z_2]_{\approx}$,

$s_7 = [([\{x_1\}, \frac{1}{2}) * \overline{((\{r_1\}, \frac{1}{2}); (\{d_1, y_1\}, 1); (\{m_1, z_1\}, \frac{1}{2}))} * \mathsf{Stop}] \| [(\{x_2\}, \frac{1}{2}) * ((\{r_2\}, \frac{1}{2}); (\{d_2, y_2\}, 1); \overline{(\{m_2, z_2\}, \frac{1}{2})}) * \mathsf{Stop}] \| [(\{a, \widehat{x_1}, \widehat{x_2}\}, \frac{1}{2}) * ((((\{\widehat{y_1}\}, 1); (\{\widehat{z_1}\}, \frac{1}{2}))[]((\{\widehat{y_2}\}, 1); \overline{(\{\widehat{z_2}\}, \frac{1}{2})})) * \mathsf{Stop}])\ \mathsf{sy}\ x_1\ \mathsf{sy}\ x_2\ \mathsf{sy}\ y_1\ \mathsf{sy}\ y_2\ \mathsf{sy}\ z_1\ \mathsf{sy}\ z_2\ \mathsf{rs}\ x_1\ \mathsf{rs}\ x_2\ \mathsf{rs}\ y_1\ \mathsf{rs}\ y_2\ \mathsf{rs}\ z_1\ \mathsf{rs}\ z_2]_{\approx}$,

$s_8 = [([\{x_1\}, \frac{1}{2}) * ((\{r_1\}, \frac{1}{2}); (\{d_1, y_1\}, 1); \overline{(\{m_1, z_1\}, \frac{1}{2})}) * \mathsf{Stop}] \| [(\{x_2\}, \frac{1}{2}) * ((\{r_2\}, \frac{1}{2}); (\{d_2, y_2\}, 1); (\{m_2, z_2\}, \frac{1}{2})) * \mathsf{Stop}] \| [(\{a, \widehat{x_1}, \widehat{x_2}\}, \frac{1}{2}) * ((((\{\widehat{y_1}\}, 1); \overline{(\{\widehat{z_1}\}, \frac{1}{2})})[]((\{\widehat{y_2}\}, 1); (\{\widehat{z_2}\}, \frac{1}{2}))) * \mathsf{Stop}])\ \mathsf{sy}\ x_1\ \mathsf{sy}\ x_2\ \mathsf{sy}\ y_1\ \mathsf{sy}\ y_2\ \mathsf{sy}\ z_1\ \mathsf{sy}\ z_2\ \mathsf{rs}\ x_1\ \mathsf{rs}\ x_2\ \mathsf{rs}\ y_1\ \mathsf{rs}\ y_2\ \mathsf{rs}\ z_1\ \mathsf{rs}\ z_2]_{\approx}$,

$s_9 = [([\{x_1\}, \frac{1}{2}) * ((\{r_1\}, \frac{1}{2}); (\{d_1, y_1\}, 1); (\{m_1, z_1\}, \frac{1}{2})) * \mathsf{Stop}] \| [(\{x_2\}, \frac{1}{2}) * ((\{r_2\}, \frac{1}{2}); (\{d_2, y_2\}, 1); \overline{(\{m_2, z_2\}, \frac{1}{2})}) * \mathsf{Stop}] \| [(\{a, \widehat{x_1}, \widehat{x_2}\}, \frac{1}{2}) * ((((\{\widehat{y_1}\}, 1); (\{\widehat{z_1}\}, \frac{1}{2}))[]((\{\widehat{y_2}\}, 1); \overline{(\{\widehat{z_2}\}, \frac{1}{2})})) * \mathsf{Stop}])\ \mathsf{sy}\ x_1\ \mathsf{sy}\ x_2\ \mathsf{sy}\ y_1\ \mathsf{sy}\ y_2\ \mathsf{sy}\ z_1\ \mathsf{sy}\ z_2\ \mathsf{rs}\ x_1\ \mathsf{rs}\ x_2\ \mathsf{rs}\ y_1\ \mathsf{rs}\ y_2\ \mathsf{rs}\ z_1\ \mathsf{rs}\ z_2]_{\approx}$.

We have $DR_T(\overline{E}) = \{s_1, s_2, s_5, s_5, s_8, s_9\}$ and $DR_V(\overline{E}) = \{s_3, s_4, s_6\}$.

The states are interpreted as follows: s_1 is the initial state, s_2: the system is activated and the memory is not requested, s_3: the memory is requested by the first processor, s_4: the memory is requested by the second processor, s_5: the memory is allocated to the first processor, s_6: the memory is requested by two processors, s_7: the memory is allocated to the second processor, s_8: the memory is allocated to the first processor and the memory is requested by the second processor, s_9: the memory is allocated to the second processor and the memory is requested by the first processor.

In Figure 8 the transition system $TS(\overline{E})$ is presented. In Figure 9 the underlying SMC $SMC(\overline{E})$ is depicted.

The average sojourn time vector is

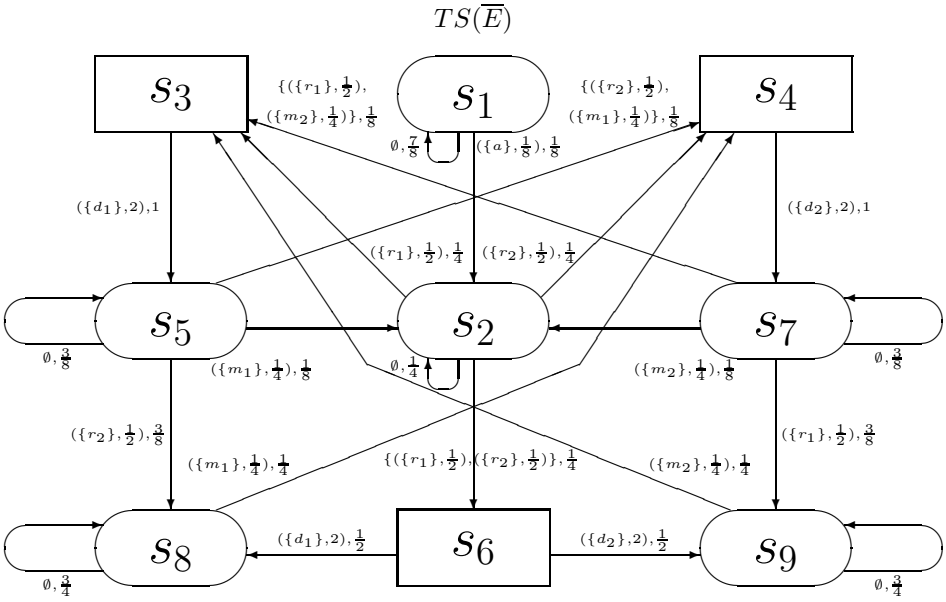

Figure 8: The transition system of the shared memory system

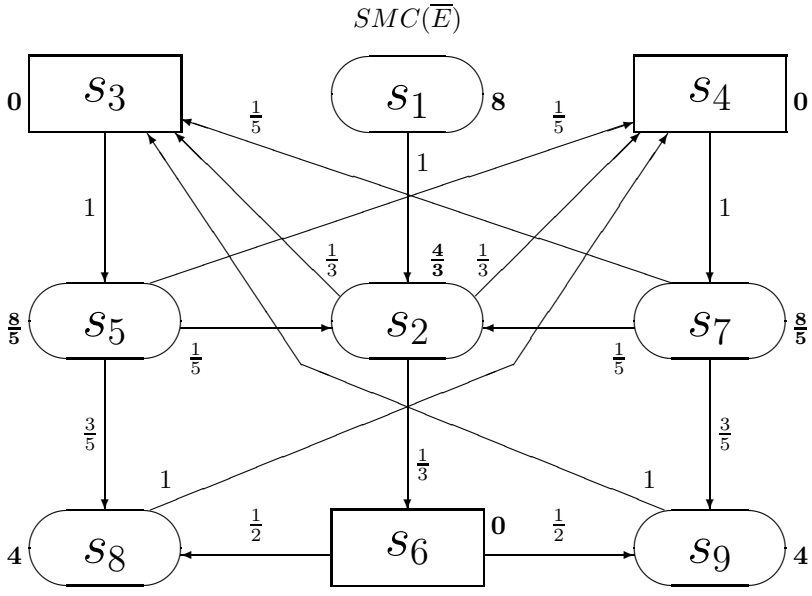

Figure 9: The underlying SMC of the shared memory system

Table 4: Transient and steady-state probabilities for the EDTMC of the shared memory system

k	0	1	2	3	4	5	6	7	8	9	10	∞
$\psi_1^*[k]$	1	0	0	0	0	0	0	0	0	0	0	0
$\psi_2^*[k]$	0	1	0	0	0.1333	0	0.0933	0.0978	0.0187	0.0969	0.0754	0.0682
$\psi_3^*[k]$	0	0	0.3333	0	0.2333	0.2444	0.0467	0.2422	0.1886	0.0982	0.2316	0.1705
$\psi_5^*[k]$	0	0	0	0.3333	0	0.2333	0.2444	0.0467	0.2422	0.1886	0.0982	0.1705
$\psi_6^*[k]$	0	0	0.3333	0	0	0.0444	0	0.0311	0.0326	0.0062	0.0323	0.0227
$\psi_8^*[k]$	0	0	0	0.1667	0.2000	0	0.1622	0.1467	0.0436	0.1616	0.1163	0.1136

$$SJ = \left(8, \frac{4}{3}, 0, 0, \frac{8}{5}, 0, \frac{8}{5}, 4, 4\right).$$

The TPM for $EDTMC(\overline{E})$ is

$$\mathbf{P}^* = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{3} & \frac{1}{3} & 0 & \frac{1}{3} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & \frac{1}{5} & 0 & \frac{1}{5} & 0 & 0 & 0 & \frac{3}{5} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & \frac{1}{5} & \frac{1}{5} & 0 & 0 & 0 & 0 & 0 & \frac{3}{5} \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

In Table 4 the transient and the steady-state probabilities $\psi_i^*[k]$ ($i \in \{1, 2, 3, 5, 6, 8\}$) for the EDTMC of the shared memory system at the time moments k ($0 \leq k \leq 10$) and $k = \infty$ are presented, and in Figure 10 the alteration diagram (evolution in time) for the transient probabilities is depicted. It is sufficient to consider the probabilities for the states $s_1, s_2, s_3, s_5, s_6, s_8$ only, since the corresponding values coincide for s_3, s_4 as well as for s_5, s_7 as well as for s_8, s_9.

The steady-state PMF for $EDTMC(\overline{E})$ is

$$\psi^* = \left(0, \frac{3}{44}, \frac{15}{88}, \frac{15}{88}, \frac{15}{88}, \frac{1}{44}, \frac{15}{88}, \frac{5}{44}, \frac{5}{44}\right).$$

The steady-state PMF ψ^* weighted by SJ is

$$\left(0, \frac{1}{11}, 0, 0, \frac{3}{11}, 0, \frac{3}{11}, \frac{5}{11}, \frac{5}{11}\right).$$

It remains to normalize the steady-state weighted PMF dividing it by the sum of its components

$$\psi^* SJ^T = \frac{17}{11}.$$

Thus, the steady-state PMF for $SMC(\overline{E})$ is

$$\varphi = \left(0, \frac{1}{17}, 0, 0, \frac{3}{17}, 0, \frac{3}{17}, \frac{5}{17}, \frac{5}{17}\right).$$

We can now calculate the main performance indices.

- The average recurrence time in the state s_2, where no processor requests the memory, called the *average system run-through*, is $\frac{1}{\varphi_2} = 17$.
- The common memory is available only in the states s_2, s_3, s_4, s_6. The steady-state probability that the memory is available is $\varphi_2 + \varphi_3 + \varphi_4 + \varphi_6 = \frac{1}{17} + 0 + 0 + 0 = \frac{1}{17}$. Then the steady-state probability that the memory is used (i.e., not available), called the *shared memory utilization*, is $1 - \frac{1}{17} = \frac{16}{17}$.
- After activation of the system, we leave the state s_1 for all, and the common memory is either requested or allocated in every remaining state, with exception of s_2. Thus, the *rate with which the shared memory necessity emerges* coincides with the rate of leaving s_2, calculated as $\frac{\varphi(s_2)}{SJ(s_2)} = \frac{1}{17} \cdot \frac{3}{4} = \frac{3}{68}$.

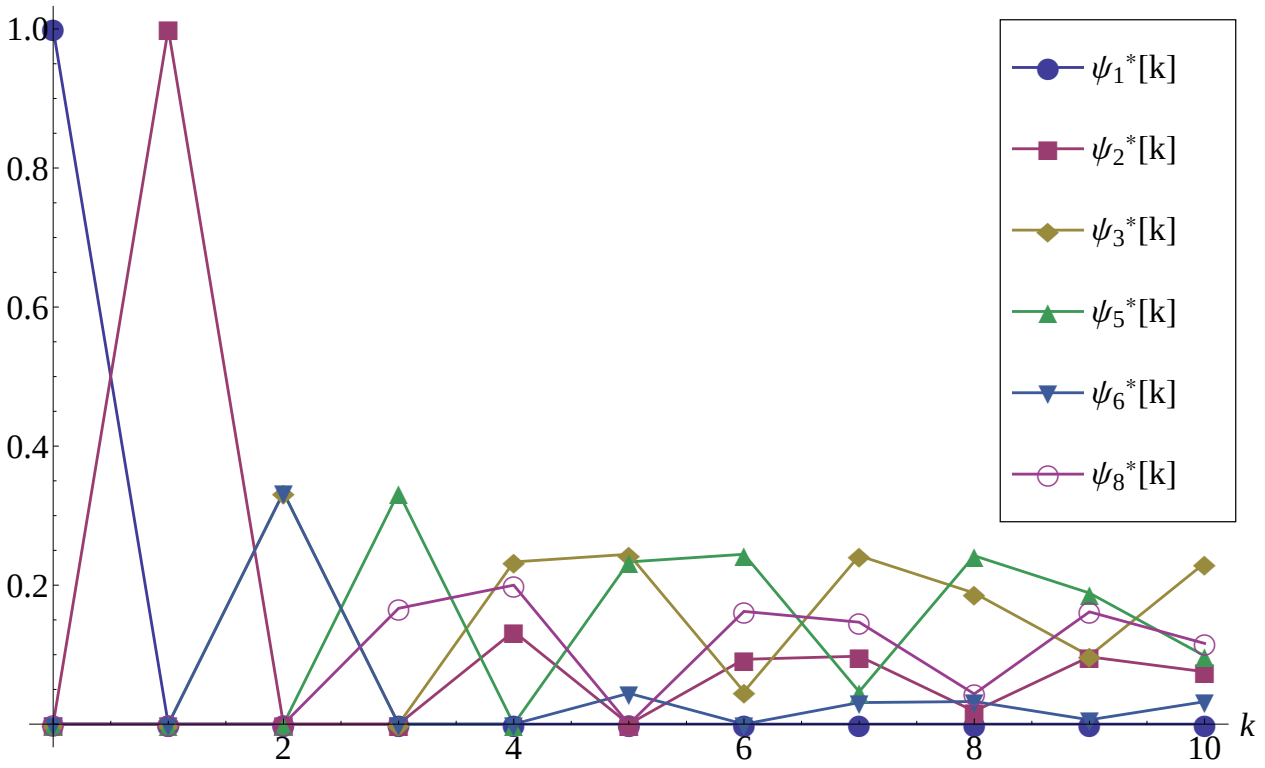

Figure 10: Transient probabilities alteration diagram for the EDTMC of the shared memory system

- The common memory request of the first processor $(\{r_1\}, \frac{1}{2})$ is only possible from the states s_2, s_7. In each of the states the request probability is the sum of the execution probabilities for all sets of activities containing $(\{r_1\}, \frac{1}{2})$. Thus, the *steady-state probability of the shared memory request from the first processor* is $\varphi_2 \sum_{\{\Gamma|(\{r_1\},\frac{1}{2})\in\Gamma\}} PT(\Gamma, s_2) + \varphi_7 \sum_{\{\Gamma|(\{r_1\},\frac{1}{2})\in\Gamma\}} PT(\Gamma, s_7) = \frac{1}{17}\left(\frac{1}{4}+\frac{1}{4}\right) + \frac{3}{17}\left(\frac{3}{8}+\frac{1}{8}\right) = \frac{2}{17}$.

In Figure 11 the marked dtsi-boxes corresponding to the dynamic expressions of two processors and shared memory are presented, i.e., $N_i = Box_{dtsi}(\overline{E_i})$ $(1 \leq i \leq 3)$. In Figure 12 the marked dtsi-box corresponding to the dynamic expression of the shared memory system is depicted, i.e., $N = Box_{dtsi}(\overline{E})$.

7 Conclusion

In this paper, we have proposed a discrete time stochastic extension dtsiPBC of a finite part of PBC enriched with iteration and immediate multiactions. The calculus has the concurrent step operational semantics based on labeled probabilistic transition systems and the denotational semantics in terms of a subclass of LDTSIPNs. A method and a case study of performance evaluation in the framework of the calculus have been presented. We have considered an example of the concurrent system with two processors and a common shared memory, for which a number of performance indices have been calculated.

Future work consists in constructing a number of stochastic equivalences on the expressions of dtsiPBC. The relations will be used to identify stochastic processes with similar behaviour which are otherwise differentiated by the semantic equivalence of the calculus, since the latter relation is too discriminate in many cases. Moreover, we plan to extend dtsiPBC with deterministically timed multiactions having a fixed time delay (including the zero one which is the case of immediate multiactions) to enhance the specification power of the calculus.

References

[1] Balbo G. *Introduction to stochastic Petri nets. Lecture Notes in Computer Science* **2090**, p. 84–155, 2001.

[2] Balbo G. *Introduction to generalized stochastic Petri nets. Lecture Notes in Computer Science* **4486**, p. 83–131, 2007.

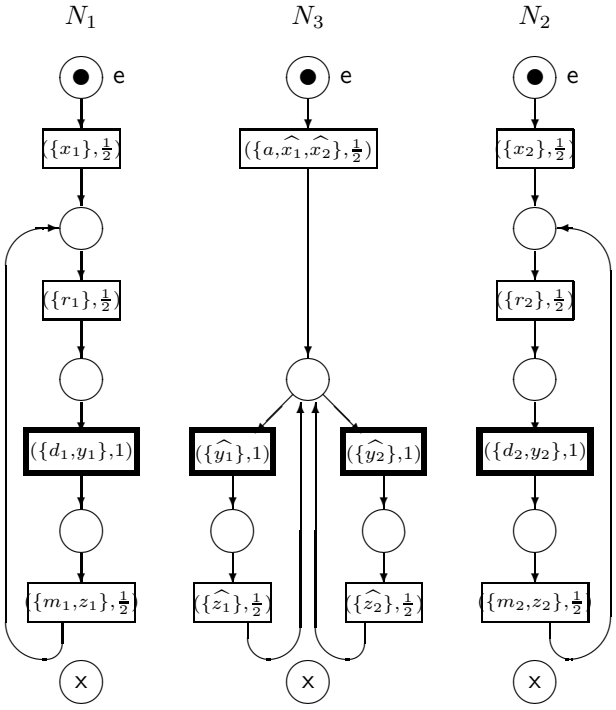

Figure 11: The marked dtsi-boxes of two processors and shared memory

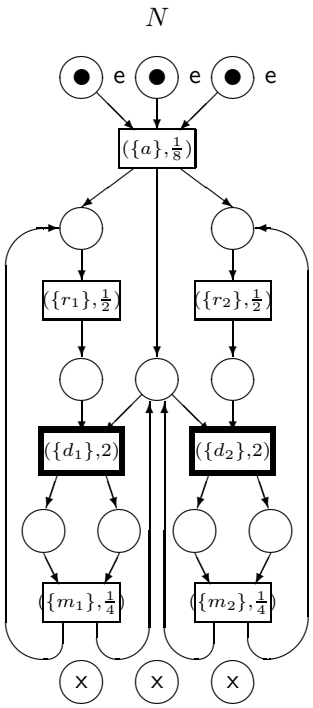

Figure 12: The marked dtsi-box of the shared memory system

[3] Best E., Devillers R., Hall J.G. *The box calculus: a new causal algebra with multi-label communication. Lecture Notes in Computer Science* **609**, p. 21–69, 1992.

[4] Best E., Devillers R., Koutny M. *Petri net algebra. EATCS Monographs on Theoretical Computer Science*, 378 p., Springer Verlag, 2001.

[5] Bernardo M., Gorrieri R. *A tutorial on EMPA: a theory of concurrent processes with nondeterminism, priorities, probabilities and time. Theoretical Computer Science* **202**, p. 1–54, July 1998, `http://www.sti.uniurb.it/bernardo/documents/tcs202.pdf`.

[6] Bergstra J.A., Klop J.W. *Algebra of communicating processes with abstraction. Theoretical Computer Science* **37**, p. 77–121, 1985.

[7] van Glabbeek R.J. *The linear time – branching time spectrum II: the semantics of sequential systems with silent moves. Extended abstract. Lecture Notes in Computer Science* **715**, p. 66–81, 1993.

[8] van Glabbeek R.J., Smolka S.A., Steffen B. *Reactive, generative, and stratified models of probabilistic processes. Information and Computation* **121(1)**, p. 59–80, 1995, `http://boole.stanford.edu/pub/prob.ps.gz`.

[9] Hillston J. *A compositional approach to performance modelling.* Cambridge University Press, Great Britain, 1996, `http://www.dcs.ed.ac.uk/pepa/book.pdf`.

[10] Hoare C.A.R. *Communicating sequential processes.* Prentice-Hall, London, 1985.

[11] Hermanns H., Rettelbach M. *Syntax, semantics, equivalences and axioms for MTIPP. Proceedings of 2^{nd} Workshop on Process Algebras and Performance Modelling*, Regensberg / Erlangen (Herzog U., Rettelbach M., eds.), *Arbeitsberichte des IMMD* **27**, p. 71–88, University of Erlangen, Germany, 1994, `http://ftp.informatik.uni-erlangen.de/local/inf7/papers/Hermanns/syntax_semantics_equivalences_axioms_for_MTIPP.ps.gz`.

[12] Koutny M. *A compositional model of time Petri nets. Lecture Notes in Computer Science* **1825**, p. 303–322, 2000.

[13] Macià H.S., Valero V.R., Cazorla D.L., Cuartero F.G. *Introducing the iteration in sPBC. Proceedings of the 24^{th} International Conference on Formal Techniques for Networked and Distributed Systems - 04 (FORTE'04)*, Madrid, Spain, *Lecture Notes in Computer Science* **3235**, p. 292–308, October 2004, `http://www.info-ab.uclm.es/retics/publications/2004/forte04.pdf`.

[14] Macià H.S., Valero V.R., Cuartero F.G., de Frutos D.E. *A congruence relation for sPBC. Formal Methods in System Design* **32(2)**, p. 85–128, Springer, The Netherlands, April 2008.

[15] Macià H.S., Valero V.R., Cuartero F.G., Ruiz M.C.D. *sPBC: a Markovian extension of Petri box calculus with immediate multiactions. Fundamenta Informaticae* **87(3–4)**, p. 367–406, IOS Press, Amsterdam, The Netherlands, 2008.

[16] Macià H., Valero V., de Frutos D. *sPBC: a Markovian extension of finite Petri box calculus. Proceedings of 9^{th} IEEE International Workshop PNPM'01*, p. 207–216, Aachen, Germany, IEEE Computer Society Press, 2001, `http://www.info-ab.uclm.es/retics/publications/2001/pnpm01.ps`.

[17] Marsan M.A. *Stochastic Petri nets: an elementary introduction. Lecture Notes in Computer Science* **424**, p. 1–29, 1990.

[18] Marsan M.A., Balbo G., Conte G., Donatelli S., Franceschinis G. *Modelling with generalized stochastic Petri nets. Wiley Series in Parallel Computing*, John Wiley and Sons, 316 p., 1995, `http://www.di.unito.it/~greatspn/GSPN-Wiley/`.

[19] Marroquín O.A., de Frutos D.E. *Extending the Petri box calculus with time. Lecture Notes in Computer Science* **2075**, p. 303–322, 2001.

[20] Milner R.A.J. *Communication and concurrency.* Prentice-Hall, 260 p., Upper Saddle River, NJ, USA, 1989.

[21] Mudge T.N., Al-Sadoun H.B. *A semi-Markov model for the performance of multiple-bus systems. IEEE Transactions on Computers* **C-34(10)**, p. 934–942, October 1985, `http://www.eecs.umich.edu/~tnm/papers/SemiMarkov.pdf`.

[22] Núñez M.G., de Frutos D.E., Llana L.D. *Acceptance trees for probabilistic processes. Lecture Notes in Computer Science* **962**, p. 249–263, 1995.

[23] Ross S.M. *Stochastic processes.* 2^{nd} edition, John Wiley and Sons, 528 p., New York, USA, April 1996.

[24] Tarasyuk I.V. *Iteration in discrete time stochastic Petri box calculus. Bulletin of the Novosibirsk Computing Center, Series Computer Science, IIS Special Issue* **24**, p. 129–148, NCC Publisher, Novosibirsk, 2006, `http://www.iis.nsk.su/persons/itar/dtsitncc.pdf`.

[25] Tarasyuk I.V. *Stochastic Petri box calculus with discrete time. Fundamenta Informaticae* **76(1–2)**, p. 189–218, IOS Press, Amsterdam, The Netherlands, February 2007, `http://www.iis.nsk.su/persons/itar/dtspbcfi.pdf`.

[26] Tarasyuk I.V. *Investigating equivalence relations in dtsPBC. Berichte aus dem Department für Informatik* **5/08**, 57 p., Carl von Ossietzky Universität Oldenburg, Germany, October 2008, `http://www.iis.nsk.su/persons/itar/dtspbcit_cov.pdf`.

[27] Zimmermann A., Freiheit J., Hommel G. *Discrete time stochastic Petri nets for modeling and evaluation of real-time systems. Proceedings of Workshop on Parallel and Distributed Real Time Systems*, San Francisco, USA, 6 p., 2001, `http://pdv.cs.tu-berlin.de/~azi/texte/WPDRTS01.pdf`.