ISSN 0361-7688, Programming and Computer Software, 2014, Vol. 40, No. 5, pp. 229–249. © Pleiades Publishing, Ltd., 2014.
Original Russian Text © I.V. Tarasyuk, H. Macià, V. Valero, 2014, published in Programmirovanie, 2014, Vol. 40, No. 5.

Performance Analysis of Concurrent Systems in Algebra dtsiPBC

I. V. Tarasyuk[a], H. Macià, and V. Valero[b]

[a] *A. P. Ershov Institute of Informatics Systems, Siberian Division, Russian Academy of Sciences, pr. akademika Lavrent'eva 6, Novosibirsk, 630090 Russia*

[b] *High School of Information Engineering, University of Castilla-La Mancha, Avda. de España s/n, 02071 Albacete, Spain*

e-mail: itar@iis.nsk.ru, hermenegilda.macia@uclm.es, valentin.valero@uclm.es

Received August 10, 2013

Abstract—Petri box calculus PBC is a well-known algebra of concurrent processes with a Petri net semantics. In the paper, an extension of PBC with discrete stochastic time and immediate multiactions, which is referred to as discrete time stochastic and immediate PBC (dtsiPBC), is considered. Performance analysis methods for concurrent and distributed systems with random time delays are investigated in the framework of the new stochastic process algebra. It is demonstrated that the performance evaluation is possible not only via the underlying semi-Markov chains of the dtsiPBC expressions but also with the use of the underlying discrete time Markov chains, and the latter analysis technique is more optimal.

Keywords: stochastic process algebras, stochastic Petri nets, Petri box calculus, discrete time, immediate multiactions, semantics, transition systems, dtsi-boxes, performance analysis, Markov chains

DOI: 10.1134/S0361768814050089

1. INTRODUCTION

Algebraic process calculi are a well-known formal model for the specification of computing systems and analysis of their behavior. 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 and widely used. Stochastic process algebras (SPAs) do not just specify actions that can occur (qualitative features), like ordinary process algebras, but they associate some quantitative parameters with actions (quantitative characteristics). The best-known SPAs are MTIPP [1], PEPA [2], and EMPA [3].

Petri box calculus (PBC) [4, 5, 6] is a flexible and expressive process algebra, which is based on the CCS calculus [7] and was developed as a tool for specification of structure of Petri nets (PNs) 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. The PBC has a step operational semantics in terms of labeled transition systems, which is based on rules of structural operational semantics. The denotational semantics of PBC was proposed via a subclass of PNs equipped with an interface and considered up to isomorphism, which is called Petri boxes.

A stochastic extension of PBC, called stochastic Petri box calculus (sPBC), was proposed in [8]. 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), which is called stochastic Petri boxes (s-boxes). In [9], the iteration operator was added to sPBC. In sPBC, performance is evaluated by analyzing the underlying stochastic process, which is continuous-time Markov chain (CTMC).

In [10], sPBC with iteration was enriched with immediate multiactions. We call the resulting calculus generalized stochastic PBC (gsPBC). gsPBC has an interleaving operational semantics based on labeled transition systems. The denotational semantics of such an sPBC extension was defined via a subclass of labeled generalized SPNs (LGSPNs), which is called generalized stochastic Petri boxes (gs-boxes). The performance analysis in gsPBC is based on the underlying semi-Markov chains (SMCs).

In [11, 12], 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 in terms of a subclass of

labeled discrete-time stochastic PNs (LDTSPNs), based on DTSPNs and called discrete time stochastic Petri boxes (dts-boxes).

In [13], dtsPBC was enriched with the iteration operator with a goal of specifying infinite processes. The underlying stochastic process, which is a discrete-time Markov chain (DTMC), was constructed and investigated to analyze performance in dtsPBC.

In this paper, we describe performance evaluation methods for computing systems in the final version of the algebra discrete-time stochastic and immediate PBC (dtsiPBC) introduced in [14]. dtsiPBC is an extension of dtsPBC with iteration by immediate multiactions having zero time delay. Immediate multiactions improve capabilities of specification: they can model instant probabilistic choices, as well as activities whose duration is insignificant compared to that of others, which allows us to get a simpler and clearer representation of systems being specified. Thus, dtsiPBC possess concurrent discrete-time semantics with geometrically distributed (like in dtsPBC) or zero delays in the states of algebraic processes. In the continuous-time semantics used in sPBC and gsPBC, parallelism is modeled by interleaving, since the probability that any two events occur simultaneously is equal to zero by the properties of continuous probability distributions. The syntax of algebra dtsiPBC is presented. Then, its step operational semantics based on labeled probabilistic transition systems is constructed. The denotational semantics is constructed on the basis of a subclass of labeled discrete-time stochastic and immediate Petri nets (LDTSIPNs), which is referred to as discrete-time stochastic and immediate Petri boxes (dtsi-boxes). To evaluate performance in dtsiPBC, we find and study a stochastic process, which is a SMC, corresponding to both semantics. We also develop an alternative, more optimal, solution method based on the corresponding DTMC.

This paper is an extended and modified version of our previous paper [15], which was published in the proceedings of an international conference on practical applications of stochastic modeling, enriched with detailed description of syntax and semantics of algebra dtsiPBC, as well as with formal definitions and use of some additional important concepts (for example, numbering of expressions, numbering function, etc.). It also includes a new alternative method for performance evaluation and supplementary illustrative examples.

The paper is organized as follows. In Section 2, the syntax of algebra dtsiPBC is presented. In Section 3, we construct the operational semantics. In Section 4, we propose the denotational semantics. The conventional and alternative methods of performance evaluation are described in Section 5. Finally, Section 6 summarizes the results obtained and outlines the research perspectives.

2. SYNTAX

In this section, we propose the syntax of dtsiPBC.

A finite *multiset* M over a set X is a mapping $M: X \longrightarrow \mathbb{N}$ such that $|\{x \in X | M(x) > 0\}| < \infty$. We denote the *set of all finite multisets* over a set X by $\mathbb{N}_f^X$. Let M, $M' \in \mathbb{N}_f^X$. The *cardinality* of 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 $\forall x \in X\, M(x) \le M'(x)$. We define $(M + M')(x) = M(x) + M'(x)$ and $(M - M')(x) = \max\{0, M(x) - M'(x)\}$. If $\forall x \in X\, M(x) \le 1$, then M can be seen as an ordinary set.

Let $Act = \{a, b, ...\}$ be a set of *elementary actions*. Then, $\widehat{Act} = \{\hat{a}, \hat{b}, ...\}$ is the set of *conjugated actions* (*conjugates*) such that $\hat{a} \ne a$ 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 $\emptyset \in \mathcal{L}$ corresponds to an internal move, i.e., to the execution of a multiaction that contains no visible action names. The *alphabet* of an multiaction $\alpha \in \mathcal{L}$ 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 *probability* of the multiaction α. This probability is interpreted as that of the independent execution of the stochastic multiaction at the next discrete time moment. Such probabilities are used to calculate those of execution of (possibly, empty) multisets of stochastic multiactions after one time unit delay. The probabilities of the stochastic multiactions are required not to be equal to 1 to avoid extra model complexity due to assigning them weights needed to make a choice when several stochastic multiactions with probability 1 can be executed from a state. In this case, some problems appear with conflicts resolving. A discussion of a similar situation in the framework of DTSPNs can be found in [16]. This decision also allows us to avoid technical difficulties related to conditioning events with zero probability. 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 $l \in \mathbb{N}\backslash\{0\}$ is a non-zero *weight* of the multiaction α. This weight is interpreted as a measure of importance (urgency, interest) or a bonus reward associated with the execution of the immediate multiaction at the current discrete time moment. Such weights are used to calculate the probabilities to execute sets of immediate multiactions instantly. The immediate multiactions have a priority over stochastic ones. One can assume that all immediate multiactions have priority 1, whereas all stochastic ones have priority 0. This means that, in a state where both kinds of multiactions can occur, immediate multiactions always occur before stochastic ones. Stochastic and immediate multi-

actions cannot be executed together in some step (concurrent execution); i.e., steps consisting of only immediate multiactions and steps including only stochastic multiactions are allowed. Let $\mathcal{IL}$ be a set of *all immediate multiactions*.

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, stochastic and immediate multiactions can also be easily differentiated, since probabilities of stochastic multiactions belong to the interval (0; 1) and weights of immediate multiactions are nonzero natural numbers. An *activity* is a stochastic or an immediate multiaction. Let $\mathcal{SIL} = \mathcal{SL} \cup \mathcal{IL}$ be the set of *all activities*. The *alphabet* of a multiset of activities $\Upsilon \in \mathbb{N}_f^{\mathcal{SIL}}$ is defined as $\mathcal{A}(\Upsilon) = \cup_{(\alpha, \kappa) \in \Upsilon} \mathcal{A}(\alpha)$. For an activity $(\alpha, \kappa) \in \mathcal{SIL}$, 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 (process expressions) 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 standard interpretations, like in other process algebras, but parallelism does not include synchronization, unlike the corresponding operation in CCS [7].

Relabeling functions $f : \mathcal{A} \longrightarrow \mathcal{A}$ are bijections preserving conjugates; i.e., $\mathcal{A}, f(\hat{x}) = \widehat{f(x)}$. The relabeling is extended to multiactions in the usual way: for $\alpha \in \mathcal{L}$, we define $f(\alpha) = \sum_{x \in \alpha} f(x)$. The relabeling is extended to the multisets of activities as follows: for $\Upsilon \in \mathbb{N}_f^{\mathcal{SIL}}$, we define $f(\Upsilon) = \sum_{(\alpha, \kappa) \in \Upsilon} (f(\alpha), \kappa)$.

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

Let $\alpha, \beta \in \mathcal{L}$ be two multiactions such that, for some elementary action $a \in Act$, we have $a \in \alpha$ and $\hat{a} \in \beta$ or $\hat{a} \in \alpha$ and $a \in \beta$. Then, synchronization of α and β by a is defined as $\alpha \oplus_a \beta = \gamma$, 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 other words, we require that $\alpha \oplus_a \beta = \alpha + \beta - \{a, \hat{a}\}$; i.e., we remove one exemplar of a and one exemplar of $\hat{a}$ from the multiset sum $\alpha + \beta$, since the synchronization of a and $\hat{a}$ produces $\emptyset$. Activities are synchronized with the use of their multiaction parts; i.e., the synchronization by a of two activities whose multiaction parts α and β possess the above-mentioned properties results in the activity with the multiaction part $\alpha \oplus_a \beta$. We may synchronize activities of the same type only: either both stochastic multiactions or both immediate ones, since immediate multiactions have a priority over stochastic ones; hence, stochastic and immediate multiactions cannot be executed together (note also that the execution of immediate multiactions takes no time, unlike that of stochastic ones). Synchronization by a means that, for a given expression with a process behavior containing two concurrent activities that can be synchronized by a, there also exists a process behavior that differs from the former only in that the two activities are replaced by the result of their synchronization.

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

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

Definition 2.1. Let $(\alpha, \kappa) \in \mathcal{SIL}$ and $a \in Act$. A *static expression* of dtsiPBC is defined as

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

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

To avoid technical difficulties with the iteration operator, we should not allow any concurrency at 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 part. Alternatively, we can use a different, safe, version of the iteration operator, but its net translation has six arguments. See also [12] for discussion on this subject.

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

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

where

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

Let *RegStatExpr* denote the set of *all regular static expressions* of dtsiPBC. Dynamic expressions specify states of the processes. As we shall see, such expressions correspond to LDTSIPNs (which are marked by default). Dynamic expressions are obtained from static ones by annotating them with upper or lower bars, which specify active components of the system at the current moment of time. $\overline{E}$ and $\underline{E}$ denote the *initial* and *final* states, respectively, of the process specified by a static expression E. The *underlying* (*base*) *static expression* of a dynamic one is obtained by removing all upper and lower bars from it.

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

$\overline{E;\, F} \Rightarrow \overline{E};\, F$	$\underline{E};\, F \Rightarrow E;\, \overline{F}$
$E;\, \underline{F} \Rightarrow \underline{E;\, F}$	$\overline{E[]F} \Rightarrow \overline{E}\,[]F$
$\overline{E[]F} \Rightarrow E\,[]\,\overline{F}$	$\underline{E}[]F \Rightarrow \underline{E[]F}$
$E[]\,\underline{F} \Rightarrow \underline{E[]F}$	$\overline{E \parallel F} \Rightarrow \overline{E} \parallel \overline{F}$
$\underline{E} \parallel \underline{F} \Rightarrow \underline{E \parallel F}$	$\overline{E[f]} \Rightarrow \overline{E}\,[f]$
$\underline{E}\,[f] \Rightarrow \underline{E\,[f]}$	$\overline{E \text{ rs } a} \Rightarrow \overline{E}$ **rs** a
$\underline{E}$ **rs** $a \Rightarrow \underline{E}$ **rs** a	$\overline{E \text{ sy } a} \Rightarrow \overline{E}$ **sy** a
$\underline{E}$ **sy** $a \Rightarrow \underline{E}$ **sy** a	$\overline{[E * F * K]} \Rightarrow [\overline{E} * F * K]$
$[\underline{E} * F * K] \Rightarrow [E * \overline{F} * K]$	$[E * \underline{F} * K] \Rightarrow [E * \overline{F} * K]$
$[E * \underline{F} * K] \Rightarrow [E * F * \overline{K}]$	$[E * F * \underline{K}] \Rightarrow \underline{[E * F * K]}$

Definition 2.3. Let $a \in Act$ and $E \in StatExpr$. A *dynamic expression* of dtsiPBC is defined as

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

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 [6]).

Definition 2.4. A dynamic expression is *regular* if its underlying static expression is regular.

Let *RegDynExpr* denote the set of *all regular dynamic expressions* of dtsiPBC.

3. OPERATIONAL SEMANTICS

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

3.1. Inaction Rules

The inaction rules for dynamic expressions describe their structural transformations that do not change states of the specified processes. As we shall see, the application of an inaction rule to a dynamic expression does not lead to any discrete time step or any transition firing in the corresponding LDTSIPN; hence, its current marking remains unchanged. Thus, an application of every inaction rule does not require any discrete time delay; i.e., the dynamic expression transformation described by the rule is accomplished instantly.

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

In Table 2, we present inaction rules for regular dynamic expressions of an arbitrary form. In this table, $E, F \in RegStatExpr$, $G, H, \tilde{G}, \tilde{H} \in RegDynExpr$ and $a \in Act$.

Definition 3.1. A regular dynamic expression G is *operative* if no inaction rule can be applied to it.

Let *OpRegDynExpr* denote the set of *all operative regular dynamic expressions* of dtsiPBC. Note that any dynamic expression can be always transformed into a (not necessarily unique) operative one by using inaction rules. In what follows, we consider regular expressions only and omit the word "regular."

Definition 3.2. Let $\approx = (\Rightarrow \cup \Leftarrow)^*$ be a structural equivalence of dynamic expressions in dtsiPBC. Thus, two dynamic expressions G and G' are *structurally equivalent*, which is denoted as $G \approx G'$, if they can be reached from each other by applying inaction rules in a forward or backward direction.

3.2. Action and Empty Loop Rules

The action rules are applied when some activities are executed. With these rules we capture the prioritization of immediate multiactions with respect to stochastic ones. We also have the empty loop rule, which is used to capture a delay of one time unit in the same state when no immediate multiactions are executable. In this case, an empty multiset of activities is executed. The action and empty loop rules will be used later to determine all multisets of activities that can be executed from the structural equivalence class of every dynamic expression (i.e., from the states of the corresponding process). This information, together with

Table 2. Inaction rules for arbitrary regular dynamic expressions

$\dfrac{G \Rightarrow \tilde{G},\ \circ \in \{;, []\}}{G \circ E \Rightarrow \tilde{G} \circ E}$	$\dfrac{G \Rightarrow \tilde{G},\ \circ \in \{;, []\}}{E \circ G \Rightarrow E \circ \tilde{G}}$	$\dfrac{G \Rightarrow \tilde{G}}{G \parallel H \Rightarrow \tilde{G} \parallel H}$
$\dfrac{H \Rightarrow \tilde{H}}{G \parallel H \Rightarrow G \parallel \tilde{H}}$	$\dfrac{G \Rightarrow \tilde{G}}{G[f] \Rightarrow \tilde{G}[f]}$	$\dfrac{G \Rightarrow \tilde{G},\ \circ \in \{\textbf{rs}, \textbf{sy}\}}{G \circ a \Rightarrow \tilde{G} \circ a}$
$\dfrac{G \Rightarrow \tilde{G}}{[G * E * F] \Rightarrow [\tilde{G} * E * F]}$	$\dfrac{G \Rightarrow \tilde{G}}{[E * G * F] \Rightarrow [E * \tilde{G} * F]}$	$\dfrac{G \Rightarrow \tilde{G}}{[E * F * G] \Rightarrow [E * F * \tilde{G}]}$

that about probabilities or weights of the activities to be executed from the process state, will be used to calculate probabilities of such executions.

The action rules with stochastic (or immediate, otherwise) multiactions describe dynamic expression transformations due to execution of non-empty multisets of stochastic (or immediate) multiactions. The rules represent possible state changes of the specified processes when some non-empty multisets of stochastic (or immediate) multiactions are executed. As we shall see, the application of an action rule with stochastic (or immediate) multiactions to a dynamic expression leads in the corresponding LDTSIPN to a discrete time step at which some stochastic transitions fire (or to the instantaneous firing of some immediate transitions) and change of the current marking, unless there is a self-loop produced by an iterative execution of a non-empty multiset, which must be one-element, i.e., a single stochastic (or immediate) multiaction, since no concurrency is allowed at the highest level of the second argument of iteration.

The empty loop rule describes dynamic expression transformations due to execution of an empty multiset of activities at a discrete time step. The rule reflects 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, the application of the empty loop rule to a dynamic expression leads to a discrete time step in the corresponding LDTSIPN at which no transitions fire and the current marking is not changed. This is a new rule that has no prototype among inaction rules of PBC, since it represents a time delay, but no notion of time exists in PBC.

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

Note that expressions of dtsiPBC can contain identical activities. To avoid technical difficulties, such as proper calculation of the state change probabilities for multiple transitions, we can always enumerate coinciding activities from left to right in the syntax of the expressions. The new activities resulted from synchronization will be annotated with concatenation of numberings of the activities they come from; hence, the numbering should have a tree structure to reflect the effect of multiple synchronizations. Now, we define the numbering that encodes a binary tree with the leaves labeled by natural numbers.

Definition 3.3. The *numbering* of expressions is defined as $\iota ::= n \mid (\iota)(\iota)$, where $n \in \mathbb{N}$.

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

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

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

After the enumeration, the multisets of activities from the expressions will become the proper sets. Suppose that the identical activities are enumerated when it is required to avoid ambiguity. This enumeration is considered to be implicit.

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

Definition 3.4. Let $G \in OpRegDynExpr$. We define the *set of all non-empty sets of activities that can potentially be executed from G* and denote it as $Can(G)$. Let $(\alpha, \kappa) \in \mathcal{SPL}$, $E, F \in RegStatExpr$, $G, H \in OpRegDynExpr$ 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 $\Upsilon \in Can(G \circ E)$, $\Upsilon \in Can(E \circ G)$ $(\circ \in \{;, []\})$, $\Upsilon \in Can(G \| H)$, $\Upsilon \in Can(H \| G)$, $f(\Upsilon) \in Can(G(f))$, $\Upsilon \in Can(G\ \mathbf{rs}\ a)$ (when $a, \hat{a} \notin \mathcal{A}(\Upsilon)$), $\Upsilon \in Can(G\ \mathbf{sy}\ a)$, $\Upsilon \in Can([G * E * F])$, $\Upsilon \in Can([E * G * F])$, $\Upsilon \in Can([E * F * G])$.

4. If $\Upsilon \in Can(G)$ and $\Xi \in Can(H)$, then $\Upsilon + \Xi \in Can(G \| H)$.

5. If $\Upsilon \in Can(G\ \mathbf{sy}\ a)$ and $(\alpha, \kappa), (\beta, \lambda) \in \Upsilon$ are different activities such that $a \in \alpha$ and $\hat{a} \in \beta$, then (a) $(\Upsilon + \{(\alpha \oplus_a \beta, \kappa \cdot \lambda)\})\backslash\{(\alpha, \kappa), (\beta, \lambda)\} \in Can(G\ \mathbf{sy}\ a)$ if $\kappa, \lambda \in (0; 1)$; (b) $\{(\alpha \oplus_a \beta, \kappa + \lambda)\})\backslash\{(\alpha, \kappa), (\beta, \lambda)\} \in Can(G\ \mathbf{sy}\ a)$ if $\kappa, \lambda \in \mathbb{N}\backslash\{0\}$.

When we synchronize the same set of activities in different orders, we obtain several activities with the same multiaction and probability or weight parts but with different numberings having the same content. In this case, we consider only one of the resulting activities to avoid introducing redundant elements. 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 the synchronization will appear in a multiset from $Can(G \,\mathbf{sy}\, a)$.

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

Let $G \in OpRegDynExpr$. Obviously, if there are only stochastic (or only immediate) multiactions in the sets from $Can(G)$, then these stochastic (or immediate) multiactions can be executed from G. Otherwise, besides stochastic ones, there are also immediate multiactions in the sets from $Can(G)$. According to the note above, there are non-empty sets of immediate multiactions in $Can(G)$ as well; i.e., $\exists \Upsilon \in Can(G)\ \Upsilon \in \mathbb{N}_f^{\mathcal{I}\mathcal{L}} \backslash \{\emptyset\}$. In this case, no stochastic multiactions can be executed from G, even if $Can(G)$ contains non-empty sets of stochastic multiactions, since immediate multiactions have a priority over stochastic ones and should be executed first.

Definition 3.5. Let $G \in OpRegDynExpr$. The *set of all non-empty sets of activities that can be executed from* G is

$$Now(G) = \begin{cases} Can(G), & (Can(G) \subseteq \mathbb{N}_f^{\mathcal{S}\mathcal{L}} \backslash \{\emptyset\}) \vee \\ & (Can(G) \subseteq \mathbb{N}_f^{\mathcal{I}\mathcal{L}} \backslash \{\emptyset\}); \\ Can(G) \cap \mathbb{N}_f^{\mathcal{I}\mathcal{L}}, & \text{otherwise.} \end{cases}$$

An expression $G \in OpRegDynExpr$ is *tangible*, which is denoted by $tang(G)$, if $Now(G) \in \mathbb{N}_f^{\mathcal{S}\mathcal{L}} \backslash \{\emptyset\}$. Otherwise, the expression G is *vanishing*, which is denoted as $vanish(G)$; in this case, $Now(G) \in \mathbb{N}_f^{\mathcal{I}\mathcal{L}} \backslash \{\emptyset\}$.

In Table 3, we define the action and empty loop rules. In this table, $(\alpha, \rho), (\beta, \chi) \in \mathcal{S}\mathcal{L}$, $(\alpha, l), (\beta, m) \in \mathcal{I}\mathcal{L}$ and $(\alpha, \kappa) \in \mathcal{S}\mathcal{I}\mathcal{L}$. Further, $E, F \in RegStatExpr$; $G, H \in OpRegDynExpr$; $\tilde{G}, \tilde{H} \in RegDynExpr$; and $a \in Act$. Moreover, $\Gamma; \Delta \in \mathbb{N}_f^{\mathcal{S}\mathcal{L}} \backslash \{\emptyset\}$; $\Gamma' \in \mathbb{N}_f^{\mathcal{S}\mathcal{L}}$; $I, J \in \mathbb{N}_f^{\mathcal{I}\mathcal{L}} \backslash \{\emptyset\}$; $I' \in \mathbb{N}_f^{\mathcal{I}\mathcal{L}}$ and $\Upsilon \in \mathbb{N}_f^{\mathcal{S}\mathcal{I}\mathcal{L}} \backslash \{\emptyset\}$. The first rule in the table is the empty loop rule **El**. The other rules are the action rules describing transformations of dynamic expressions built by means of certain algebraic operations. If we cannot merge a rule with stochastic multiactions and a rule with immediate multiactions for some operation, then we get coupled action rules. In such cases, the names of the action rules with immediate multiactions have suffix '**i**'.

The preconditions in rules **El**, **C**, **P1**, **I2**, and **I3** are needed to ensure that (possibly empty) sets of stochastic multiactions are executed only from *tangible* operative dynamic expressions, such that all structurally equivalent operative dynamic expressions are tangible as well. For example, if $init(G)$ in rule **C**, then $G = \bar{F}$ for some static expression F and $G[]E = \bar{F}[]E \approx F[]\bar{E}$. Hence, it should be guaranteed that $tang(F[]E)$, which holds if and only if $tang(\bar{E})$.

Synchronization rule **Sy2** establishes that the synchronization of two stochastic multiactions is made by taking the product of their probabilities, since we are assume that both must occur for the synchronization to happen.

In rule **Sy2i**, we sum the weights of two synchronized immediate multiactions, since the weights can be interpreted as rewards [17]; thus, we collect the rewards. Moreover, this means that synchronized execution of immediate multiactions is more important than separate execution of each of them. Since execution of immediate multiactions takes no time, we prefer to execute as many synchronized immediate multiactions during a step as possible to get more significant progress in behavior. Under behavioral progress, we understand an advance in executing activities, which does not always imply a progress in time, as in the case when the activities are immediate multiactions. This aspect will be used later, when evaluating performance via analysis of the embedded discrete time Markov chains (EDTMCs) of expressions. Since every state change in EDTMC takes one unit of (local) time, the greater advance in operation of the EDTMC allows one to calculate performance indices faster.

Observe also that we do not have self-synchronization, i.e., synchronization of an activity with itself, since all the (enumerated) activities executed together are considered to be different. This allows us to avoid many technical difficulties described in [12].

3.3. Transition Systems

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

Definition 3.6. The *derivation set* $DR(G)$ of a dynamic expression G is the minimal set such that $[G]_{\approx} \in DR(G)$, or, if $[H]_{\approx} \in DR(G)$ and $\exists \Gamma\ H \xrightarrow{\Gamma} \tilde{H}$, $[\tilde{H}]_{\approx} \in DR(G)$.

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

The set of *all multisets of activities executable in* s is defined as $Exec(s) = \{\Upsilon \mid \exists H \in s\, \exists \tilde{H}\ H \xrightarrow{\Upsilon} \tilde{H}\}$. Note that, if $\Upsilon \in Exec(s) \backslash \{\emptyset\}$, then $\exists H \in s$, $\Upsilon \in Can(H)$. The state s is *tangible* if $Exec(s) \subseteq \mathbb{N}_f^{\mathcal{S}\mathcal{L}}$. For tangible states, we may have $Exec(s) = \emptyset$. Otherwise, the state s is *vanishing*, and, in this case, $Exec(s) \subseteq \mathbb{N}_f^{\mathcal{I}\mathcal{L}} \backslash \{\emptyset\}$. 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 $DR_V(G)$. Clearly, $DR(G) = DR_T(G) \uplus DR_V(G)$ ($\uplus$ denotes disjoint union).

Table 3. Action and empty loop rules

El $\dfrac{tang(G)}{G \xrightarrow{\emptyset} G}$	**B** $\overline{(\alpha, \kappa)} \xrightarrow{\{(\alpha,\kappa)\}} \underline{(\alpha, \kappa)}$	**S** $\dfrac{G \xrightarrow{\Upsilon} \tilde{G}}{G;\, E \xrightarrow{\Upsilon} \tilde{G};\, E; \quad E;\, G \xrightarrow{\Upsilon} E;\, \tilde{G}}$
C $\dfrac{G \xrightarrow{\Gamma} \tilde{G},\ \neg init(G) \vee (init(G) \wedge tang(\overline{E}))}{G[]E \xrightarrow{\Gamma} \tilde{G}[]E \quad E[]G \xrightarrow{\Gamma} E[]\tilde{G}}$	**Ci** $\dfrac{G \xrightarrow{I} \tilde{G}}{G[]E \xrightarrow{I} \tilde{G}[]E \quad E[]G \xrightarrow{I} E[]\tilde{G}}$	**P1** $\dfrac{G \xrightarrow{\Gamma} \tilde{G},\ tang(H)}{G \parallel H \xrightarrow{\Gamma} \tilde{G} \parallel H \quad H \parallel G \xrightarrow{\Gamma} H \parallel \tilde{G}}$
P1i $\dfrac{G \xrightarrow{I} \tilde{G}}{G \parallel H \xrightarrow{I} \tilde{G} \parallel H \quad H \parallel G \xrightarrow{I} H \parallel \tilde{G}}$	**P2** $\dfrac{G \xrightarrow{\Gamma} \tilde{G},\ H \xrightarrow{\Delta} \tilde{H}}{G \parallel H \xrightarrow{\Gamma+\Delta} \tilde{G} \parallel \tilde{H}}$	**P2i** $\dfrac{G \xrightarrow{I} \tilde{G},\ H \xrightarrow{J} \tilde{H}}{G \parallel H \xrightarrow{I+J} \tilde{G} \parallel \tilde{H}}$
L $\dfrac{G \xrightarrow{\Upsilon} \tilde{G}}{G[f] \xrightarrow{f(\Upsilon)} \tilde{G}[f]}$	**Rs** $\dfrac{G \xrightarrow{\Upsilon} \tilde{G},\ a,\ \hat{a} \notin \mathcal{A}(\Upsilon)}{G \text{ rs } a \xrightarrow{\Upsilon} \tilde{G} \text{ rs } a}$	**I1** $\dfrac{G \xrightarrow{\Upsilon} \tilde{G}}{[G * E * F] \xrightarrow{\Upsilon} [\tilde{G} * E * F]}$
I2 $\dfrac{G \xrightarrow{\Gamma} \tilde{G},\ \neg init(G) \vee (init(G) \wedge tang(\overline{F}))}{[E * G * F] \xrightarrow{\Gamma} [E * \tilde{G} * F]}$	**I2i** $\dfrac{G \xrightarrow{I} \tilde{G}}{[E * G * F] \xrightarrow{I} [E * \tilde{G} * F]}$	**I3** $\dfrac{G \xrightarrow{\Gamma} \tilde{G},\ \neg init(G) \vee (init(G) \wedge tang(\overline{F}))}{[E * F * G] \xrightarrow{\Gamma} [E * F * \tilde{G}]}$
I3i $\dfrac{G \xrightarrow{I} \tilde{G}}{[E * F * G] \xrightarrow{I} [E * F * \tilde{G}]}$	**Sy1** $\dfrac{G \xrightarrow{\Upsilon} \tilde{G}}{G \textbf{ sy } a \xrightarrow{\Upsilon} \tilde{G} \textbf{ sy } a}$	
Sy2 $\dfrac{G \textbf{ sy } a \xrightarrow{\Gamma' + \{(\alpha,\rho)\} + \{(\beta,\chi)\}} \tilde{G} \textbf{ sy } a,\ a \in \alpha,\ \hat{a} \in \beta}{G \textbf{ sy } a \xrightarrow{\Gamma' + \{(\alpha \oplus_a \beta,\ \rho \cdot \chi)\}} \tilde{G} \textbf{ sy } a}$	**Sy2i** $\dfrac{G \textbf{ sy } a \xrightarrow{I' + \{(\alpha,l)\} + \{(\beta,m)\}} \tilde{G} \textbf{ sy } a,\ a \in \alpha,\ \hat{a} \in \beta}{G \textbf{ sy } a \xrightarrow{I' + \{(\alpha \oplus_a \beta,\ l+m)\}} \tilde{G} \textbf{ sy } a}$	

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

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

If $\Upsilon = \emptyset$ and $s \in DR_T(G)$, we define

$$PF(\emptyset, s) = \begin{cases} \prod\limits_{\{(\beta,\chi)\}\in Exec(s)} (1-\chi), \quad Exec(s) \neq \{\emptyset\}; \\ 1, \quad Exec(s) = \{\emptyset\}. \end{cases}$$

If $s \in DR_T(G)$ and $Exec(s) \neq \{\emptyset\}$, then $PF(\Upsilon, s)$ can be interpreted as a *joint* probability of independent events (in the probability theory sense; i.e., the probability of intersection of these events is equal to the product of their probabilities). Each such an event consists in the positive or negative decision to be executed for a particular stochastic multiaction. Every executable stochastic multiaction decides probabilistically (using its probabilistic part) and independently (from others) whether it wants to be executed in s. If Υ is a set of all executable stochastic multiactions that have decided to be executed in s and $\Upsilon \in Exec(s)$, then Υ is ready for execution in s. The multiplication in the definition is used because it reflects the probability of the intersection of independent events. Alternatively, when $\Upsilon \neq \emptyset$, $PF(\Upsilon, s)$ can be interpreted as the probability to execute *exclusively* the set of stochastic multiactions Υ in s, i.e., the probability of *intersection* of two events calculated using the conditional probability formula in the form $\mathcal{P}(X \cap Y) = \mathcal{P}(X|Y)\mathcal{P}(Y)$. The event X consists in the execution of Υ in s. The event Y consists in the non-execution in s of all executable stochastic multiactions that do not belong to Υ. Since the above-mentioned non-executions are obviously independent events, the probability of Y is the product of the probabilities of the non-executions: $\mathcal{P}(Y) = \prod_{\{\{(\beta,\chi)\}\in Exec(s)|(\beta,\chi)\notin\Upsilon\}} (1-\chi)$. The conditioning of X by Y makes the executions of the stochastic multiactions from Υ independent, since all of them can be executed in parallel in s by the definition of $Exec(s)$. Hence, the probability to execute Υ *under condition* that no executable stochastic multiactions not belonging to Υ are executed in s is the product of probabilities of these stochastic multiactions: $\mathcal{P}(X|Y) = \prod_{(\alpha,\rho)\in\Upsilon} \rho$. Thus, the probability that Υ is executed *and* no executable stochastic multiactions not belonging to Υ are executed in s is equal to the probability of X conditioned by Y multiplied by the probability of Y: $\mathcal{P}(X \cap Y) = \mathcal{P}(X|Y)\mathcal{P}(Y) = \prod_{(\alpha,\rho)\in\Upsilon} \rho \cdot \prod_{\{\{(\beta,\chi)\}\in Exec(s)|(\beta,\chi)\notin\Upsilon\}} (1-\chi)$. When $\Upsilon \neq \emptyset$, $PF(\Upsilon, s)$, can be interpreted as the probability not to execute in s any executable stochastic multiactions; thus, $PF(\emptyset, s) = \prod_{\{(\beta,\chi)\}\in Exec(s)} (1-\chi)$. When only an empty set of activities can be executed in s, i.e., $Exec(s) = \{\emptyset\}$, we have $PF(\emptyset, s) = 1$, since we stay in s in this case. Note that, for $s \in DR_T(G)$, we have $PF(\Upsilon, s) \in (0; 1]$; hence, we can stay in s at the next time moment with a certain positive probability.

If $s \in DR_V(G)$, then $PF(\Upsilon, s)$ can be interpreted as the *overall* (*cumulative*) weight of the immediate multiactions from Υ, i.e., the sum of all their weights. The summation here is used since the weights can be seen as the rewards collected [17]. In addition, this means that concurrent execution of the immediate multiactions has more importance than that of every single one. The weights of immediate multiactions can be also interpreted as bonus rewards of transitions [18]. The rewards are summed when immediate multiactions are executed in parallel, because all of them "operated" thereby. Since execution of immediate multiactions takes no time, we prefer to execute as many parallel immediate multiactions in a step as possible to get more progress in behavior. This aspect will be used later, when evaluating performance on the basis of the EDTMCs of expressions. Note that this reasoning is the same as that used to define the probability of synchronized immediate multiactions in rule **Sy2i**. Another reason is that our approach is analogous to the definition of the probability of conflicting immediate transitions in GSPNs [19]. The only difference is that we have a step semantics and, for every set of immediate multiactions executed in parallel, use its cumulative weight. To get the analogy with GSPNs possessing interleaving semantics, we interpret weights of immediate transitions of GSPNs as the cumulative weights of the sets of immediate multiactions of dtsiPBC.

Note that the definition of $PF(\Upsilon, s)$ (as well as the definitions of other probability functions to be presented) is based on the assumed implicit enumeration of activities.

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

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

If $s \in DR_T(G)$, then $PT(\Upsilon, s)$ can be interpreted as the *conditional* probability to execute Υ in s calculated using the conditional probability formula in the form

$\mathcal{P}(Z|W) = \frac{\mathcal{P}(Z \cap W)}{\mathcal{P}(W)}$. The event Z consists in the exclusive execution of Υ in s; hence, $\mathcal{P}(Z) = PF(\Upsilon, s)$. The event W consists in the exclusive execution of any multiset (including the empty one) $\Xi \in Exec(s)$ in s. Thus, $W = \cup_j Z_j$, where, $\forall j$, Z_j are mutually exclusive (incompatible) events (in the probability theory sense, i.e., intersection of these events is the empty event) and $\exists i$, $Z = Z_i$. We have $\mathcal{P}(W) = \sum_j \mathcal{P}(Z_j) = \sum_{\Xi \in Exec(s)} PF(\Xi, s)$, because the summation reflects the probability of the union of mutually exclusive events. Since $Z \cap W = Z_i \cap (\cup_j Z_j) = Z_i = Z$, we have $\mathcal{P}(Z|W) = \frac{\mathcal{P}(Z)}{\mathcal{P}(W)} = \frac{PF(\Upsilon, s)}{\sum_{\Xi \in Exec(s)} PF(\Xi, s)}$. $PF(\Upsilon, s)$ can be also viewed as the *potential* probability to execute Υ in s, since we have $PF(\Upsilon, s) = PT(\Upsilon, s)$ only when *all* sets (including the empty one) consisting of the executable stochastic multiactions can be executed in s. In this case, all stochastic multiactions mentioned can be executed in parallel in s, and we have $\sum_{\Xi \in Exec(s)} PF(\Xi, s) = 1$, since this sum collects products of *all* combinations of the probability parts of the stochastic multiactions and the negations of these parts. However, in general, for example, for two stochastic multiactions (α, ρ) and (β, χ) executable in s, it may happen that they cannot be executed in s together, in parallel, i.e., $\emptyset, \{(\alpha, \rho)\}, \{(\beta, \chi)\} \in Exec(s)$, but $\{(\alpha, \rho)\}, \{(\beta, \chi)\} \notin Exec(s)$. Note that, for $s \in DR_{\mathcal{T}}(G)$, we have $PT(\emptyset, s) \in (0; 1]$; hence, there is a non-zero probability to stay in the state s at the next time moment, and the residence time in s is at least one discrete time unit.

If $s \in DR_{\mathcal{V}}(G)$, then $PT(\Upsilon, s)$ can be interpreted as the weight of the set of immediate multiactions Υ that is ready for execution in s *normalized* by the weights of *all* sets executable in s.

The advantage of our two-stage approach to the definition of the probability to execute a set of activities is that the resulting probability formula $PT(\Upsilon, s)$ is valid both for (multisets of) stochastic and immediate multiactions. It allows one to unify the notation used in what follows for constructing the operational semantics and analyzing performance.

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 it always defines a probability distribution.

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

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

Since $PM(s, \tilde{s})$ is the probability to move from s to $\tilde{s}$ by executing an *arbitrary* set of activities (including the empty one), we use summation in the definition. We have

$$\forall s \in DR(G) \sum_{\{\tilde{s} | \exists H \in s\ \exists \tilde{H} \in \tilde{s}\ \exists \Upsilon\ H \xrightarrow{\Upsilon} \tilde{H}\}} PM(s, \tilde{s})$$

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

$$= \sum_{\Upsilon \in Exec(s)} PT(\Upsilon, s) = 1.$$

Example 3.1. Let $E = (\{a\}, \rho)[](\{a\}, \chi)$. Then, $DR(\overline{E})$ consists of the equivalence classes $s_1 = [\overline{E}]_{\approx}$ and $s_2 = [\underline{E}]_{\approx}$. We have $DR_{\mathcal{T}}(\overline{E}) = \{s_1, s_2\}$. Let us show how to calculate the execution probabilities. Since $Exec(s_1) = \{\emptyset, \{(\{a\}, \rho)\}, \{(\{a\}, \chi)\}\}$, we obtain $PF(\{(\{a\}, \rho)\}, s_1) = \rho(1-\chi)$, $PF(\{(\{a\}, \chi)\}, s_1) = \chi(1-\rho)$, and $PF(\emptyset, s_1) = (1-\rho)(1-\chi)$. Then, $\sum_{\Xi \in Exec(s_1)} PF(\Xi, s_1) = \rho(1-\chi) + \chi(1-\rho) + (1-\rho)(1-\chi) = 1 - \rho\chi$. Thus, $PT(\{(\{a\}, \rho)\}, s_1) = \frac{\rho(1-\chi)}{1-\rho\chi}$, $PT(\{(\{a\}, \chi)\}, s_1) = \frac{\chi(1-\rho)}{1-\rho\chi}$, and $PT(\emptyset, s_1) = PM(s_1, s_1) = \frac{(1-\rho)(1-\chi)}{1-\rho\chi}$. Further, $Exec(s_2) = \{\emptyset\}$; hence, $\sum_{\Xi \in Exec(s_2)} PF(\Xi, s_2) = PF(\emptyset, s_2) = 1$ and $PT(\emptyset, s_2) = PM(s_2, s_2) = \frac{1}{1} = 1$. Finally, $PM(s_1, s_2) = PT(\{(\{a\}, \rho)\}, s_1) + PT(\{(\{a\}, \chi)\}, s_1) = \frac{\rho(1-\chi)}{1-\rho\chi} + \frac{\chi(1-\rho)}{1-\rho\chi} = \frac{\rho + \chi - 2\rho\chi}{1-\rho\chi}$.

Let $E' = (\{a\}, l)[](\{a\}, m)$. Then, $DR(\overline{E'})$ consists of the equivalence classes $s_1' = [\overline{E'}]_{\approx}$ and $s_2' = [\underline{E'}]_{\approx}$. We have $DR_{\mathcal{T}}(\overline{E'}) = \{s_2'\}$ and $DR_{\mathcal{V}}(\overline{E'}) = \{s_1'\}$. Let us show how to calculate the execution probabilities. Since $Exec(s_1') = \{\{(\{a\}, l)\}, \{(\{a\}, m)\}\}$, we obtain $PF(\{(\{a\}, l)\}, s_1') = l$ and $PF(\{(\{a\}, m)\}, s_1') = m$. Then, $\sum_{\Xi \in Exec(s_1')} PF(\Xi, s_1') = l + m$. Thus, $PT(\{(\{a\}, l)\}, s_1') = \frac{l}{l+m}$ and $PT(\{(\{a\}, m)\}, s_1') = \frac{m}{l+m}$. Further, $Exec(s_2') = \{\emptyset\}$; hence, $\sum_{\Xi \in Exec(s_2')} PF(\Xi, s_2') = PF(\emptyset, s_2') = 1$ and $PT(\emptyset, s_2') = PM(s_2', s_2') = \frac{1}{1} = 1$. Finally,

$PM(s_1', s_2') = PT(\{(\{a\}, l)\}, s_1') + PT(\{(\{a\}, m)\}, s_1') = \frac{l}{l+m} + \frac{m}{l+m} = 1.$

Definition 3.7. 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

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

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 remark after the definition of $PT(\Upsilon, s)$. Thus, we have defined a *generative* model of probabilistic processes [20]. This follows from the fact that the sum of the probabilities of the transitions with all possible labels, rather than of only those with the same labels (up to enumeration of activities they include), like in the *reactive* models, should be equal to 1. In addition, we do not have a nested probabilistic choice as in the *stratified* models.

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

Note that for tangible states, $\mathcal{P}$ can be the empty set, and its execution does not change the current state (i.e., the equivalence class), since we have a loop transition $s \xrightarrow{\emptyset}_{\mathcal{P}} s$ from a tangible state s to itself. This corresponds to the application of the empty loop rule to expressions from the equivalence class. We have to keep track of such executions, which are called *empty loops*, because they have non-zero probabilities. This follows from the definition of $PF(\emptyset, s)$ and the fact that multiaction probabilities cannot be equal to 1 since they belong to the interval (0; 1). For vanishing states, Υ cannot be the empty set, since we must execute some immediate multiactions from them at the current moment.

The step probabilities belong to the interval (0; 1], with value 1 corresponding to the case where we cannot leave a tangible state s and there exists only one transition from it, namely, the empty loop $s \xrightarrow{\emptyset}_1 s$, or if there is just a single transition from a vanishing state to any other one.

We write $s \xrightarrow{\Upsilon} \tilde{s}$ if $\exists \mathcal{P}\ s \xrightarrow{\Upsilon}_{\mathcal{P}} \tilde{s}$ and $s \longrightarrow \tilde{s}$ if $\exists \Upsilon\ s \xrightarrow{\Upsilon} \tilde{s}$.

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

Definition 3.8. Let G, G' be dynamic expressions and $TS(G) = (S_G, L_G, \mathcal{T}_G, s_G)$, $TS(G') = (S_{G'}, L_{G'}, \mathcal{T}_{G'}, s_{G'})$ be their transition systems. A mapping $\beta : S_G \longrightarrow S_{G'}$ is an *isomorphism* between $TS(G)$ and $TS(G')$, which is denoted as $\beta : TS(G) \simeq TS(G')$, if β is a bijection such that $\beta(s_G) = s_{G'}$ and $\forall s, \tilde{s} \in S_G\ \forall \Upsilon\ s \xrightarrow{\Upsilon}_{\mathcal{P}} \tilde{s} \Leftrightarrow \beta(s) \xrightarrow{\Upsilon}_{\mathcal{P}} \beta(\tilde{s})$. Two transition systems $TS(G)$ and $TS(G')$ are *isomorphic*, which is denoted as $TS(G) \simeq TS(G')$ if $\exists \beta$: $TS(G) \simeq TS(G')$.

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

Definition 3.9. Two dynamic expressions G and G' are *equivalent with respect to the transition systems*, which is denoted as $G =_{ts} G'$, if $TS(G) \simeq TS(G')$.

Example 3.2. The expression $\texttt{Stop} = \left(\{g\}, \frac{1}{2}\right)$ **rs** g specifies the non-terminating process performing only empty loops with probability 1. Let $E = [(\{a\}, \rho) * ((\{b\}, \chi); ((\{c\}, l); (\{d\}, \theta))[](\{e\}, m); (\{f\}, \phi)))) * \texttt{Stop}]$. $DR(\overline{E})$ consists of the elements

$s_1 = [\overline{(\{a\}, \rho)} * ((\{b\}, \chi); ((\{c\}, l); (\{d\}, \theta))[](\{e\}, m); (\{f\}, \phi)))) * \texttt{Stop}]_\approx$,

$s_2 = [(\{a\}, \rho) * \overline{((\{b\}, \chi)}; ((\{c\}, l); (\{d\}, \theta))[](\{e\}, m); (\{f\}, \phi)))) * \texttt{Stop}]_\approx$,

$s_3 = [(\{a\}, \rho) * ((\{b\}, \chi); \overline{((\{c\}, l); (\{d\}, \theta))[](\{e\}, m); (\{f\}, \phi))))} * \texttt{Stop}]_\approx$,

$s_4 = [(\{a\}, \rho) * ((\{b\}, \chi); ((\{c\}, l); \overline{(\{d\}, \theta))}[](\{e\}, m); (\{f\}, \phi)))) * \texttt{Stop}]_\approx$,

$s_5 = [(\{a\}, \rho) * ((\{b\}, \chi); ((\{c\}, l); (\{d\}, \theta))[](\{e\}, m); \overline{(\{f\}, \phi))))} * \texttt{Stop}]_\approx$,

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

In Fig. 1, 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 multisets 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 PNs (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 (LDTSIPNs), a subclass of DTSPNs [16] (we do not allow the transition probabilities to be equal to 1) extended with transition labeling and immediate transitions. LDTSIPNs resemble in part discrete time deterministic and stochastic PNs (DTDSPNs) [21], as well as discrete deterministic and stochastic PNs (DDSPNs) [22]. DTDSPNs and DTSPNs are extensions of DTSPNs with deterministic transitions (having fixed delay that can be zero), inhibitor arcs and guards. In addition, while stochastic transitions of DTDSPNs, like those of DTSPNs, have geometrically distributed delays, delays of stochastic transitions of DTSPNs have discrete-time phase distributions. At the same time, LDTSIPNs are not subsumed by DTDSPNs or DTSPNs, since LDTSIPNs have a step semantics, whereas DTDSPNs and DDSPNs have interleaving semantics. LDTSIPNs are somewhat similar to labeled weighted DTSPNs (LWDTSPNs) from [23], but, in LWDTSPNs, there are no immediate transitions, all (stochastic) transitions have weights, the transition probabilities may be equal to 1, and only maximal fireable subsets of the enabled transitions are fired.

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 $T_N = T_{sN} \uplus T_{iN}$ are finite sets of *places* and *stochastic and immediate transitions*, respectively, such that $P_N \cup T_N \neq \emptyset$ and $P_N \cap T_N = \emptyset$; that
- $W_N : (P_N \times T_N) \cup (T_N \times P_N) \longrightarrow \mathbb{N}$ is a function providing *weights of arcs* between places and transitions;
- $\Omega_N : T_N \longrightarrow (0; 1) \cup (\mathbb{N}\backslash\{0\})$ is a *transition probability and weight* function associating stochastic transitions with probabilities and immediate transitions with weights;
- $L_N : T_N \longrightarrow \mathcal{L}$ is a transition *labeling* function assigning multiactions to transitions;
- $M_N \in \mathbb{N}_f^{P_N}$ is an *initial marking*.

The graphical representation of LDTSIPNs is similar to that of standard labeled PNs supplemented with indications of probabilities or weights written near the corresponding transitions. Square boxes of normal thickness depict stochastic transitions, and those with thick borders represent immediate transitions. If the probabilities or weights are not indicated, they are considered to be of no importance in the corresponding examples.

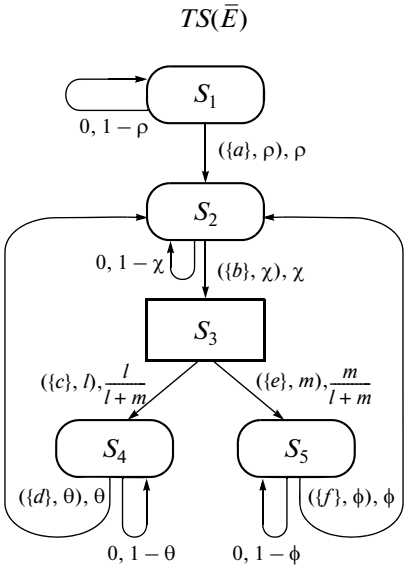

Fig. 1. The transition system of $\bar{E}$ for $E = [(\{a\}, \rho) * ((\{b\}, \chi); ((\{c\}, l); (\{d\}, \theta))[](\{e\}, m); (\{f\}, \phi)))) * \texttt{Stop}]$.

Now, we consider the semantics of LDTSIPNs.

Let N be an LDTSIPN and $M, \tilde{M} \in \mathbb{N}_f^{P_N}$.

Immediate transitions have a priority over stochastic ones; thus, immediate transitions always fire first, if they can. Suppose that all stochastic transitions have priority 0 and all immediate ones have priority 1.

A transition $t \in T_N$ is *enabled* in M if ${}^\bullet t \subseteq M$ and one of the following conditions holds: $t \in T_{iN}$ or $\forall u \in T_N \, {}^\bullet u \in M \Rightarrow u \in T_{sN}$. In other words, a transition is enabled in a marking if it has enough tokens in its input places and is immediate, or, when it is stochastic, there exists no immediate transition with enough tokens in its input places. Let $Ena(M)$ be the set of *all transitions enabled in M*. By definition, it follows that $Ena(M) \subseteq T_{iN}$ or $Ena(M) \subseteq T_{sN}$. A set of transitions $U \subseteq Ena(M)$ is *enabled* in a marking M if ${}^\bullet U \subseteq M$. Firings of transitions are atomic (instantaneous) operations, and transitions may fire concurrently in steps. We assume that all transitions participating in a step should differ; hence, only sets (not multisets) of transitions may fire. Thus, we do not allow self-concurrency to avoid some technical difficulties in calculating probabilities for multisets of transitions, as will be seen after the following formal definitions. Moreover, we do not need to consider self-concurrency, since denotational semantics of expressions will be defined via dtsi-boxes which are safe LDTSIPNs (hence, no self-concurrency is possible).

A marking M is *tangible*, which is denoted by $tang(M)$, if $Ena(M) \subseteq T_{sN}$, in particular, if $Ena(M) = \emptyset$. Otherwise, the marking M is *vanishing*, which is denoted by $vanish(M)$ and, in this case, $Ena(M) \subseteq T_{iN}$

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 that the set of stochastic transitions U is ready for firing in M* or the *weight of the set of immediate transitions U that is ready for firing in M* is

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

If $U = \emptyset$ and $tang(M)$, we define

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

Let $U \subseteq Ena(M)$, $U \neq \emptyset$, and ${}^\bullet U \subseteq M$ or $U = \emptyset$ and $tang(M)$. Besides U, some other sets of transitions may be ready for firing in M; hence, a kind of conditioning or normalization is needed to calculate the firing probability. The concurrent firing of the transitions from U changes the marking M to $\tilde{M} - {}^\bullet U + U^\bullet$, which is denoted as $M \xrightarrow{U}_{\mathcal{P}} M$, where $\mathcal{P} = PT(U, M)$ is the *probability that the set of transitions U fires in M* defined as $PT(U, M) = \dfrac{PF(U, M)}{\sum_{\{V | {}^\bullet V \subseteq M\}} PF(V, M)}$.

Note that, in the case of $U = \emptyset$ and $tang(M)$, we have $M = \tilde{M}$.

The advantage of our two-stage approach to definition of the probability that a set of transitions fires is that the resulting probability formula $PT(U, M)$ is valid both for (sets of) stochastic and immediate transitions. This allows one to unify the notation used further for constructing the denotational semantics and analyzing performance.

Note that, for all markings of an LDTSIPN N, the sum of outgoing probabilities (i.e., probabilities to change the current marking) is equal to 1. More formally, $\forall M \in \mathbb{N}_f^{P_N} \sum_{\{U | {}^\bullet U \subseteq M\}} PT(U, M) = 1$. This obviously follows from the definition of $PT(U, M)$ and guarantees that this function defines a probability distribution.

We write $M \xrightarrow{U} \tilde{M}$ if $\exists \mathcal{P}\ M \xrightarrow{U}_{\mathcal{P}} \tilde{M}$ and $M \longrightarrow \tilde{M}$ if $\exists U\ M \xrightarrow{U} \tilde{M}$.

The *probability to move from M to $\tilde{M}$ by firing an arbitrary set of transitions* is $PM(M, \tilde{M}) = \sum_{\{U | M \xrightarrow{U} \tilde{M}\}} PT(U, M)$.

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

Definition 4.2. Let N be an LDTSIPN.

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

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, $RS(N) = RS_T(N) \uplus RS_V(N)$.

4.2. Algebra of dtsi-Boxes

Now, we introduce discrete-time stochastic and immediate Petri boxes and the corresponding 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 $P_N \cap T_N = \emptyset$;
- $W_N : (P_N \times T_N) \cup (T_N \times P_N) \longrightarrow \mathbb{N}$ is a function providing *weights of arcs* between places and transitions;
- Λ_N is the *place and transition labeling* function such that $\Lambda_N|_{P_N} : P_N \longrightarrow \{\mathtt{e}, \mathtt{i}, \mathtt{x}\}$ (it specifies *entry*, *internal*, and *exit* places, respectively); $\Lambda_N|_{T_N} : T_N \longrightarrow \{\varrho \mid \varrho \subseteq \mathbb{N}_f^{\mathcal{SSL}} \times \mathcal{SSL}\}$ (it associates transitions with the *relabeling relations* on activities).

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

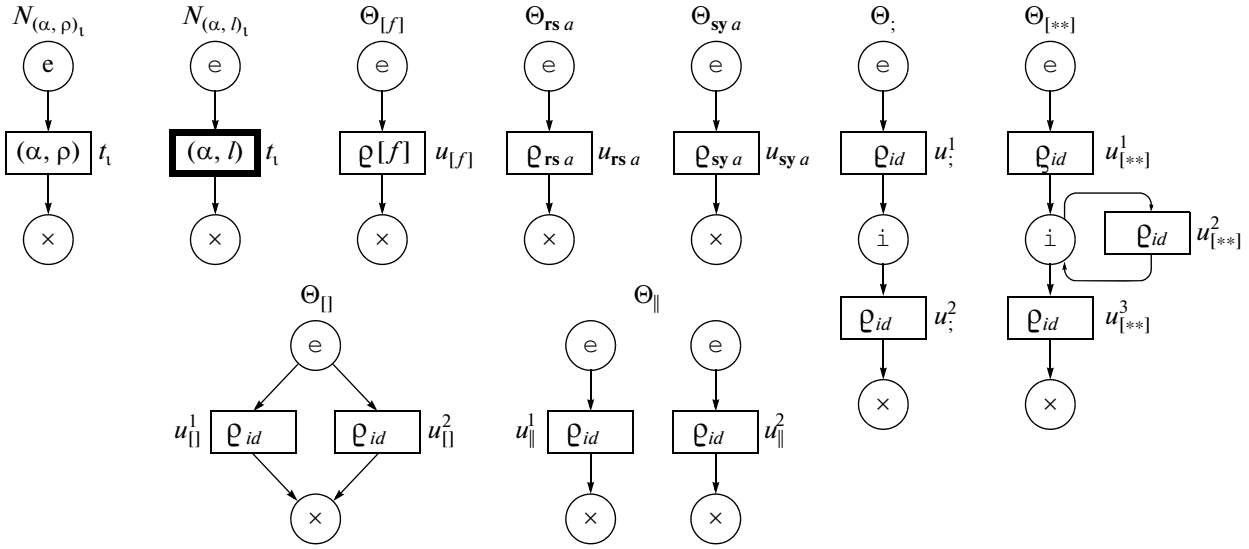

Fig. 2. The plain and operator dtsi-boxes.

A dtsi-box is *plain* if $\forall t \in T_N\ \Lambda_N(t) \in \mathcal{SJL}$, i.e., $\Lambda_N(t)$ is a constant relabeling relation to be defined later. A *marked plain dtsi-box* is a pair (N, M_N), where N is a plain dtsi-box and $M_N \in \mathbb{N}_f^{P_N}$ is its marking. We will use the following notation: $\overline{N} = (N, {}^\circ N)$ and $\underline{N} = (N, N^\circ)$. Note that a marked plain dtsi-box $(P_N, T_N, W_N, \Lambda_N, M_N)$ can 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 $L_N(t) = \mathcal{L}(\Lambda_N(t))$. Behavior of marked dtsi-boxes follows from the firing rule of LDTSIPNs. A plain dtsi-box N is *n-bounded* ($n \in \mathbb{N}$) if $\overline{N}$ is *bounded*, i.e., $\forall M \in RS(\overline{N})$ $\forall p \in P_N\ M(p) \le n$, and it is *safe* if it is 1-bounded. A plain dtsi-box N is *clean* if $\forall M \in RS(\overline{N})\ {}^\circ N \subseteq M \Rightarrow M = {}^\circ N$ 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.

The structure of the plain dtsi-box corresponding to a static expression is constructed like in PBC [5, 6]; 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. The operator dtsi-boxes specify n-ary functions from plain dtsi-boxes to plain dtsi-boxes (we have $1 \le n \le 3$ in dtsiPBC). Thus, the resulting plain dtsi-boxes are safe and clean. In the definition of the denotational semantics, we will apply standard constructions used for PBC. Let Θ denote an *operator box* and u denote the *transition name* from the PBC setting.

The relabeling relations $\varrho \subseteq \mathbb{N}_f^{\mathcal{SJL}} \times \mathcal{SJL}$ are defined as follows:

• $\varrho_{id} = \{(\{(\alpha, \kappa)\}, (\alpha, \kappa)) \mid (\alpha, \kappa) \in \mathcal{SJL}\}$ is the *identity relabeling* keeping the interface as is;

• $\varrho_{(\alpha,\kappa)}\ \{(\emptyset, (\alpha, \kappa))\}$ is the *constant relabeling* that can be identified with the activity $(\alpha, \kappa) \in \mathcal{SJL}$ itself;

• $\varrho_{[f]} = \{(\{(\alpha, \kappa)\}, (f(\alpha), \kappa)) \mid (\alpha, \kappa) \in \mathcal{SJL}\}$;

• $\varrho_{\mathbf{rs}\,a} = \{(\{(\alpha, \kappa)\}, (\alpha, \kappa)) \mid (\alpha, \kappa) \in \mathcal{SJL}, a, \hat{a} \notin \alpha\}$;

• $\varrho_{\mathbf{sy}\,a}$ is the least relabeling relation containing ϱ_{id} such that, if $a \in \alpha$, $\hat{a} \in \beta$, and $(\Upsilon, (\alpha, \kappa)), (\Xi, (\alpha, \kappa)) \in \varrho_{\mathbf{sy}\,a}$ then

– $(\Upsilon + \Xi\{(\alpha \oplus_a \beta, \kappa \cdot \lambda)\}) \in \varrho_{\mathbf{sy}\,a}$ if $\kappa, \lambda \in (0; 1)$;

– $(\Upsilon + \Xi\{(\alpha \oplus_a \beta, \kappa + \lambda)\}) \in \varrho_{\mathbf{sy}\,a}$ if $\kappa, \lambda \in \mathbb{N}\backslash\{0\}$.

The plain dtsi-boxes $N_{(\alpha,\rho)_\iota}$, $N_{(\alpha,l)_\iota}$ and operator dtsi-boxes are shown in Fig. 2. The label i associated with internal places is usually omitted.

In the case of the iteration, a decision is to be made on the selection of the operator box that will be used for it, since we have two proposals in plain PBC for that purpose [6]. 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 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, this particular situation cannot occur due to the syntactical restriction introduced for regular terms. Therefore, the net obtained will be always safe.

To construct the semantic function that associates a plain dtsi-box with every static expression of dtsiPBC, we introduce the *enumeration* function $Enu : T_N \longrightarrow Num$, which associates the numberings with transitions of a plain dtsi-box N in accordance with those of activities. In the case of synchronization, this

function associates with the resulting new transition the concatenation of the parenthesized numberings of the transitions it came from.

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 a plain dtsi-box corresponding to a static expression E, and let $Enu_E : T_E \longrightarrow Num$ be the enumeration function for $Box_{dtsi}(E)$. We will use similar 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: $Enu(t) = Enu_E(t)$, $t \in T_E$.

• $Box_{dtsi}(E \textbf{ rs } a) = \Theta_{\textbf{rs}\,a}(Box_{dtsi}(E))$. Since we remove all transitions labeled with multiactions containing a or $\hat{a}$, this does not change the numbering of the remaining transitions: $Enu(t) = Enu_E(t)$, $t \in T_E$, $a, \hat{a} \notin \mathcal{L}(\Lambda_E(t))$.

• $Box_{dtsi}(E \textbf{ sy } a) = \Theta_{\textbf{sy}\,a}(Box_{dtsi}(E))$. Note that $\forall v, w \in T_E$ such that $\Lambda_E(v) = (\alpha, \kappa)$, $\Lambda_E(v) = (\beta, \lambda)$, and $a \in \alpha$, $\hat{a} \in \beta$, the new transition t resulting from synchronization of v and w has the label $\Lambda(t) = (\alpha \oplus_a \beta, \kappa \cdot \lambda)$ and the numbering $Enu(t) = (Enu_E(v))(Enu_E(w))$. Thus, the enumeration function is defined as

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

According to the definition of $\varrho_{\textbf{sy}\,a}$, the synchronization is possible only when all 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 transition from the resulting ones in the plain dtsi-box to avoid introducing redundant transitions.

For example, if 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, $Enu(t) = (Enu_E(v))(Enu_E(w)) \neq (Enu_E(w))(Enu_E(v)) = Enu(u)$, whereas $Cont(Enu(t)) = Cont(Enu(v)) \cup Cont(Enu(w)) = Cont(Enu(u))$. Then, only one transition t (or u) will appear in $Box_{dtsi}(E \textbf{ sy } a)$.

• $Box_{dtsi}([E * F * K]) = \Theta_{[* *]}(Box_{dtsi}(E), Box_{dtsi}(F), Box_{dtsi}(K))$. 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. $Box_{dtsi}((\alpha, \kappa)_\iota) = N_{(\alpha, \kappa)_\iota}$;
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. $Box_{dtsi}(E \circ a) = \Theta_{\circ a}(Box_{dtsi}(E))$, $\circ \in \{\textbf{rs}, \textbf{sy}\}$;
5. $Box_{dtsi}([E * F * K]) = \Theta_{[* *]}(Box_{dtsi}(E), Box_{dtsi}(F), Box_{dtsi}(K))$.

The dtsi-boxes of dynamic expressions can be defined as well. For $E \in RegStatExpr$, we set $Box_{dtsi}(\overline{E}) = \overline{Box_{dtsi}(E)}$ and $Box_{dtsi}(\underline{E}) = \underline{Box_{dtsi}(E)}$. Note that any dynamic expression can be decomposed into overlined or underlined static expressions or those without any lines. The definition of dtsi-boxes for arbitrary dynamic expressions should also be compositional. Hence, it is required to apply the net operations to the dtsi-boxes of the three above-specified types in which the only places with tokens are input and output ones or the places do not contain tokens at all. The operations are applied to the dtsi-boxes with tokens in the same way as to those that do not contain tokens, but the composed dtsi-boxes will retain the tokens in their places.

Let $\simeq$ denote isomorphism between transition systems and reachability graphs that binds their initial states. The corresponding definitions are omitted since they are similar to the isomorphism of the transition systems. Names of transitions of the dtsi-box corresponding to a static expression can be identified with the enumerated activities of the latter.

Theorem 4.1. For any static expression E,

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

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

The quantitative (probabilistic) behavior is the same by the following reasons. First, the activities of any 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 the expression from Example 3.2. In Fig. 3, the marked dtsi-box $N = Box_{dtsi}(\bar{E})$ and its reachability graph $RG(N)$ are depicted. It is easy to see that $TS(\bar{E})$ and $RG(N)$ are isomorphic.

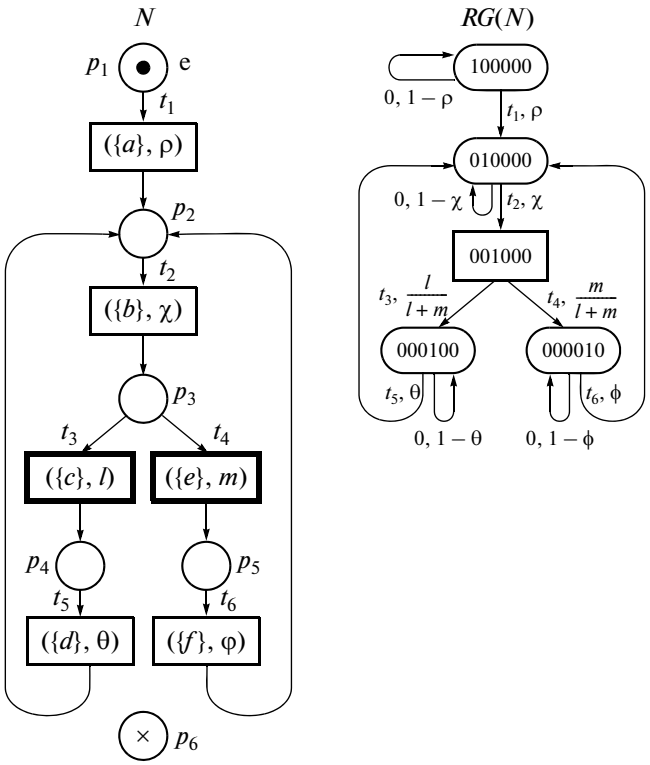

Fig. 3. The marked dtsi-box $N = Box_{dtsi}(\bar{E})$ for $E = [(\{a\}, \rho) * ((\{b\}, \chi); ((\{c\}, l); (\{d\}, \theta))[](\{e\}, m); (\{f\}, \phi)))) * \texttt{Stop}]$ and its reachability graph.

5. PERFORMANCE EVALUATION

In this section, we demonstrate how Markov chains corresponding to the expressions and dtsi-boxes can be constructed and used then for performance evaluation. The standard technique for analyzing Markov chains consists in studying their transient and stable behavior and on subsequent calculation of the performance indices (measures) based on the transient or steady-state probabilities.

5.1. Analysis of the Underlying Stochastic Process

For a dynamic expression G, a discrete random variable is associated with every tangible state $s \in DR_T(G)$. The variable captures a residence time in the state. One can interpret staying in a state at 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 with the parameter $1 - PM(s, s)$, since the probability to stay in s for $k - 1$ time moments and leave it at the moment $k \geq 1$ is $PM(s, s)^{k-1}(1 - PM(s, s))$ (the residence time is k in this case, and this formula defines the probability mass function (PMF) of residence time in s). Hence, the probability distribution function (PDF) of residence time in s is $1 - PM(s, s)^{k-1} (k \geq 1)$ (the probability that the residence time in s is less than k). The mean value formula for the geometrical distribution allows us to calculate the average sojourn time in s as $\dfrac{1}{1 - PM(s, s)}$. Clearly, the average sojourn time in a vanishing state is zero. Let $s \in DR(G)$.

The *average sojourn time in the state s* is

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

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

The *sojourn time variance in the state s* is

$$VAR(s) = \begin{cases} \dfrac{PM(s, s)}{(1 - PM(s, s))^2}, & s \in DR_T(G); \\ 0, & s \in DR_V(G). \end{cases}$$

The *sojourn time variance vector* of G, which is denoted by VAR, has the elements $VAR(s)$, $s \in DR(G)$.

To evaluate performance of the system specified by a dynamic expression G, we should investigate the stochastic process associated with it. This process is the underlying semi-Markov chain $SMC(G)$ [17], which can be analyzed by extracting from it the embedded (absorbing) discrete-time Markov chain $EDTMC(G)$ corresponding to G. The construction of the latter is similar to that used in the context of the generalized stochastic PNs (GSPNs) in [19], as well as in the frameworks of discrete-time deterministic and stochastic PNs (DTDSPNs) in [21] and discrete deterministic and stochastic PNs (DDSPNs) in [22]. The $EDTMC(G)$ only describes state changes of $SMC(G)$, while ignoring its time characteristics. Thus, to construct the EDTMC, we should abstract from all time aspects of behavior of the SMC, i.e., from the sojourn time in its states. The (local) sojourn time in every state of the EDTMC is equal to one discrete time unit. It is well-known that every SMC is fully described by the EDTMC and the state sojourn time distributions (the latter can be specified by the vector of PDFs of the residence time in the states) [24].

Let G be a dynamic expression and $s, \tilde{s} \in DR(G)$. The transition system $TS(G)$ can have self-loops, i.e., loops going from a state to itself, that have a non-zero probability. Obviously, the current state remains unchanged in this case.

Let $s \longrightarrow \tilde{s}$. The *probability to stay in s due to k* $(k \geq 1)$ *self-loops* is $(PM(s, s))^k$.

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

$$\left\{ \begin{array}{l} PM(s, \tilde{s}) \sum\limits_{k=0}^{\infty} (PM(s, s))^k = \dfrac{PM(s, \tilde{s})}{1 - PM(s, s)}, \quad s \to s; \\ PM(s, \tilde{s}), \quad \text{otherwise}; \end{array} \right\}$$

$$= SL(s) PM(s, \tilde{s}),$$

where

$$SL(s) = \begin{cases} \dfrac{1}{1 - PM(s, s)}, & s \to s; \\ 1, & \text{otherwise}; \end{cases}$$

is the *self-loops abstraction factor*. The *self-loops abstraction vector of G*, which is denoted by SL, has the elements $SL(s)$, $s \in DR(G)$. The value $k = 0$ in the summation above corresponds to the case where no self-loops occur. Note that $\forall s \in DR_T(G)\ SL(s) = \dfrac{1}{1 - PM(s, s)} = SJ(s)$; hence, $\forall s \in DR_T(G)\ PM^*(s, \tilde{s}) = SJ(s) PM(s, \tilde{s})$, since there always exists an empty loop (which is a self-loop) $s \xrightarrow{\emptyset} s$ from every tangible state s. Empty loops are not possible from vanishing states; hence, $\forall s \in DR_V(G)\ PM^*(s, \tilde{s}) = \dfrac{PM(s, \tilde{s})}{1 - PM(s, s)}$, when there are non-empty self-loops (produced by iteration) from s, or $PM^*(s, \tilde{s}) = PM(s, \tilde{s})$, when there are no self-loops from s.

Note that, after abstraction from the probabilities of transitions that do not change the states, the remaining transition probabilities are normalized. In order to calculate transition probabilities $PT(\Upsilon, s)$, it is required 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 arrive at two-stage normalization.

Note that $PM^*(s, \tilde{s})$ specifies a probability distribution, since, $\forall s \in DR(G)$ such that s is not a terminal state, i.e., there are transitions to differrent states after possible self-loops from it, we have $\sum_{\{\tilde{s} | s \to \tilde{s},\, s \neq \tilde{s}\}} PM^*(s, \tilde{s}) = \dfrac{1}{1 - PM(s, s)} \sum_{\{\tilde{s} | s \to \tilde{s},\, s \neq \tilde{s}\}} PM(s, \tilde{s}) = \dfrac{1}{1 - PM(s, s)} (1 - PM(s, s)) = 1$.

We consider self-loops followed only by a state-changing step just for convenience. Alternatively, we could take a state-changing step followed by self-loops or a state-changing step preceded and followed by self-loops. In all these three cases, our sequence begins or/and ends with the loops that 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, which resembles in some sense a construction of branching bisimulation [25] taking 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 $DR(G)$, the initial state $[G]_{\approx}$, and the transitions $s \twoheadrightarrow_{\mathcal{P}} \tilde{s}$ if $s \longrightarrow \tilde{s}$ and $s \neq \tilde{s}$, where $\mathcal{P} = PM^*(s, \tilde{s})$.

The *underlying SMC* of G denoted by $SMC(G)$ has the *EDTMC* $EDTMC(G)$ and the sojourn time in every $s \in DR_T(G)$ is geometrically distributed with the parameter $1 - PM(s, \tilde{s})$, while the sojourn time in every $s \in DR_V(G)$ is equal to zero.

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

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

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

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

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

where $\mathbf{I}$ is the identity matrix of order n and $\mathbf{0}$ is the row vector of n zero values, and $\mathbf{1}$ is the n-dimensional row vector of ones.

When $EDTMC(G)$ has a 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 $\psi^*(s_i)$ $(1 \leq i \leq n)$ by the average sojourn time $SJ(s_i)$ in the state s_i, after which we normalize the resulting values. Recall that, for a vanishing state $s \in$

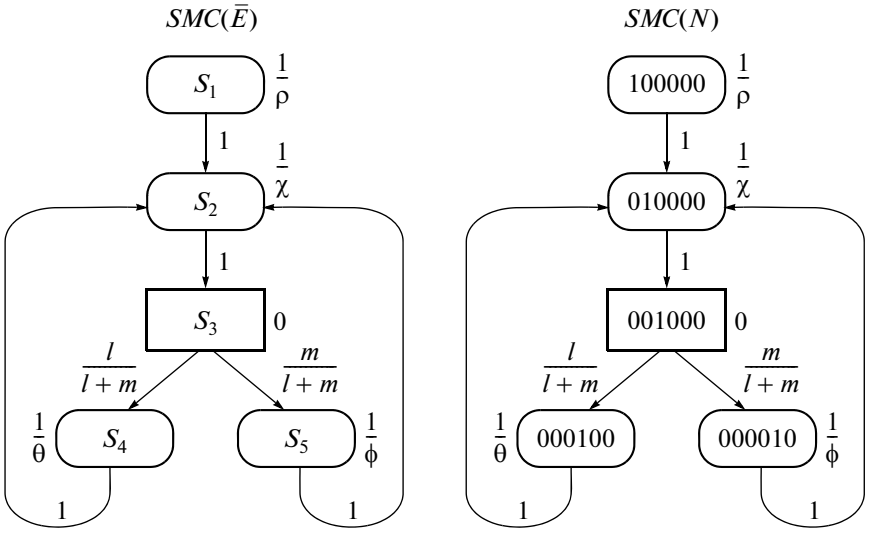

Fig. 4. The underlying *SMC* of $\bar{E}$ and $N = Box_{dtsi}(\bar{E})$ for $E = [(\{a\}, \rho) * ((\{b\}, \chi); ((\{c\}, l); (\{d\}, \theta))[]((\{e\}, m); (\{f\}, \phi)))) * \texttt{Stop}]$.

$DR_V(G)$, we have $SJ(s) = 0$. Thus, the steady-state PMF $\varphi = \varphi(s_1), \ldots, \varphi(s_n)$ for $SMC(G)$ is

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

Thus, to calculate φ, we apply abstracting from self-loops to get $\mathbf{P}^*$ and ψ^* followed by weighting by SJ and normalization. $EDTMC(G)$ has no self-loops, unlike $SMC(G)$; hence, the behavior of $EDTMC(G)$ is stabilized faster than that of $SMC(G)$ (if both possess single steady states), since $\mathbf{P}^*$ has only zero elements at the main diagonal.

Example 5.1. Let E be the expression from Example 3.2. In Fig. 4, the underlying SMC $SMC(\bar{E})$ is presented. The average sojourn time in the states of the underlying SMC is indicated next to them in bold font.

The average sojourn time vector of $\bar{E}$ is

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

The sojourn time variance vector of $\bar{E}$ is

$$VAR = \left(\frac{1-\rho}{\rho^2}, \frac{1-\chi}{\chi^2}, 0, \frac{1-\theta}{\theta^2}, \frac{1-\phi}{\phi^2}\right).$$

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

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

The steady-state PMF for $EDTMC(E)$ 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 $\left(0, \frac{1}{3\chi}, 0, \frac{l}{3\theta(l+m)}, \frac{m}{3\theta(l+m)}\right)$.

It remains to normalize the steady-state weighted PMF by dividing it by the sum of its components: $\psi^* SJ^T = \dfrac{\theta\phi(l+m) + \chi(\phi l + \theta m)}{3\chi\theta\phi(l+m)}$.

Thus, the steady-state *PMF* for $SMC(\bar{E})$ is $\varphi = \dfrac{1}{\theta\phi(l+m) + \chi(\phi l + \theta m)} (0, \theta\phi(l+m), 0, \chi\phi l, \chi\phi m)$.

If $l = m$ and $\theta = \phi$, we have

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

Let G be a dynamic expression and $s, \tilde{s} \in DR(G)$, $S, \tilde{S} \subseteq DR(G)$. The following standard *performance indices (measures)* can be calculated based on the steady-state PMF for $SMC(G)$ [26, 27].

• The *average recurrence* (*return*) *time to the state s* (the number of discrete time units required for this) is $\frac{1}{\varphi(s)}$.

• The *fraction of residence time in the state s* is $\varphi(s)$.

• 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 *relative fraction of residence time in the set of states S with respect to that in* $\tilde{S}$ is $\frac{\sum_{s \in S} \varphi(s)}{\sum_{\tilde{s} \in \tilde{S}} \varphi(\tilde{s})}$.

• The *rate of leaving state s* is $\frac{\varphi(s)}{SJ(s)}$.

• The *steady-state probability to perform a step with an activity* (α, κ) is $\sum_{s \in DR(F)} \varphi(s) \sum_{\{\Upsilon | (\alpha, \kappa) \in \Upsilon\}} PT(\Upsilon, s)$.

• The *probability of the event determined by a reward function r on the states* is $\sum_{s \in DR(F)} \varphi(s) r(s)$.

Let $N = (P_N, T_N, W_N, \Omega_N, L_N, M_N)$ be an *LDTSIPN* and $M, \tilde{M} \in \mathbb{N}_f^{P_N}$. Then, the average sojourn time $SJ(M)$, the sojourn time variance $VAR(M)$, the probabilities $PM^*(M, \tilde{M})$, the transition relation $M \twoheadrightarrow_{\mathcal{P}} \tilde{M}$, 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 have mentioned earlier, every marked plain dtsi-box can 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 binds their initial states.

Proposition 5.1. For any static expression E, $SMC(\bar{E}) \simeq SMC(Box_{dtsi}(\bar{E}))$.

Proof. The proposition is proved by means of Theorem 4.1 and definitions of the underlying SMCs for dynamic expressions and LDTSIPNs and by taking into account the following argumentation. First, for the associated SMCs, the average sojourn time in the states is the same, since it is defined via similar probability functions. Second, the transition probabilities of the associated SMCs are sums of those belonging to the transition systems or reachability graphs. □

Example 5.2. Let E be the expression from Example 3.2. Figure 4 shows the underlying SMC $SMC(N)$. Clearly, $SMC(\bar{E})$ and $SMC(N)$ are isomorphic.

5.2. Alternative Solution Methods

Let us consider DTMCs of expressions based on the state change probabilities $PM(s, \tilde{s})$.

Definition 5.2. Let G be a dynamic expression. The *discrete-time Markov chain* (*DTMC*) of G, which is denoted as $DTMC(G)$, has the state space $DR(G)$, the initial state $[G]_{\approx}$, and transitions $s \longrightarrow_{\mathcal{P}} \tilde{s}$, where $\mathcal{P} = PM(s, \tilde{s})$.

DTMCs of static expressions can be defined as well. For $E \in RegStatExpr$, let $DTMC(E) = DTMC(\underline{E})$.

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

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

The steady-state PMF ψ for $DTMC(G)$ is defined like the corresponding notion for $EDTMC(G)$. Let us determine a relationship between steady-state PMFs for $DTMC(G)$ and $EDTMC(G)$. The following theorem proposes the equation that relates the above-mentioned steady-state PMFs. First, we introduce some helpful notation. For a vector $v = (v_1, \ldots, v_n)$, let $Diag(v)$ be the diagonal matrix of order n with the elements $Diag_{ij}(v)$ $(1 \le i, j \le n)$ defined as

$$Diag_{ij}(v) = \begin{cases} v_i, & i = j; \\ 0, & \text{otherwise.} \end{cases} \qquad (1 \le i, j \le n)$$

Theorem 5.1. Let G be a dynamic expression and SL be its self-loops abstraction vector. Then, the steady-state PMFs ψ for $DTMC(G)$ and ψ^* for $EDTMC(G)$ are related as follows: $\forall s \in DR(G)$

$$\psi(s) = \frac{\psi^*(s) SL(s)}{\sum_{\tilde{s} \in DR(G)} \psi^*(\tilde{s}) SL(\tilde{s})}.$$

Proof. Let PSL be a row vector with the elements

$$PSL(s) = \begin{cases} PM(s, s), & s \to s; \\ 0, & \text{otherwise.} \end{cases}$$

By the definition of $PM^*(s, \tilde{s})$, we have $\mathbf{P}^* = Diag(SL)(\mathbf{P} - Diag(PSL))$. Further, $\psi^*(\mathbf{P}^* - \mathbf{I}) = \mathbf{0}$ and $\psi^* \mathbf{P}^* = \psi^*$. After the replacement of $\mathbf{P}^*$ by $Diag(SL)(\mathbf{P} - Diag(PSL))$, we obtain $\psi^* Diag(SL)(\mathbf{P} - Diag(PSL)) = \psi^*$ and $\psi^* Diag(SL)\mathbf{P} = \psi^*(Diag(SL)Diag(PSL) + \mathbf{I})$.

Note that, $\forall s \in DR(G)$, we have

$$SL(s)PSL(s) + 1$$

$$= \begin{cases} SL(s)PM(s, s) + 1 = \dfrac{PM(s, s)}{1 - PM(s, s)} + 1 \\ = \dfrac{1}{1 - PM(s, s)}, \quad s \to s; \\ SL(s) \cdot 0 + 1 = 1, \quad \text{otherwise.} \end{cases} = SL(s).$$

Hence, $Diag(SL)Diag(PSL) + \mathbf{I} = Diag(SL)$. Thus, $\psi^* Diag(SL)\mathbf{P} = \psi^* Diag(SL)$. Then, for $v = \psi^*(Diag(SL)$, we have $v\mathbf{P} = v$ and $v(\mathbf{P} - \mathbf{I}) = \mathbf{0}$.

In order to calculate ψ on the basis of v, we must normalize it by dividing its elements by their sum, since we should have $\psi \mathbf{1}^T = 1$ as a result, i.e., $\psi = \frac{1}{v\mathbf{1}^T} v = \frac{1}{\psi^* Diag(SL)\mathbf{1}^T} \psi^* Diag(SL)$. Thus, the elements of ψ are calculated as follows: $\forall s \in DR(G)\ \psi(s) = \frac{\psi^*(s)SL(s)}{\sum_{\tilde{s} \in DR(G)} \psi^*(\tilde{s})SL(\tilde{s})}$. It is easy to check that ψ is a solution of the equation system $\begin{cases} \psi(\mathbf{P} - \mathbf{I}) = \mathbf{0} \\ \psi \mathbf{1}^T = 1, \end{cases}$; hence, it is indeed the steady-state *PMF* for *DTMC*(*G*). □

The following proposition relates the steady-state PMFs for *SMC*(*G*) and *DTMC*(*G*).

Proposition 5.2. Let G be a dynamic expression, φ be a steady-state PMF for *SMC*(*G*), and ψ be a steady-state PMF for DTMC(G). Then, $\forall s \in DR(G)$,

$$\varphi(s) = \begin{cases} \dfrac{\psi(s)}{\sum\limits_{\tilde{s} \in DR_T(G)} \psi(\tilde{s})}, & s \in DR_T(G); \\ 0, & s \in DR_V(G). \end{cases}$$

Proof. Let $s \in DR_T(G)$. Recall that $\forall s \in DR_T(G)$ $SL(s) = SJ(s)$ and $\forall s \in DR_V(G)$ $SJ(s) = 0$. Then, by Theorem 5.1, we have

$$\frac{\psi(s)}{\sum\limits_{\tilde{s} \in DR_T(G)} \psi(\tilde{s})} = \frac{\dfrac{\psi^*(s)SL(s)}{\sum\limits_{\tilde{s} \in DR(G)} \psi^*(\tilde{s})SL(\tilde{s})}}{\sum\limits_{\tilde{s} \in DR_T(G)} \left(\dfrac{\psi^*(\tilde{s})SL(\tilde{s})}{\sum\limits_{\bar{s} \in DR(G)} \psi^*(\bar{s})SL(\bar{s})} \right)}$$

$$= \frac{\psi^*(s)SL(s)}{\sum\limits_{\tilde{s} \in DR(G)} \psi^*(\tilde{s})SL(\tilde{s})} \cdot \frac{\sum\limits_{\bar{s} \in DR(G)} \psi^*(\bar{s})SL(\bar{s})}{\sum\limits_{\tilde{s} \in DR_T(G)} \psi^*(\tilde{s})SL(\tilde{s})}$$

$$= \frac{\psi^*(s)SL(s)}{\sum\limits_{\tilde{s} \in DR_T(G)} \psi^*(\tilde{s})SL(\tilde{s})} = \frac{\psi^*(s)SJ(s)}{\sum\limits_{\tilde{s} \in DR_T(G)} \psi^*(\tilde{s})SJ(\tilde{s})}$$

$$= \frac{\psi^*(s)SJ(s)}{\sum\limits_{\tilde{s} \in DR(G)} \psi^*(\tilde{s})SJ(\tilde{s})} = \varphi(s). \ \square$$

Thus, to calculate φ, it is sufficient to apply normalization to some elements of ψ (corresponding to the tangible states) instead of abstracting from self-loops to get $\mathbf{P}^*$ and, then, ψ^* followed by weighting by *SJ* and normalization. Hence, using *DTMC*(*G*) instead of *EDTMC*(*G*) allows one to avoid such a multistage analysis. The optimization of the performance estimate we obtained is based on specific features of the SMCs corresponding to algebraic expressions, with the residence time in the states of these SMCs being distributed geometrically or equal to zero.

Example 5.3. Let E be the expression from Example 3.5. The TPM for $DTMC(\bar{E})$ is

$$\mathbf{P} = \begin{pmatrix} 1-\rho & \rho & 0 & 0 & 0 \\ 0 & 1-\chi & \chi & 0 & 0 \\ 0 & 0 & 0 & \dfrac{l}{1+m} & \dfrac{m}{l+m} \\ 0 & \theta & 0 & 1-\theta & 0 \\ 0 & \phi & 0 & 0 & 1-\phi \end{pmatrix}.$$

The steady-state PMF for $DTMC(\bar{E})$ is

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

Recall that $DR_T(\bar{E}) = \{s_1, s_2, s_4, s_5\}$ and $DR_V(\bar{E}) = \{s_3\}$. Hence, $\sum_{\tilde{s} \in DR_T(\bar{E})} \psi(\tilde{s}) = \psi(s_1) + \psi(s_2) + \psi(s_4) + \psi(s_5) = \frac{\theta\phi(l+m) + \chi(\phi l + \theta m)}{\theta\phi(1+\chi)(l+m) + \chi(\phi l + \theta m)}$.

By Proposition 5.2, we have

$$\varphi(s_1) = 0 \cdot \frac{\theta\phi(1+\chi)(l+m) + \chi(\phi l + \theta m)}{\theta\phi(l+m) + \chi(\phi l + \theta m)} = 0,$$

$$\varphi(s_2) = \frac{\theta\phi(l+m)}{\theta\phi(1+\chi)(l+m) + \chi(\phi l + \theta m)} \cdot \frac{\theta\phi(1+\chi)(l+m) + \chi(\phi l + \theta m)}{\theta\phi(l+m) + \chi(\phi l + \theta m)} = \frac{\theta\phi(l+m)}{\theta\phi(l+m) + \chi(\phi l + \theta m)},$$

$$\varphi(s_3) = 0,$$

$$\varphi(s_4) = \frac{\chi\phi l}{\theta\phi(1+\chi)(l+m) + \chi(\phi l + \theta m)} \cdot \frac{\theta\phi(1+\chi)(l+m) + \chi(\phi l + \theta m)}{\theta\phi(l+m) + \chi(\phi l + \theta m)} = \frac{\chi\phi l}{\theta\phi(l+m) + \chi(\phi l + \theta m)},$$

$$\varphi(s_5) = \frac{\chi\phi m}{\theta\phi(1+\chi)(l+m) + \chi(\phi l + \theta m)} \cdot \frac{\theta\phi(1+\chi)(l+m) + \chi(\phi l + \theta m)}{\theta\phi(l+m) + \chi(\phi l + \theta m)} = \frac{\chi\phi m}{\theta\phi(l+m) + \chi(\phi l + \theta m)}.$$

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

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

This coincides with the result obtained in Example 5.1 with the use of ψ^* and *SJ*.

6. CONCLUSIONS

In this paper, we have considered a discrete-time stochastic extension dtsiPBC of algebra PBC enriched with iteration and immediate multiactions. In dtsiPBC, simultaneous (concurrent) execution of activities is possible owing to the step discrete-time semantics of the calculus; therefore, it is appropriate for specification and analysis of behavior of concurrent systems with random time delays. Standard and alternative methods for evaluating performance of modeled systems are described. The latter methods take into account specific features of the underlying stochastic process of algebraic expressions, such as zero delays in vanishing states of the corresponding SMC. This makes it possible to calculate performance measures in a simpler and more optimal way, which is very important in modeling complex large-scale concurrent systems, the state spaces of which grow drastically when the number of their components increases.

One of the directions of our future work is construction of an equivalence relation that preserves functionality and performance and is a congruence with respect to algebraic operations in dtsiPBC, i.e., the equivalence that withstands their application to equivalent subprocesses while bottom-up design of concurrent systems.. We also plan to extend dtsiPBC with a recursion operator, aiming to specify and analyze behavior of a wider class of infinite processes with a discrete stochastic time.

ACKNOWLEDGMENTS

This work was supported in part by Spanish government, project "Modeling and formal analysis of contracts and Web services with distributed resources", project no. TIN2012-36812-C02-02. The first author was also supported in part by Deutsche Forschungsgemeinschaft (DFG), grant BE 1267/14-1, and Russian Foundation for Basic Research (RFBR), grant 14-01-91334.

REFERENCES

1. Hermanns, H. and Rettelbach, M., Syntax, semantics, equivalences and axioms for MTIPP, *Proc. of the 2nd Workshop on Process Algebras and Performance Modelling (PAPM'94)*, Regensberg, 1994, pp. 71–88.
2. Hillston, J., *A Compositional Approach to Performance Modelling*, Cambridge (UK): Cambridge Univ. Press, UK, 1996.
3. Bernardo, M. and Gorrieri, R., A tutorial on EMPA: a theory of concurrent processes with nondeterminism, priorities, probabilities and time, *Theor. Comput. Sci.*, 1998, vol. 202, pp. 1–54.
4. Best, E., Devillers, R., and Hall, J.G., The box calculus: a new causal algebra with multi-label communication, *Lect. Notes Comp. Sci.*, 1992, vol. 609. pp. 21–69.
5. Best, E. and Koutny, M., A refined view of the box algebra, *Lect. Notes Comp. Sci.*, 1995, vol. 935, pp. 1–20.
6. Best, E., Devillers, R., and Koutny, M., *Petri Net Algebra*, EATCS Monographs on Theoretical Comp. Sci., Springer, 2001.
7. Milner, R.A.J., *Communication and Concurrency*, Upper Saddle River, NJ: Prentice-Hall, 1989.
8. Macià, H., Valero, V., and de Frutos, D., sPBC: a Markovian extension of finite Petri box calculus, *Proc. of the 9th IEEE Int. Workshop on Petri Nets and Performance Models (PNPM'01)*, Aachen: IEEE Comput. Society, 2001. pp. 207–216.
9. Macià, H., Valero, V., Cazorla, D., and Cuartero, F., Introducing the iteration in sPBC, *Lect. Notes Comp. Sci.*, 2004, vol. 3235, pp. 292–308.
10. Macià, H., Valero, V., Cuartero, F., and Ruiz, M.C., sPBC: a Markovian extension of Petri box calculus with immediate multiactions, *Fundamenta Informaticae*, 2008, vol. 87, nos. 3–4, pp. 367–406.
11. Tarasyuk, I.V., Discrete time stochastic Petri box calculus. Oldenburg, Germany, 2005 (Berichte aus dem Department für Informatik. Carl von Ossietzky Univ. Oldenburg. no. 3/05).
12. Tarasyuk, I.V., Stochastic Petri box calculus with discrete time, *Fundamenta Informaticae*, 2007, vol. 76, nos. 1–2, pp. 189–218.
13. Tarasyuk, I.V., Iteration in discrete time stochastic Petri box calculus, *Bulletin of the Novosibirsk Computing Center, Series Comp. Sci., IIS Special Issue*, 2006, vol. 24, pp. 129–148.
14. Tarasyuk, I.V., Macià, H., and Valero, V., Discrete time stochastic Petri box calculus with immediate multiactions, *Technical Report*, Department of Computer Systems, High School of Information Engineering, Univ. of Castilla-La Mancha. no. DIAB-10-03-1.
15. Tarasyuk, I.V., Macià, H., Valero, V., Discrete time stochastic Petri box calculus with immediate multiactions dtsiPBC, in *Electronic Notes in Theoretical Computer Science*, Elsevier, 2013, vol. 296, pp. 229–252.
16. Molloy, M.K., Discrete time stochastic Petri nets, *IEEE Trans. Software Eng.*, 1985, vol. 11, no. 4. pp. 417–423.
17. Ross, S.M., *Stochastic Processes*, New York: Wiley, 1996.
18. Bernardo, M. and Bravetti, M., Reward based congruences: can we aggregate more?, *Lect. Notes Comp. Sci.,* 2001, vol. 2165, pp. 136–151.
19. Balbo, G., Introduction to generalized stochastic Petri nets, *Lect. Notes Comp. Sci.*, 2007, vol. 4486. pp. 83–131.
20. van Glabbeek, R.J., Smolka, S.A., and Steffen, B., Reactive, generative, and stratified models of probabilistic processes, *Information Computation*, 1995, vol. 121, no. 1, pp. 59–80.
21. Zimmermann, A., Freiheit, J., and Hommel, G., Discrete time stochastic Petri nets for modeling and evalu-

ation of real-time systems, *Proc. of the 9th Int. Workshop on Parallel and Distributed Real Time Systems (WPDRTS'01),* San Francisco, 2001, pp. 282–286.

22. Zijal, R., Ciardo, G., and Hommel, G., Discrete deterministic and stochastic Petri nets, *Proc. of the 9th ITG/GI Professional Meeting "Messung, Modellierung und Bewertung von Rechen- und Kommunikationssystemen"*(MMB'97), VDE-Verlag, Berlin, 1997, pp. 103–117.
23. Buchholz P. and Tarasyuk, I.V., Net and algebraic approaches to probabilistic modeling, *Joint Novosibirsk Computing Center and Institute of Informatics Systems Bulletin, Series Comp. Sci.*, Novosibirsk, 2001, vol. 15, pp. 31–64.
24. Haverkort, B.R., Markovian models for performance and dependability evaluation, *Lect. Notes Comp. Sci.*, 2001, vol. 2090, pp. 38–83.
25. van Glabbeek, R.J., The linear time – branching time spectrum II: the semantics of sequential systems with silent moves (extended abstract), *Lect. Notes Comp. Sci.*, 1993, vol. 715, pp. 66–81.
26. Mudge, T.N. and Al-Sadoun, H.B., A semi-Markov model for the performance of multiple-bus systems, *IEEE Trans. Computers*, 1985, vol. C-34, no. 10, pp. 934–942.
27. Katoen, J.-P., Quantitative and qualitative extensions of event structures, *Ph. D.* Thesis, Enschede, The Netherlands, 1996, (CTIT Ph. D.-thesis series. Centre for Telematics and Information Technology, University of Twente, no. 96–09).

Translated by A. Pesterev