S©MR ISSN 1813-3304

СИБИРСКИЕ ЭЛЕКТРОННЫЕ
МАТЕМАТИЧЕСКИЕ ИЗВЕСТИЯ

Siberian Electronic Mathematical Reports
http://semr.math.nsc.ru

Том 12, стр. 513–551 (2015) УДК 004.423.4, 519.217.2, 519.681.2, 519.681.3
MSC 60J10, 60K15, 68Q85

STOCHASTIC PROCESS REDUCTION FOR PERFORMANCE EVALUATION IN DTSIPBC

I.V. TARASYUK, H. MACIÀ, V. VALERO

Abstract. Petri box calculus (PBC) is a well-known algebra of concurrent processes with a Petri net semantics. In the paper, we consider an extension of PBC with discrete stochastic time and immediate multiactions, which is referred to as discrete time stochastic and immediate PBC (dtsiPBC). 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 algebraic process expressions but also by exploring the reduced discrete time Markov chains, obtained from the semi-Markov chains by eliminating their states with zero residence time (called vanishing states). The latter approach simplifies performance analysis of large systems due to abstraction from many instantaneous activities, such as those used to specify logical conditions, probabilistic branching, as well as urgent or internal (unobservable) work.

Keywords: stochastic process algebras, stochastic Petri nets, Petri box calculus, discrete time, immediate multiactions, operational semantics, transition systems, performance analysis, Markov chains, vanishing states, reduction.

Tarasyuk I.V., Macià H., Valero V., Stochastic process reduction for performance evaluation in dtsiPBC.

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. I.V. Tarasyuk 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.

Received January, 29, 2015, published September, 14, 2015.

1. Introduction

Algebraic process calculi are a well-known formal model for the specification of computing systems and analysis of their behaviour. In such process algebras (PAs), systems and processes are specified by formulas, and verification of their properties is accomplished at a syntactic level via equivalences, axioms, and inference rules. In the last decades, stochastic extensions of PAs were proposed and widely used. Stochastic process algebras (SPAs) do not just specify actions that can occur (qualitative features), like ordinary process algebras, but they associate with actions some quantitative parameters (quantitative characteristics), such as rates or probabilities, related to the distributions of the random action delays. Some well-known SPAs are MTIPP [10], PEPA [11] and EMPA [3].

Petri box calculus (PBC) [4, 6, 5] is a flexible and expressive process algebra, based on the CCS calculus [20], and 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 PNs. Formulas of PBC are combined not from single (visible or invisible) actions and variables, like in CCS, but from multisets of elementary actions and their conjugates, called multiactions. The empty multiset of actions is interpreted as the silent multiaction specifying some invisible activity. PBC has a step operational semantics in terms of labeled transition systems, constructed from the rules of the classical structural operational semantics (SOS). The denotational semantics of PBC was defined via a subclass of PNs, equipped with an interface and considered up to isomorphism, called Petri boxes.

A stochastic extension of PBC, called stochastic Petri box calculus (sPBC), was proposed in [16]. In sPBC, delays of stochastic multiactions follow negative exponential distribution. Each multiaction is equipped with a rate that is a parameter of the corresponding exponential distribution. The instantaneous execution of a stochastic multiaction is possible only after the corresponding stochastic time delay. Just a finite part of PBC was initially used for the stochastic enrichment: in its former version, sPBC had neither refinement nor recursion nor iteration. The calculus has an interleaving operational semantics in terms of transition systems, labeled with multiactions and their rates. Its denotational semantics was defined in terms of a subclass of labeled continuous time stochastic PNs (LCTSPNs), based on CTSPNs [18] and called stochastic Petri boxes (s-boxes). In [14], the iteration operation was added to sPBC. In sPBC, performance is evaluated by analyzing the underlying stochastic process, which is a continuous time Markov chain (CTMC).

In [15], sPBC with iteration was enriched with immediate multiactions, having a deterministic zero time delay. We call the resulting calculus generalized stochastic PBC (gsPBC). gsPBC has an interleaving operational semantics via transition systems, labeled with stochastic or immediate multiactions, together with their rates or probabilities, respectively. The denotational semantics of gsPBC was defined via a subclass of labeled generalized stochastic PNs (LGSPNs), based on GSPNs [19, 1] and called generalized stochastic Petri boxes (gs-boxes). The performance analysis in gsPBC is accomplished via the underlying semi-Markov chains (SMCs). Note that 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.

In [24, 26], a discrete time stochastic extension dtsPBC of finite PBC was presented. In dtsPBC, the residence time in the process states is geometrically distributed. A step operational semantics of dtsPBC was constructed via labeled probabilistic transition systems. Its denotational semantics was defined in terms of a subclass of labeled discrete time stochastic PNs (LDTSPNs), based on DTSPNs [21] and called discrete time stochastic Petri boxes (dts-boxes). In [25, 27], dtsPBC was enriched with the iteration operator with the 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 investigate performance evaluation methods for computing systems in the algebra discrete time stochastic and immediate PBC (dtsiPBC), initially introduced in [28]. dtsiPBC is an extension of dtsPBC with iteration by immediate multiactions, having zero delay. Immediate multiactions improve capabilities of specification: they can model instantaneous probabilistic choices, as well as activities whose duration is insignificant compared to those of others. This 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 sojourn time in the states of algebraic processes. The syntax of the algebra dtsiPBC is presented. Then, its step operational semantics, based on labeled probabilistic transition systems, is constructed. Here we do not consider the denotational semantics of the calculus, defined via a subclass of labeled discrete time stochastic and immediate PNs (LDTSPNs with immediate transitions, LDTSIPNs), called dtsi-boxes, since it was extensively described in our previous publications [28, 29, 30]. In those papers, a consistency of the operational and denotational semantics of dtsiPBC was proved, hence, all the results obtained for the former can be easily transferred to the latter.

To evaluate performance in dtsiPBC, we study the underlying stochastic process of its algebraic process expressions, which is an SMC. In addition, the alternative solution methods are developed, based on the underlying discrete time Markov chain (DTMC) and its reduction (RDTMC) by eliminating vanishing states (i.e. those with zero residence time). The method based on the RDTMC is the main contribution of the present paper. With a running illustrative example of the generalized shared memory system, we demonstrate how to apply the developed specification and analysis techniques to realistic concurrent systems. The case study also shows that the novel approach exploiting RDTMCs of the process expressions simplifies performance analysis of the specified systems due to abstracting from their activities with zero or negligible durations, while preserving the steady-state behaviour and the corresponding performance measures.

Comparing with our previous works about dtsiPBC [29, 30], the present paper proposes a novel original performance analysis method, based on RDTMCs of the process expressions, while in [29, 30], just SMCs and DTMCs were taken for that purpose. Further, here we consider a case study of the generalized shared memory system, which is an extension of the standard shared memory system, described in [29], allowing multiactions in the system specification to have arbitrary probabilities and weights, whereas in [30], no application example was considered at all. These generalized probabilities and weights can be seen as variable parameters to be adjusted later for optimization of the system performance.

The paper is organized as follows. In Section 2, the syntax of algebra dtsiPBC is presented. In Section 3, we construct its operational semantics via labeled probabilistic transition systems. The conventional (SMC-based) and alternative (DTMC- and RDTMC-based) methods of performance evaluation are described in Section 4. Section 5 summarizes the results obtained and outlines the research perspectives.

2. Syntax

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

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

We denote the *set of all finite multisets* over a set X by $\mathbb{N}^X_{fin}$. Let $M, M' \in \mathbb{N}^X_{fin}$. 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 $\forall x \in X,\ M(x) \leq M'(x)$. We define $(M + M')(x) = M(x) + M'(x)$ and $(M - M')(x) = \max\{0, M(x) - M'(x)\}$. When $\forall x \in X,\ M(x) \leq 1,\ M$ can be interpreted as a proper set and denoted by $M \subseteq X$. The *set of all subsets* of X is denoted by 2^X.

Let $Act = \{a, b, \ldots\}$ be the set of *elementary actions*. Then $\widehat{Act} = \{\hat{a}, \hat{b}, \ldots\}$ is the set of *conjugated actions (conjugates)* such that $\hat{a} \neq a$ and $\hat{\hat{a}} = a$. Let $\mathcal{A} = Act \cup \widehat{Act}$ be the set of *all actions*, and $\mathcal{L} = \mathbb{N}^{\mathcal{A}}_{fin}$ be the set of *all multiactions*. Note that $\emptyset \in \mathcal{L}$, this corresponds to an internal move, i.e. the execution of a multiaction that contains no visible action names. The *alphabet* of $\alpha \in \mathcal{L}$ is defined as $\mathcal{A}(\alpha) = \{x \in \mathcal{A} \mid \alpha(x) > 0\}$.

A *stochastic multiaction* is a pair (α, ρ), where $\alpha \in \mathcal{L}$ and $\rho \in (0; 1)$ is the *probability* of the multiaction α. This probability is interpreted as that of independent execution of the stochastic multiaction at the next discrete time moment. Such probabilities are used to calculate those to execute (possibly empty) sets of stochastic multiactions after one time unit delay. The probabilities of stochastic multiactions are required not to be equal to 1 to avoid extra model complexity due to assigning with 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. See [21] for the discussion on DTSPNs. This decision also allows us to avoid technical difficulties related to conditioning events with probability 0. Another reason is that not allowing probability 1 for stochastic multiactions excludes a source of potential periodicity (hence, non-ergodicity) in the underlying SMCs of the algebraic expressions. 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 $l \in \mathbb{N}_{\geq 1} = \{1, 2, \ldots\}$ is the non-zero *weight* of the multiaction α. This weight is interpreted as a measure of importance (urgency, interest) or a bonus reward associated with execution of the immediate multiaction at the current discrete time moment. Such weights are used to calculate the probabilities to execute sets of immediate multiactions instantly. 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 multiactions cannot participate together in some step (concurrent execution), i.e. the steps consisting only of immediate multiactions or those including only stochastic multiactions are allowed. Let $\mathcal{IL}$ be the set of *all immediate multiactions*.

Let us note that the same multiaction $\alpha \in \mathcal{L}$ may have different probabilities and weights in the same specification. It is easy to differentiate between probabilities and weights, hence, between stochastic and immediate multiactions, since the probabilities of stochastic multiactions belong to the interval $(0; 1)$, and the weights of immediate multiactions are non-zero (positive) natural numbers from $\mathbb{N}_{\geq 1}$. 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 an activity $(\alpha, \kappa) \in \mathcal{SIL}$ is defined as $\mathcal{A}(\alpha, \kappa) = \mathcal{A}(\alpha)$. The *alphabet* of a multiset of activities $\Upsilon \in \mathbb{N}_{fin}^{\mathcal{SIL}}$ is defined as $\mathcal{A}(\Upsilon) = \cup_{(\alpha,\kappa)\in\Upsilon}\mathcal{A}(\alpha)$.

Activities are combined into formulas (process expressions) by the operations:

;	:	*sequential execution*,
[]	:	*choice*,
$\|$	:	*parallelism*,
$[f]$	:	*relabeling* of actions,
rs	:	*restriction* over a single action,
sy	:	*synchronization* on an action and its conjugate,
$[**]$	:	*iteration* with three arguments: initialization, body and termination.

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

Relabeling functions $f : \mathcal{A} \to \mathcal{A}$ are bijections preserving conjugates, i.e. $\forall x \in \mathcal{A},\ f(\hat{x}) = \widehat{f(x)}$. Relabeling is extended to multiactions in the usual way: for $\alpha \in \mathcal{L}$ we define $f(\alpha) = \sum_{x\in\alpha} f(x)$. Relabeling is extended to activities: for $(\alpha, \kappa) \in \mathcal{SIL}$, we define $f(\alpha, \kappa) = (f(\alpha), \kappa)$. Relabeling is extended to the multisets of activities as follows: for $\Upsilon \in \mathbb{N}_{fin}^{\mathcal{SIL}}$ we define $f(\Upsilon) = \sum_{(\alpha,\kappa)\in\Upsilon}(f(\alpha), \kappa)$. Remember that sums are considered with the multiplicity when applied to multisets: for example, $f(\alpha) = \sum_{x\in\alpha} f(x) = \sum_{x\in\mathcal{A}} \alpha(x)f(x)$.

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

Let $\alpha, \beta \in \mathcal{L}$ be two multiactions such that for some elementary action $a \in Act$ we have $a \in \alpha$ and $\hat{a} \in \beta$, or $\hat{a} \in \alpha$ and $a \in \beta$. Then, synchronization of α and β by a is defined as $\alpha \oplus_a \beta = \gamma$, 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 properties mentioned above, 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 behaviour containing two concurrent activities that can be synchronized by a, there exists also the process behaviour that differs from the former only in that the two activities are replaced by the result of their synchronization.

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

Static expressions specify the structure of processes. As we shall see, the expressions correspond to unmarked LDTSIPNs (note that LDTSIPNs are marked by definition). Remember that a marking is the allocation of tokens in the places of a PN and markings are used to describe dynamic behaviour of PNs in terms of transition firings.

Definition 2. *Let* $(\alpha, \kappa) \in \mathcal{SIL}$ *and* $a \in Act$. *A* static expression *of dtsiPBC is defined as*

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

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

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

To avoid 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, 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 [5] for discussion on this subject. Remember that a PN is *n-bounded* ($n \in \mathbb{N}$) if for all its reachable (from the initial marking by the sequences of transition firings) markings there are at most n tokens in every place, and a PN is *safe* if it is 1-bounded.

Definition 3. *Let* $(\alpha, \kappa) \in \mathcal{SIL}$ *and* $a \in Act$. *A* regular static expression *of dtsiPBC is defined as*

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

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

Dynamic expressions specify the states of processes. As we shall see, the 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 the active components of the system at the current moment of time. The dynamic expression with upper bar (the overlined one) $\overline{E}$ denotes the *initial*, and that with lower bar (the underlined one) $\underline{E}$ denotes the *final* state of the process specified by a static expression E. The *underlying static expression* of a dynamic one is obtained by removing all upper and lower bars from it.

Definition 4. *Let* $E \in StatExpr$ *and* $a \in Act$. *A* dynamic expression *of dtsiPBC is defined as*

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

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 (it is 2-bounded in the worst case [5]).

Definition 5. *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 in the form of $G \Rightarrow \widetilde{G}$ which do not change the states of the specified processes. The goal of these syntactic transformations is to obtain the well-structured resulting expressions, called operative ones, to which no inaction rules can be further applied. However, inaction rules do not bring associated either time elapsing or activities execution, thus, the corresponding marking in the associated LDTSIPN remains unchanged by their application.

Thus, the application of inaction rules does not take 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 regular dynamic expressions in the form of overlined and underlined static ones. In this table, $E, F, K \in RegStatExpr$ and $a \in Act$.

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

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

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

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

TABLE 2. Inaction rules for arbitrary regular dynamic expressions

$\dfrac{G \Rightarrow \widetilde{G},\ \circ \in \{;, []\}}{G \circ E \Rightarrow \widetilde{G} \circ E}$	$\dfrac{G \Rightarrow \widetilde{G},\ \circ \in \{;, []\}}{E \circ G \Rightarrow E \circ \widetilde{G}}$	$\dfrac{G \Rightarrow \widetilde{G}}{G \| H \Rightarrow \widetilde{G} \| H}$
$\dfrac{H \Rightarrow \widetilde{H}}{G \| H \Rightarrow G \| \widetilde{H}}$	$\dfrac{G \Rightarrow \widetilde{G}}{G[f] \Rightarrow \widetilde{G}[f]}$	$\dfrac{G \Rightarrow \widetilde{G},\ \circ \in \{\mathsf{rs}, \mathsf{sy}\}}{G \circ a \Rightarrow \widetilde{G} \circ a}$
$\dfrac{G \Rightarrow \widetilde{G}}{[G * E * F] \Rightarrow [\widetilde{G} * E * F]}$	$\dfrac{G \Rightarrow \widetilde{G}}{[E * G * F] \Rightarrow [E * \widetilde{G} * F]}$	$\dfrac{G \Rightarrow \widetilde{G}}{[E * F * G] \Rightarrow [E * F * \widetilde{G}]}$

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

Note that any dynamic expression can be always transformed into a (not necessarily unique) operative one by using the inaction rules. In the following, we consider regular expressions only and omit the word "regular".

Definition 7. *Let* $\approx = (\Rightarrow \cup \Leftarrow)^*$ *be a structural equivalence of dynamic expressions in dtsiPBC. Thus, two dynamic expressions* G *and* G' *are* structurally equivalent*, denoted by* $G \approx G'$*, if they can be reached from each other by applying the inaction rules in 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 discrete time unit in the same state when no immediate multiactions are executable. In this case, the empty multiset of activities is executed. The action and empty loop rules will be used later to determine all multisets of activities which can be executed from the structural equivalence class of every dynamic expression (i.e. from the state of the corresponding process). This information together with that about probabilities or weights of the activities to be executed from the current process state will be used to calculate the probabilities of such executions.

The action rules with stochastic (or immediate, otherwise) multiactions describe dynamic expression transformations in the form of $G \stackrel{\Gamma}{\to} \widetilde{G}$ (or $G \stackrel{I}{\to} \widetilde{G}$) due to execution of non-empty multisets Γ of stochastic (or I of 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 tick at which some stochastic transitions fire (or to the instantaneous firing of some immediate transitions) and possible change of the current marking. The current marking remains unchanged only if there is a self-loop produced by the iterative execution of a non-empty multiset, which must be one-element, i.e. the single stochastic (or immediate) multiaction. The reason is the regularity requirement that allows no concurrency at the highest level of the second argument of iteration.

The empty loop rule describes dynamic expression transformations in the form of $G \stackrel{\emptyset}{\to} G$ due to execution of the empty multiset of activities at a discrete time tick. The rule reflects a non-zero probability to stay in the current state at the next moment, which is a 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 tick 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. The PBC rule $G \xrightarrow{\emptyset} G$ from [6, 5] in our setting would correspond to the rule $G \Rightarrow G$ describing the stay in the current state when no time elapses. Since we do not need the latter rule to transform dynamic expressions into operative ones and it can destroy the definition of operative expressions, we do not have it.

Thus, the application of action rules with stochastic multiactions or the empty loop rule takes 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 the proper calculation of the state change probabilities for multiple transitions, we can always enumerate coinciding activities from left to right in the syntax of expressions. The new activities resulted from synchronization will be annotated with concatenation of numberings of the activities they come from, hence, the numbering should have a tree structure to reflect the effect of multiple synchronizations. Now we define the numbering which encodes a binary tree with the leaves labeled by natural numbers.

Definition 8. *The* numbering *of expressions is defined as* $\iota ::= \ n \mid (\iota)(\iota),\ n \in \mathbb{N}$.

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

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

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

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

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

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

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

Definition 9. *Let $G \in OpRegDynExpr$. We now define the* set of all non-empty multisets of activities which can be potentially executed from G, *denoted by* $Can(G)$. *Let $(\alpha,\kappa) \in \mathcal{SIL}$, $E, F \in RegStatExpr$, $H \in OpRegDynExpr$ and $a \in Act$.*

(1) *If $final(G)$ then $Can(G) = \emptyset$.*
(2) *If $G = \overline{(\alpha,\kappa)}$ then $Can(G) = \{\{(\alpha,\kappa)\}\}$.*
(3) *If $\Upsilon \in Can(G)$ then $\Upsilon \in Can(G \circ E)$, $\Upsilon \in Can(E \circ G)$ $(\circ \in \{;, []\})$, $\Upsilon \in Can(G \| H)$, $\Upsilon \in Can(H \| G)$, $f(\Upsilon) \in Can(G[f])$, $\Upsilon \in Can(G \text{ sy } a)$ (when $a, \hat{a} \notin \mathcal{A}(\Upsilon)$), $\Upsilon \in Can(G \text{ sy } a)$, $\Upsilon \in Can([G * E * F])$, $\Upsilon \in Can([E * G * F])$, $\Upsilon \in Can([E * F * G])$.*
(4) *If $\Upsilon \in Can(G)$ and $\Xi \in Can(H)$ then $\Upsilon + \Xi \in Can(G \| H)$.*
(5) *If $\Upsilon \in Can(G \text{ sy } a)$ and $(\alpha,\kappa), (\beta,\lambda) \in \Upsilon$ are different activities such that $a \in \alpha$, $\hat{a} \in \beta$, then*
   (a) *$(\Upsilon + \{(\alpha \oplus_a \beta, \kappa \cdot \lambda)\}) \setminus \{(\alpha,\kappa),(\beta,\lambda)\} \in Can(G \text{ sy } a)$, if $\kappa, \lambda \in (0;1)$;*
   (b) *$(\Upsilon + \{(\alpha \oplus_a \beta, \kappa + \lambda)\}) \setminus \{(\alpha,\kappa),(\beta,\lambda)\} \in Can(G \text{ sy } a)$, if $\kappa, \lambda \in \mathbb{N}_{\geq 1}$.*

When we synchronize the same multiset 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. Then we only consider a single one of the resulting activities to avoid introducing redundant ones.

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

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

Let $G \in OpRegDynExpr$. Obviously, if there are only stochastic (or only immediate) multiactions in the multisets 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 multisets from $Can(G)$. By the note above, there are non-empty multisets of immediate multiactions in $Can(G)$ as well, i.e. $\exists \Upsilon \in Can(G)$, $\Upsilon \in \mathbb{N}_{fin}^{\mathcal{IL}} \setminus \{\emptyset\}$. In this case, no stochastic multiactions can be executed from G, even if $Can(G)$ contains non-empty multisets of stochastic multiactions, since immediate multiactions have a priority over stochastic ones, and should be executed first.

Definition 10. *Let $G \in OpRegDynExpr$. The* set of all non-empty multisets of activities which can be executed from G *is*

$$Now(G) = \begin{cases} Can(G), & (Can(G) \subseteq \mathbb{N}_{fin}^{\mathcal{SL}} \setminus \{\emptyset\}) \vee (Can(G) \subseteq \mathbb{N}_{fin}^{\mathcal{IL}} \setminus \{\emptyset\}); \\ Can(G) \cap \mathbb{N}_{fin}^{\mathcal{IL}}, & \text{otherwise.} \end{cases}$$

An expression $G \in OpRegDynExpr$ is *tangible*, denoted by $tang(G)$, if $Now(G) \subseteq \mathbb{N}_{fin}^{\mathcal{SL}} \setminus \{\emptyset\}$. Otherwise, the expression G is *vanishing*, denoted by $vanish(G)$, and in this case $Now(G) \subseteq \mathbb{N}_{fin}^{\mathcal{IL}} \setminus \{\emptyset\}$.

In Table 3, we define the action and empty loop rules. In this table, $(\alpha,\rho), (\beta,\chi) \in \mathcal{SL}$, $(\alpha,l), (\beta,m) \in \mathcal{IL}$ and $(\alpha,\kappa) \in \mathcal{SIL}$. Further, $E, F \in RegStatExpr$, $G, H \in$

$OpRegDynExpr$, $\widetilde{G}, \widetilde{H} \in RegDynExpr$ and $a \in Act$. Moreover, $\Gamma, \Delta \in \mathbb{N}^{\mathcal{SL}}_{fin} \setminus \{\emptyset\}$, $\Gamma' \in \mathbb{N}^{\mathcal{SL}}_{fin}$, $I, J \in \mathbb{N}^{\mathcal{IL}}_{fin} \setminus \{\emptyset\}$, $I' \in \mathbb{N}^{\mathcal{IL}}_{fin}$ and $\Upsilon \in \mathbb{N}^{\mathcal{SIL}}_{fin} \setminus \{\emptyset\}$. The first rule in the table is the empty loop rule **El**. The other rules are the action rules, describing transformations of dynamic expressions, which are built using particular algebraic operations. If we cannot merge a rule with stochastic multiactions and a rule with immediate multiactions for some operation then we get the coupled action rules. Then the names of the action rules with immediate multiactions have a suffix '**i**'.

TABLE 3. Action and empty loop rules

El $\dfrac{tang(G)}{G \xrightarrow{\emptyset} G}$ **B** $\overline{(\alpha,\kappa)} \overset{\{(\alpha,\kappa)\}}{\longrightarrow} \underline{(\alpha,\kappa)}$ **S** $\dfrac{G \xrightarrow{\Upsilon} \widetilde{G}}{G; E \xrightarrow{\Upsilon} \widetilde{G}; E,\ E; G \xrightarrow{\Upsilon} E; \widetilde{G}}$

C $\dfrac{G \xrightarrow{\Gamma} \widetilde{G},\ \neg init(G) \vee (init(G) \wedge tang(\overline{E}))}{G[]E \xrightarrow{\Gamma} \widetilde{G}[]E,\ E[]G \xrightarrow{\Gamma} E[]\widetilde{G}}$ **Ci** $\dfrac{G \xrightarrow{I} \widetilde{G}}{G[]E \xrightarrow{I} \widetilde{G}[]E,\ E[]G \xrightarrow{I} E[]\widetilde{G}}$

P1 $\dfrac{G \xrightarrow{\Gamma} \widetilde{G},\ tang(H)}{G\|H \xrightarrow{\Gamma} \widetilde{G}\|H,\ H\|G \xrightarrow{\Gamma} H\|\widetilde{G}}$ **P1i** $\dfrac{G \xrightarrow{I} \widetilde{G}}{G\|H \xrightarrow{I} \widetilde{G}\|H,\ H\|G \xrightarrow{I} H\|\widetilde{G}}$

P2 $\dfrac{G \xrightarrow{\Gamma} \widetilde{G},\ H \xrightarrow{\Delta} \widetilde{H}}{G\|H \xrightarrow{\Gamma+\Delta} \widetilde{G}\|\widetilde{H}}$ **P2i** $\dfrac{G \xrightarrow{I} \widetilde{G},\ H \xrightarrow{J} \widetilde{H}}{G\|H \xrightarrow{I+J} \widetilde{G}\|\widetilde{H}}$

L $\dfrac{G \xrightarrow{\Upsilon} \widetilde{G}}{G[f] \xrightarrow{f(\Upsilon)} \widetilde{G}[f]}$ **Rs** $\dfrac{G \xrightarrow{\Upsilon} \widetilde{G},\ a, \hat{a} \notin \mathcal{A}(\Upsilon)}{G\ \mathsf{rs}\ a \xrightarrow{\Upsilon} \widetilde{G}\ \mathsf{rs}\ a}$

I1 $\dfrac{G \xrightarrow{\Upsilon} \widetilde{G}}{[G * E * F] \xrightarrow{\Upsilon} [\widetilde{G} * E * F]}$ **I2** $\dfrac{G \xrightarrow{\Gamma} \widetilde{G},\ \neg init(G) \vee (init(G) \wedge tang(\overline{F}))}{[E * G * F] \xrightarrow{\Gamma} [E * \widetilde{G} * F]}$

I2i $\dfrac{G \xrightarrow{I} \widetilde{G}}{[E * G * F] \xrightarrow{I} [E * \widetilde{G} * F]}$ **I3** $\dfrac{G \xrightarrow{\Gamma} \widetilde{G},\ \neg init(G) \vee (init(G) \wedge tang(\overline{F}))}{[E * F * G] \xrightarrow{\Gamma} [E * F * \widetilde{G}]}$

I3i $\dfrac{G \xrightarrow{I} \widetilde{G}}{[E * F * G] \xrightarrow{I} [E * F * \widetilde{G}]}$ **Sy1** $\dfrac{G \xrightarrow{\Upsilon} \widetilde{G}}{G\ \mathsf{sy}\ a \xrightarrow{\Upsilon} \widetilde{G}\ \mathsf{sy}\ a}$

Sy2 $\dfrac{G\ \mathsf{sy}\ a \xrightarrow{\Gamma'+\{(\alpha,\rho)\}+\{(\beta,\chi)\}} \widetilde{G}\ \mathsf{sy}\ a,\ a \in \alpha,\ \hat{a} \in \beta}{G\ \mathsf{sy}\ a \xrightarrow{\Gamma'+\{(\alpha \oplus_a \beta, \rho \cdot \chi)\}} \widetilde{G}\ \mathsf{sy}\ a}$

Sy2i $\dfrac{G\ \mathsf{sy}\ a \xrightarrow{I'+\{(\alpha,l)\}+\{(\beta,m)\}} \widetilde{G}\ \mathsf{sy}\ a,\ a \in \alpha,\ \hat{a} \in \beta}{G\ \mathsf{sy}\ a \xrightarrow{I'+\{(\alpha \oplus_a \beta, l+m)\}} \widetilde{G}\ \mathsf{sy}\ a}$

Almost all the rules in Table 3 (excepting **El**, **P2**, **P2i**, **Sy2** and **Sy2i**) resemble those of gsPBC [15], but the former correspond to execution of multisets of activities, not of single activities, as in the latter, and our rules have simpler preconditions (if any), since all immediate multiactions in dtsiPBC have the same priority level, unlike those of gsPBC. The preconditions in rules **El**, **C**, **P1**, **I2** and **I3** are needed to ensure that (possibly empty) multisets of stochastic multiactions are executed only from *tangible* operative dynamic expressions, such that all operative dynamic expressions structurally equivalent to them are tangible as well. For example, if

$init(G)$ in rule **C** then $G = \overline{F}$ for some static expression F and $G[]E = \overline{F}[]E \approx F[]\overline{E}$. Hence, it should be guaranteed that $tang(F[]\overline{E})$, which holds iff $tang(\overline{E})$. The case $E[]G$ is treated similarly. Further, in rule **P1**, assuming that $tang(G)$, it should be guaranteed that $tang(G\|H)$ and $tang(H\|G)$, which holds iff $tang(H)$. The preconditions in rules **I2** and **I3** are analogous to that in rule **C**.

Rule **El** corresponds to one discrete time unit delay while executing no activities and therefore it has no analogues among the rules of gsPBC that adopts the continuous time model. Rules **P2** and **P2i** have no similar rules in gsPBC, since interleaving semantics of the algebra allows no simultaneous execution of activities. Rules **P2** and **P2i** have in PBC the analogous rule **PAR** that is used to construct step semantics of the calculus, but the former two rules correspond to execution of multisets of activities, unlike that of multisets of multiactions in the latter rule.

Rules **Sy2** and **Sy2i** differ from the corresponding synchronization rules in gsPBC, since the probability or the weight of synchronization in the former rules and the rate or the weight of synchronization in the latter rules are calculated in two distinct ways. Rule **Sy2** establishes that the synchronization of two stochastic multiactions is made by taking the product of their probabilities, since we are considering that both must occur for the synchronization to happen, so this corresponds to the probability of the independent event intersection, but the real situation is more complex, since these stochastic multiactions can also be executed in parallel. Nevertheless, when scoping (the combined operation consisting of synchronization followed by restriction over the same action [5]) is applied over a parallel execution, we get as final result just the simple product of the probabilities, since no normalization is needed there. In rule **Sy2i**, we sum the weights of two synchronized immediate multiactions, since the weights can be interpreted as the rewards [23], thus, we collect the rewards. Moreover, we express that the synchronized execution of immediate multiactions has more importance than that of every single one. The weights of immediate multiactions can also be seen as bonus rewards of transitions [2]. The rewards are summed during synchronized execution of immediate multiactions, since in this case all the synchronized activities can be seen as "operated". We prefer to collect more rewards, thus, the transitions providing greater rewards will have a preference and they will be executed with a greater probability.

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 rather cumbersome and unexpected behaviour, as well as many technical difficulties [5].

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

Definition 11. *The* derivation set *of a dynamic expression G, denoted by $DR(G)$, is the minimal set such that*

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

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

The set of *all multisets of activities executable in* s is defined as $Exec(s) = \{\Upsilon \mid \exists H \in s,\ \exists \widetilde{H},\ H \xrightarrow{\Upsilon} \widetilde{H}\}$.

It can be proved by induction on the structure of expressions that $\Upsilon \in Exec(s) \setminus \{\emptyset\}$ implies $\exists H \in s,\ \Upsilon \in Now(H)$. The reverse statement does not hold in general.

The state s is *tangible*, if $Exec(s) \subseteq \mathbb{N}_{fin}^{\mathcal{SL}}$. For tangible states we may have $Exec(s) = \{\emptyset\}$. Otherwise, the state s is *vanishing*, and in this case $Exec(s) \subseteq \mathbb{N}_{fin}^{\mathcal{IL}} \setminus \{\emptyset\}$. The set of *all tangible states from* $DR(G)$ is denoted by $DR_T(G)$, and the set of *all vanishing states from* $DR(G)$ is denoted by $DR_V(G)$. Obviously, $DR(G) = DR_T(G) \uplus DR_V(G)$, where $\uplus$ denotes disjoint union.

Note that if $\Upsilon \in Exec(s)$ then by rules **P2**, **P2i**, **Sy2**, **Sy2i** and definition of $Exec(s)$ $\forall \Xi \subseteq \Upsilon,\ \Xi \neq \emptyset$ we have $\Xi \in Exec(s)$.

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

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

In the case $\Upsilon = \emptyset$ and $s \in DR_T(G)$ we define

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

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

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

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

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 defines a probability distribution.

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

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

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

Definition 12. *Let* G *be a dynamic expression. The* (labeled probabilistic) transition system *of* G *is a quadruple* $TS(G) = (S_G, L_G, \mathcal{T}_G, s_G)$*, where*

- *the set of* states *is* $S_G = DR(G)$*;*
- *the set of* labels *is* $L_G = \mathbb{N}_{fin}^{\mathcal{SIL}} \times (0; 1]$*;*

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

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

Note that for tangible states Υ can be the empty multiset, and its execution does not change the current state (i.e. the equivalence class), since we have a loop transition $s \stackrel{\emptyset}{\to}_{\mathcal{P}} s$ from a tangible state s to itself. This corresponds to the application of the empty loop rule to the expressions from the equivalence class. We have to keep track of such executions, 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 as they belong to the interval $(0; 1)$. For vanishing states Υ cannot be the empty multiset, since we must execute some immediate multiactions from them at the current time moment.

The step probabilities belong to the interval $(0; 1]$, being 1 in the case when we cannot leave a tangible state s and the only transition leaving it is the empty loop one, $s \stackrel{\emptyset}{\to}_1 s$, or if there is just a single transition from a vanishing state to any other one.

We write $s \stackrel{\Upsilon}{\to} \tilde{s}$ if $\exists \mathcal{P},\ s \stackrel{\Upsilon}{\to}_{\mathcal{P}} \tilde{s}$ and $s \to \tilde{s}$ if $\exists \Upsilon,\ s \stackrel{\Upsilon}{\to} \tilde{s}$.

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

Definition 13. *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 $\beta : S_G \to S_{G'}$ is an* isomorphism *between $TS(G)$ and $TS(G')$, written $\beta : TS(G) \simeq TS(G')$, if*

(1) *β is a bijection such that $\beta(s_G) = s_{G'}$;*

(2) *$\forall s, \tilde{s} \in S_G,\ \forall \Upsilon,\ s \stackrel{\Upsilon}{\to}_{\mathcal{P}} \tilde{s} \ \Leftrightarrow\ \beta(s) \stackrel{\Upsilon}{\to}_{\mathcal{P}} \beta(\tilde{s})$.*

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

Definition 14. *Let G be a dynamic expression. The* operational semantics *of dtsiPBC is a mapping TS_{dtsi} from $OpRegDynExpr$ into the domain of isomorphism classes (i.e. equivalence classes w.r.t. $\simeq$) of labeled probabilistic transition systems, defined as follows: $TS_{dtsi}(G) = [TS(G)]_{\simeq}$.*

Let E be a static expression. The operational semantics of dtsiPBC is extended to $RegStatExpr$ as follows: $TS_{dtsi}(E) = [TS(\overline{E})]_{\simeq}$.

Example 1. *Consider a model of two processors accessing a common shared memory described in* [19, 1] *in the continuous time setting on GSPNs. We shall analyze this shared memory system in the discrete time stochastic setting of dtsiPBC, where*

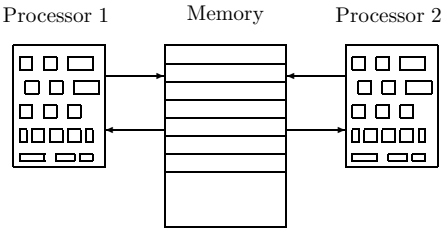

FIG. 1. The diagram of the shared memory system

concurrent execution of activities is possible, while no two transitions of a GSPN may fire simultaneously (in parallel). Our model parameterizes that from [29]. *The model behaves as follows. After activation of the system (turning the computer on), two processors are active, and the common memory is available. Each processor can request an access to the memory after which the instantaneous decision is made. When the decision is made in favour of a processor, it starts acquisition of the memory and the other processor should wait until the former one ends its memory operations, and the system returns to the state with both active processors and available common memory. The diagram of the system is depicted in Fig. 1.*

The meaning of actions from the syntax of dtsiPBC expressions which will specify the system modules is as follows. The action a corresponds to the system activation. The actions r_i $(1 \leq i \leq 2)$ *represent the common memory request of processor* i. *The instantaneous actions* d_i *correspond to the decision on the memory allocation in favour of the processor* i. *The actions* m_i *represent the common memory access of processor* i. *The other actions are used for communication purposes only via synchronization, and we abstract from them later using restriction. The expression* $\mathsf{Stop} = (\{g\}, \frac{1}{2})$ rs g *specifies a non-terminating process that performs only empty loops with probability* 1, *since its behaviour cannot contain* g *or* $\hat{g}$ *by definition of the restriction operator. We take general values for all multiaction probabilities and weights in the specification. Let all stochastic multiactions have the same generalized probability* $\rho \in (0; 1)$, *and all immediate ones have the same generalized weight* $l \in \mathbb{N}_{\geq 1}$. *The specification* K *of the generalized shared memory system is as follows.*

The static expression of the first processor is

$$K_1 = [(\{x_1\}, \rho) * ((\{r_1\}, \rho); (\{d_1, y_1\}, l); (\{m_1, z_1\}, \rho)) * \mathsf{Stop}].$$

The static expression of the second processor is

$$K_2 = [(\{x_2\}, \rho) * ((\{r_2\}, \rho); (\{d_2, y_2\}, l); (\{m_2, z_2\}, \rho)) * \mathsf{Stop}].$$

The static expression of the shared memory is

$$K_3 = [(\{a, \widehat{x_1}, \widehat{x_2}\}, \rho) * ((((\{\widehat{y_1}\}, l); (\{\widehat{z_1}\}, \rho))[]((\{\widehat{y_2}\}, l); (\{\widehat{z_2}\}, \rho))) * \mathsf{Stop}].$$

The static expression of the generalized shared memory system is

$$K = (K_1 \| K_2 \| K_3)\ \mathsf{sy}\ x_1\ \mathsf{sy}\ x_2\ \mathsf{sy}\ y_1\ \mathsf{sy}\ y_2\ \mathsf{sy}\ z_1\ \mathsf{sy}\ z_2\ \mathsf{rs}\ x_1\ \mathsf{rs}\ x_2\ \mathsf{rs}\ y_1\ \mathsf{rs}\ y_2\ \mathsf{rs}\ z_1\ \mathsf{rs}\ z_2.$$

As a result of the synchronization of immediate multiactions $(\{d_i, y_i\}, l)$ *and* $(\{\widehat{y_i}\}, l)$ *we get* $(\{d_i\}, 2l)$ $(1 \leq i \leq 2)$. *The synchronization of stochastic multiactions*

$(\{m_i, z_i\}, \rho)$ and $(\{\widehat{z_i}\}, \rho)$ produces $(\{m_i\}, \rho^2)$ $(1 \le i \le 2)$. The result of synchronization of $(\{a, \widehat{x_1}, \widehat{x_2}\}, \rho)$ with $(\{x_1\}, \rho)$ is $(\{a, \widehat{x_2}\}, \rho^2)$, and that of synchronization of $(\{a, \widehat{x_1}, \widehat{x_2}\}, \rho)$ with $(\{x_2\}, \rho)$ is $(\{a, \widehat{x_1}\}, \rho^2)$. After synchronizing $(\{a, \widehat{x_2}\}, \rho^2)$ and $(\{x_2\}, \rho)$, as well as $(\{a, \widehat{x_1}\}, \rho^2)$ and $(\{x_1\}, \rho)$, we get the same activity $(\{a\}, \rho^3)$.

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

$$\tilde{s}_1 = [([\overline{(\{x_1\}, \rho)} * ((\{r_1\}, \rho); (\{d_1, y_1\}, l); (\{m_1, z_1\}, \rho)) * \mathsf{Stop}]\|$$
$$[\overline{(\{x_2\}, \rho)} * ((\{r_2\}, \rho); (\{d_2, y_2\}, l); (\{m_2, z_2\}, \rho)) * \mathsf{Stop}]\|$$
$$[(\{a, \widehat{x_1}, \widehat{x_2}\}, \rho) * ((((\{\widehat{y_1}\}, l); (\{\widehat{z_1}\}, \rho))[]((\{\widehat{y_2}\}, l); (\{\widehat{z_2}\}, \rho))) * \mathsf{Stop}])$$
$$\mathsf{sy}\ x_1\ \mathsf{sy}\ x_2\ \mathsf{sy}\ y_1\ \mathsf{sy}\ y_2\ \mathsf{sy}\ z_1\ \mathsf{sy}\ z_2\ \mathsf{rs}\ x_1\ \mathsf{rs}\ x_2\ \mathsf{rs}\ y_1\ \mathsf{rs}\ y_2\ \mathsf{rs}\ z_1\ \mathsf{rs}\ z_2]_{\approx},$$

$$\tilde{s}_2 = [([(\{x_1\}, \rho) * \overline{(\{r_1\}, \rho)}; (\{d_1, y_1\}, l); (\{m_1, z_1\}, \rho)) * \mathsf{Stop}]\|$$
$$[(\{x_2\}, \rho) * \overline{(\{r_2\}, \rho)}; (\{d_2, y_2\}, l); (\{m_2, z_2\}, \rho)) * \mathsf{Stop}]\|$$
$$[(\{a, \widehat{x_1}, \widehat{x_2}\}, \rho) * ((((\{\widehat{y_1}\}, l); (\{\widehat{z_1}\}, \rho))[]((\{\widehat{y_2}\}, l); (\{\widehat{z_2}\}, \rho))) * \mathsf{Stop}])$$
$$\mathsf{sy}\ x_1\ \mathsf{sy}\ x_2\ \mathsf{sy}\ y_1\ \mathsf{sy}\ y_2\ \mathsf{sy}\ z_1\ \mathsf{sy}\ z_2\ \mathsf{rs}\ x_1\ \mathsf{rs}\ x_2\ \mathsf{rs}\ y_1\ \mathsf{rs}\ y_2\ \mathsf{rs}\ z_1\ \mathsf{rs}\ z_2]_{\approx},$$

$$\tilde{s}_3 = [([(\{x_1\}, \rho) * (\{r_1\}, \rho); \overline{(\{d_1, y_1\}, 1)}; (\{m_1, z_1\}, \rho)) * \mathsf{Stop}]\|$$
$$[(\{x_2\}, \rho) * \overline{(\{r_2\}, \rho)}; (\{d_2, y_2\}, l); (\{m_2, z_2\}, \rho)) * \mathsf{Stop}]\|$$
$$[(\{a, \widehat{x_1}, \widehat{x_2}\}, \rho) * (((\overline{(\{\widehat{y_1}\}, 1)}; (\{\widehat{z_1}\}, \rho))[]((\{\widehat{y_2}\}, l); (\{\widehat{z_2}\}, \rho))) * \mathsf{Stop}])$$
$$\mathsf{sy}\ x_1\ \mathsf{sy}\ x_2\ \mathsf{sy}\ y_1\ \mathsf{sy}\ y_2\ \mathsf{sy}\ z_1\ \mathsf{sy}\ z_2\ \mathsf{rs}\ x_1\ \mathsf{rs}\ x_2\ \mathsf{rs}\ y_1\ \mathsf{rs}\ y_2\ \mathsf{rs}\ z_1\ \mathsf{rs}\ z_2]_{\approx},$$

$$\tilde{s}_4 = [([(\{x_1\}, \rho) * (\overline{(\{r_1\}, \rho)}; (\{d_1, y_1\}, l); (\{m_1, z_1\}, \rho)) * \mathsf{Stop}]\|$$
$$[(\{x_2\}, \rho) * ((\{r_2\}, \rho); \overline{(\{d_2, y_2\}, 1)}; (\{m_2, z_2\}, \rho)) * \mathsf{Stop}]\|$$
$$[(\{a, \widehat{x_1}, \widehat{x_2}\}, \rho) * ((((\{\widehat{y_1}\}, l); (\{\widehat{z_1}\}, \rho))[](\overline{(\{\widehat{y_2}\}, 1)}; (\{\widehat{z_2}\}, \rho))) * \mathsf{Stop}])$$
$$\mathsf{sy}\ x_1\ \mathsf{sy}\ x_2\ \mathsf{sy}\ y_1\ \mathsf{sy}\ y_2\ \mathsf{sy}\ z_1\ \mathsf{sy}\ z_2\ \mathsf{rs}\ x_1\ \mathsf{rs}\ x_2\ \mathsf{rs}\ y_1\ \mathsf{rs}\ y_2\ \mathsf{rs}\ z_1\ \mathsf{rs}\ z_2]_{\approx},$$

$$\tilde{s}_5 = [([([(\{x_1\}, \rho) * (((\{r_1\}, \rho); (\{d_1, y_1\}, l); \overline{(\{m_1, z_1\}, \rho)}) * \mathsf{Stop}]\|$$
$$[(\{x_2\}, \rho) * \overline{((\{r_2\}, \rho)}; (\{d_2, y_2\}, l); (\{m_2, z_2\}, \rho)) * \mathsf{Stop}]\|$$
$$[(\{a, \widehat{x_1}, \widehat{x_2}\}, \rho) * ((((\{\widehat{y_1}\}, l); \overline{(\{\widehat{z_1}\}, \rho)})[]((\{\widehat{y_2}\}, l); (\{\widehat{z_2}\}, \rho))) * \mathsf{Stop}])$$
$$\mathsf{sy}\ x_1\ \mathsf{sy}\ x_2\ \mathsf{sy}\ y_1\ \mathsf{sy}\ y_2\ \mathsf{sy}\ z_1\ \mathsf{sy}\ z_2\ \mathsf{rs}\ x_1\ \mathsf{rs}\ x_2\ \mathsf{rs}\ y_1\ \mathsf{rs}\ y_2\ \mathsf{rs}\ z_1\ \mathsf{rs}\ z_2]_{\approx},$$

$$\tilde{s}_6 = [([(\{x_1\}, \rho) * ((\{r_1\}, \rho); \overline{(\{d_1, y_1\}, 1)}; (\{m_1, z_1\}, \rho)) * \mathsf{Stop}]\|$$
$$[(\{x_2\}, \rho) * ((\{r_2\}, \rho); \overline{(\{d_2, y_2\}, 1)}; (\{m_2, z_2\}, \rho)) * \mathsf{Stop}]\|$$
$$[(\{a, \widehat{x_1}, \widehat{x_2}\}, \rho) * (((\overline{(\{\widehat{y_1}\}, 1)}; (\{\widehat{z_1}\}, \rho))[](\overline{(\{\widehat{y_2}\}, 1)}; (\{\widehat{z_2}\}, \rho))) * \mathsf{Stop}])$$
$$\mathsf{sy}\ x_1\ \mathsf{sy}\ x_2\ \mathsf{sy}\ y_1\ \mathsf{sy}\ y_2\ \mathsf{sy}\ z_1\ \mathsf{sy}\ z_2\ \mathsf{rs}\ x_1\ \mathsf{rs}\ x_2\ \mathsf{rs}\ y_1\ \mathsf{rs}\ y_2\ \mathsf{rs}\ z_1\ \mathsf{rs}\ z_2]_{\approx},$$

$$\tilde{s}_7 = [([(\{x_1\}, \rho) * \overline{(\{r_1\}, \rho)}; (\{d_1, y_1\}, l); (\{m_1, z_1\}, \rho)) * \mathsf{Stop}]\|$$
$$[(\{x_2\}, \rho) * ((\{r_2\}, \rho); (\{d_2, y_2\}, l); \overline{(\{m_2, z_2\}, \rho)}) * \mathsf{Stop}]\|$$
$$[(\{a, \widehat{x_1}, \widehat{x_2}\}, \rho) * ((((\{\widehat{y_1}\}, l); (\{\widehat{z_1}\}, \rho))[]((\{\widehat{y_2}\}, l); \overline{(\{\widehat{z_2}\}, \rho)})) * \mathsf{Stop}])$$
$$\mathsf{sy}\ x_1\ \mathsf{sy}\ x_2\ \mathsf{sy}\ y_1\ \mathsf{sy}\ y_2\ \mathsf{sy}\ z_1\ \mathsf{sy}\ z_2\ \mathsf{rs}\ x_1\ \mathsf{rs}\ x_2\ \mathsf{rs}\ y_1\ \mathsf{rs}\ y_2\ \mathsf{rs}\ z_1\ \mathsf{rs}\ z_2]_{\approx},$$

$$\tilde{s}_8 = [([(\{x_1\}, \rho) * ((\{r_1\}, \rho); (\{d_1, y_1\}, l); \overline{(\{m_1, z_1\}, \rho)}) * \mathsf{Stop}]\|$$
$$[(\{x_2\}, \rho) * ((\{r_2\}, \rho); (\{d_2, y_2\}, l); (\{m_2, z_2\}, \rho)) * \mathsf{Stop}]\|$$
$$[(\{a, \widehat{x_1}, \widehat{x_2}\}, \rho) * ((((\{\widehat{y_1}\}, l); \overline{(\{\widehat{z_1}\}, \rho)})[]((\{\widehat{y_2}\}, l); (\{\widehat{z_2}\}, \rho))) * \mathsf{Stop}])$$
$$\mathsf{sy}\ x_1\ \mathsf{sy}\ x_2\ \mathsf{sy}\ y_1\ \mathsf{sy}\ y_2\ \mathsf{sy}\ z_1\ \mathsf{sy}\ z_2\ \mathsf{rs}\ x_1\ \mathsf{rs}\ x_2\ \mathsf{rs}\ y_1\ \mathsf{rs}\ y_2\ \mathsf{rs}\ z_1\ \mathsf{rs}\ z_2]_{\approx},$$

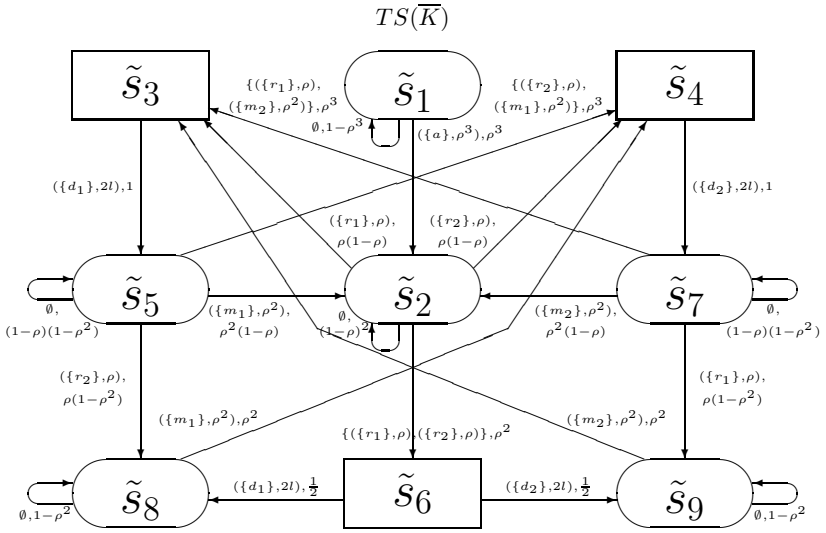

FIG. 2. The transition system of the generalized shared memory system

$$\tilde{s}_9 = [([[(\{x_1\},\rho) * ((\{r_1\},\rho);(\{d_1,y_1\},l);(\{m_1,z_1\},\rho)) * \mathsf{Stop}]\|$$
$$[(\{x_2\},\rho) * ((\{r_2\},\rho);(\{d_2,y_2\},l);\overline{(\{m_2,z_2\},\rho)}) * \mathsf{Stop}]\|$$
$$[(\{a,\widehat{x_1},\widehat{x_2}\},\rho) * ((((\{\widehat{y_1}\},l);(\{\widehat{z_1}\},\rho))[]((\{\widehat{y_2}\},l);\overline{(\{\widehat{z_2}\},\rho)})) * \mathsf{Stop}])$$
$$\mathsf{sy}\ x_1\ \mathsf{sy}\ x_2\ \mathsf{sy}\ y_1\ \mathsf{sy}\ y_2\ \mathsf{sy}\ z_1\ \mathsf{sy}\ z_2\ \mathsf{rs}\ x_1\ \mathsf{rs}\ x_2\ \mathsf{rs}\ y_1\ \mathsf{rs}\ y_2\ \mathsf{rs}\ z_1\ \mathsf{rs}\ z_2]_{\approx}.$$

We have $DR_T(\overline{K}) = \{\tilde{s}_1, \tilde{s}_2, \tilde{s}_5, \tilde{s}_5, \tilde{s}_8, \tilde{s}_9\}$ *and* $DR_V(\overline{K}) = \{\tilde{s}_3, \tilde{s}_4, \tilde{s}_6\}$. *The interpretation of the states is:* $\tilde{s}_1$ *is the initial state,* $\tilde{s}_2$*: the system is activated and the memory is not requested,* $\tilde{s}_3$*: the memory is requested by the first processor,* $\tilde{s}_4$*: the memory is requested by the second processor,* $\tilde{s}_5$*: the memory is allocated to the first processor,* $\tilde{s}_6$*: the memory is requested by two processors,* $\tilde{s}_7$*: the memory is allocated to the second processor,* $\tilde{s}_8$*: the memory is allocated to the first processor and the memory is requested by the second processor,* $\tilde{s}_9$*: the memory is allocated to the second processor and the memory is requested by the first processor.*

In Fig. 2, the transition system $TS(\overline{K})$ *is presented. The tangible states are depicted in ovals, the vanishing ones are depicted in boxes. To simplify the graphical representation, the singleton multisets of activities are written without braces. Note that, in step semantics, we may execute the following activities in parallel:* $(\{r_1\},\rho)$, $(\{r_2\},\rho)$*, as well as* $(\{r_1\},\rho),(\{m_2\},\rho^2)$*, and* $(\{r_2\},\rho),(\{m_1\},\rho^2)$.

4. Performance evaluation

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

We are interested in the expressions specifying the processes with infinite behaviour, i.e. in those with the iteration operator, since this is a unique source of infiniteness within dtsiPBC. Note that the presence of iteration does not guarantee infiniteness of behaviour, since there may exist a deadlock within the body (second argument) of iteration when the corresponding subprocess does not reach its final state by some reason. In particular, if the body of iteration contains the Stop

expression, then iteration will be "broken". On the other hand, the iteration body can be left after a finite number of its repeated executions and then the iteration termination is started. To avoid executing any activities after the iteration body, we take Stop as the termination argument of iteration.

In the framework of Markov chains, the most common systems for performance analysis are *ergodic* (irreducible, positive recurrent and aperiodic) ones. For ergodic Markov chains, the steady-state probabilities exist and can be determined. Here we consider only the process expressions such that their underlying Markov chains contain exactly one closed communication class of states, and this class should also be ergodic to ensure uniqueness of the stationary distribution. Remember that a communication class of states is their equivalence class w.r.t. communication, i.e. a maximal subset of communicating states. If a Markov chain contains several closed communication classes that are all ergodic then several stationary distributions may exist, which depend on the initial probability mass function (PMF). There is an analytical method to determine stationary probabilities for Markov chains of this kind as well [13]. We shall see that the underlying Markov chain of every process expression will have only one initial PMF (that at the time moment 0), hence, the stationary distribution will be unique in this case too. The general steady-state probabilities are then calculated as the sum of the stationary probabilities of all the ergodic classes of states, weighted by the probabilities to enter these classes, starting from the initial state and passing through some transient states.

4.1. **Analysis of the underlying SMC.** For a dynamic expression G, a discrete random variable is associated with every tangible state $s \in DR_T(G)$. The variable captures the 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 in 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 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 $\frac{1}{1-PM(s,s)}$. Obviously, the average sojourn time in a vanishing state is zero. Let $s \in DR(G)$.

The *average sojourn time in the state* s is

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

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

The *sojourn time variance in the state* s is

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

The *sojourn time variance vector* of G, 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. The process is the underlying semi-Markov chain (SMC) [23, 13], denoted by $SMC(G)$, which can be analyzed by extracting from it the embedded (absorbing) discrete time Markov chain (EDTMC) corresponding to G, denoted by $EDTMC(G)$. The construction of the latter is analogous to that applied in the context of generalized stochastic PNs (GSPNs) in [19, 1]. $EDTMC(G)$ only describes the state changes of $SMC(G)$ while ignoring its time characteristics. Thus, to construct the EDTMC, we should abstract from all time aspects of behaviour of the SMC, i.e. from the sojourn time in its states. The (local) sojourn time in every state of the EDTMC is deterministic and it 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) [9].

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

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

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

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

$$PM^*(s,\tilde{s}) = \left\{ \begin{array}{ll} PM(s,\tilde{s}) \sum_{k=0}^{\infty} (PM(s,s))^k = \frac{PM(s,\tilde{s})}{1-PM(s,s)}, & s \to s; \\ PM(s,\tilde{s}), & \text{otherwise;} \end{array} \right\} =$$

$$SL(s) PM(s,\tilde{s}), \text{ where } SL(s) = \left\{ \begin{array}{ll} \frac{1}{1-PM(s,s)}, & s \to s; \\ 1, & \text{otherwise.} \end{array} \right.$$

Here $SL(s)$ is the *self-loops abstraction factor in the state s*. The *self-loops abstraction vector* of G, denoted by SL, has the elements $SL(s),\ s \in DR(G)$. The value $k = 0$ in the summation above corresponds to the case when no self-loops occur. Note that $\forall s \in DR_T(G),\ SL(s) = \frac{1}{1-PM(s,s)} = SJ(s)$, hence, $\forall s \in DR_T(G),\ PM^*(s,\tilde{s}) = SJ(s)PM(s,\tilde{s})$, since we always have the 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, $\forall s \in DR_V(G),\ PM^*(s,\tilde{s}) = \frac{PM(s,\tilde{s})}{1-PM(s,s)}$, 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 $PM^*(s,\tilde{s})$ defines a probability distribution, since $\forall s \in DR(G)$ such that s is not a terminal state, i.e. there are transitions to different states after possible self-loops from it, we have $\sum_{\{\tilde{s} | s \to \tilde{s},\ s \neq \tilde{s}\}} PM^*(s,\tilde{s}) =$
$\frac{1}{1-PM(s,s)} \sum_{\{\tilde{s} | s \to \tilde{s},\ s \neq \tilde{s}\}} PM(s,\tilde{s}) = \frac{1}{1-PM(s,s)}(1 - PM(s,s)) = 1.$

Definition 15. *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 $s \to \tilde{s}$ 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 $1 - PM(s,s)$ while the sojourn time in every $s \in DR_V(G)$ is zero.*

Let G be a dynamic expression. The elements $\mathcal{P}^*_{ij}$ $(1 \leq i, j \leq n = |DR(G)|)$ 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 calculated as

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

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 a row vector of n values 0, $\mathbf{1}$ is that of n values 1.

If $EDTMC(G)$ has a single steady-state distribution then $\psi^* = \lim_{k\to\infty} \psi^*[k]$.

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. Remember that for a vanishing state $s \in DR_V(G)$ we have $SJ(s) = 0$.

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

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

Thus, to calculate φ, we apply abstraction from self-loops to get $\mathbf{P}^*$ and then ψ^*, followed by weighting by SJ and normalization. $EDTMC(G)$ has no self-loops, unlike $SMC(G)$, hence, the behaviour of $EDTMC(G)$ stabilizes quicker than that of $SMC(G)$ (if each of them has a single steady-state distribution), since $\mathbf{P}^*$ has only zero elements at the main diagonal.

Let G be a dynamic expression and $s, \tilde{s} \in DR(G),\ S, \widetilde{S} \subseteq DR(G)$. The following standard *performance indices (measures)* can be calculated, based on the steady-state PMF φ for $SMC(G)$ and the average sojourn time vector SJ of G [22, 8, 12].

- The *average recurrence (return) time in the state s* (i.e. the number of discrete time units or steps required for this) is $\frac{1}{\varphi(s)}$.
- The *fraction of residence time in the state s* is $\varphi(s)$.
- The *fraction of residence time in the set of states S* 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 $\widetilde{S}$* is $\frac{\sum_{s\in S} \varphi(s)}{\sum_{\tilde{s}\in \widetilde{S}} \varphi(\tilde{s})}$.
- The *rate of leaving the state s* is $\frac{\varphi(s)}{SJ(s)}$.

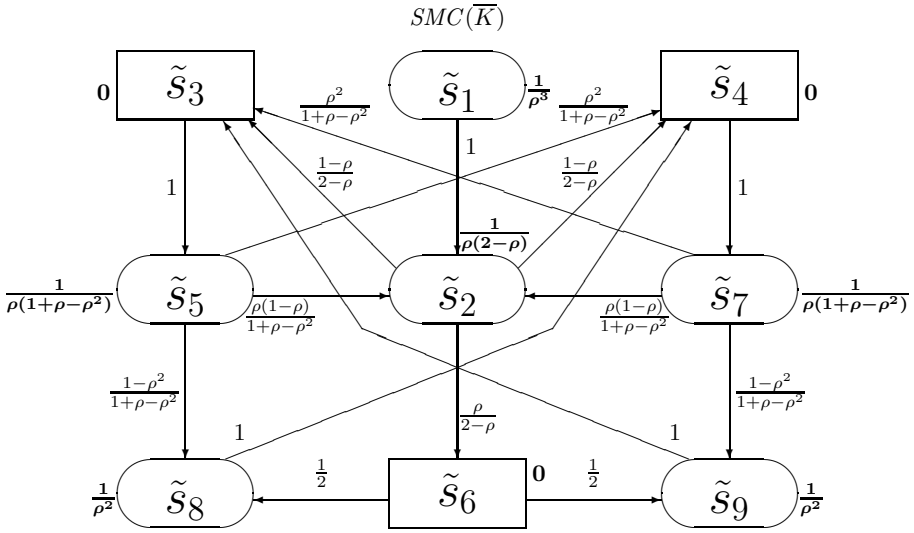

FIG. 3. The underlying SMC of the generalized shared memory system

- The *steady-state probability to perform a step with an activity* (α, κ) is $\sum_{s \in DR(G)} \varphi(s) \sum_{\{\Upsilon | (\alpha,\kappa) \in \Upsilon\}} PT(\Upsilon, s)$.
- The *probability of the event determined by a reward function* r *on the states* is $\sum_{s \in DR(G)} \varphi(s) r(s)$, where $\forall s \in DR(G)\ 0 \leq r(s) \leq 1$.

Example 2. *Let* K *be from Example 1. In Fig. 3, the underlying SMC* $SMC(\overline{K})$ *is depicted. The average sojourn time in the states of the underlying SMC is written next to them in bold font.*

The average sojourn time vector of $\overline{K}$ *is*

$$\widetilde{SJ} = \left(\frac{1}{\rho^3}, \frac{1}{\rho(2-\rho)}, 0, 0, \frac{1}{\rho(1+\rho-\rho^2)}, 0, \frac{1}{\rho(1+\rho-\rho^2)}, \frac{1}{\rho^2}, \frac{1}{\rho^2} \right).$$

The sojourn time variance vector of $\overline{K}$ *is*

$$\widetilde{VAR} = \left(\tfrac{1-\rho^3}{\rho^6}, \tfrac{(1-\rho)^2}{\rho^2(2-\rho)^2}, 0, 0, \tfrac{(1-\rho)^2(1+\rho)}{\rho^2(1+\rho-\rho^2)^2}, 0, \tfrac{(1-\rho)^2(1+\rho)}{\rho^2(1+\rho-\rho^2)^2}, \tfrac{1-\rho^2}{\rho^4}, \tfrac{1-\rho^2}{\rho^4} \right).$$

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

$$\widetilde{\mathbf{P}}^* = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1-\rho}{2-\rho} & \frac{1-\rho}{2-\rho} & 0 & \frac{\rho}{2-\rho} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & \frac{\rho(1-\rho)}{1+\rho-\rho^2} & 0 & \frac{\rho^2}{1+\rho-\rho^2} & 0 & 0 & 0 & \frac{1-\rho^2}{1+\rho-\rho^2} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & \frac{\rho(1-\rho)}{1+\rho-\rho^2} & \frac{\rho^2}{1+\rho-\rho^2} & 0 & 0 & 0 & 0 & 0 & \frac{1-\rho^2}{1+\rho-\rho^2} \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

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

$$\tilde{\psi}^* = \tfrac{1}{2(6+3\rho-9\rho^2+2\rho^3)}(0, 2\rho(2-3\rho-\rho^2), 2+\rho-3\rho^2+\rho^3, 2+\rho-3\rho^2+\rho^3,$$
$$2+\rho-3\rho^2+\rho^3, 2\rho^2(1-\rho), 2+\rho-3\rho^2+\rho^3, 2-\rho-\rho^2, 2-\rho-\rho^2).$$

The steady-state PMF $\tilde{\psi}^*$ weighted by $\widetilde{SJ}$ is

$$\frac{1}{2\rho^2(6+3\rho-9\rho^2+2\rho^3)}(0, 2\rho^2(1-\rho), 0, 0, \rho(2-\rho), 0, \rho(2-\rho), 2-\rho-\rho^2, 2-\rho-\rho^2).$$

We normalize the steady-state weighted PMF, dividing it by the sum of its components

$$\tilde{\psi}^* \widetilde{SJ}^T = \frac{2+\rho-\rho^2-\rho^3}{\rho^2(6+3\rho-9\rho^2+2\rho^3)}.$$

The steady-state PMF for SMC($\overline{K}$) is

$$\tilde{\varphi} = \frac{1}{2(2+\rho-\rho^2-\rho^3)}(0, 2\rho^2(1-\rho), 0, 0, \rho(2-\rho), 0, \rho(2-\rho), 2-\rho-\rho^2, 2-\rho-\rho^2).$$

We can now calculate the main performance indices.

- *The average recurrence time in the state $\tilde{s}_2$, where no processor requests the memory, called the* average system run-through*, is $\frac{1}{\tilde{\varphi}_2} = \frac{2+\rho-\rho^2-\rho^3}{\rho^2(1-\rho)}$.*
- *The common memory is available only in the states $\tilde{s}_2, \tilde{s}_3, \tilde{s}_4, \tilde{s}_6$. The steady-state probability that the memory is available is $\tilde{\varphi}_2 + \tilde{\varphi}_3 + \tilde{\varphi}_4 + \tilde{\varphi}_6 = \frac{\rho^2(1-\rho)}{2+\rho-\rho^2-\rho^3} + 0 + 0 + 0 = \frac{\rho^2(1-\rho)}{2+\rho-\rho^2-\rho^3}$. The steady-state probability that the memory is used (i.e. not available), called the* shared memory utilization*, is $1 - \frac{\rho^2(1-\rho)}{2+\rho-\rho^2-\rho^3} = \frac{2+\rho-2\rho^2}{2+\rho-\rho^2-\rho^3}$.*
- *After activation of the system, we leave the state $\tilde{s}_1$ for ever, and the common memory is either requested or allocated in every remaining state, with exception of $\tilde{s}_2$. Thus, the* rate with which the necessity of shared memory emerges *coincides with the rate of leaving $\tilde{s}_2$, $\frac{\tilde{\varphi}_2}{\widetilde{SJ}_2} = \frac{\rho^2(1-\rho)}{2+\rho-\rho^2-\rho^3} \cdot \frac{\rho(2-\rho)}{1} = \frac{\rho^3(1-\rho)(2-\rho)}{2+\rho-\rho^2-\rho^3}$.*
- *The common memory request of the first processor $(\{r_1\}, \rho)$ is only possible from the states $\tilde{s}_2, \tilde{s}_7$. In each of the states, the request probability is the sum of the execution probabilities for all sets of activities containing $(\{r_1\}, \rho)$. The* steady-state probability of the shared memory request from the first processor *is $\tilde{\varphi}_2 \sum_{\{\Upsilon|(\{r_1\},\rho)\in\Upsilon\}} PT(\Upsilon, \tilde{s}_2) + \tilde{\varphi}_7 \sum_{\{\Upsilon|(\{r_1\},\rho)\in\Upsilon\}} PT(\Upsilon, \tilde{s}_7) = \frac{\rho^2(1-\rho)}{2+\rho-\rho^2-\rho^3}(\rho(1-\rho)+\rho^2) + \frac{\rho(2-\rho)}{2(2+\rho-\rho^2-\rho^3)}(\rho(1-\rho^2)+\rho^3) = \frac{\rho^2(2+\rho-2\rho^2)}{2(2+\rho-\rho^2-\rho^3)}$.*

Example 3. *Let us take $\rho = \frac{1}{2}$ and $l = 1$ in the specification K from Example 1. The resulting specification E describes a special case of the latter, which we call the* standard *shared memory system. In Table 4, the transient and the steady-state probabilities $\psi_i^*[k]$ $(i \in \{1, 2, 3, 5, 6, 8\})$ for the EDTMC of the standard shared memory system at the time moments $k \in \{0, 10, 20, 30, 40, 50\}$ and $k = \infty$ are presented, and in Fig. 4, the alteration diagram (evolution in time) for the transient probabilities is depicted. It is sufficient to consider the probabilities only for the*

TABLE 4. Transient and steady-state probabilities for the EDTMC of the standard shared memory system

k	0	10	20	30	40	50	∞
$\psi_1^*[k]$	1	0	0	0	0	0	0
$\psi_2^*[k]$	0	0.0754	0.0677	0.0680	0.0683	0.0681	0.0682
$\psi_3^*[k]$	0	0.2316	0.1554	0.1741	0.1696	0.1707	0.1705
$\psi_5^*[k]$	0	0.0982	0.1859	0.1672	0.1711	0.1703	0.1705
$\psi_6^*[k]$	0	0.0323	0.0202	0.0234	0.0226	0.0228	0.0227
$\psi_8^*[k]$	0	0.1163	0.1147	0.1130	0.1139	0.1136	0.1136

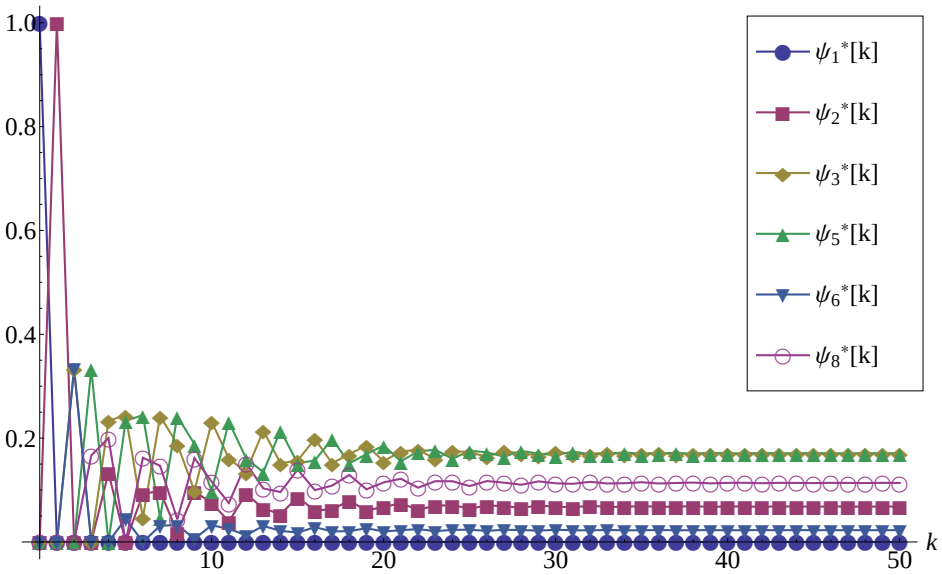

FIG. 4. Transient probabilities alteration diagram for the EDTMC of the standard shared memory system

states $s_1, s_2, s_3, s_5, s_6, s_8$ *of the standard shared memory system, since the corresponding values coincide for* s_3, s_4, *as well as for* s_5, s_7, *and for* s_8, s_9.

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

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

4.2. **Analysis of the DTMC.** Consider an alternative solution method that explores the DTMCs of expressions based on the state change probabilities $PM(s, \tilde{s})$.

Definition 16. *Let* G *be a dynamic expression. The* discrete time Markov chain (DTMC) *of* G, *denoted by* $DTMC(G)$, *has the state space* $DR(G)$, *the initial state* $[G]_{\approx}$ *and the transitions* $s \to_{\mathcal{P}} \tilde{s}$, *where* $\mathcal{P} = PM(s, \tilde{s})$.

Let G be a dynamic expression. The elements $\mathcal{P}_{ij}$ $(1 \le i, j \le n = |DR(G)|)$ of (one-step) transition probability matrix (TPM) $\mathbf{P}$ for $DTMC(G)$ 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 proposition describes the equation that relates the mentioned steady-state PMFs.

We introduce a helpful notation. For a vector $v = (v_1, \ldots, v_n)$, let $Diag(v)$ be a diagonal matrix of order n with the elements $Diag_{ij}(v)$ $(1 \le i, j \le n)$, defined as

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

Proposition 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 vector with the elements

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

By definition of $PM^*(s, \tilde{s})$, we have $\mathbf{P}^* = Diag(SL)(\mathbf{P} - Diag(PSL))$. Further,

$$\psi^*(\mathbf{P}^* - \mathbf{I}) = \mathbf{0} \text{ and } \psi^*\mathbf{P}^* = \psi^*.$$

After replacement of $\mathbf{P}^*$ by $Diag(SL)(\mathbf{P} - Diag(PSL))$ we obtain

$$\psi^* Diag(SL)(\mathbf{P} - Diag(PSL)) = \psi^* \text{ and}$$
$$\psi^* Diag(SL)\mathbf{P} = \psi^*(Diag(SL)Diag(PSL) + \mathbf{I}).$$

Note that $\forall s \in DR(G)$, we have $SL(s)PSL(s) + 1 =$

$$\left\{ \begin{array}{ll} SL(s)PM(s, s) + 1 = \frac{PM(s,s)}{1-PM(s,s)} + 1 = \frac{1}{1-PM(s,s)}, & s \to s; \\ SL(s) \cdot 0 + 1 = 1, & \text{otherwise;} \end{array} \right\} = SL(s).$$

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

$$v\mathbf{P} = v \text{ and } v(\mathbf{P} - \mathbf{I}) = \mathbf{0}.$$

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:

$$\psi = \frac{1}{v\mathbf{1}^T} v = \frac{1}{\psi^* Diag(SL)\mathbf{1}^T} \psi^* Diag(SL).$$

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 next proposition relates the steady-state PMFs for $SMC(G)$ and $DTMC(G)$.

Proposition 2. *Let G be a dynamic expression, φ be the steady-state PMF for $SMC(G)$ and ψ be the steady-state PMF for $DTMC(G)$. Then $\forall s \in DR(G)$,*

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

Proof. Let $s \in DR_T(G)$. Remember that $\forall s \in DR_T(G),\ SL(s) = SJ(s)$ and $\forall s \in DR_V(G),\ SJ(s) = 0$. Then, by Proposition 1, we have

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

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

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

□

Thus, to calculate φ, one can only 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 multistage analysis, but the payment for it is more time-consuming numerical and more complex analytical calculation of ψ with respect to ψ^*. The reason is that $DTMC(G)$ has self-loops, unlike $EDTMC(G)$, hence, the behaviour of $DTMC(G)$ stabilizes slower than that of $EDTMC(G)$ (if each of them has a single steady-state distribution) and $\mathbf{P}$ is more dense matrix than $\mathbf{P}^*$, since $\mathbf{P}$ may additionally have non-zero elements at the main diagonal. Nevertheless, Proposition 2 is very important, since the relationship between φ and ψ it discovers will be used in Proposition 3 to relate the steady-state PMFs for $SMC(G)$ and the reduced $DTMC(G)$.

Example 4. *Let K be from Example 1. In Fig. 5, the DTMC $RDTMC(\overline{K})$ is presented.*

The TPM for $DTMC(\overline{K})$ is

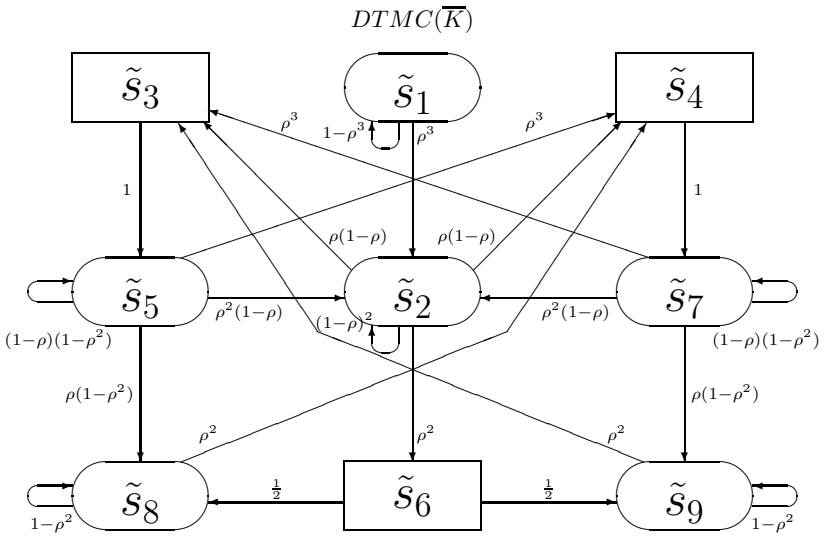

FIG. 5. The DTMC of the generalized shared memory system

$$\widetilde{\mathbf{P}} = \begin{pmatrix} 1-\rho^3 & \rho^3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & (1-\rho)^2 & \rho(1-\rho) & \rho(1-\rho) & 0 & \rho^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & \rho^2(1-\rho) & 0 & \rho^3 & (1-\rho)(1-\rho^2) & 0 & 0 & \rho(1-\rho^2) & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & \rho^2(1-\rho) & \rho^3 & 0 & 0 & 0 & (1-\rho)(1-\rho^2) & 0 & \rho(1-\rho^2) \\ 0 & 0 & 0 & \rho^2 & 0 & 0 & 0 & 1-\rho^2 & 0 \\ 0 & 0 & \rho^2 & 0 & 0 & 0 & 0 & 0 & 1-\rho^2 \end{pmatrix}.$$

The steady-state PMF for $DTMC(\overline{K})$ *is*

$$\tilde{\psi} = \tfrac{1}{2(2+\rho+\rho^2-2\rho^4)}(0, 2\rho^2(1-\rho), \rho^2(2+\rho-3\rho^2+\rho^3), \rho^2(2+\rho-3\rho^2+\rho^3), \\ \rho(2-\rho), 2\rho^4(1-\rho), \rho(2-\rho), 2-\rho-\rho^2, 2-\rho-\rho^2).$$

Remember that $DR_T(\overline{K}) = \{\tilde{s}_1, \tilde{s}_2, \tilde{s}_5, \tilde{s}_5, \tilde{s}_8, \tilde{s}_9\}$ *and* $DR_V(\overline{K}) = \{\tilde{s}_3, \tilde{s}_4, \tilde{s}_6\}$. *Hence,*

$$\sum_{\tilde{s}\in DR_T(\overline{K})} \tilde{\psi}(\tilde{s}) = \tilde{\psi}(\tilde{s}_1) + \tilde{\psi}(\tilde{s}_2) + \tilde{\psi}(\tilde{s}_5) + \tilde{\psi}(\tilde{s}_7) + \tilde{\psi}(\tilde{s}_8) + \tilde{\psi}(\tilde{s}_9) = \frac{2+\rho-\rho^2-\rho^3}{2+\rho+\rho^2-2\rho^4}.$$

By Proposition 2, we have

TABLE 5. Transient and steady-state probabilities for the DTMC of the standard shared memory system

k	0	10	20	30	40	50	∞
$\psi_1[k]$	1	0.2631	0.0692	0.0182	0.0048	0.0013	0
$\psi_2[k]$	0	0.0829	0.0569	0.0501	0.0483	0.0478	0.0476
$\psi_3[k]$	0	0.0677	0.0836	0.0878	0.0889	0.0892	0.0893
$\psi_5[k]$	0	0.0996	0.1315	0.1399	0.1421	0.1427	0.1429
$\psi_6[k]$	0	0.0220	0.0146	0.0126	0.0121	0.0120	0.0119
$\psi_8[k]$	0	0.1487	0.2146	0.2319	0.2365	0.2377	0.2381

$$
\begin{aligned}
&\tilde{\varphi}(\tilde{s}_1) = 0 \cdot \tfrac{2+\rho+\rho^2-2\rho^4}{2+\rho-\rho^2-\rho^3} = 0,\\
&\tilde{\varphi}(\tilde{s}_2) = \tfrac{\rho^2(1-\rho)}{2+\rho+\rho^2-2\rho^4} \cdot \tfrac{2+\rho+\rho^2-2\rho^4}{2+\rho-\rho^2-\rho^3} = \tfrac{\rho^2(1-\rho)}{2+\rho-\rho^2-\rho^3},\\
&\tilde{\varphi}(\tilde{s}_3) = 0,\\
&\tilde{\varphi}(\tilde{s}_4) = 0,\\
&\tilde{\varphi}(\tilde{s}_5) = \tfrac{\rho(2-\rho)}{2(2+\rho+\rho^2-2\rho^4)} \cdot \tfrac{2+\rho+\rho^2-2\rho^4}{2+\rho-\rho^2-\rho^3} = \tfrac{\rho(2-\rho)}{2(2+\rho-\rho^2-\rho^3)},\\
&\tilde{\varphi}(\tilde{s}_6) = 0,\\
&\tilde{\varphi}(\tilde{s}_7) = \tfrac{\rho(2-\rho)}{2(2+\rho+\rho^2-2\rho^4)} \cdot \tfrac{2+\rho+\rho^2-2\rho^4}{2+\rho-\rho^2-\rho^3} = \tfrac{\rho(2-\rho)}{2(2+\rho-\rho^2-\rho^3)},\\
&\tilde{\varphi}(\tilde{s}_8) = \tfrac{2-\rho-\rho^2}{2(2+\rho+\rho^2-2\rho^4)} \cdot \tfrac{2+\rho+\rho^2-2\rho^4}{2+\rho-\rho^2-\rho^3} = \tfrac{2-\rho-\rho^2}{2(2+\rho-\rho^2-\rho^3)},\\
&\tilde{\varphi}(\tilde{s}_9) = \tfrac{2-\rho-\rho^2}{2(2+\rho+\rho^2-2\rho^4)} \cdot \tfrac{2+\rho+\rho^2-2\rho^4}{2+\rho-\rho^2-\rho^3} = \tfrac{2-\rho-\rho^2}{2(2+\rho-\rho^2-\rho^3)}.
\end{aligned}
$$

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

$$\tilde{\varphi} = \frac{1}{2(2+\rho-\rho^2-\rho^3)}(0, 2\rho^2(1-\rho), 0, 0, \rho(2-\rho), 0, \rho(2-\rho), 2-\rho-\rho^2, 2-\rho-\rho^2).$$

This coincides with the result obtained in Example 2 with the use of $\tilde{\psi}^*$ and $\widetilde{SJ}$.

Example 5. *Let E be from Example 3. In Table 5, the transient and the steady-state probabilities $\psi_i[k]$ ($i \in \{1,2,3,5,6,8\}$) for the DTMC of the standard shared memory system at the time moments $k \in \{0,5,10,\ldots,50\}$ and $k=\infty$ are presented, and in Fig. 6, the alteration diagram (evolution in time) for the transient probabilities is depicted. It is sufficient to consider the probabilities for the states $s_1, s_2, s_3, s_5, s_6, s_8$ only, since the corresponding values coincide for s_3, s_4, as well as for s_5, s_7, and for s_8, s_9.*

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

$$\psi = \left(0, \frac{1}{21}, \frac{5}{56}, \frac{5}{56}, \frac{1}{7}, \frac{1}{84}, \frac{1}{7}, \frac{5}{21}, \frac{5}{21}\right).$$

4.3. **Analysis of the reduced DTMC.** Let us now consider the method from [8, 19, 1] that eliminates vanishing states from the EMC (EDTMC, in our terminology) corresponding to the underlying SMC of every GSPN N. The TPM for the resulting *reduced* EDTMC (REDTMC) has smaller size than that for the EDTMC. The method demonstrates that there exists a transformation of the underlying SMC of

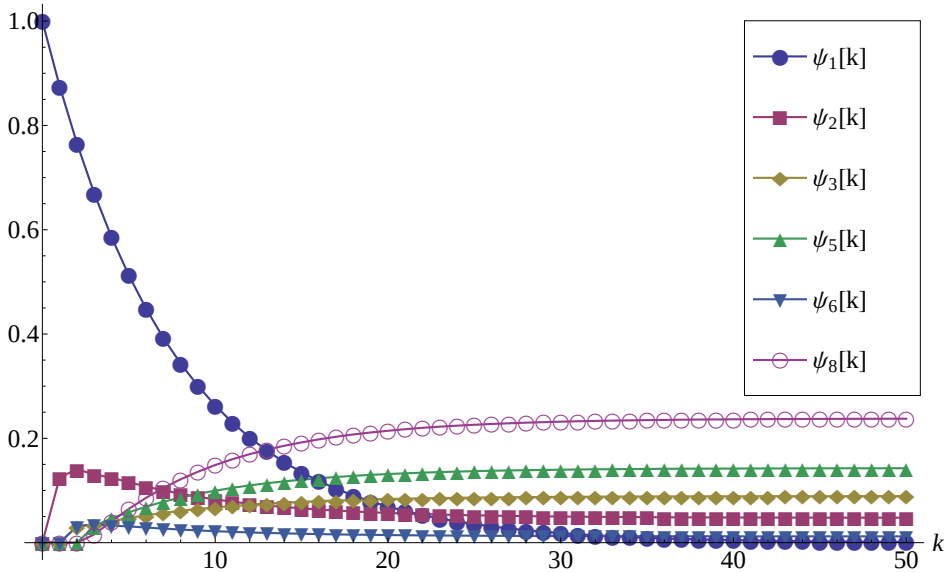

FIG. 6. Transient probabilities alteration diagram for the DTMC of the standard shared memory system

N into a CTMC, whose states are the tangible markings of N. This CTMC, which is essentially the *reduced* underlying SMC (RSMC) of N, is constructed on the basis of the REDTMC. The CTMC can then be directly solved to get the transient and steady-state PMFs over the tangible markings of N. In [8], the program and computational complexities of such an *elimination* method, based on the REDTMC, were evaluated and compared with those of the *preservation* method that does not eliminate vanishing states and based on the EDTMC. The preservation method corresponds in dtsiPBC to the analysis of the underlying SMCs of expressions.

The elimination method can be easily transferred to dtsiPBC, hence, for every dynamic expression G, we can find a DTMC (since the sojourn time in the tangible states from $DR(G)$ is discrete and geometrically distributed) with the states from $DR_T(G)$, which can be directly solved to find the transient and the steady-state PMFs over the tangible states. We shall demonstrate that such a *reduced* DTMC (RDTMC) of G, denoted by $RDTMC(G)$, can be constructed from $DTMC(G)$, using the method analogous to that designed in [19, 1] in the framework of GSPNs to transform EDTMC into REDTMC. Since the sojourn time in the vanishing states is zero, the state transitions of $RDTMC(G)$ occur in the moments of the global discrete time associated with $SMC(G)$, unlike those of $EDTMC(G)$, which happen only when the current state changes to some *different* one, irrespective of the global time. Therefore, in our case, we can skip the stages of constructing the REDTMC of G, denoted by $REDTMC(G)$, from $EDTMC(G)$, and recovering RSMC of G, denoted by $RSMC(G)$, (which is the sought-for DTMC) from $REDTMC(G)$, since we shall have $RSMC(G) = RDTMC(G)$. This will be the second alternative solution method that we propose, namely, exploring the RDTMCs of expressions.

Let G be a dynamic expression and $\mathbf{P}$ be the TPM for $DTMC(G)$. We reorder the states from $DR(G)$ so that the first rows and columns of $\mathbf{P}$ will correspond to the

states from $DR_V(G)$ and the last ones will correspond to the states from $DR_T(G)$. Let $|DR(G)| = n$ and $|DR_T(G)| = m$. The resulting matrix can be decomposed as follows:

$$\mathbf{P} = \begin{pmatrix} \mathbf{C} & \mathbf{D} \\ \mathbf{E} & \mathbf{F} \end{pmatrix}.$$

The elements of the $(n-m) \times (n-m)$ submatrix $\mathbf{C}$ are the probabilities to move from vanishing to vanishing states, and those of the $(n-m) \times m$ submatrix $\mathbf{D}$ are the probabilities to move from vanishing to tangible states. The elements of the $m \times (n-m)$ submatrix $\mathbf{E}$ are the probabilities to move from tangible to vanishing states, and those of the $m \times m$ submatrix $\mathbf{F}$ are the probabilities to move from tangible to tangible states.

The TPM $\mathbf{P}^{\diamond}$ for $RDTMC(G)$ is the $m \times m$ matrix, calculated as

$$\mathbf{P}^{\diamond} = \mathbf{F} + \mathbf{EGD},$$

where the elements of the matrix $\mathbf{G}$ are the probabilities to move from vanishing to vanishing states in any number of state changes, without traversal of tangible states.

If there are no loops among vanishing states then for any vanishing state there exists a value $l \in \mathbb{N}$ such that every sequence of state changes that starts in a vanishing state and is longer than l should reach a tangible state. Thus, $\exists l \in \mathbb{N},\ \forall k > l,\ \mathbf{C}^k = \mathbf{0}$ and $\sum_{k=0}^{\infty} \mathbf{C}^k = \sum_{k=0}^{l} \mathbf{C}^k$.

If there are loops among vanishing states then all such loops are supposed to be of "transient" rather than "absorbing" type, since the latter is treated as a specification error to be corrected, like in [19, 1]. Remember that $SMC(G)$ has a single closed communication class of states, which is also ergodic. The ergodic class cannot consist only of vanishing states to avoid "absorbing" loops among them, hence, it contains tangible states as well. Thus, any sequence of vanishing state changes that starts in the ergodic class will reach a tangible state at some time moment. All the states that do not belong to the ergodic class should be transient. Hence, any sequence of vanishing state changes that starts in a transient vanishing state will some time reach either a transient tangible state or a state from the ergodic class [13]. In the latter case, a tangible state will be reached as well, as argued above. Thus, every sequence of vanishing state changes in $SMC(G)$ that starts in a vanishing state will exit the set of all vanishing states in the future. This implies that the probabilities to move from vanishing to vanishing states in $k \in \mathbb{N}$ state changes, without traversal of tangible states, will lead to 0 when k tends to ∞. Then we have $\lim_{k\to\infty} \mathbf{C}^k = \lim_{k\to\infty} (\mathbf{I} - (\mathbf{I} - \mathbf{C}))^k = \mathbf{0}$, hence, $\mathbf{I} - \mathbf{C}$ is a non-singular matrix, i.e. its determinant is not equal to zero. Thus, the inverse matrix of $\mathbf{I} - \mathbf{C}$ exists and may be expressed by a Neumann series as $\sum_{k=0}^{\infty} (\mathbf{I} - (\mathbf{I} - \mathbf{C}))^k = \sum_{k=0}^{\infty} \mathbf{C}^k = (\mathbf{I} - \mathbf{C})^{-1}$.

Therefore,

$$\mathbf{G} = \sum_{k=0}^{\infty} \mathbf{C}^k = \begin{cases} \sum_{k=0}^{l} \mathbf{C}^k, & \exists l \in \mathbb{N},\ \forall k > l,\ \mathbf{C}^k = \mathbf{0}, & \text{no vanishing states loops;} \\ (\mathbf{I} - \mathbf{C})^{-1}, & \lim_{k\to\infty} \mathbf{C}^k = \mathbf{0}, & \text{vanishing states loops;} \end{cases}$$

where $\mathbf{0}$ is the square matrix consisting only of zeros and $\mathbf{I}$ is the identity matrix, both of order $n - m$.

For $1 \leq i, j \leq m$ and $1 \leq k, l \leq n-m$, let $\mathcal{F}_{ij}$ be the elements of the matrix $\mathbf{F}$, $\mathcal{E}_{ik}$ be those of $\mathbf{E}$, $\mathcal{G}_{kl}$ be those of $\mathbf{G}$ and $\mathcal{D}_{lj}$ be those of $\mathbf{D}$. By definition, the elements $\mathcal{P}^{\diamond}_{ij}$ of the matrix $\mathbf{P}^{\diamond}$ are calculated as

$$\mathcal{P}^{\diamond}_{ij} = \mathcal{F}_{ij} + \sum_{k=1}^{n-m}\sum_{l=1}^{n-m} \mathcal{E}_{ik}\mathcal{G}_{kl}\mathcal{D}_{lj} = \mathcal{F}_{ij} + \sum_{k=1}^{n-m} \mathcal{E}_{ik} \sum_{l=1}^{n-m} \mathcal{G}_{kl}\mathcal{D}_{lj} = \mathcal{F}_{ij} + \sum_{l=1}^{n-m} \mathcal{D}_{lj} \sum_{k=1}^{n-m} \mathcal{E}_{ik}\mathcal{G}_{kl},$$

i.e. $\mathcal{P}^{\diamond}_{ij}$ $(1 \leq i, j \leq m)$ is the total probability to move from the tangible state s_i to the tangible state s_j in any number of steps, without traversal of tangible states, but possibly going through vanishing states.

Let $s, \tilde{s} \in DR_T(G)$ such that $s = s_i,\ \tilde{s} = s_j$. The *probability to move from s to $\tilde{s}$ in any number of steps, without traversal of tangible states* is

$$PM^{\diamond}(s, \tilde{s}) = \mathcal{P}^{\diamond}_{ij}.$$

Definition 17. *Let G be a dynamic expression and $[G]_{\approx} \in DR_T(G)$. The* reduced discrete time Markov chain (RDTMC) *of G, denoted by $RDTMC(G)$, has the state space $DR_T(G)$, the initial state $[G]_{\approx}$ and the transitions $s \hookrightarrow_{\mathcal{P}} \tilde{s}$, where $\mathcal{P} = PM^{\diamond}(s, \tilde{s})$.*

Let us now try to define $RSMC(G)$ as a "restriction" of $SMC(G)$ to its tangible states. Since the sojourn time in the tangible states of $SMC(G)$ is discrete and geometrically distributed, we can see that $RSMC(G)$ is a DTMC with the state space $DR_T(G)$, the initial state $[G]_{\approx}$ and the transitions whose probabilities collect all those in $SMC(G)$ to move from the tangible to the tangible states, directly or indirectly, namely, by going through its vanishing states only. Thus, $RSMC(G)$ has the transitions $s \hookrightarrow_{\mathcal{P}} \tilde{s}$, where $\mathcal{P} = PM^{\diamond}(s, \tilde{s})$, hence, we get $RSMC(G) = RDTMC(G)$.

Let $DR_T(G) = \{s_1, \ldots, s_m\}$ and $[G]_{\approx} \in DR_T(G)$. Then the transient ($k$-step, $k \in \mathbb{N}$) PMF $\psi^{\diamond}[k] = (\psi^{\diamond}[k](s_1), \ldots, \psi^{\diamond}[k](s_m))$ for $RDTMC(G)$ is calculated as

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

where $\psi^{\diamond}[0] = (\psi^{\diamond}[0](s_1), \ldots, \psi^{\diamond}[0](s_m))$ is the initial PMF, defined as

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

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

The steady-state PMF $\psi^{\diamond} = (\psi^{\diamond}(s_1), \ldots, \psi^{\diamond}(s_m))$ for $RDTMC(G)$ is a solution of the equation system

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

where $\mathbf{I}$ is the identity matrix of order m and $\mathbf{0}$ is a row vector of m values 0, $\mathbf{1}$ is that of m values 1.

If $RDTMC(G)$ has a single steady-state distribution then $\psi^{\diamond} = \lim_{k\to\infty} \psi^{\diamond}[k]$.

The zero sojourn time in the vanishing states guarantees that the state transitions of $RDTMC(G)$ occur in the moments of the global discrete time associated with $SMC(G)$, i.e. every such state transition occurs after one time unit delay. Hence, the sojourn time in the tangible states is the same for $RDTMC(G)$ and $SMC(G)$.

The state transition probabilities of $RDTMC(G)$ are those to move from tangible to tangible states in any number of steps, without traversal of the tangible states. Therefore, $RDTMC(G)$ and $SMC(G)$ have the same transient behaviour over the tangible states, thus, the transient analysis of $SMC(G)$ is possible to accomplish using $RDTMC(G)$.
The next proposition relates the steady-state PMFs for $SMC(G)$ and $RDTMC(G)$. It proves that the stationary probabilities of the tangible states coincide for them.

Proposition 3. *Let G be a dynamic expression, φ be the steady-state PMF for $SMC(G)$ and $\psi^\diamond$ be the steady-state PMF for $RDTMC(G)$. Then $\forall s \in DR(G)$,*

$$\varphi(s) = \begin{cases} \psi^\diamond(s), & s \in DR_T(G); \\ 0, & s \in DR_V(G). \end{cases}$$

Proof. To simplify the proof, we use the following unified notation. $\mathbf{I}$ denotes the identity matrices of any size. $\mathbf{0}$ denotes square matrices and row vectors of any size and length of values 0. $\mathbf{1}$ denotes row vectors of any size and length of values 1.

Let $\mathbf{P}$ be the reordered TPM for $DTMC(G)$ and ψ be the steady-state PMF for $DTMC(G)$, i.e. ψ is a solution of the equation system

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

Let $|DR(G)| = n$ and $|DR_T(G)| = m$. The decomposed $\mathbf{P}$, $\mathbf{P} - \mathbf{I}$ and ψ are

$$\mathbf{P} = \begin{pmatrix} \mathbf{C} & \mathbf{D} \\ \mathbf{E} & \mathbf{F} \end{pmatrix}, \ \mathbf{P} - \mathbf{I} = \begin{pmatrix} \mathbf{C} - \mathbf{I} & \mathbf{D} \\ \mathbf{E} & \mathbf{F} - \mathbf{I} \end{pmatrix} \text{ and } \psi = (\psi_V, \psi_T),$$

where $\psi_V = (\psi_1, \ldots, \psi_{n-m})$ is the subvector of ψ with the steady-state probabilities of vanishing states and $\psi_T = (\psi_{n-m+1}, \ldots, \psi_n)$ is that with the steady-state probabilities of tangible states.

Then the equation system for ψ is decomposed as follows:

$$\begin{cases} \psi_V(\mathbf{C} - \mathbf{I}) + \psi_T \mathbf{E} = \mathbf{0} \\ \psi_V \mathbf{D} + \psi_T(\mathbf{F} - \mathbf{I}) = \mathbf{0} \\ \psi_V \mathbf{1}^T + \psi_T \mathbf{1}^T = 1 \end{cases}.$$

Let $\mathbf{P}^\diamond$ be the TPM for $RDTMC(G)$. Then $\psi^\diamond$ is a solution of the equation system

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

We have

$$\mathbf{P}^\diamond = \mathbf{F} + \mathbf{EGD},$$

where the matrix $\mathbf{G}$ can have two different forms, depending on whether the loops among vanishing states exist, hence, we consider the two following cases.

(1) There exist *no loops among vanishing states*. We have $\exists l \in \mathbb{N}, \ \forall k > l, \ \mathbf{C}^k = \mathbf{0}$ and $\mathbf{G} = \sum_{k=0}^{l} \mathbf{C}^k$.

Let us right-multiply the first equation of the decomposed equation system for ψ by $\mathbf{G}$:

$$\psi_V(\mathbf{CG} - \mathbf{G}) + \psi_T \mathbf{EG} = \mathbf{0}.$$

Taking into account that $\mathbf{G} = \sum_{k=0}^{l} \mathbf{C}^k$, we get

$$\psi_V \left(\sum_{k=1}^{l} \mathbf{C}^k + \mathbf{C}^{l+1} - \mathbf{C}^0 - \sum_{k=1}^{l} \mathbf{C}^k \right) + \psi_T \mathbf{EG} = \mathbf{0}.$$

Since $\mathbf{C}^0 = \mathbf{I}$ and $\mathbf{C}^{l+1} = \mathbf{0}$, we obtain

$$-\psi_V + \psi_T \mathbf{EG} = \mathbf{0} \text{ and } \psi_V = \psi_T \mathbf{EG}.$$

Let us substitute ψ_V with $\psi_T \mathbf{EG}$ in the second equation of the decomposed equation system for ψ:

$$\psi_T \mathbf{EGD} + \psi_T(\mathbf{F} - \mathbf{I}) = \mathbf{0} \text{ and } \psi_T(\mathbf{F} + \mathbf{EGD} - \mathbf{I}) = \mathbf{0}.$$

Since $\mathbf{F} + \mathbf{EGD} = \mathbf{P}^\diamond$, we have

$$\psi_T(\mathbf{P}^\diamond - \mathbf{I}) = \mathbf{0}.$$

(2) There exist *loops among vanishing states*. We have $\lim_{\to\infty} \mathbf{C}^k = \mathbf{0}$ and $\mathbf{G} = (\mathbf{I} - \mathbf{C})^{-1}$.

Let us right-multiply the first equation of the decomposed equation system for ψ by $\mathbf{G}$:

$$-\psi_V(\mathbf{I} - \mathbf{C})\mathbf{G} + \psi_T \mathbf{EG} = \mathbf{0}.$$

Taking into account that $\mathbf{G} = (\mathbf{I} - \mathbf{C})^{-1}$, we get

$$-\psi_V + \psi_T \mathbf{EG} = \mathbf{0} \text{ and } \psi_V = \psi_T \mathbf{EG}.$$

Let us substitute ψ_V with $\psi_T \mathbf{EG}$ in the second equation of the decomposed equation system for ψ:

$$\psi_T \mathbf{EGD} + \psi_T(\mathbf{F} - \mathbf{I}) = \mathbf{0} \text{ and } \psi_T(\mathbf{F} + \mathbf{EGD} - \mathbf{I}) = \mathbf{0}.$$

Since $\mathbf{F} + \mathbf{EGD} = \mathbf{P}^\diamond$, we have

$$\psi_T(\mathbf{P}^\diamond - \mathbf{I}) = \mathbf{0}.$$

The third equation $\psi_V \mathbf{1}^T + \psi_T \mathbf{1}^T = 1$ of the decomposed equation system for ψ implies that if ψ_V has nonzero elements then the sum of the elements of ψ_T is less than one. We normalize ψ_T by dividing its elements by their sum:

$$v = \frac{1}{\psi_T \mathbf{1}^T} \psi_T.$$

It is easy to check that v is a solution of the equation system

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

hence, it is the steady-state PMF for $RDTMC(G)$ and we have

$$\psi^\diamond = v = \frac{1}{\psi_T \mathbf{1}^T} \psi_T.$$

Note that $\forall s \in DR_T(G),\ \psi_T(s) = \psi(s)$. Then the elements of $\psi^\diamond$ are calculated as follows: $\forall s \in DR_T(G)$,

$$\psi^{\diamond}(s) = \frac{\psi_T(s)}{\sum_{\tilde{s}\in DR_T(G)} \psi_T(\tilde{s})} = \frac{\psi(s)}{\sum_{\tilde{s}\in DR_T(G)} \psi(\tilde{s})}.$$

By Proposition 2, $\forall s \in DR_T(G),\ \varphi(s) = \frac{\psi(s)}{\sum_{\tilde{s}\in DR_T(G)} \psi(\tilde{s})}$.
Therefore, $\forall s \in DR_T(G)$,

$$\varphi(s) = \frac{\psi(s)}{\sum_{\tilde{s}\in DR_T(G)} \psi(\tilde{s})} = \psi^{\diamond}(s).$$

□

Thus, to calculate φ, one can just take all the elements of $\psi^{\diamond}$ as the steady-state probabilities of 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 $RDTMC(G)$ instead of $EDTMC(G)$ allows one to avoid such a multistage analysis, but constructing $\mathbf{P}^{\diamond}$ also requires some efforts, including calculating matrix powers or inverse matrices. Note that $RDTMC(G)$ has self-loops, unlike $EDTMC(G)$, hence, the behaviour of $RDTMC(G)$ may stabilize slower than that of $EDTMC(G)$ (if each of them has a single steady-state distribution). On the other hand, $\mathbf{P}^{\diamond}$ is smaller and denser matrix than $\mathbf{P}^*$, since $\mathbf{P}^{\diamond}$ has additional non-zero elements not only at the main diagonal, but also many of them outside it. Therefore, in most cases, we have less time-consuming numerical calculation of $\psi^{\diamond}$ with respect to ψ^*. At the same time, the complexity of the analytical calculation of $\psi^{\diamond}$ with respect to ψ^* depends on the model structure, such as the number of vanishing states and loops among them, but usually it is lower, since the matrix size reduction plays an important role in many cases. Hence, for the system models with many immediate activities we normally have a significant simplification of the solution. At the abstraction level of SMCs, the elimination of vanishing states decreases their impact to the solution complexity while allowing immediate activities to specify a comprehensible logical structure of systems at the higher level of transition systems.

Example 6. *Let K be from Example 1. Remember that $DR_T(\overline{K}) = \{\tilde{s}_1, \tilde{s}_2, \tilde{s}_5, \tilde{s}_7, \tilde{s}_8, \tilde{s}_9\}$ and $DR_V(\overline{K}) = \{\tilde{s}_3, \tilde{s}_4, \tilde{s}_6\}$. We reorder the elements of $DR(\overline{K})$, by moving vanishing states to the first positions: $\tilde{s}_3, \tilde{s}_4, \tilde{s}_6, \tilde{s}_1, \tilde{s}_2, \tilde{s}_5, \tilde{s}_7, \tilde{s}_8, \tilde{s}_9$. The reordered TPM for $DTMC(\overline{K})$ is*

$$\widetilde{\mathbf{P}}_r = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 0 & 1-\rho^3 & \rho^3 & 0 & 0 & 0 & 0 \\ \rho(1-\rho) & \rho(1-\rho) & \rho^2 & 0 & (1-\rho)^2 & 0 & 0 & 0 & 0 \\ 0 & \rho^3 & 0 & 0 & \rho^2(1-\rho) & (1-\rho)(1-\rho^2) & 0 & \rho(1-\rho^2) & 0 \\ \rho^3 & 0 & 0 & 0 & \rho^2(1-\rho) & 0 & (1-\rho)(1-\rho^2) & 0 & \rho(1-\rho^2) \\ 0 & \rho^2 & 0 & 0 & 0 & 0 & 0 & 1-\rho^2 & 0 \\ \rho^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1-\rho^2 \end{pmatrix}.$$

The result of the decomposing $\widetilde{\mathbf{P}}_r$ are the matrices

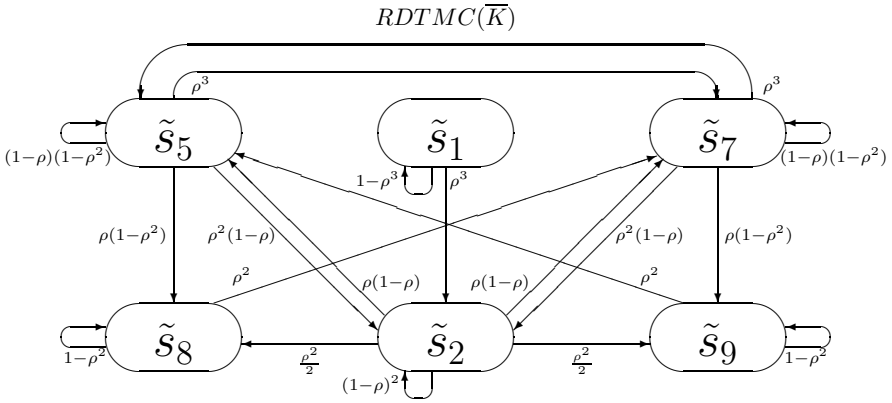

FIG. 7. The reduced DTMC of the generalized shared memory system

$$\widetilde{\mathbf{C}} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix},\ \widetilde{\mathbf{D}} = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \end{pmatrix},\ \widetilde{\mathbf{E}} = \begin{pmatrix} 0 & 0 & 0 \\ \rho(1-\rho) & \rho(1-\rho) & \rho^2 \\ 0 & \rho^3 & 0 \\ \rho^3 & 0 & 0 \\ 0 & \rho^2 & 0 \\ \rho^2 & 0 & 0 \end{pmatrix},$$

$$\widetilde{\mathbf{F}} = \begin{pmatrix} 1-\rho^3 & \rho^3 & 0 & 0 & 0 & 0 \\ 0 & (1-\rho)^2 & 0 & 0 & 0 & 0 \\ 0 & \rho^2(1-\rho) & (1-\rho)(1-\rho^2) & 0 & \rho(1-\rho^2) & 0 \\ 0 & \rho^2(1-\rho) & 0 & (1-\rho)(1-\rho^2) & 0 & \rho(1-\rho^2) \\ 0 & 0 & 0 & 0 & 1-\rho^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1-\rho^2 \end{pmatrix}.$$

Since $\widetilde{\mathbf{C}}^1 = \mathbf{0}$, *we have* $\forall k > 0,\ \widetilde{\mathbf{C}}^k = \mathbf{0}$, *hence,* $l = 0$ *and there are no loops among vanishing states. Then*

$$\widetilde{\mathbf{G}} = \sum_{k=0}^{l} \widetilde{\mathbf{C}}^k = \widetilde{\mathbf{C}}^0 = \mathbf{I}.$$

Further, the TPM for $RDTMC(\overline{K})$ *is*

$$\widetilde{\mathbf{P}}^{\diamond} = \widetilde{\mathbf{F}} + \widetilde{\mathbf{E}}\widetilde{\mathbf{G}}\widetilde{\mathbf{D}} = \widetilde{\mathbf{F}} + \widetilde{\mathbf{E}}\mathbf{I}\widetilde{\mathbf{D}} = \widetilde{\mathbf{F}} + \widetilde{\mathbf{E}}\widetilde{\mathbf{D}} =$$

$$\begin{pmatrix} 1-\rho^3 & \rho^3 & 0 & 0 & 0 & 0 \\ 0 & (1-\rho)^2 & \rho(1-\rho) & \rho(1-\rho) & \frac{\rho^2}{2} & \frac{\rho^2}{2} \\ 0 & \rho^2(1-\rho) & (1-\rho)(1-\rho^2) & \rho^3 & \rho(1-\rho^2) & 0 \\ 0 & \rho^2(1-\rho) & \rho^3 & (1-\rho)(1-\rho^2) & 0 & \rho(1-\rho^2) \\ 0 & 0 & 0 & \rho^2 & 1-\rho^2 & 0 \\ 0 & 0 & \rho^2 & 0 & 0 & 1-\rho^2 \end{pmatrix}.$$

In Fig. 7, the reduced DTMC $RDTMC(\overline{K})$ *is presented.*
The steady-state PMF for $RDTMC(\overline{K})$ *is*

TABLE 6. Transient and steady-state probabilities for the RDTMC of the standard shared memory system

k	0	10	20	30	40	50	∞
$\psi_1^\diamond[k]$	1	0.2631	0.0692	0.0182	0.0048	0.0013	0
$\psi_2^\diamond[k]$	0	0.0931	0.0679	0.0612	0.0594	0.0590	0.0588
$\psi_3^\diamond[k]$	0	0.1307	0.1644	0.1733	0.1756	0.1763	0.1765
$\psi_5^\diamond[k]$	0	0.1912	0.2670	0.2870	0.2922	0.2936	0.2941

$$\tilde{\psi}^\diamond = \frac{1}{2(2+\rho-\rho^2-\rho^3)}(0, 2\rho^2(1-\rho), \rho(2-\rho), \rho(2-\rho), 2-\rho-\rho^2, 2-\rho-\rho^2).$$

Note that $\tilde{\psi}^\diamond = (\tilde{\psi}^\diamond(\tilde{s}_1), \tilde{\psi}^\diamond(\tilde{s}_2), \tilde{\psi}^\diamond(\tilde{s}_5), \tilde{\psi}^\diamond(\tilde{s}_7), \tilde{\psi}^\diamond(\tilde{s}_8), \tilde{\psi}^\diamond(\tilde{s}_9))$. *By Proposition 3, we have*

$$\tilde{\varphi}(\tilde{s}_1) = 0, \quad \tilde{\varphi}(\tilde{s}_2) = \tfrac{\rho^2(1-\rho)}{2+\rho-\rho^2-\rho^3}, \quad \tilde{\varphi}(\tilde{s}_5) = \tfrac{\rho(2-\rho)}{2(2+\rho-\rho^2-\rho^3)}, \quad \tilde{\varphi}(\tilde{s}_7) = \tfrac{\rho(2-\rho)}{2(2+\rho-\rho^2-\rho^3)},$$
$$\tilde{\varphi}(\tilde{s}_8) = \tfrac{2-\rho-\rho^2}{2(2+\rho-\rho^2-\rho^3)}, \qquad \tilde{\varphi}(\tilde{s}_9) = \tfrac{2-\rho-\rho^2}{2(2+\rho-\rho^2-\rho^3)}.$$

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

$$\tilde{\varphi} = \frac{1}{2(2+\rho-\rho^2-\rho^3)}(0, 2\rho^2(1-\rho), 0, 0, \rho(2-\rho), 0, \rho(2-\rho), 2-\rho-\rho^2, 2-\rho-\rho^2).$$

This coincides with the result obtained in Example 2 with the use of $\tilde{\psi}^*$ *and* $\widetilde{SJ}$.

Example 7. *Let E be from Example 3. In Table 6, the transient and the steady-state probabilities* $\psi_i^\diamond[k]$ $(i \in \{1,2,3,5\})$ *for the RDTMC of the standard shared memory system at the time moments* $k \in \{0, 10, 20, 30, 40, 50\}$ *and* $k = \infty$ *are presented, and in Fig. 8, the alteration diagram (evolution in time) for the transient probabilities is depicted. It is sufficient to consider the probabilities for the states* s_1, s_2, s_5, s_8 *only, since the corresponding values coincide for* s_5, s_7, *as well as for* s_8, s_9.

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

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

Note that our reduction of the underlying SMC by eliminating its vanishing states, resulting in the reduced DTMC, resembles the reduction from [17] by removing instantaneous states of stochastically discontinuous Markov reward chains. The latter are "limits" of continuous time Markov chains with state rewards and fast transitions when the rates (speeds) of these transitions tend to infinity, making them immediate. By analogy with this work, we would consider DTMCs extended with instantaneous states instead of SMCs with geometrically distributed or zero sojourn time in the states. However, within dtsiPBC, we have decided to take SMCs as the underlying stochastic process to be able in the perspective to consider not only geometrically distributed and zero residence time in the states, but arbitrary fixed time delays as well.

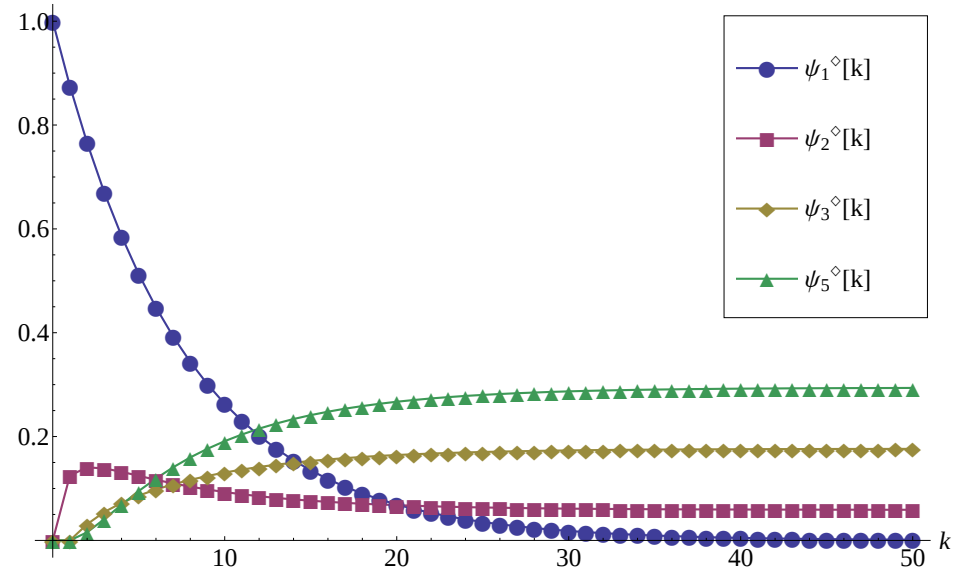

FIG. 8. Transient probabilities alteration diagram for the RDTMC of the standard shared memory system

5. Conclusion

In this paper, we have presented a discrete time stochastic extension dtsiPBC of a finite part of PBC, enriched with iteration and immediate multiactions. The calculus has a concurrent step operational semantics, based on labeled probabilistic transition systems. A novel method of performance evaluation in the framework of the calculus has been proposed that explores a reduction of the underlying stochastic process of the algebraic expressions. The method takes into account specific features of the SMCs corresponding to the expressions, such as zero sojourn time in the vanishing states. The reduced stochastic process is the RDTMC, which is the reduction of both the underlying SMC and DTMC, accomplished with eliminating vanishing states. The proposed analysis technique makes it possible to calculate performance indices in a simpler and more optimal way. This is is very important in modeling complex and large-scale concurrent systems, the state spaces of which grow drastically when the number of their components increases. As a running example, a case study of the generalized shared memory system has been presented with the goal to demonstrate the specification, modeling and performance analysis in dtsiPBC. With this real-world example we have demonstrated that the mentioned new RDTMC-based approach makes easier the performance analysis due to abstracting from the activities with zero or negligibly small durations, which do not affect the stationary behaviour and corresponding performance measures.

The advantage of our framework is twofold. First, one can specify in it concurrent composition and synchronization of (multi)actions, whereas this is not possible in classical Markov chains. Second, algebraic formulas represent processes in a more compact way than Petri nets and allow one to apply syntactic transformations and comparisons. Process algebras are compositional by definition and their operations naturally correspond to operators of programming languages. Hence, it is much

easier to construct a complex model in the algebraic setting than in PNs. The complexity of PNs generated for practical models in the literature demonstrates that it is not straightforward to construct such PNs directly from the system specifications. dtsiPBC is well suited for the discrete time applications, such as business processes, neural and transportation networks, computer and communication systems, whose discrete states change with a global time tick, as well as for those, in which the distributed architecture or the concurrency level should be preserved while modeling and analysis (remember that, in step semantics, we have additional transitions due to concurrent executions). Strong points of dtsiPBC are the flexible multiaction labels, immediate multiactions, the powerful operations, as well as the step operational and Petri net denotational semantics allowing for concurrent execution of activities (transitions).

One of the directions of our future work is a construction of a bisimulation equivalence that preserves functionality and performance. This equivalence may be applied to construct quotients of the transition systems of the algebraic expressions and thereby decrease the number of the process states at the functional level (related to transition systems) before applying the vanishing states reduction at the performance level (related to SMCs). In Example 1, the following states of $TS(\overline{K})$ may be related by such a bisimulation: $\tilde{s}_3, \tilde{s}_4$, as well as $\tilde{s}_5, \tilde{s}_7$, and $\tilde{s}_8, \tilde{s}_9$. We also plan to extend the calculus with deterministically timed multiactions, having a fixed time delay (including the zero one which is the case of immediate multiactions) to enhance expressiveness of the calculus and to extend application area of the associated analysis techniques. The resulting SPA will be a concurrent discrete time analogue of SM-PEPA [7], whose underlying stochastic model is an SMC. Moreover, recursion operation could be added to dtsiPBC to extend further specification power of the algebra towards a wider class of infinite processes with a discrete stochastic time.

References

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

[2] Bernardo M., Bravetti M. *Reward based congruences: can we aggregate more? Lecture Notes in Computer Science* **2165**, p. 136–151, 2001, `http://www.cs.unibo.it/~bravetti/papers/papm01b.ps`.

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

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

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

[6] Best E., Koutny M. *A refined view of the box algebra. Lecture Notes in Computer Science* **935**, p. 1–20, 1995, `http://parsys.informatik.uni-oldenburg.de/~best/publications/pn95.ps.gz`.

[7] Bradley J.T. *Semi-Markov PEPA: modelling with generally distributed actions. International Journal of Simulation* **6(3–4)**, p. 43–51, February 2005, `http://pubs.doc.ic.ac.uk/semi-markov-pepa/semi-markov-pepa.pdf`.

[8] Ciardo G., Muppala J.K., Trivedi K.S. *On the solution of GSPN reward models. Performance Evaluation* **12(4)**, p. 237–253, July 1991, `http://people.ee.duke.edu/~kst/spn_papers/gspnrew.ps`.

[9] Haverkort B.R. *Markovian models for performance and dependability evaluation. Lecture Notes in Computer Science* **2090**, p. 38–83, 2001, http://www-i2.informatik.rwth-aachen.de/Teaching/Seminar/VOSS2005/have01.pdf.

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

[11] Hillston J. *A compositional approach to performance modelling.* 158 p., Cambridge University Press, UK, 1996, http://www.dcs.ed.ac.uk/pepa/book.pdf.

[12] Katoen J.-P. *Quantinative and qualitative extensions of event structures. Ph. D. thesis, CTIT Ph. D.-thesis series* **96-09**, 303 p., Centre for Telematics and Information Technology, University of Twente, Enschede, The Netherlands, 1996.

[13] Kulkarni V.G. *Modeling and analysis of stochastic systems. Texts in Statistical Science*, 563 p., Chapman and Hall / CRC Press, 2009.

[14] Marià H., Valero V., Cazorla D., Cuartero F. *Introducing the iteration in sPBC. Lecture Notes in Computer Science* **3235**, p. 292–308, 2004, http://www.info-ab.uclm.es/retics/publications/2004/forte04.pdf.

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

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

[17] Markovski J., Sokolova A., Trčka N., de Vink E.P. *Compositionality for Markov reward chains with fast and silent transitions. Performance Evaluation* **66**, p. 435–452, 2009.

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

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

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

[21] Molloy M.K. *Discrete time stochastic Petri nets. IEEE Transactions on Software Engineering* **11(4)**, p. 417–423, 1985.

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

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

[24] Tarasyuk I.V. *Discrete time stochastic Petri box calculus. Berichte aus dem Department für Informatik* **3/05**, 25 p., Carl von Ossietzky Universität Oldenburg, Germany, November 2005 (ISSN 1867-9218), http://itar.iis.nsk.su/files/itar/pages/dtspbcib_cov.pdf.

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

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

[27] Tarasyuk I.V. *Equivalence relations for modular performance evaluation in dtsPBC. Mathematical Structures in Computer Science* **24(1)**, p. 78–154 (e240103), Cambridge University Press, Cambridge, UK, February 2014, http://itar.iis.nsk.su/files/itar/pages/dtsdphms.pdf.

[28] Tarasyuk I.V., Marià H., Valero V. *Discrete time stochastic Petri box calculus with immediate multiactions. Technical Report* **DIAB-10-03-1**, 25 p., Department of Computer Systems, High School of Computer Science Engineering, University of Castilla - La Mancha,

Albacete, Spain, March 2010, http://itar.iis.nsk.su/files/itar/pages/dtsipbc.pdf, http://www.dsi.uclm.es/descargas/technicalreports/DIAB-10-03-1/dtsipbc.pdf.

[29] Tarasyuk I.V., Macià H., Valero V. *Discrete time stochastic Petri box calculus with immediate multiactions dtsiPBC. Proc.* 6^{th} *International Workshop on Practical Applications of Stochastic Modelling - 12 (PASM'12) and* 11^{th} *International Workshop on Parallel and Distributed Methods in Verification - 12 (PDMC'12), Electronic Notes in Theoretical Computer Science* **296**, p. 229–252, Elsevier, August 2013, http://itar.iis.nsk.su/files/itar/pages/dtsipbcentcs.pdf.

[30] Tarasyuk I.V., Macià H., Valero V. *Performance analysis of concurrent systems in algebra dtsiPBC. Programming and Computer Software* **40(5)**, p. 229–249, Pleiades Publishing, Ltd., September 2014.

I.V. Tarasyuk
A.P. Ershov Institute of Informatics Systems,
Siberian Branch of the Russian Academy of Sciences,
6, Acad. Lavrentiev pr.,
630090 Novosibirsk, Russian Federation
E-mail address: itar@iis.nsk.su

H. Macià, V. Valero
High School of Informatics Engineering,
University of Castilla - La Mancha,
Avda. de España s/n,
02071 Albacete, Spain
E-mail address: Hermenegilda.Macia@uclm.es, Valentin.Valero@uclm.es