RESEARCH ARTICLE

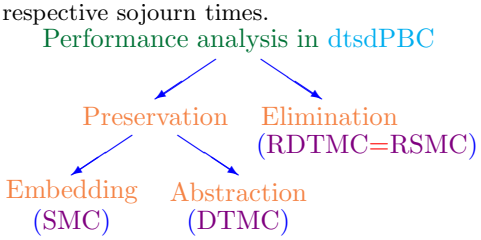
Embedding and elimination for performance analysis in stochastic process algebra dtsdPBC

I. V. Tarasyuk[a]

[a]Laboratory for Theory of Concurrent Processes, A.P. Ershov Institute of Informatics Systems, Siberian Branch of the Russian Academy of Sciences, Acad. Lavrentiev pr. 6, 630090 Novosibirsk, Russian Federation

ARTICLE HISTORY
Compiled August 28, 2025

ABSTRACT
dtsdPBC extends the well-known algebra of parallel processes, Petri box calculus (PBC), by incorporating discrete time stochastic and deterministic delays. To analyze performance in this extended calculus, the underlying semi-Markov chains, and the related (complete) and reduced discrete time Markov chains of the process expressions are built. The semi-Markov chains are extracted using the embedding method, which constructs the embedded discrete time Markov chains and calculates the sojourn time distributions in the states. The reductions of the discrete time Markov chains are obtained through the elimination method, which removes the vanishing states (those with zero sojourn times) and recalculates the transition probabilities among the tangible states (those with positive sojourn times). We prove that the reduced semi-Markov chain coincides with the reduced discrete time Markov chain, by demonstrating that an additional embedding into the reduced semi-Markov chain is needed for the reduced embedded discrete time Markov chain to match the embedded reduced discrete time Markov chain, and by comparing the respective sojourn times.



KEYWORDS
Petri box calculus; stochastic and deterministic delays; Markov chain; performance analysis; embedding; elimination

1. Introduction

Process calculi, like CSP [1], ACP [2] and CCS [3] are well-known formal models for specification of computing systems and analysis of their behaviour. In such process algebras (PAs), formulas describe processes, and verification of the functionality properties of their behaviour is accomplished at a syntactic level via equivalences, axioms and inference rules. In order to represent stochastic timing and analyze the

CONTACT I. V. Tarasyuk. Email: itar@iis.nsk.su

performance properties, stochastic extensions of PAs were proposed, like MTIPP [4], PEPA [5,6] and EMPA [7]. Such stochastic process algebras (SPAs) specify actions which can occur (qualitative features) and associate with the actions the distribution parameters of their random delays (quantitative characteristics).

1.1. Petri box calculus (PBC)

Petri box calculus (PBC) [8–11] is a flexible and expressive process algebra developed as a tool for specification of the Petri nets (PNs) structure and their interrelations. Its goal was also to propose a compositional semantics for high level constructs of concurrent programming languages in terms of elementary PNs. Formulas of PBC are combined from multisets of elementary actions and their conjugates, called multiactions (*basic formulas*). The empty multiset of actions is interpreted as the silent multiaction specifying an invisible activity. The operational semantics of PBC is of step type, since its SOS rules have transitions with (multi)sets of activities, corresponding to simultaneous executions of activities (steps). A denotational semantics of PBC was proposed via a subclass of PNs with an interface and considered up to isomorphism, called Petri boxes. The extensions of PBC with a deterministic, a nondeterministic or a stochastic model of time exist.

1.2. Time extensions of PBC

A time extension of PBC with a nondeterministic time model, called time Petri box calculus (tPBC), was proposed in [12]. In tPBC, timing information is added by associating time intervals with instantaneous *actions*. tPBC has a step time operational semantics in terms of labeled transition systems. Its denotational semantics was defined in terms of a subclass of labeled time Petri nets (LtPNs), based on tPNs [13] and called time Petri boxes (ct-boxes).

Another time enrichment of PBC, called Timed Petri box calculus (TPBC), was defined in [14,15]. It accommodates a deterministic model of time. In contrast to tPBC, multiactions of TPBC are not instantaneous, but have time durations. TPBC has a step timed operational semantics in terms of labeled transition systems. The denotational semantics of TPBC was defined in terms of a subclass of labeled Timed Petri nets (LTPNs), based on TPNs [16] and called Timed Petri boxes (T-boxes).

The third time extension of PBC, called arc time Petri box calculus (atPBC), was constructed in [17,18]. It implements a nondeterministic time. In atPBC, multiactions are associated with time delay intervals. atPBC has a step time operational semantics in terms of labeled transition systems. Its denotational semantics was defined on a subclass of labeled arc time Petri nets (atPNs), based of those from [19,20], where time restrictions are associated with the arcs, called arc time Petri boxes (at-boxes). tPBC, TPBC and atPBC, all adapt the discrete time approach, but TPBC has no immediate (multi)actions (those with zero delays).

1.3. Stochastic extensions of PBC

A stochastic extension of PBC, called stochastic Petri box calculus (sPBC), was proposed in [21–23]. In sPBC, multiactions have stochastic delays that follow (negative) exponential distribution. Each multiaction is equipped with a rate that is a parameter of the corresponding exponential distribution. The (instantaneous) execution of a

stochastic multiaction is possible only after the corresponding stochastic time delay. The calculus has an interleaving operational semantics defined via transition systems labeled with multiactions and their rates. Its denotational semantics was defined on a subclass of labeled continuous time stochastic PNs, based on CTSPNs [24,25] and called stochastic Petri boxes (s-boxes). In sPBC, performance of the processes is evaluated by analyzing their underlying continuous time Markov chains (CTMCs).

sPBC was enriched with immediate multiactions having zero delay in [26,27]. We call such an extension generalized sPBC (gsPBC). An interleaving operational semantics of gsPBC was constructed via transition systems labeled with stochastic or immediate multiactions together with their rates or probabilities. A denotational semantics of gsPBC was defined via a subclass of labeled generalized stochastic PNs, based on GSPNs [24,25,28] and called generalized stochastic Petri boxes (gs-boxes). The performance analysis in gsPBC is based on semi-Markov chains (SMCs).

In [29–32], we presented a discrete time stochastic extension dtsPBC of the algebra PBC. In dtsPBC, the residence time in the process states is geometrically distributed. A step operational semantics of dtsPBC was constructed via labeled probabilistic transition systems. Its denotational semantics was defined in terms of a subclass of labeled discrete time stochastic PNs (LDTSPNs), based on DTSPNs [33,34] and called discrete time stochastic Petri boxes (dts-boxes). The performance evaluation in dtsPBC is accomplished via the underlying discrete time Markov chains (DTMCs).

In [35–39], a calculus dtsiPBC was proposed as an extension with immediate multiactions of dtsPBC. Immediate multiactions increase the specification capability: they can model logical conditions, probabilistic branching, instantaneous probabilistic choices and activities whose durations are negligible in comparison with those of others. They are also used to specify urgent activities and the ones that are not relevant for performance evaluation. The step operational semantics of dtsiPBC was constructed with the use of labeled probabilistic transition systems. Its denotational semantics was defined in terms of a subclass of labeled discrete time stochastic and immediate PNs (LDTSIPNs), called dtsi-boxes. The corresponding stochastic process, the underlying SMC, was constructed and investigated, with the purpose of performance evaluation. In addition, the alternative solution methods were developed, based on the underlying ordinary and reduced DTMCs.

In [40–43], we defined dtsdPBC, an extension of dtsiPBC with deterministic multiactions. In dtsdPBC, besides the probabilities from the real-valued interval $(0; 1)$, applied to calculate discrete time delays of stochastic multiactions, also non-negative integers are used to specify fixed delays of deterministic multiactions (including zero delay, which is the case of immediate multiactions). To resolve conflicts among deterministic multiactions, they are additionally equipped with positive real-valued weights. As argued in [44–46], a combination of deterministic and stochastic delays fits well to model technical systems with constant (fixed) durations of the regular non-random activities and probabilistically distributed (stochastic) durations of the randomly occurring activities. dtsdPBC has a step operational semantics, defined via labeled probabilistic transition systems. The denotational semantics of dtsdPBC was defined in terms of a subclass of labeled discrete time stochastic and deterministic Petri nets (LDTSDPNs), called dtsd-boxes.

1.4. Our contributions

As a basis model, we take *discrete time stochastic and deterministic Petri box calculus* (dtsdPBC) [40–43], featuring a step operational semantics. Here we do not consider the Petri net denotational semantics of the calculus, since it was extensively described in [41]. In that paper, a consistency of the operational and denotational semantics with respect to step stochastic bisimulation equivalence was proved. Hence, all the results established for the former can be readily transferred to the latter up to that equivalence.

In [42], with the *embedding* method, based on the embedded DTMC (EDTMC) specifying the state change probabilities, we constructed and solved the underlying stochastic process, which is a semi-Markov chain (SMC). The obtained stationary probability masses and average sojourn times in the states of the SMC were used to calculate the performance measures (indices) of interest. The alternative solution techniques were also developed, called *abstraction* and *elimination*, that are based respectively on the corresponding discrete time Markov chain (DTMC) and its reduction (RDTMC) by eliminating vanishing states (those with zero sojourn times).

In this paper, we formally prove that the reduced SMC (RSMC) coincides with the RDTMC. Interestingly, the proof of this very intuitive fact appears to be rather involved. First, we demonstrate that an additional embedding (into RSMC) of the reduced EDTMC is needed to coincide with the embedded RDTMC. Second, we calculate the respective sojourn time distributions in the tangible states (those with positive sojourn times) and check their coincidence. Hence, constructing the RDTMC is more optimal than building the RSMC, since the former technique involves only one computationally costly embedding. Thus, the main contributions of the paper are:

- Coincidence of the embedded reduced EDTMC with the embedded RDTMC.
- Identity of the respective sojourn times, hence, the RSMC and the RDTMC.

1.5. Structure of the paper

In Section 2, the syntax of algebra dtsdPBC is proposed. In Section 3, the operational semantics of the calculus in terms of labeled probabilistic transition systems is presented. In Section 4, the underlying stochastic process (SMC) is defined, the alternative solution method via the corresponding RDTMC is outlined, and coincidence of the reduced SMC (RSMC) with the RDTMC is established. Section 5 discusses the results obtained and outlines future research. The concluding Section 6 summarizes the presented and planned work.

2. Syntax

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

2.1. Activities and operations

Multiset is a set with allowed identical elements.

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

The *set of all finite multisets* over a set X is $\mathbb{N}^X_{fin}$. Let $M, M' \in \mathbb{N}^X_{fin}$. The *cardinality*

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

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

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

A *deterministic multiaction* is a pair $(\alpha, \natural_l^\theta)$, where $\alpha \in \mathcal{L},\ \theta \in \mathbb{N}$ is the non-negative integer-valued *(fixed) delay* and $l \in \mathbb{R}_{>0} = (0; \infty)$ is the positive real-valued *weight* of the multiaction α. This weight is interpreted as a measure of importance (urgency, interest) or a bonus reward associated with execution of the deterministic multiaction at the moment when the corresponding delay has expired. Such weights are used to calculate the probabilities to execute sets of deterministic multiactions after their delays. An *immediate multiaction* is a deterministic multiaction with the delay 0 while a *waiting multiaction* is a deterministic multiaction with a positive delay. In case of no conflicts among waiting multiactions, whose remaining times to execute (RTEs) are equal to one time unit, they are executed with probability 1 at the next moment. Deterministic multiactions have a priority over stochastic ones while immediate multiactions have a priority over waiting ones. Different types of multiactions cannot participate together in some step (parallel execution). Let $\mathcal{DL}$ be the set of *all deterministic multiactions*, $\mathcal{IL}$ be the set of *all immediate multiactions* and $\mathcal{WL}$ be the set of *all waiting multiactions*. We have $\mathcal{DL} = \mathcal{IL} \cup \mathcal{WL}$.

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

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

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

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

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

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

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

2.2. Process expressions

Static expressions specify the structure of processes, i.e. how activities are combined by operations to construct the composite process-algebraic formulas. As for the PN intuition, static expressions correspond to unmarked LDTSDPNs [40,41]. A marking is the allocation of tokens in the places of a PN. Markings are used to describe dynamic behaviour of PNs in terms of transition firings.

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

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

Definition 2.2. Let $(\alpha, \kappa) \in \mathcal{SDL}$, $(\alpha, \natural_l^\theta) \in \mathcal{WL}$, $\delta \in \{1, \ldots, \theta\}$ and $a \in Act$. A *static expression* of dtsdPBC is

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

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

To avoid technical difficulties with the iteration operator, we should not allow concurrency at the highest level of the second argument of iteration. This is not a severe

restriction, since we can always prefix parallel expressions by an activity with the empty multiaction part.

Definition 2.3. Let $(\alpha,\kappa) \in \mathcal{SDL}$, $(\alpha,\natural_l^\theta) \in \mathcal{WL}$, $\delta \in \{1,\ldots,\theta\}$ and $a \in Act$. A *regular static expression* of dtsdPBC is

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

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

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

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

Dynamic expressions specify the states of processes, i.e. particular stages of the process behaviour. As for the Petri net intuition, dynamic expressions correspond to marked LDTSDPNs [40,41]. Dynamic expressions are obtained from static ones, by annotating them with upper or lower bars which specify the active components of the system at the current moment of time. The dynamic expression with upper bar (the overlined one) $\overline{E}$ denotes the *initial*, and that with lower bar (the underlined one) $\underline{E}$ denotes the *final* state of the process specified by a static expression E.

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

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

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

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

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

Let G be a dynamic expression. The *underlying static (line-free) expression* $\lfloor G \rfloor$ of

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

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

G is obtained by removing from it all upper and lower bars.

Definition 2.5. A dynamic expression G is *regular* if $\lfloor G \rfloor$ is regular.

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

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

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

3. Operational semantics

In this section, we define the operational semantics via 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 those syntactic transformations is to obtain the well-structured resulting expressions called operative ones to which no inaction rules can be further applied. The application of an inaction rule to a dynamic expression does not lead to any discrete time tick or any transition firing in the corresponding LDTSDPN [40,41], hence, its current marking stays unchanged.

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

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

In Table 2, we introduce inaction rules for regular dynamic expressions in the arbitrary form, where $E, F \in$ *RegStatExpr*, $G, H, \widetilde{G}, \widetilde{H} \in$ *RegDynExpr* and $a \in$ *Act*. For brevity, two distinct inaction rules with the same premises are sometimes collated, resulting in the inaction rules with double conclusion.

Table 2. Inaction rules for arbitrary regular dynamic expressions.

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

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

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

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

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

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

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

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

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

Thus, the set of *all enabled deterministic multiactions of* s is $EnaDet(s) = EnaImm(s) \cup EnaWait(s)$ and the set of *all enabled activities of* s is $Ena(s) = EnaSto(s) \cup EnaDet(s) = EnaSto(s) \cup EnaImm(s) \cup EnaWait(s)$. Then $Ena(s) = Ena([G]_\approx)$ is an algebraic analogue of the set of all transitions enabled at the initial marking of the LDTSDPN [40,41] corresponding to G. The activities, resulted from synchronization, are not present in the syntax of the dynamic expressions. Their enabledness status can be recovered by observing that of the pair of synchronized activities from the syntax (they both should be enabled for enabling their synchronous product), even if they are affected by restriction after the synchronization.

Definition 3.3. An operative dynamic expression G is *saturated (with the values of timers)*, if each enabled waiting multiaction of $[G]_\approx$, being superscribed with the value of its timer and possibly overlined, is the subexpression of G.

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

Proposition 3.4 ([40,41]). *Any operative dynamic expression can be transformed into the saturated one by a forward or a backward applying the inaction rules.*

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

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

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

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

3.2. Action and empty move rules

The action rules are applied when some activities are executed. With these rules we capture the prioritization among different types of multiactions. We also have the empty move rule, used to capture a delay of one discrete time unit when no immediate or waiting multiactions are executable. In this case, the empty multiset of activities is executed. The action and empty move rules will be used later to determine all multisets of activities which can be executed from the structural equivalence class of every dynamic expression (i.e. from the state of the corresponding process). This information together with that about probabilities or delays and weights of the activities to be executed from the current process state will be used to calculate the probabilities of such executions.

The action rules with stochastic (immediate or waiting, respectively) multiactions describe dynamic expression transformations in the form of $G \stackrel{\Gamma}{\to} \widetilde{G}$ ($G \stackrel{I}{\to} \widetilde{G}$ or $G \stackrel{W}{\to} \widetilde{G}$, respectively) due to execution of non-empty multisets Γ of stochastic (I of immediate or W of waiting, respectively) multiactions. The rules represent possible state changes of the specified processes when some non-empty multisets of stochastic (immediate or waiting, respectively) multiactions are executed. The application of an action rule with stochastic (immediate or waiting, respectively) multiactions to a dynamic expression leads in the corresponding LDTSDPN [40,41] to a discrete time tick at which some stochastic or waiting transitions fire (or to the instantaneous firing of some immediate transitions) and possible change of the current marking. The current marking stays unchanged only if there is a self-loop produced by the iterative execution of a non-empty multiset, which must be one-element, since we allow no concurrency at the

highest level of the second argument of iteration.

The empty move rule (applicable only when no immediate or waiting multiactions can be executed from the current state) describes dynamic expression transformations in the form of $G \xrightarrow{\emptyset} \circlearrowleft G$, called the *empty moves*, due to execution of the empty multiset of activities at a discrete time tick. When no timer values are decremented within G with the empty multiset execution at the next moment (for example, if G contains no stamped waiting multiactions), we have $\circlearrowleft G = G$. In such a case, the empty move from G is in the form of $G \xrightarrow{\emptyset} G$, called the *empty loop*. The application of the empty move rule to a dynamic expression leads to a discrete time tick in the corresponding LDTSDPN [40,41] at which no transitions fire and the current marking is not changed, but the timer values of the waiting transitions enabled at the marking (if any) are decremented by one. This is a new rule that has no prototype among inaction rules of PBC, since it represents a time delay.

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

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

Definition 3.5. The *numbering* of expressions is $\iota ::= \ n \mid (\iota)(\iota)$, where $n \in \mathbb{N}$.

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

The new activities resulting from synchronizations in different orders should be considered up to permutation of their numbering. In this way, we shall recognize different instances of the same activity. If we compare the contents of different numberings, i.e. the sets of natural numbers in them, we shall identify the mentioned instances. The *content* of a numbering $\iota \in Num$ is $Cont(\iota) = \begin{cases} \{\iota\}, & \iota \in \mathbb{N}; \\ Cont(\iota_1) \cup Cont(\iota_2), & \iota = (\iota_1)(\iota_2). \end{cases}$
After the enumeration, the multisets of activities from the expressions become proper sets. We suppose that the identical activities are enumerated when needed to avoid ambiguity. This enumeration is considered to be implicit.

Definition 3.6. Let $G \in OpRegDynExpr$. We define the *set of all non-empty multisets of activities which can be potentially executed from* G, denoted by $Can(G)$. Let $(\alpha, \kappa) \in \mathcal{SDL},\ E, F \in RegStatExpr,\ H \in OpRegDynExpr$ and $a \in Act$.

(1) If $final(G)$ then $Can(G) = \emptyset$.
(2) If $G = \overline{(\alpha, \kappa)^\delta}$ and $\kappa = \natural_l^\theta,\ \theta \in \mathbb{N}_{\geq 2},\ l \in \mathbb{R}_{>0},\ \delta \in \{2, \ldots, \theta\}$, then $Can(G) = \emptyset$.
(3) If $G = \overline{(\alpha, \kappa)}$ and $\kappa \in (0; 1)$ or $\kappa = \natural_l^0,\ l \in \mathbb{R}_{>0}$, then $Can(G) = \{\{(\alpha, \kappa)\}\}$.
(4) If $G = \overline{(\alpha, \kappa)^1}$ and $\kappa = \natural_l^\theta,\ \theta \in \mathbb{N}_{\geq 1},\ l \in \mathbb{R}_{>0}$, then $Can(G) = \{\{(\alpha, \kappa)\}\}$.
(5) If $\Upsilon \in Can(G)$ then $\Upsilon \in Can(G \circ E),\ \Upsilon \in Can(E \circ G)$ $(\circ \in \{;, []\})$,

$\Upsilon \in Can(G\|H),\ \Upsilon \in Can(H\|G),\ f(\Upsilon) \in Can(G[f]),\ \Upsilon \in Can(G\ \mathsf{rs}\ a)$ (when $a, \hat{a} \notin \mathcal{A}(\Upsilon)$), $\Upsilon \in Can(G\ \mathsf{sy}\ a),\ \Upsilon \in Can([G * E * F]),$ $\Upsilon \in Can([E * G * F]),\ \Upsilon \in Can([E * F * G])$.

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

(7) If $\Upsilon \in Can(G\ \mathsf{sy}\ a)$ and $(\alpha, \kappa), (\beta, \lambda) \in \Upsilon$ are different, $a \in \alpha,\ \hat{a} \in \beta$, then
   (a) $\Upsilon - \{(\alpha, \kappa), (\beta, \lambda)\} + \{(\alpha \oplus_a \beta, \kappa \cdot \lambda)\} \in Can(G\ \mathsf{sy}\ a)$ if $\kappa, \lambda \in (0; 1)$;
   (b) $\Upsilon - \{(\alpha, \kappa), (\beta, \lambda)\} + \{(\alpha \oplus_a \beta, \natural^\theta_{l+m})\} \in Can(G\ \mathsf{sy}\ a)$ if $\kappa = \natural^\theta_l,\ \lambda = \natural^\theta_m,$ $\theta \in \mathbb{N},\ l, m \in \mathbb{R}_{>0}$.

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

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

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

Definition 3.7. Let $G \in OpRegDynExpr$. The *set of all non-empty multisets of activities which can be executed from* G is

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

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

Let $G \in RegDynExpr$. We write $stang([G]_\approx)$, if $\forall H \in [G]_\approx \cap OpRegDynExpr$

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

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

We use the next abbreviations in the names of the rules: '**E**' for '**E**mpty move', '**B**' for '**B**asis case', '**S**' for '**S**equence', '**C**' for '**C**hoice', '**P**' for '**P**arallel', '**L**' for 're**L**abeling', '**R**' for '**R**estriction', '**I**' for '**I**teraton' and '**Sy**' for '**Sy**nchronization'. The first rule is the empty move rule **E**. The other rules are the action rules, describing transformations of dynamic expressions, which are built using particular algebraic operations. If we cannot merge the rules with stochastic, immediate ans waiting multiactions in one rule for some operation then we get the coupled action rules. In such cases, the names of the action rules with stochastic multiactions have a suffix '**s**', those with immediate multiactions have a suffix '**i**', and those with waiting multiactions have a suffix '**w**'. The rules in Table 3 are explained in [40,41].

Notice that the timers of all waiting multiactions that lose their enabledness when a state change occurs become inactive (turned off) and their values become irrelevant while the timers of all those preserving their enabledness continue running with their stored values. Hence, we adapt the *enabling memory* policy [25,28,47,48] when the process states are changed and the enabledness of deterministic multiactions is possibly modified (immediate multiactions may be seen as those with the timers displaying a single value 0, so we do not need to store their values). Then the timer values of waiting multiactions are taken as the enabling memory variables.

Like in [12], we are interested in the dynamic expressions, inferred by applying the inaction rules (also in the reverse direction) and action rules from the overlined static expressions, such that no stamped (superscribed with the timer values) waiting multiaction is a subexpression of them. The reason is to ensure that time proceeds uniformly and only enabled waiting multiactions are stamped. We call such dynamic expressions reachable, by analogy with the reachable states of LDTSDPNs [40,41].

Definition 3.8. A dynamic expression G is *reachable*, if there exists a static expression E without timer value superscripts, such that $\overline{E} \approx G$ or $\overline{E} \approx G_0 \stackrel{\Upsilon_1}{\to} H_1 \approx G_1 \stackrel{\Upsilon_2}{\to} \ldots \stackrel{\Upsilon_n}{\to} H_n \approx G$ for some $\Upsilon_1, \ldots, \Upsilon_n \in \mathbb{N}_{fin}^{\mathcal{SDL}}$.

We now consider the enabledness of the stamped waiting multiactions.

Proposition 3.9 ([40,41])**.** *Let G be a reachable dynamic expression. Then only waiting multiactions from $EnaWait([G]_{\approx})$ are stamped in G.*

3.3. Transition systems

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

Let G be a dynamic expression and $s = [G]_{\approx}$. The set of *all multisets of activities*

Table 3. Action and empty move rules.

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

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

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

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

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

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

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

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

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

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

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

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

executable in s is defined as $Exec(s) = \{\Upsilon \mid \exists H \in s\ \exists \widetilde{H}\ H \xrightarrow{\Upsilon} \widetilde{H}\}$. Here $H \xrightarrow{\Upsilon} \widetilde{H}$ is an inference by the rules from Table 3. It can be proved by induction on the structure of expressions that $\Upsilon \in Exec(s) \setminus \{\emptyset\}$ implies $\exists H \in s\ \Upsilon \in Now(H)$. The reverse statement does not hold, since the preconditions in the action rules disable executions of the activities with the lower-priority types from every $H \in s$, see [40,41].

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

Definition 3.10. The *derivation set* of a dynamic expression G, denoted by $DR(G)$, is the minimal set such that

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

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

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

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

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

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

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

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

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

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

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

Definition 3.11. 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 $L_G = \mathbb{N}_{fin}^{\mathcal{SDL}} \times (0;1]$;
- 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 (parallel executions) that occur at discrete time moments with some (one-step) probability and consist of multisets of activities. Every step consisting of stochastic (waiting, respectively) multiactions or the empty step (consisting of the empty multiset of activities) occurs instantly after one discrete time unit delay. Each step consisting of immediate multiactions occurs instantly without any delay. The step can change the current state to a different one. The states are the structural equivalence classes of dynamic expressions obtained by application of action rules starting from the expressions belonging to $[G]_{\approx}$. A transition $(s, (\Upsilon, \mathcal{P}), \tilde{s}) \in \mathcal{T}_G$ will be written as $s \stackrel{\Upsilon}{\to}_{\mathcal{P}} \tilde{s}$. It is interpreted as: the probability to change from state s to $\tilde{s}$ as a result of executing Υ is $\mathcal{P}$.

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

The step probabilities belong to the interval $(0;1]$, being 1 when the only transition from an s-tangible state s is the empty move one $s \stackrel{\emptyset}{\to}_1 \circlearrowleft s$, or if there is a single transition from a w-tangible or a vanishing state. 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 is a coincidence of systems up to renaming of their components.

Definition 3.12. Let for dynamic expressions G, G', $TS(G) = (S_G, L_G, \mathcal{T}_G, s_G)$, $TS(G') = (S_{G'}, L_{G'}, \mathcal{T}_{G'}, s_{G'})$. A mapping $\beta : S_G \to S_{G'}$ is an *isomorphism* between

$TS(G)$ and $TS(G')$, denoted by $\beta : TS(G) \simeq TS(G')$, if

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

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

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

Definition 3.13. Two dynamic expressions G and G' are *equivalent with respect to transition systems*, denoted by $G =_{ts} G'$, if $TS(G) \simeq TS(G')$.

Example 3.14. The expression $\mathsf{Stop} = (\{h\}, \frac{1}{2})$ rs h specifies the non-terminating process that performs only empty loops with probability 1. Let $E = [(\{a\}, \rho) * (((\{b\}, \natural_k^1); ((((\{c\}, \natural_l^0); (\{d\}, \theta))[]((\{e\}, \natural_m^0); (\{f\}, \phi))[] \; (\{g\}, \natural_{l+m}^0))) * \mathsf{Stop}]$, where $\rho, \theta, \phi \in (0; 1)$ and $k, l, m \in \mathbb{R}_{>0}$. $DR(\overline{E})$ consists of the equivalence classes

$$s_1 = [[\overline{(\{a\}, \rho)} * (((\{b\}, \natural_k^1); ((((\{c\}, \natural_l^0); (\{d\}, \theta))[]((\{e\}, \natural_m^0); (\{f\}, \phi))[](\{g\}, \natural_{l+m}^0))) * \mathsf{Stop}]]_{\approx},$$

$$s_2 = [[(\{a\}, \rho) * \overline{((\{b\}, \natural_k^1)^1}; ((((\{c\}, \natural_l^0); (\{d\}, \theta))[]((\{e\}, \natural_m^0); (\{f\}, \phi))[](\{g\}, \natural_{l+m}^0))) * \mathsf{Stop}]]_{\approx},$$

$$s_3 = [[(\{a\}, \rho) * (((\{b\}, \natural_k^1); (\overline{((\{c\}, \natural_l^0)}; (\{d\}, \theta))[]((\{e\}, \natural_m^0); (\{f\}, \phi))[](\{g\}, \natural_{l+m}^0))) * \mathsf{Stop}]]_{\approx} =$$

$$[[(\{a\}, \rho) * (((\{b\}, \natural_k^1); ((((\{c\}, \natural_l^0); (\{d\}, \theta))[](\overline{(\{e\}, \natural_m^0)}; (\{f\}, \phi))[](\{g\}, \natural_{l+m}^0))) * \mathsf{Stop}]]_{\approx} =$$

$$[[(\{a\}, \rho) * (((\{b\}, \natural_k^1); ((((\{c\}, \natural_l^0); (\{d\}, \theta))[]((\{e\}, \natural_m^0); (\{f\}, \phi))[]\overline{(\{g\}, \natural_{l+m}^0)})) * \mathsf{Stop}]]_{\approx},$$

$$s_4 = [[(\{a\}, \rho) * (((\{b\}, \natural_k^1); ((((\{c\}, \natural_l^0); \overline{(\{d\}, \theta)})[]((\{e\}, \natural_m^0); (\{f\}, \phi))[](\{g\}, \natural_{l+m}^0))) * \mathsf{Stop}]]_{\approx},$$

$$s_5 = [[(\{a\}, \rho) * (((\{b\}, \natural_k^1); ((((\{c\}, \natural_l^0); (\{d\}, \theta))[]((\{e\}, \natural_m^0); \overline{(\{f\}, \phi))[](\{g\}, \natural_{l+m}^0)})) * \mathsf{Stop}]]_{\approx}.$$

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

In Figure 1, the transition system $TS(\overline{E})$ is presented. The s-tangible and w-tangible states are depicted in ordinary and double ovals, respectively, and the vanishing ones are depicted in boxes.

This example demonstrates an infinite iteration loop. The loop is preceded with the iteration initiation, modeled by a stochastic multiaction $(\{a\}, \rho)$. The iteration body that corresponds to the loop consists of a waiting multiaction $(\{b\}, \natural_k^1)$, followed (via sequential composition) by the probabilistic choice, modeled via three conflicting immediate multiactions $(\{c\}, \natural_l^0)$, $(\{e\}, \natural_m^0)$, $(\{g\}, \natural_{l+m}^0)$, such as the first and second are followed by different stochastic multiactions $(\{d\}, \theta)$ and $(\{f\}, \phi)$, whereas the third has no follower. The iteration termination Stop demonstrates an empty behaviour, assuring that the iteration does not reach its final state after any number of repeated executions of its body.

Example 3.15. Let us interpret E from Example 3.14 as a specification of the travel system. A tourist visits regularly new cities. After seeing the sights of the current city, he goes to the next city by the nearest train or bus available at the city station. Buses depart less frequently than trains, but the next city is quicker reached by bus than by train. We suppose that the stay duration in every city (being a constant), the departure numbers of trains and buses, as well as their speeds *do not depend* on a particular city, bus or train. The travel route has been planned so that the distances between successive cities *coincide*.

The meaning of actions and activities from the syntax of E is as follows. The action a corresponds to the system activation after planning the travel route that takes a time, geometrically distributed with a parameter ρ, the probability of the corresponding stochastic multiaction $(\{a\}, \rho)$. The action b represents coming to the city

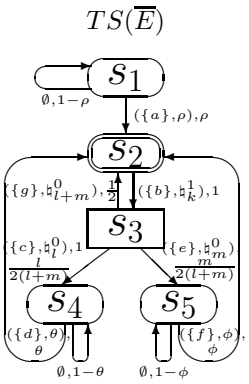


Figure 1. The transition system of $\overline{E}$ for $E = [(\{a\},\rho) * ((\{b\},\natural^1_k);((((\{c\},\natural^0_l);(\{d\},\theta))[](\{e\},\natural^0_m);(\{f\},\phi))[](\{g\},\natural^0_{l+m}))) * \mathsf{Stop}]$.

station after completion of looking round the current city that takes (for every city) a fixed time equal to 1 (hour), the time delay of the corresponding waiting multiaction $(\{b\},\natural^1_k)$ with (resolving no choice) weight k. The actions c and e correspond to the urgent (in zero time) getting on bus and train, respectively, and thus model the choice between these two transport facilities. The weights of the two corresponding immediate multiactions $(\{c\},\natural^0_l)$ and $(\{e\},\natural^0_m)$ suggest that every l departures of buses take the same time as m departures of trains ($l < m$), hence, a bus departs with the probability $\frac{l}{l+m}$ while a train departs with the probability $\frac{m}{l+m}$. The actions d and f correspond to coming in a city by bus and train, respectively, that takes a time, geometrically distributed with the parameters θ and ϕ, respectively ($\theta > \phi$), the probabilities of the corresponding stochastic multiactions $(\{d\},\theta)$ and $(\{f\},\phi)$. The action g specifies instantaneous coming back to the (current) city (i.e. not getting on any transport) from the station. The weight of the corresponding immediate multiaction $(\{g\},\natural^0_{l+m})$ suggests that choosing *no* transport facility has the same probability as choosing *any* transport facility and equals $\frac{l+m}{2(l+m)} = \frac{1}{2} = \frac{l}{2(l+m)} + \frac{m}{2(l+m)}$, where $2(l+m) = l+m+(l+m)$ is the overall weight of all possible outcomes at the city station (bus departure, train departure and coming back to the city).

The meaning of states from $DR(\overline{E})$ is as follows. The s-tangible state s_1 corresponds to staying at home and planning the future travel. The w-tangible state s_2 means residence in a city for exactly one time unit (hour). The vanishing state s_3 with zero residence time represents instantaneous stay at the city station, signifying that the tourist does not wait there as for departure of the transport, as before coming back to the city. The s-tangible states s_4 and s_5 correspond to going by bus and train, respectively.

4. Performance evaluation

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

4.1. Analysis of the underlying SMC (embedding)

For a dynamic expression G, a discrete random variable $\xi(s)$ is associated with every tangible state $s \in DR_T(G)$. The variable captures the residence (sojourn) 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 $\xi(s)$ is 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 $k \geq 1$, called the probability mass function (PMF) of the residence time in s, is $p_{\xi(s)}(k) = \mathsf{P}(\xi(s) = k) = PM(s,s)^{k-1}(1 - PM(s,s))$ $(k \in \mathbb{N}_{\geq 1})$ (the residence time in s is k in this case). Hence, the probability distribution function (PDF) of the residence time in s is $F_{\xi(s)}(k) = \mathsf{P}(\xi(s) < k) = 1 - PM(s,s)^{k-1}$ $(k \in \mathbb{N}_{\geq 1})$ (the probability that the residence time in s is less than k).

The deterministic residence time 1 in a tangible state s can be interpreted as a random variable $\xi(s)$ that is geometrically distributed with the parameter $1 = 1 - PM(s,s)$. In that case, $PM(s,s) = 0$ and $k = 1$ is the only residence time value with a positive probability. Hence, $p_{\xi(s)}(1) = PM(s,s)^{1-1}(1 - PM(s,s)) = 0^0 \cdot 1 = 1$, i.e. the probability that the residence time is 1 equals 1.

Further, the residence time ∞ in an absorbing tangible state s can be interpreted as a random variable $\xi(s)$ that is geometrically distributed with the parameter $0 = 1 - PM(s,s)$. In that case, $PM(s,s) = 1$ and there exists no finite residence time value with a positive probability. Hence, $p_{\xi(s)}(k) = PM(s,s)^{k-1}(1 - PM(s,s)) = 1^{k-1} \cdot 0 = 0$ $(k \in \mathbb{N}_{\geq 1})$, i.e. the probability that the residence time is k equals 0 for every $k \geq 1$. Then we cannot leave s for a different state after any number of time ticks and we stay in s for infinite time.

The mean value formula for the geometrical distribution allows us to calculate the average sojourn time in $s \in DR_T(G)$ as $SJ(s) = \frac{1}{1-PM(s,s)}$. The average sojourn time in each vanishing state $s \in DR_V(G)$ is $SJ(s) = 0$. 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 $SJ(s)$, $s \in DR(G)$.

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) [49–56], 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 [24,25,28], and also in the framework of discrete time deterministic and stochastic PNs (DTDSPNs) in [44–46,57–59], as well as within discrete deterministic and stochastic PNs (DDSPNs) [60,61]. $EDTMC(G)$ 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 residence time in the states) [50,53,54,62].

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. Clearly, 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 $s \neq \tilde{s}$, i.e. $PM(s,s) < 1$. 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.

Let $s \in DR_T(G)$. If there are self-loops from s (i.e. if $s \to s$) then $PM(s,s) > 0$ and $SL(s) = \frac{1}{1-PM(s,s)} = SJ(s)$. Otherwise, if there exist no self-loops from s then $PM(s,s) = 0$ and $SL(s) = 1 = \frac{1}{1-PM(s,s)} = SJ(s)$. Thus, $\forall s \in DR_T(G)\ SL(s) = SJ(s)$, hence, $\forall s \in DR_T(G)$ with $PM(s,s) < 1$ it holds $PM^*(s,\tilde{s}) = SJ(s) PM(s,\tilde{s})$. Note that the self-loops from tangible states are of the empty or non-empty type, the latter produced by iteration, since empty loops are not possible from w-tangible states, but they are possible from s-tangible states, while non-empty loops are possible from both s-tangible and w-tangible states.

Let $s \in DR_V(G)$. We have $\forall s \in DR_V(G)\ SL(s) \neq SJ(s) = 0$ and $\forall s \in DR_V(G)$ with $PM(s,s) < 1$ it holds $PM^*(s,\tilde{s}) = SL(s) PM(s,\tilde{s})$. If there exist self-loops from s then $PM^*(s,\tilde{s}) = \frac{PM(s,\tilde{s})}{1-PM(s,s)}$ when $PM(s,s) < 1$. Otherwise, if there exist no self-loops from s then $PM^*(s,\tilde{s}) = PM(s,\tilde{s})$. Note that the self-loops from vanishing states are always of the non-empty type, produced by iteration, since empty loops are not possible from vanishing states.

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

Then $PM^*(s,\tilde{s})$ defines a probability distribution, since $\forall s \in DR(G)$ such that s is not an absorbing state (i.e. $PM(s,s) < 1$ and there are transitions to different states after possible self-loops from it) we have $\sum_{\{\tilde{s} | s \to \tilde{s},\ s \neq \tilde{s}\}} PM^*(s,\tilde{s}) = \frac{1}{1-PM(s,s)} \sum_{\{\tilde{s} | s \to \tilde{s},\ s \neq \tilde{s}\}} PM(s,\tilde{s}) = \frac{1}{1-PM(s,s)}(1 - PM(s,s)) = 1$.

We decided to consider self-loops followed only by a state-changing step just for convenience. Alternatively, we could take a state-changing step followed by self-loops or a state-changing step preceded and followed by self-loops. In all these three cases our sequence begins or/and ends with the loops which do not change states. At the same time, the overall probabilities of the evolutions can differ, since self-loops have positive probabilities. To avoid inconsistency of definitions and too complex description, we consider sequences ending with a state-changing step. It resembles in some sense a construction of branching bisimulation [63] taking self-loops instead of silent transitions. Further, we shall not abstract from self-loops with probabilities 1 while constructing EDTMCs, in order to maintain a probability distribution among transitions (actually,

a single transition to the same state) from every state with such a self-loop.

Definition 4.1. Let G be a dynamic expression. The *embedded (absorbing) discrete time Markov chain (EDTMC)* of G, denoted by $EDTMC(G)$, has the state space $DR(G)$, the initial state $[G]_{\approx}$ and the transitions $s \twoheadrightarrow_{\mathcal{P}} \tilde{s}$, if $s \to \tilde{s}$ and $s \neq \tilde{s}$, where $\mathcal{P} = PM^*(s, \tilde{s})$; or $s \twoheadrightarrow_1 s$, if $PM(s, s) = 1$.

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 $1 - PM(s, s)$ (in particular, the sojourn time is 1 when $PM(s, s) = 0$, and ∞ when $PM(s, s) = 1$) while the sojourn time in every $s \in DR_V(G)$ is equal to 0.

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

$$\mathcal{P}^*_{ij} = \begin{cases} PM^*(s_i, s_j), & s_i \to s_j,\ i \neq j; \\ 1, & PM(s_i, s_i) = 1,\ i = j; \\ 0, & \text{otherwise.} \end{cases}$$

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.

Note that the vector ψ^* exists and is unique if $EDTMC(G)$ is ergodic. Then $EDTMC(G)$ has a single steady state, and we have $\psi^* = \lim_{k \to \infty} \psi^*[k]$.

The steady-state PMF for the underlying semi-Markov chain $SMC(G)$ is calculated via multiplication of every $\psi^*(s_i)$ $(1 \leq i \leq n)$ by the average sojourn time $SJ(s_i)$ in the state s_i, after which we normalize the resulting values. Remember that for each tangible state $s \in DR_T(G)$ we have $SJ(s) \geq 1$, and for each 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 with probabilities less

than 1 to get $\mathbf{P}^*$ and then ψ^*, followed by weighting by SJ and normalization. We call that technique *embedding*, since the embedded DTMC (EDTMC) is used to specify the SMC state change probabilities. $EDTMC(G)$ has no self-loops with probabilities less than 1, unlike $SMC(G)$, hence, the behaviour of $EDTMC(G)$ may stabilize quicker than that of $SMC(G)$ (if each of them has a single steady state), since $\mathbf{P}^*$ has only zero (excepting the states having self-loops with probabilities 1) elements at the main diagonal.

Example 4.2. Let E be from Example 3.14. In Figure 2, the underlying SMC $SMC(\overline{E})$ is presented. The average sojourn times in the states of the underlying SMC are written next to them in bold font.

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

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

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

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

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

$$\psi^* = \left(0, \frac{2}{5}, \frac{2}{5}, \frac{l}{5(l+m)}, \frac{m}{5(l+m)}\right).$$

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

$$\left(0, \frac{2}{5}, 0, \frac{l}{5\theta(l+m)}, \frac{m}{5\phi(l+m)}\right).$$

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

$$\psi^* SJ^T = \frac{2\theta\phi(l+m) + \phi l + \theta m}{5\theta\phi(l+m)}.$$

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

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

Let G be a dynamic expression and $s, \tilde{s} \in DR(G),\ S, \widetilde{S} \subseteq DR(G)$. The next 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 [64,65].

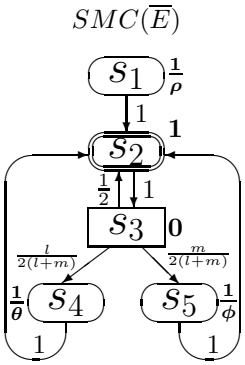


Figure 2. The underlying SMC of $\overline{E}$ for $E = [(\{a\}, \rho) * ((\{b\}, \natural_k^1); ((((\{c\}, \natural_l^0); (\{d\}, \theta))[](\{e\}, \natural_m^0); (\{f\}, \phi))[] (\{g\}, \natural_{l+m}^0))) * \mathsf{Stop}]$.

- The *average recurrence (return) time in the state* s (i.e. the number of discrete time units or steps required for this) is $ReturnTime(s) = \frac{1}{\varphi(s)}$.
- The *fraction of residence time in the state* s is $TimeFract(s) = \varphi(s)$.
- 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 $TimeFract(S) = \sum_{s \in S} \varphi(s)$.
- The *relative fraction of residence time in the set of states* S *with respect to that in* $\widetilde{S}$ is $RltTimeFract(S, \widetilde{S}) = \frac{\sum_{s \in S} \varphi(s)}{\sum_{\tilde{s} \in \widetilde{S}} \varphi(\tilde{s})}$.
- The *exit/entrance frequency (rate of leaving/entering, average number of exits/entrances per unit of time) the state* s is $ExitFreq(s) = \frac{\varphi(s)}{SJ(s)}$.
- The *steady-state probability to perform a step with a multiset of activities* Ξ is $ActsProb(\Xi) = \sum_{s \in DR(G)} \varphi(s) \sum_{\{\Upsilon | \Xi \subseteq \Upsilon\}} PT(\Upsilon, s)$.
- The *probability of the event determined by a reward function* r *on the states* is $Prob(r) = \sum_{s \in DR(G)} \varphi(s) r(s)$, where $\forall s \in DR(G)\ 0 \leq r(s) \leq 1$.

Example 4.3. We now calculate the performance indices for the travel system from Example 3.15. They are based on the steady-state PMF for $SMC(\overline{E})$ $\varphi = \frac{1}{2\theta\phi(l+m)+\phi l+\theta m}(0, 2\theta\phi(l+m), 0, \phi l, \theta m)$ and the average sojourn time vector of $\overline{E}$ $SJ = \left(\frac{1}{\rho}, 1, 0, \frac{1}{\theta}, \frac{1}{\phi}\right)$ from Example 4.2.

- The average time between comings to the successive cities (mean sightseeing and travel time) is $ReturnTime(s_2) = \frac{1}{\varphi(s_2)} = 1 + \frac{\phi l+\theta m}{2\theta\phi(l+m)}$.
- The fraction of time spent in a city (sightseeing time fraction) is $TimeFract(s_2) = \varphi(s_2) = \frac{2\theta\phi(l+m)}{2\theta\phi(l+m)+\phi l+\theta m}$.
- The fraction of time spent in a transport (travel time fraction) is $TimeFract(\{s_4, s_5\}) = \varphi(s_4) + \varphi(s_5) = \frac{\phi l+\theta m}{2\theta\phi(l+m)+\phi l+\theta m}$.
- The relative fraction of time spent in a city with respect to that spent in transport (sightseeing relative to travel time fraction) is $RltTimeFract(\{s_2\}, \{s_4, s_5\}) = \frac{\varphi(s_2)}{\varphi(s_4)+\varphi(s_5)} = \frac{2\theta\phi(l+m)}{\phi l+\theta m}$.
- The rate of leaving/entering a city (departure/arrival rate) is $ExitFreq(s_2) = \frac{\varphi(s_2)}{SJ(s_2)} = \frac{2\theta\phi(l+m)}{\theta\phi(l+m)+\phi l+\theta m}$.

4.2. Analysis of the reduced DTMC (elimination)

Let us now consider the method from [25,28,47,66–69] 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 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 both the transient and the steady-state PMFs over the tangible markings of N. In [68], 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 for GSPNs corresponds in dtsdPBC to the analysis of the underlying SMCs of expressions, called the *embedding* approach.

Definition 4.4. 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 $\mathcal{P}_{ij}$ $(1 \leq i, j \leq n = |DR(G)|)$ of (one-step) transition probability matrix (TPM) $\mathbf{P}$ for $DTMC(G)$ are defined as

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

Example 4.5. Let E be from Example 3.14. In Figure 3, the DTMC $DTMC(\overline{E})$ is presented. The TPM for $DTMC(\overline{E})$ is

$$\mathbf{P} = \begin{pmatrix} 1-\rho & \rho & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & \frac{l}{2(l+m)} & \frac{m}{2(l+m)} \\ 0 & \theta & 0 & 1-\theta & 0 \\ 0 & \phi & 0 & 0 & 1-\phi \end{pmatrix}.$$

The elimination method for GSPNs can be easily transferred to dtsdPBC, 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 [25,28,47,69] in the framework of GSPNs to transform EDTMC into REDTMC. Since the sojourn time in the vanishing states is zero, the state changes 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

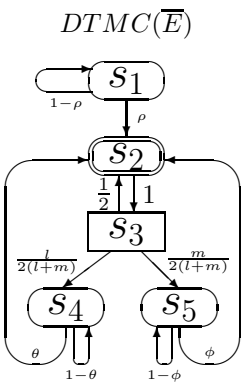


Figure 3. The DTMC of $\overline{E}$ for $E = [(\{a\}, \rho) * ((\{b\}, \natural_k^1); ((((\{c\}, \natural_l^0); (\{d\}, \theta))[](\{e\}, \natural_m^0); (\{f\}, \phi))[]$ $(\{g\}, \natural_{l+m}^0))) * \mathsf{Stop}]$.

have $RSMC(G) = RDTMC(G)$.

Let G be a dynamic expression and $\mathbf{P}$ be the TPM for $DTMC(G)$. We reorder the states from $DR(G)$ such that the first rows and columns of the modified matrix $\mathbf{P}_r$ correspond to the states from $DR_V(G)$ and the last ones correspond to the states from $DR_T(G)$. Let $|DR(G)| = n$ and $|DR_T(G)| = m$. The reordered matrix can be decomposed as follows:

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

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 [28,47]. We have earlier required that $SMC(G)$ has a single closed communication (which is also ergodic) class of states. A communication class of states is their equivalence class w.r.t. communication relation, i.e. a maximal subset of communicating states. A communication class of states is closed if only the states belonging to it are accessible from every its state.

The ergodic class cannot consist of vanishing states only, 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 [50–56]. 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. As a result, 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,

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

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

For $1 \le i, j \le m$ and $1 \le k, l \le n - m$, 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. $\mathcal{P}^{\diamond}_{ij}$ $(1 \le i, j \le m)$ 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 4.6. 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 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, i.e. by going through its vanishing states only. Thus, $RSMC(G)$ should have the transitions $s \hookrightarrow_{\mathcal{P}} \tilde{s}$, where $\mathcal{P} = PM^{\diamond}(s, \tilde{s})$, resulting in $RSMC(G) = RDTMC(G)$.

Note that $RDTMC(G)$ is constructed from $DTMC(G)$ as follows. All vanishing states and all transitions to, from and between them are removed. All transitions between tangible states are preserved. The probabilities of transitions between tangible states may become greater and new transitions between tangible states may be added, both iff there exist moves between these tangible states in any number of steps, going through vanishing states only. Thus, for each sequence of transitions between two tangible states in $DTMC(G)$ there exists a (possibly shorter, since the eventual

passed through vanishing states are removed) sequence between the same states in $RDTMC(G)$ and vice versa. If $DTMC(G)$ is irreducible then all its states (including tangible ones) communicate, hence, all states of $RDTMC(G)$ communicate as well and it is irreducible. Since both $DTMC(G)$ and $RDTMC(G)$ are finite, they are positive recurrent. Thus, in case of irreducibility of $DTMC(G)$, each of them has a single stationary PMF. Then $DTMC(G)$ and/or $RDTMC(G)$ may be periodic, thus having a unique stationary distribution, but no steady-state (limiting) one. For example, it may happen that $DTMC(G)$ is aperiodic while $RDTMC(G)$ is periodic due to removing vanishing states from the former.

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.

Note that the vector $\psi^{\diamond}$ exists and is unique if $RDTMC(G)$ is ergodic. Then $RDTMC(G)$ has a single steady state, and we have $\psi^{\diamond} = \lim_{k\to\infty} \psi^{\diamond}[k]$.

The zero sojourn time in the vanishing states guarantees that the state changes of $RDTMC(G)$ occur in the moments of the global discrete time associated with $SMC(G)$, i.e. every such state change 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 change probabilities of $RDTMC(G)$ are those to move from tangible to tangible states in any number of steps, without traversal of the tangible states. Then $RDTMC(G)$ and $SMC(G)$ have the same transient behaviour over the tangible states, thus, the transient analysis of $SMC(G)$ is possible using $RDTMC(G)$.

The next proposition relates steady-state PMFs for $SMC(G)$ and $RDTMC(G)$ by proving that their steady-state probabilities of the tangible states coincide.

Proposition 4.7 ([42]). *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 $\forall s \in DR(G)$*

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

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 with proba-

bilities less than 1 to get $\mathbf{P}^*$ and then ψ^*, followed by weighting by SJ and normalization. We call that technique *elimination*, since we eliminate the vanishing states. 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 $RDTMC(G)$ may have self-loops with probabilities less than 1, 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). On the other hand, $\mathbf{P}^\diamond$ is generally smaller and denser matrix than $\mathbf{P}^*$, since $\mathbf{P}^\diamond$ may have 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 4.8. Let E be from Example 3.14. Remember that $DR_T(\overline{E}) = DR_{ST}(\overline{E}) \cup DR_{WT}(\overline{E}) = \{s_1, s_2, s_4, s_5\}$ and $DR_V(\overline{E}) = \{s_3\}$. We reorder the states from $DR(\overline{E})$, by moving vanishing states to the first positions: s_3, s_1, s_2, s_4, s_5.

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

$$\mathbf{P}_r = \begin{pmatrix} 0 & 0 & \frac{1}{2} & \frac{l}{2(l+m)} & \frac{m}{2(l+m)} \\ 0 & 1-\rho & \rho & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & \theta & 1-\theta & 0 \\ 0 & 0 & \phi & 0 & 1-\phi \end{pmatrix}.$$

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

$$\mathbf{C}{=}0,\ \mathbf{D}{=}\left(0, \frac{1}{2}, \frac{l}{2(l+m)}, \frac{m}{2(l+m)}\right),\ \mathbf{E}{=}\begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix},\ \mathbf{F}{=}\begin{pmatrix} 1-\rho & \rho & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & \theta & 1-\theta & 0 \\ 0 & \phi & 0 & 1-\phi \end{pmatrix}.$$

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

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

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

$$\mathbf{P}^\diamond = \mathbf{F} + \mathbf{EGD} = \mathbf{F} + \mathbf{EID} = \mathbf{F} + \mathbf{ED} = \begin{pmatrix} 1-\rho & \rho & 0 & 0 \\ 0 & \frac{1}{2} & \frac{l}{2(l+m)} & \frac{m}{2(l+m)} \\ 0 & \theta & 1-\theta & 0 \\ 0 & \phi & 0 & 1-\phi \end{pmatrix}.$$

$RDTMC(\overline{E})$

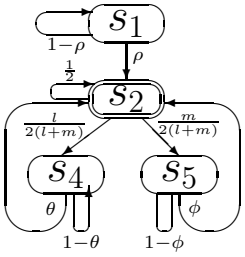


Figure 4. The reduced DTMC of $\overline{E}$ for $E = [(\{a\}, \rho) * ((\{b\}, \natural_k^1); ((((\{c\}, \natural_l^0); (\{d\}, \theta))[]((\{e\}, \natural_m^0); (\{f\}, \phi))[]$ $(\{g\}, \natural_{l+m}^0))) * \mathsf{Stop}]$.

In Figure 4, the reduced DTMC $RDTMC(\overline{E})$ is presented. The steady-state PMF for $RDTMC(\overline{E})$ is

$$\psi^\diamond = \frac{1}{2\theta\phi(l+m) + \phi l + \theta m}(0, 2\theta\phi(l+m), \phi l, \theta m).$$

Note that $\psi^\diamond = (\psi^\diamond(s_1), \psi^\diamond(s_2), \psi^\diamond(s_4), \psi^\diamond(s_5))$. By Proposition 4.7, we have

$$\begin{array}{lll} \varphi(s_1) = 0, & \varphi(s_2) = \frac{2\theta\phi(l+m)}{2\theta\phi(l+m)+\phi l+\theta m}, & \varphi(s_3) = 0, \\ \varphi(s_4) = \frac{\phi l}{2\theta\phi(l+m)+\phi l+\theta m}, & \varphi(s_5) = \frac{\theta m}{2\theta\phi(l+m)+\phi l+\theta m}. & \end{array}$$

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

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

This coincides with the result obtained in Example 4.2 with the use of ψ^* and SJ.

Example 4.9. In Figure 5, the reduced underlying SMC $RSMC(\overline{E})$ is depicted. The average sojourn times in the states of the reduced underlying SMC are written next to them in bold font. In spite of the equality $RSMC(\overline{E}) = RDTMC(\overline{E})$, the graphical representation of $RSMC(\overline{E})$ differs from that of $RDTMC(\overline{E})$, since the former is based on the $REDTMC(\overline{E})$, where each state is decorated with the *positive* average sojourn time of $RSMC(\overline{E})$ in it. $REDTMC(\overline{E})$ is constructed from $EDTMC(\overline{E})$ in the similar way as $RDTMC(\overline{E})$ is obtained from $DTMC(\overline{E})$. By construction, the residence time in each state of $RSMC(\overline{E})$ is geometrically distributed. Hence, the associated parameter of geometrical distribution is uniquely recovered from the average sojourn time in the state.

Let us now formally prove that RSMC coincides with RDTMC. Although this assertion is very intuitive, its proof is rather involved.

The relation between $DTMC$ and $RDTMC$ is obtained using the transition function $PM^\diamond(s, \tilde{s})$, based on $PM(s, \tilde{s})$. The relation between $RDTMC$ and the embedded $RDTMC$ ($ERDTMC$) is obtained using the transition function $(PM^\diamond)^*(s, \tilde{s})$, based on $PM^\diamond(s, \tilde{s})$. The relation between $EDTMC$ and the reduced $EDTMC$ ($REDTMC$) is obtained using the transition function $(PM^*)^\diamond(s, \tilde{s})$, based on $PM^*(s, \tilde{s})$.

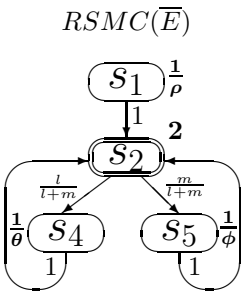


Figure 5. The reduced SMC of $\overline{E}$ for $E = [(\{a\}, \rho) * ((\{b\}, \natural_k^1); ((((\{c\}, \natural_l^0); (\{d\}, \theta))[]((\{e\}, \natural_m^0); (\{f\}, \phi))[]$ $(\{g\}, \natural_{l+m}^0))) * \mathsf{Stop}]$.

Let G be a dynamic expression. We shall prove that the TPM $(\mathbf{P}^\diamond)^*$ for the embedded $RDTMC(G)$ ($ERDTMC(G)$), (forwardly) constructed by reduction (eliminating vanishing states) of $DTMC(G)$, followed by embedding $ERDTMC(G)$ into $RDTMC(G)$, *coincides with* the (finally) embedded TPM $((\mathbf{P}^*)^\diamond)^*$, (reversely) constructed by embedding $EDTMC(G)$ into $SMC(G)$, followed by reduction $REDTMC(G)$ of $EDTMC(G)$, and final embedding $EREDTMC(G)$ into $RSMC(G)$. The final embedding in the reverse construction is needed, since new self-loops may arise after reducing $EDTMC(G)$, i.e. $REDTMC(G)$ may become not an EDTMC, but a DTMC featuring self-loops with probabilities less than 1.

Note that for $s, \tilde{s} \in DR_T(G)$, we have $(PM^\diamond)^*(s, \tilde{s}) = SL^\diamond(s)PM^\diamond(s, \tilde{s})$ in $ERDTMC(G)$. Here $SL^\diamond(s)$ is the self-loops abstraction factor in s in $RDTMC(G)$. This corresponds to a different expression $(PM^*)^\diamond(s, \tilde{s}) = (SL \cdot PM)^\diamond(s, \tilde{s})$ in $REDTMC(G)$. In particular, $SL^\diamond(s) > SL(s)$ when $PM^\diamond(s, s) > PM(s, s)$, being the reason for a new self-loop associated with s in $RDTMC(G)$. As we shall see, in that case $(PM^\diamond)^*(s, \tilde{s}) > (PM^*)^\diamond(s, \tilde{s})$.

The following theorem relates those finally embedded reduced embedded TPM $((\mathbf{P}^*)^\diamond)^*$ (i.e. the TPM for $EREDTMC(G)$) and embedded reduced TPM $(\mathbf{P}^\diamond)^*$ (the TPM for $ERDTMC(G)$).

Theorem 4.10. *Let G be a dynamic expression, $(\mathbf{P}^\diamond)^*$ results from embedding the TPM $\mathbf{P}^\diamond$ for $RDTMC(G)$, and $((\mathbf{P}^*)^\diamond)^*$ results from reduction and final embedding the TPM $\mathbf{P}^*$ for $EDTMC(G)$. Then*

$$((\mathbf{P}^*)^\diamond)^* = (\mathbf{P}^\diamond)^*.$$

Proof. Let $\mathbf{P}_r$ be the reordered (by moving vanishing states to the first positions) TPM for $DTMC(G)$. Like in Section 4, we reorder the states from $DR(G)$ so that the first rows and columns of $\mathbf{P}_r$ 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 $|DR_T(G)| = m$. Then the reordered TPM for $DTMC(G)$ can be decomposed as

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

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 $\mathbf{G} = \sum_{k=0}^{\infty} \mathbf{C}^k$ are the probabilities to move from vanishing to vanishing states in any number of state changes, without traversal of tangible states, in $DTMC(G)$. We define the matrix $\mathbf{H} = \mathbf{EGD}$. For $s, \tilde{s} \in DR_T(G)$, let $PM_F(s, \tilde{s})$ and $PM_H(s, \tilde{s})$ be the probabilities to change from s to $\tilde{s}$ for the submatrix $\mathbf{F}$ and matrix $\mathbf{H}$, respectively.

In a similar way, the reordered TPM for $EDTMC(G)$ can be decomposed as

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

The elements of the submatrices of $\mathbf{P}_r^*$ resemble those of the submatrices of $\mathbf{P}_r$.

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

$$(\mathbf{P}^*)^{\diamond} = \mathbf{F}^* + \mathbf{E}^*\mathbf{G}'\mathbf{D}^*,$$

where the elements of the matrix $\mathbf{G}' = \sum_{k=0}^{\infty} (\mathbf{C}^*)^k$ are the probabilities to move from vanishing to vanishing states in any number of state changes, without traversal of tangible states, in $EDTMC(G)$. We define the matrix $\mathbf{H}' = \mathbf{E}^*\mathbf{G}'\mathbf{D}^*$. For $s, \tilde{s} \in DR_T(G)$, let $PM_{H'}(s, \tilde{s})$ be the probability to change from s to $\tilde{s}$ for the matrix $\mathbf{H}'$.

By the proof of Proposition 3 from [42], we have $\mathbf{P}_r^* = Diag(SL_r)(\mathbf{P}_r - \mathbf{I}) + \mathbf{I}$, where SL_r is the reordered (by moving vanishing states to the first positions) self-loops abstraction vector of G in $DTMC(G)$. Let SL_C and SL_F be the self-loops abstraction subvectors of G for the submatrices $\mathbf{C}$ and $\mathbf{F}$, respectively, i.e. the 'head' of length $n - m$ and the 'tail' of length m, taken from the vector SL_r, with the next elements: $\forall s \in DR_V(G)\ SL_C(s) = SL_r(s)$ and $\forall s \in DR_T(G)\ SL_F(s) = SL_r(s)$. Then we have

$$\mathbf{P}_r^* = \begin{pmatrix} Diag(SL_C) & \mathbf{0} \\ \mathbf{0} & Diag(SL_F) \end{pmatrix} \begin{pmatrix} \mathbf{C} - \mathbf{I} & \mathbf{D} \\ \mathbf{E} & \mathbf{F} - \mathbf{I} \end{pmatrix} + \begin{pmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{pmatrix} = \begin{pmatrix} Diag(SL_C)(\mathbf{C} - \mathbf{I}) + \mathbf{I} & Diag(SL_C)\mathbf{D} \\ Diag(SL_F)\mathbf{E} & Diag(SL_F)(\mathbf{F} - \mathbf{I}) + \mathbf{I} \end{pmatrix}.$$

Hence, $\mathbf{C}^* = Diag(SL_C)(\mathbf{C} - \mathbf{I}) + \mathbf{I}$, $\mathbf{D}^* = Diag(SL_C)\mathbf{D}$, $\mathbf{E}^* = Diag(SL_F)\mathbf{E}$, $\mathbf{F}^* = Diag(SL_F)(\mathbf{F} - \mathbf{I}) + \mathbf{I}$. Then $(\mathbf{P}^*)^{\diamond} = \mathbf{F}^* + \mathbf{E}^*\mathbf{G}'\mathbf{D}^* = Diag(SL_F)(\mathbf{F} - \mathbf{I}) + \mathbf{I} + Diag(SL_F)\mathbf{E}\mathbf{G}'Diag(SL_C)\mathbf{D} = Diag(SL_F)((\mathbf{F} + \mathbf{E}\mathbf{G}'Diag(SL_C)\mathbf{D}) - \mathbf{I}) + \mathbf{I}$. Let us explore the matrix $\mathbf{G}'Diag(SL_C)$. The matrix $\mathbf{G}'$ can have two different forms, depending on whether the loops among vanishing states exist in $EDTMC(G)$, hence, we consider two cases.

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

Then there are no loops among *different* vanishing states in $DTMC(G)$ (but self-loops may exist in vanishing states), since no loop among *different*

states is removed and all self-loops (in the non-absorbing states) are removed in $EDTMC(G)$, with respect to $DTMC(G)$.

Let there are no self-loops in vanishing states in $DTMC(G)$. In such a case, $\forall s \in DT_V(G)\ SL_C(s) = SL(s) = 1$ and $Diag(SL_C) = \mathbf{I}$. We have $\mathbf{C}^* = Diag(SL_C)(\mathbf{C}-\mathbf{I})+\mathbf{I} = \mathbf{I}(\mathbf{C}-\mathbf{I})+\mathbf{I} = \mathbf{C}$ and $\mathbf{G}' = \sum_{k=0}^{l}(\mathbf{C}^*)^k = \sum_{k=0}^{l}\mathbf{C}^k = \mathbf{G}$. Thus, $\mathbf{G}'Diag(SL_C) = \mathbf{GI} = \mathbf{G}$.

Let there are self-loops in vanishing states in $DTMC(G)$. In such a case, $\mathbf{G} = (\mathbf{I}-\mathbf{C})^{-1}$. Note that $\mathbf{C} \neq \mathbf{I} \neq \mathbf{C}^*$, since there exist no absorbing vanishing states in $DTMC(G)$. It is easy to prove by induction on $l \in \mathbb{N}$ that $\mathbf{G}'(\mathbf{I}-\mathbf{C}^*) = \left(\sum_{k=0}^{l}(\mathbf{C}^*)^k\right)(\mathbf{I}-\mathbf{C}^*) = \mathbf{I}-(\mathbf{C}^*)^{l+1}$. Since $(\mathbf{C}^*)^{l+1} = \mathbf{0}$, we get $\mathbf{G}'(\mathbf{I}-\mathbf{C}^*) = \mathbf{I}-\mathbf{0} = \mathbf{I}$. In a similar way, we show that $(\mathbf{I}-\mathbf{C}^*)\mathbf{G}' = \mathbf{I}$. We have $\lim_{k\to\infty}(\mathbf{C}^*)^k = \mathbf{0}$. Hence, $\mathbf{G}' = (\mathbf{I}-\mathbf{C}^*)^{-1} = (\mathbf{I}-Diag(SL_C)(\mathbf{C}-\mathbf{I})-\mathbf{I})^{-1} = (Diag(SL_C)(\mathbf{I}-\mathbf{C}))^{-1} = (\mathbf{I}-\mathbf{C})^{-1}Diag(SL_C)^{-1} = \mathbf{G}Diag(SL_C)^{-1}$. Thus, $\mathbf{G}'Diag(SL_C) = \mathbf{G}Diag(SL_C)^{-1}Diag(SL_C) = \mathbf{G}$.

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

Then there are loops among vanishing states in $DTMC(G)$, since no loop among states is removed and self-loops are possibly added in $DTMC(G)$, with respect to $EDTMC(G)$. Hence, $\lim_{k\to\infty}(\mathbf{C})^k = \mathbf{0}$ and $\mathbf{G} = (\mathbf{I}-\mathbf{C})^{-1}$.

We have $\mathbf{G}' = (\mathbf{I}-\mathbf{C}^*)^{-1} = (\mathbf{I}-Diag(SL_C)(\mathbf{C}-\mathbf{I})-\mathbf{I})^{-1} = (Diag(SL_C)(\mathbf{I}-\mathbf{C}))^{-1} = (\mathbf{I}-\mathbf{C})^{-1}Diag(SL_C)^{-1} = \mathbf{G}Diag(SL_C)^{-1}$. Thus, $\mathbf{G}'Diag(SL_C) = \mathbf{G}Diag(SL_C)^{-1}Diag(SL_C) = \mathbf{G}$.

In the both cases above, we get $\mathbf{G}'Diag(SL_C) = \mathbf{G}$. Hence, $(\mathbf{P}^*)^\diamond = Diag(SL_F)((\mathbf{F}+\mathbf{EG}'Diag(SL_C)\mathbf{D})-\mathbf{I})+\mathbf{I} = Diag(SL_F)((\mathbf{F}+\mathbf{EGD})-\mathbf{I})+\mathbf{I} = Diag(SL_F)(\mathbf{P}^\diamond-\mathbf{I})+\mathbf{I}$.

Let $s, \tilde{s} \in DR_T(G)$. The EDTMC for $RDTMC(G)$ is denoted by $ERDTMC(G)$ and has the probabilities $(PM^\diamond)^*(s,\tilde{s})$ to change from s to $\tilde{s}$. The RDTMC for $EDTMC(G)$ is denoted by $REDTMC(G)$ and has the probabilities $(PM^*)^\diamond(s,\tilde{s})$ to change from s to $\tilde{s}$. The EDTMC for $REDTMC(G)$ is denoted by $EREDTMC(G)$ and has the probabilities $((PM^*)^\diamond)^*(s,\tilde{s})$ to change from s to $\tilde{s}$.

Let SL_H and $SL_{H'}$ be the self-loops abstraction vectors of G for the matrices $\mathbf{H}$ and $\mathbf{H}'$, respectively. We have $(\mathbf{P}^*)^\diamond = \mathbf{F}^*+\mathbf{H}' = \mathbf{F}^*+Diag(SL_F)\mathbf{EGD} = \mathbf{F}^*+Diag(SL_F)\mathbf{H}$. Hence, $\mathbf{H}' = Diag(SL_F)\mathbf{H}$ and $\forall s,\tilde{s} \in DR_T(G)\ PM_{H'}(s,\tilde{s}) = SL_F(s)PM_H(s,\tilde{s})$. Since there are no self-loops in $\mathbf{F}^*$, we conclude that $(SL^*)^\diamond = SL_{H'}$ is the self-loops abstraction vector of G in $REDTMC(G)$.

- Let $PM_F(s,s)+PM_H(s,s) = PM^\diamond(s,s) < 1$ and $PM_F(s,s), PM_H(s,s) > 0$, i.e. s is non-absorbing in $RDTMC(G)$ and there exist self-loops associated with s in $DTMC(G)$ and *extra* self-loops (in addition to those inherited from $DTMC(G)$) in $RDTMC(G)$.

  In $ERDTMC(G)$, we have $(PM^\diamond)^*(s,\tilde{s}) = SL^\diamond(s)PM^\diamond(s,\tilde{s}) = \frac{PM^\diamond(s,\tilde{s})}{1-PM^\diamond(s,s)} = \frac{PM^\diamond(s,\tilde{s})}{1-PM_F(s,s)-PM_H(s,s)} = \frac{\frac{PM^\diamond(s,\tilde{s})}{1-PM_F(s,s)}}{1-\frac{PM_H(s,\tilde{s})}{1-PM_F(s,s)}} = \frac{SL_F(s)PM^\diamond(s,\tilde{s})}{1-SL_F(s)PM_H(s,s)}$.

  Then the self-loops abstraction factor in s in $RDTMC(G)$ is $SL^\diamond(s) = \frac{SL_F(s)}{1-SL_F(s)PM_H(s,s)} = SL_F(s)SL_{H'}(s)$, where $SL_{H'}(s) = \frac{1}{1-SL_F(s)PM_H(s,s)}$ is the self-loops abstraction factor in s in $REDTMC(G)$. Thus, $(PM^\diamond)^*(s,\tilde{s}) = SL_F(s)SL_{H'}(s)PM^\diamond(s,\tilde{s})$.

  In $EREDTMC(G)$, we have $((PM^*)^\diamond)^*(s,\tilde{s}) = (SL^*)^\diamond(s)(PM^*)^\diamond(s,\tilde{s}) =$

$SL_{H'}(s)(PM^*)^\diamond(s,\tilde{s}) = SL_{H'}(s)SL_F(s)PM^\diamond(s,\tilde{s}) = (PM^\diamond)^*(s,\tilde{s})$.

The other three cases (no self-loops associated with s in $DTMC(G)$, no extra self-loops associated with s in $RDTMC(G)$, or no *any* self-loops associated with s in $RDTMC(G)$) are treated analogously, by replacing $PM_F(s,s)$ or/and $PM_H(s,s)$ with zeros.

- Let $PM_F(s,s) + PM_H(s,s) = PM^\diamond(s,s) = 1$ and $PM_F(s,s), PM_H(s,s) > 0$, i.e. s is absorbing in $RDTMC(G)$ and there exist self-loops associated with s in $DTMC(G)$ and *extra* self-loops (in addition to those inherited from $DTMC(G)$) in $RDTMC(G)$.

  In $ERDTMC(G)$, we have $(PM^\diamond)^*(s,s) = 1$ by definition of the EDTMC, since $PM^\diamond(s,s) = 1$.

  In $REDTMC(G)$, the probability of a self-loop associated with s is $(PM^*)^\diamond(s,s) = PM_{H'}(s,s) = SL_F(s)PM_H(s,s) = \frac{PM_H(s,s)}{1-PM_F(s,s)} = \frac{1-PM_F(s,s)}{1-PM_F(s,s)} = 1$.

  In $EREDTMC(G)$, we have $((PM^*)^\diamond)^*(s,s) = 1 = (PM^\diamond)^*(s,s)$ by definition of the EDTMC, since $(PM^*)^\diamond(s,s) = 1$.

  The other three cases (no self-loops associated with s in $DTMC(G)$, no extra self-loops associated with s in $RDTMC(G)$, or no *any* self-loops associated with s in $RDTMC(G)$) are treated analogously, by replacing $PM_F(s,s)$ or/and $PM_H(s,s)$ with zeros.

Thus, $((\mathbf{P}^*)^\diamond)^* = (\mathbf{P}^\diamond)^*$ and $EREDTMC(G) = ERDTMC(G)$. □

Thus, reduction before embedding is more optimal computationally for DTMCs of the process expressions, since only one embedding is needed in that case. This is very important when the DTMCs have many loops from tangible states via (one or more) vanishing states only. Such loops remain after the first embedding, and they become self-loops after the subsequent reduction, which are removed just after the second embedding.

We can now explain the inequality presented above Theorem 4.10, by using its proof. Let $s, \tilde{s} \in DR_T(G)$. Then $SL_F(s)SL_{H'}(s) = SL^\diamond(s) > SL(s) = SL_F(s)$ implies $SL_{H'}(s) > 1$. Thus, $(PM^\diamond)^*(s,\tilde{s}) = SL^\diamond(s)PM^\diamond(s,\tilde{s}) = SL_F(s)SL_{H'}(s)PM^\diamond(s,\tilde{s}) > SL_F(s)PM^\diamond(s,\tilde{s}) = (PM^*)^\diamond(s,\tilde{s})$.

Definition 4.11. Let G be a dynamic expression, $s \in DR_T(G)$ while $SL_F(s)$ is the self-loops abstraction factor in s for the submatrix $\mathbf{F}$ (from the equation $\mathbf{P}^\diamond = \mathbf{F} + \mathbf{EGD}$ calculating the TPM for $RDTMC(G)$) and $SL_{H'}(s)$ is the self-loops abstraction factor in s in $REDTMC(G)$ (for the matrix $\mathbf{H'} = Diag(SL_F)\mathbf{EGD}$, whose elements are the probabilities to move from tangible to tangible states, via any *positive* number of vanishing states, without traversal of tangible states, in $EDTMC(G)$).

The *reduced SMC (RSMC)* of G, denoted by $RSMC(G)$, has the EDTMC $EREDTMC(G)$ and the sojourn time in every $s \in DR_T(G)$ is geometrically distributed with the parameter $\frac{1}{SL_F(s)SL_{H'}(s)}$.

The following proposition demonstrates coincidence of RSMC and RDTMC.

Proposition 4.12. *Let G be a dynamic expression. Then $RSMC(G)$=$RDTMC(G)$.*

Proof. By Theorem 4.10, $EREDTMC(G) = ERDTMC(G)$. The sojourn time in every $s \in DR_T(G)$ is geometrically distributed with the parameter $\frac{1}{SL_F(s)SL_{H'}(s)} = \frac{1}{SL^\diamond(s)} = 1 - PM^\diamond(s,s) = 1 - PM_F(s,s) - PM_H(s,s)$, where $SL_{H'}(s) = \frac{1}{1-SL_F(s)PM_H(s,s)}$. Here $PM_H(s,s)$ is the self-loop probability in s for the matrix $\mathbf{H} = \mathbf{EGD}$ (from the

equation $\mathbf{P}^{\diamond} = \mathbf{F} + \mathbf{EGD}$ calculating the TPM for $RDTMC(G)$). Remember that $SL^{\diamond}(s)$ is the self-loops abstraction factor in s in $RDTMC(G)$. Hence, $RSMC(G) = RDTMC(G)$. □

Example 4.13. Let E be from Example 3.14. The TPMs for $RDTMC(\overline{E})$ and $ERDTMC(\overline{E})$ are

$$\mathbf{P}^{\diamond} = \begin{pmatrix} 1-\rho & \rho & 0 & 0 \\ 0 & \frac{1}{2} & \frac{l}{2(l+m)} & \frac{m}{2(l+m)} \\ 0 & \theta & 1-\theta & 0 \\ 0 & \phi & 0 & 1-\phi \end{pmatrix}, \ (\mathbf{P}^{\diamond})^* = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{l}{l+m} & \frac{m}{l+m} \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}.$$

The TPMs for $REDTMC(\overline{E})$ and $EREDTMC(\overline{E})$ are

$$(\mathbf{P}^*)^{\diamond} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & \frac{1}{2} & \frac{l}{2(l+m)} & \frac{m}{2(l+m)} \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}, \ ((\mathbf{P}^*)^{\diamond})^* = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{l}{l+m} & \frac{m}{l+m} \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}.$$

The self-loops abstraction subvector of $\overline{E}$ for the submatrix $\mathbf{F}$ (see Example 4.8) is $SL_F = \left(\frac{1}{\rho}, 1, \frac{1}{\theta}, \frac{1}{\phi}\right)$. The self-loops abstraction vector of $\overline{E}$ in $REDTMC(\overline{E})$ (for the matrix $\mathbf{H}'$, see below) is $(SL^*)^{\diamond} = SL_{H'} = (1, 2, 1, 1)$. The self-loops abstraction vector of $\overline{E}$ in $RDTMC(\overline{E})$ is $SL^{\diamond} = \mathbf{1}Diag(SL_F)Diag(SL_{H'}) = \left(\frac{1}{\rho}, 2, \frac{1}{\theta}, \frac{1}{\phi}\right)$, where $\mathbf{1}$ is a row vector of n values 1.

The elements of the matrix $\mathbf{H}'$ are the probabilities to move from tangible to tangible states, via any *positive* number of vanishing states, without traversal of tangible states, in $EDTMC(G)$. We have $\mathbf{H}' = Diag(SL_F)\mathbf{H}$, where elements of the matrix $\mathbf{H} = \mathbf{EGD}$ (see Example 4.8) are the probabilities to move from tangible to tangible states, via any *positive* number of vanishing states, without traversal of tangible states, in $DTMC(G)$. The matrices $\mathbf{H}$ and $\mathbf{H}'$ are

$$\mathbf{H} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & \frac{l}{2(l+m)} & \frac{m}{2(l+m)} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \ \mathbf{H}' = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & \frac{l}{2(l+m)} & \frac{m}{2(l+m)} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

Then it is easy to check that

$$((\mathbf{P}^*)^{\diamond})^* = Diag(SL_{H'})Diag(SL_F)(\mathbf{P}^{\diamond} - \mathbf{I}) + \mathbf{I} = Diag(SL^{\diamond})(\mathbf{P}^{\diamond} - \mathbf{I}) + \mathbf{I} = (\mathbf{P}^{\diamond})^*.$$

5. Discussion

In this paper, we have considered a discrete time stochastic extension dtsdPBC of PBC, enriched with deterministic multiactions. The calculus has a parallel step operational semantics, based on labeled probabilistic transition systems and a denotati-

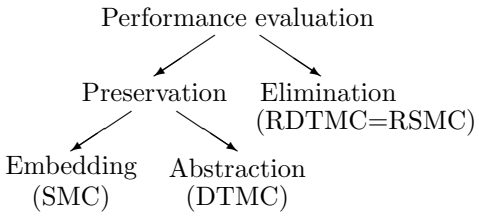


Figure 6. Performance analysis methods in dtsdPBC.

onal semantics in terms of a subclass of LDTSDPNs [40,41]. A technique of performance evaluation within the calculus has been presented (*embedding*) that explores the underlying stochastic process, which is a semi-Markov chain (SMC). In such an SMC, the sojourn time in every tangible state is geometrically distributed (being one or infinity, as special cases) while the sojourn time in every vanishing state is zero. The corresponding discrete time Markov chain (DTMC) or its reduction (RDTMC) by eliminating vanishing states may alternatively be studied for that purpose (the *abstraction* and *elimination* methods). Since both *embedding* and *abstraction* preserve vanishing states, they are classified as the *preservation* methods [42].

We have formally proved that the reduced SMC (RSMC) coincides with the RDTMC. The proof of this very intuitive fact appeared to be rather involved. First, we have shown that an additional embedding (into RSMC) of the reduced EDTMC is needed to coincide with the embedded RDTMC. Second, we have calculated the respective sojourn times in the tangible states (those with positive sojourn times) and have checked their coincidence. It is more optimal to construct the RDTMC than to build the RSMC, since the former approach involves only one embedding that requires a lot of computations. This is especially important for the DTMCs having many loops starting from (and ending in the same) tangible states and going via (one or more) vanishing states only, since such loops are not removed by the first embedding. Figure 6 depicts the performance analysis methods in dtsdPBC with our coincidence result.

In the continuous time setting, the embedding of the EDTMC into an SMC with only exponential and zero sojourn times is not hard to construct. Such SMCs resemble CTMCs, in that their behaviour is not changed at presence of self-loops (the average sojourn times are kept). The EDTMC transition probabilities are obtained by normalizing the rates of the transitions from each state to the different ones. Only the embedding of the reduced EDTMC into the RSMC (actually being a CTMC) requires more computations. In our discrete time setting, we cannot simply abstract from self-loops, since each of them takes one time unit and thereby influences the average sojourn time. Then avoiding the additional embedding allows us to decrease the computation costs.

Future work consists in constructing a congruence relation for dtsdPBC, i.e. the equivalence that withstands application of all operations of the algebra. A possible candidate is a stronger version of the equivalence with respect to transition systems, with two extra transitions skip and redo, like in sPBC [23]. The recursion operation could be added to dtsdPBC to increase specification power of the algebra. We also plan to extend dtsdPBC with discrete phase type multiaction delays that are described by arbitrary finite absorbing DTMCs and include geometric and non-Markovian (like deterministic) delays as special cases.

6. Conclusion

To sum up, we have extended Petri box calculus by incorporating discrete time stochastic and deterministic delays, allowing for more precise modeling of parallel processes. We have introduced a performance evaluation technique based on semi-Markov chains and have demonstrated that the reduced semi-Markov chain is identical to the reduced discrete time Markov chain. This finding simplifies computations in complex systems, particularly those with many loops through vanishing states. Future work will focus on enhancing the calculus by introducing a congruence relation, recursion operation, and extending the delay types to include more general cases.

Disclosure statement

No potential conflict of interest was reported by the author(s).

References

[1] Hoare CAR. Communicating sequential processes. Prentice-Hall, London, UK; 1985.

[2] Bergstra JA, Klop JW. Algebra of communicating processes with abstraction. Theor Comput Sci. 1985;37:77–121.

[3] Milner RAJ. Communication and concurrency. Prentice-Hall, Upper Saddle River, NJ, USA; 1989.

[4] Hermanns H, Rettelbach M. Syntax, semantics, equivalences and axioms for MTIPP. In: Herzog U, Rettelbach M, editors. Proc. 2^{nd} Int. Workshop on Process Algebra and Performance Modelling (PAPM) 1994; (Arbeitsberichte des IMMD; Vol. 27); Regensberg / Erlangen, Germany. Universität Erlangen-Nürnberg, Germany; 1994. p. 71–88.

[5] Hillston J. A compositional approach to performance modelling [dissertation]. Edinburgh, UK: Department of Computer Science, University of Edinburgh; 1994.

[6] Hillston J. A compositional approach to performance modelling. Cambridge University Press, Cambridge, UK; 1996.

[7] Bernardo M, Gorrieri R. A tutorial on EMPA: a theory of concurrent processes with nondeterminism, priorities, probabilities and time. Theor Comput Sci. 1998;202:1–54.

[8] Best E, Devillers R, Hall JG. The box calculus: a new causal algebra with multi-label communication. In: Rozenberg G, editor. Advances in Petri Nets (APN) 1992; (Lect. Notes Comp. Sci.; Vol. 609). Springer; 1992. p. 21–69.

[9] Best E, Koutny M. A refined view of the box algebra. In: Michelis GD, Diaz M, editors. Proc. 16^{th} Int. Conf. on Application and Theory of Petri Nets (ICATPN) 1995; (Lect. Notes Comp. Sci.; Vol. 935). Springer; 1995. p. 1–20.

[10] Best E, Devillers R, Koutny M. Petri net algebra. Springer; 2001. EATCS Monographs on Theor. Comput. Sci.

[11] Best E, Devillers R. Petri net primer: a compendium on the core model, analysis, and synthesis. 1^{st} ed. Springer International Publishing, Cham, Switzerland / Birkhäuser, Germany; 2024. Computer Science Foundations and Applied Logic Series (CSFAL).

[12] Koutny M. A compositional model of time Petri nets. In: Nielsen M, Simpson D, editors. Proc. 21^{st} Int. Conf. on Application and Theory of Petri Nets (ICATPN) 2000; (Lect. Notes Comp. Sci.; Vol. 1825). Springer; 2000. p. 303–322.

[13] Merlin PM, Farber DJ. Recoverability of communication protocols: implications of a theoretical study. IEEE Trans Communications. 1976;24:1036–1043.

[14] Marroquín A O, de Frutos E D. TPBC: timed Petri box calculus. Madrid, Spain: Departamento de Sistemas Infofmáticos y Programación, Universidad Complutense de Madrid; 2000. Technical report. In Spanish.

[15] Marroquín A O, de Frutos E D. Extending the Petri box calculus with time. In: Colom JM, Koutny M, editors. Proc. 22^{nd} Int. Conf. on Applications and Theory of Petri Nets (ICATPN) 2001; (Lect. Notes Comp. Sci.; Vol. 2075). Springer; 2001. p. 303–322.

[16] Ramchandani C. Perfomance evaluation of asynchronous concurrent systems by timed Petri nets [dissertation]. Cambridge, USA: Massachusetts Institute of Technology; 1973.

[17] Niaouris A. An algebra of Petri nets with arc-based time restrictions. In: Liu Z, Araki K, editors. Proc. 1^{st} Int. Colloquium on Theoretical Aspects of Computing (ICTAC) 2004; (Lect. Notes Comp. Sci.; Vol. 3407). Springer; 2005. p. 447–462.

[18] Niaouris A, Koutny M. An algebra of timed-arc Petri nets. Newcastle upon Tyne, UK: School of Computer Science, University of Newcastle upon Tyne; 2005. Technical Report CS-TR-895. Available from: http://www.cs.ncl.ac.uk/publications/trs/papers/895.pdf.

[19] Bolognesi T, Lucidi F, Trigila S. From timed Petri nets to timed LOTOS. In: Logrippo L, Probert RL, Ural H, editors. Proc. IFIP WG6.1 10^{th} Int. Symposium on Protocol Specification, Testing and Verification (PSTV) 1990; Ottawa, Ontario, Canada. Elsevier Science Publishers (North-Holland), Amsterdam, The Netherlands; 1990. p. 395–408.

[20] Hanish HM. Analysis of place/transition nets with timed-arcs and its application to batch process control. In: Marsan MA, editor. Proc. 14^{th} Int. Conf. on Application and Theory of Petri Nets (ICATPN) 1993; (Lect. Notes Comp. Sci.; Vol. 691). Springer; 1993. p. 282–299.

[21] Macià S H, Valero R V, de Frutos E D. sPBC: a Markovian extension of finite Petri box calculus. In: Proc. 9^{th} IEEE Int. Workshop on Petri Nets and Performance Models (PNPM) 2001; Aachen, Germany. IEEE Computer Society Press; 2001. p. 207–216.

[22] Macià S H, Valero R V, Cazorla L DC, et al. Introducing the iteration in sPBC. In: de Frutos E D, Núñez G M, editors. Proc. 24^{th} Int. Conf. on Formal Techniques for Networked and Distributed Systems (FORTE) 2004; (Lect. Notes Comp. Sci.; Vol. 3235). Springer; 2004. p. 292–308.

[23] Macià S H, Valero R V, Cuartero G F, et al. A congruence relation for sPBC. Form Methods Syst Des. 2008;32:85–128.

[24] Marsan MA. Stochastic Petri nets: an elementary introduction. In: Rozenberg G, editor. Advances in Petri Nets (APN) 1989; (Lect. Notes Comp. Sci.; Vol. 424). Springer; 1990. p. 1–29.

[25] Balbo G. Introduction to stochastic Petri nets. In: Brinksma E, Hermanns H, Katoen JP, editors. Proc. 1^{st} EEF/Euro Summer School of Trends in Comp. Sci. 2000; (Lect. Notes Comp. Sci.; Vol. 2090). Springer; 2001. p. 84–155.

[26] Macià S H, Valero R V, Cuartero G F, et al. sPBC: a Markovian extension of Petri box calculus with immediate multiactions. Fundam Inform. 2008;87:367–406.

[27] Macià S H, Valero R V, Cuartero G F, et al. Modelling a video conference system with sPBC. Appl Math Inf Sci. 2016;10:475–493.

[28] Balbo G. Introduction to generalized stochastic Petri nets. In: Bernardo M, Hillston J, editors. Formal Methods for Performance Evaluation. Proc. 7^{th} Int. School on Formal Methods for the Design of Computer, Communication, and Software Systems (SFM) 2007; (Lect. Notes Comp. Sci.; Vol. 4486). Springer; 2007. p. 83–131.

[29] Tarasyuk IV. Discrete time stochastic Petri box calculus. Oldenburg, Germany: Carl von Ossietzky Universität Oldenburg; 2005. Berichte aus dem Department für Informatik 3/05. Available from: http://itar.iis.nsk.su/files/itar/pages/dtspbcib_cov.pdf.

[30] Tarasyuk IV. Iteration in discrete time stochastic Petri box calculus. Bull Nov Comp Center, Comp Science, IIS Special Issue. 2006;24:129–148. Available from: http://bulletin.iis.nsk.su/files/article/tarasyuk_2.pdf.

[31] Tarasyuk IV. Stochastic Petri box calculus with discrete time. Fundam Inform. 2007; 76:189–218.

[32] Tarasyuk IV. Equivalence relations for modular performance evaluation in dtsPBC. Math Struct Comp Sci. 2014;24:78–154 (e240103).

[33] Molloy MK. On the integration of the throughput and delay measures in distributed processing models [dissertation]. Los Angeles, CA, USA: University of California; 1981.

[34] Molloy MK. Discrete time stochastic Petri nets. IEEE Trans Software Eng. 1985;11:417–423.
[35] Tarasyuk IV, Macià S H, Valero R V. Discrete time stochastic Petri box calculus with immediate multiactions. Albacete, Spain: Department of Computer Systems, High School of Computer Science Engineering, University of Castilla - La Mancha; 2010. Technical Report DIAB-10-03-1. Available from: http://www.dsi.uclm.es/descargas/technicalreports/DIAB-10-03-1/dtsipbc.pdf.
[36] Tarasyuk IV, Macià S H, Valero R V. Discrete time stochastic Petri box calculus with immediate multiactions dtsiPBC. In: Bradley J, Heljanko K, Knottenbelt W, et al., editors. Proc. 6^{th} Int. Workshop on Practical Applications of Stochastic Modelling (PASM) 2012 and 11^{th} Int. Workshop on Parallel and Distributed Methods in Verification (PDMC) 2012; (Electronic Notes in Theor. Comp. Sci.; Vol. 296); London, UK. Elsevier; 2013. p. 229–252.
[37] Tarasyuk IV, Macià S H, Valero R V. Performance analysis of concurrent systems in algebra dtsiPBC. Programming and Computer Software. 2014;40:229–249.
[38] Tarasyuk IV, Macià S H, Valero R V. Stochastic process reduction for performance evaluation in dtsiPBC. Siberian Electronic Mathematical Reports. 2015;12:513–551.
[39] Tarasyuk IV, Macià S H, Valero R V. Stochastic equivalence for performance analysis of concurrent systems in dtsiPBC. Siberian Electronic Mathematical Reports. 2018;15:1743–1812.
[40] Tarasyuk IV. Discrete time stochastic and deterministic Petri box calculus. Ithaca, NY, USA: Computing Research Repository, Cornell University Library; 2019. CoRR abs/1905.00456.
[41] Tarasyuk IV. Discrete time stochastic and deterministic Petri box calculus dtsdPBC. Siberian Electronic Mathematical Reports. 2020;17:1598–1679.
[42] Tarasyuk IV. Performance evaluation in stochastic process algebra dtsdPBC. Siberian Electronic Mathematical Reports. 2021;18:1105–1145.
[43] Tarasyuk IV. Performance preserving equivalence for stochastic process algebra dtsdPBC. Siberian Electronic Mathematical Reports. 2023;20:646–699.
[44] Zijal R, German R. A new approach to discrete time stochastic Petri nets. In: Cohen G, Quadrat JP, editors. Proc. 11^{th} Int. Conf. on Analysis and Optimization of Systems, Discrete Event Systems (DES) 1994; (Lecture Notes in Control and Information Sciences; Vol. 199); Sophia-Antipolis, France. Springer; 1994. p. 198–204.
[45] Zijal R. Discrete time deterministic and stochastic Petri nets. In: Hommel G, editor. Proc. Int. Workshop on Quality of Communication-Based Systems 1994; Technical University of Berlin, Germany. Kluwer Academic Publishers; 1995. p. 123–136.
[46] Zijal R. Analysis of discrete time deterministic and stochastic Petri nets [dissertation]. Berlin, Germany: Technical University of Berlin; 1997.
[47] Marsan MA, Balbo G, Conte G, et al. Modelling with generalized stochastic Petri nets. John Wiley and Sons; 1995. Wiley Series in Parallel Computing.
[48] van der Aalst WMP, van Hee KM, Reijers HA. Analysis of discrete-time stochastic Petri nets. Statistica Neerlandica. 2000;54:237–255.
[49] Ross SM. Stochastic processes. 2^{nd} ed. John Wiley and Sons, New York, USA; 1996.
[50] Stewart WJ. Probability, Markov chains, queues, and simulation. the mathematical basis of performance modeling. Princeton University Press, Princeton, NJ, USA; 2009.
[51] Kulkarni VG. Modeling and analysis of stochastic systems. 2^{nd} ed. Chapman and Hall / CRC Press; 2010. Texts in Statistical Science.
[52] Borovkov AA. Probability theory. 5^{th} ed. Springer London; 2013. Universitext (UTX).
[53] Trivedi KS. Probability and statistics with reliability, queuing, and computer science applications. 2^{nd} ed. John Wiley and Sons, Hoboken, NJ, USA; 2016.
[54] Lakatos L, Szeidl L, Telek M. Introduction to queueing systems with telecommunication applications. 2^{nd} ed. Springer Nature, Cham, Switzerland; 2019.
[55] Ross SM. Introduction to probability models. 12^{th} ed. Academic Press, Elsevier, UK; 2019.

[56] Shiryaev AN. Probability-2. 3^{rd} ed. (Graduate Texts in Mathematics (GTM); Vol. 95). Springer, New York; 2019.

[57] Zimmermann A, Freiheit J, German R, et al. Petri net modelling and performability evaluation with TimeNET 3.0. In: Haverkort BR, Bohnenkamp HC, Smith CU, editors. Proc. 11^{th} Int. Conf. on Modelling Techniques and Tools for Computer Performance Evaluation (TOOLS) 2000; (Lect. Notes Comp. Sci.; Vol. 1786). Springer; 2000. p. 188–202.

[58] Zimmermann A, Freiheit J, Hommel G. Discrete time stochastic Petri nets for modeling and evaluation of real-time systems. In: Proc. 9^{th} Int. Workshop on Parallel and Distributed Real Time Systems (WPDRTS) 2001; San Francisco, USA; 2001. p. 282–286.

[59] Zimmermann A. Modeling and evaluation of stochastic Petri nets with TimeNET 4.1. In: Gaujal B, Jean-Marie A, Jorswieck E, et al., editors. Proc. 6^{th} Int. ICST Conf. on Performance Evaluation Methodologies and Tools (VALUETOOLS) 2012; Cargèse, France. IEEE Computer Society Press; 2012. p. 1–10.

[60] Zijal R, Ciardo G. Discrete deterministic and stochastic Petri nets. Hampton, VA, USA: Institute for Computer Applications in Science and Engineering (ICASE), NASA Langley Research Center; 1996. ICASE Report 96-72. Available from: http://www.cs.odu.edu/~mln/ltrs-pdfs/icase-1996-72.pdf.

[61] Zijal R, Ciardo G, Hommel G. Discrete deterministic and stochastic Petri nets. In: Irmscher K, Mittasch C, Richter K, editors. Proc. 9^{th} ITG/GI Professional Meeting on Measuring, Modeling and Evaluation of Computer and Communication Systems (MMB) 1997, Volume 1; Freiberg, Germany. VDE-Verlag, Berlin, Germany; 1997. p. 103–117.

[62] Haverkort BR. Markovian models for performance and dependability evaluation. In: Brinksma E, Hermanns H, Katoen JP, editors. Proc. 1^{st} EEF/Euro Summer School of Trends in Comp. Sci. 2000; (Lect. Notes Comp. Sci.; Vol. 2090). Springer; 2001. p. 38–83.

[63] van Glabbeek RJ. The linear time – branching time spectrum II: the semantics of sequential systems with silent moves. Extended abstract. In: Best E, editor. Proc. 4^{th} Int. Conf. on Concurrency Theory (CONCUR) 1993; (Lect. Notes Comp. Sci.; Vol. 715). Springer; 1993. p. 66–81.

[64] Mudge TN, Al-Sadoun HB. A semi-Markov model for the performance of multiple-bus systems. IEEE Trans Computers. 1985;C-34:934–942.

[65] Katoen JP. Quantinative and qualitative extensions of event structures [dissertation]. Enschede, The Netherlands: Centre for Telematics and Information Technology, University of Twente; 1996.

[66] Chiola G. A software package for the analysis of generalized stochastic Petri net models. In: Proc. 1^{st} Int. Workshop on Timed Petri Nets 1985; Turin, Italy. IEEE Computer Society Press; 1985. p. 136–143.

[67] Ciardo G, Muppala JK, Trivedi KS. Analysis of deterministic and stochastic Petri nets. In: Proc. 3^{rd} Int. Workshop on Petri Nets and Performance Models (PNPM) 1989; Kyoto, Japan. IEEE Computer Society Press; 1989. p. 142–151.

[68] Ciardo G, Muppala JK, Trivedi KS. On the solution of GSPN reward models. Perform Eval. 1991;12:237–253.

[69] Bause F, Kritzinger PS. Stochastic Petri nets: an introduction to the theory. 2^{nd} ed. Friedrich Vieweg and Sohn, Braunschweig / Wiesbaden, Germany; 2002.