S©MR ISSN 1813-3304

СИБИРСКИЕ ЭЛЕКТРОННЫЕ
МАТЕМАТИЧЕСКИЕ ИЗВЕСТИЯ

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

Том 18, стр. 1–42 (2021) УДК 004.423.4, 519.217.2, 519.681.2, 519.681.3
DOI 10.33048/semi.2021.18.xxx MSC 18C10, 60J10, 60J20, 60K15, 68Q55, 68Q85

PERFORMANCE EVALUATION IN STOCHASTIC PROCESS ALGEBRA DTSDPBC

I.V. TARASYUK

Abstract. We consider discrete time stochastic and deterministic Petri box calculus (dtsdPBC), recently proposed by I.V. Tarasyuk. dtsdPBC is a discrete time extension with stochastically and deterministically timed multiactions of the well-known Petri box calculus (PBC), presented by E. Best, R. Devillers, J.G. Hall and M. Koutny. In dtsdPBC, stochastic multiactions have (conditional) probabilities of execution at the next time moment while deterministic multiactions have non-negative integers associated that specify fixed (including zero) delays. dtsdPBC features a step operational semantics via labeled probabilistic transition systems.

In order to evaluate performance, the underlying semi-Markov chains (SMCs) are investigated, which are extracted from the transition systems of the process expressions. It is demonstrated that the performance analysis is alternatively possible by exploring the corresponding discrete time Markov chains (DTMCs) and their reductions (RDTMCs), obtained by eliminating the states with zero residence time (vanishing states). The method based on DTMCs permits to avoid building the embedded DTMC (EDTMC) and weighting the probability masses in the states by their average sojourn times. The method based on RDTMCs simplifies performance analysis of large systems due to abstracting from the non-stop transit (vanishing) states where only instantaneous activities are executed, resulting in a smaller model that can easier be solved directly.

Keywords: stochastic process algebra, Petri box calculus, discrete time, stochastic multiaction, deterministic multiaction, transition system, operational semantics, Markov chain, performance evaluation.

Tarasyuk, I.V., Performance evaluation in stochastic process algebra dtsdPBC.

Received Month, xx, 2021, published Month, xx, 2021.

1. Introduction

Algebraic process calculi like CSP [22], ACP [6] and CCS [37] are well-known formal models for specification of computing systems and analysis of their behaviour. In such process algebras (PAs), systems and processes are specified by formulas, and verification of their properties is accomplished at a syntactic level via equivalences, axioms and inference rules. In recent decades, stochastic extensions of PAs were proposed, such as MTIPP [20], PEPA [21] and EMPA [8]. Unlike standard PAs, stochastic process algebras (SPAs) do not just specify actions which can occur (qualitative features), but they associate with the actions the distribution parameters of their random time delays (quantitative characteristics).

1.1. **Petri box calculus.** Petri box calculus (PBC) [9, 11, 10] is a flexible and expressive process algebra developed as a tool for specification of the 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 not from single (visible or invisible) actions and variables, like in CCS, but 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 equipped with an interface and considered up to isomorphism, called Petri boxes. For more detailed comparison of PBC with other process algebras and the reasoning about importance of non-interleaving semantics see [9, 10]. The extensions of PBC with a deterministic, a nondeterministic or a stochastic model of time were presented.

1.2. **Time extensions of Petri box calculus.** A time extension of PBC with a nondeterministic time model, called time Petri box calculus (tPBC), was proposed in [24]. In tPBC, timing information is added by associating time intervals (the earliest and the latest firing time) 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 [36] and called time Petri boxes (ct-boxes).

Another time enrichment of PBC, called Timed Petri box calculus (TPBC), was defined in [32, 33], it accommodates a deterministic model of time. In contrast to tPBC, multiactions of TPBC are not instantaneous, but have time durations. Additionally, in TPBC there exist no "illegal" multiaction occurrences, unlike tPBC. The complexity of "illegal" occurrences mechanism was one of the main intentions to construct TPBC though this calculus appeared to be more complicated than tPBC. 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 [43] and called Timed Petri boxes (T-boxes). tPBC and TPBC differ in ways they capture time information, and they are not in competition but complement each other.

The third time extension of PBC, called arc time Petri box calculus (atPBC), was constructed in [41, 42], and it implements a nondeterministic time. In atPBC, multiactions are associated with time delay intervals. atPBC possesses a step time

operational semantics in terms of labeled transition systems. Its denotational semantics was defined on a subclass of labeled arc time Petri nets (atPNs), based of those from [12, 18], where time restrictions are associated with the arcs, called arc time Petri boxes (at-boxes). tPBC, TPBC and atPBC, all adopt the discrete time approach, but TPBC has no immediate (multi)actions.

1.3. **Stochastic extensions of Petri box calculus.** A stochastic extension of PBC, called stochastic Petri box calculus (sPBC), was proposed in [30, 26, 27]. 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 in terms of a subclass of labeled continuous time stochastic PNs, based on CTSPNs [34, 2] 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 [28, 29]. 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 [34, 2, 3] and called generalized stochastic Petri boxes (gs-boxes). The performance analysis in gsPBC is based on semi-Markov chains (SMCs).

In [45, 46, 47, 48], a discrete time stochastic extension dtsPBC of finite PBC was presented. In dtsPBC, the residence time in the process states is geometrically distributed. A step operational semantics of dtsPBC was constructed via labeled probabilistic transition systems. Its denotational semantics was defined in terms of a subclass of labeled discrete time stochastic PNs (LDTSPNs), based on DTSPNs [38, 39] and called discrete time stochastic Petri boxes (dts-boxes). The performance evaluation in dtsPBC is accomplished via the underlying discrete time Markov chains (DTMCs) of the algebraic processes.

In [50, 51, 52, 53], we presented a calculus dtsiPBC, 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. 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), based on the extension of DTSPNs [38, 39] with transition labeling and immediate transitions, 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.

1.4. **Our contributions.** As a basis model, we take an extension of dtsiPBC with deterministic multiactions, called *discrete time stochastic and deterministic Petri box calculus* (dtsdPBC) [49]. It enhances the expressiveness of dtsiPBC and extends the application area of the associated specification and analysis techniques.

In dtsdPBC, besides the probabilities from the real-valued interval $(0;1)$ that are used to calculate discrete time delays of stochastic multiactions, also non-negative integers are used to specify fixed time 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 [59, 55, 56], 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, constructed with the use of labeled probabilistic transition systems. The denotational semantics of dtsdPBC is defined in terms of an interface-featured subclass of labeled discrete time stochastic and deterministic Petri nets (LDTSPNs with deterministic transitions, LDTSDPNs), based on the extension of DTSPNs [38, 39] with transition labeling and deterministic transitions, called dtsd-boxes. Here we do not consider the denotational semantics of the calculus, since it was extensively described in our previous publication [49]. 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 transferred to the latter up to that equivalence.

The main result of this paper is the performance analysis methods in the framework of dtsdPBC. To evaluate performance, we construct and solve the underlying stochastic process, which is a semi-Markov chain (SMC). The obtained stationary probability masses in the states of the SMC are used to calculate the performance measures (indices) of interest. The alternative solution techniques are also developed, based on the corresponding discrete time Markov chain (DTMC) and its reduction (RDTMC) by eliminating vanishing states with zero sojourn (residence) times. The approach based on the DTMC allows one to avoid the costly intermediate stages of building the embedded DTMC (EDTMC), weighting the probability masses in the states by their average sojourn times and final normalization. The approach based on the RDTMC simplifies performance analysis of large systems due to abstracting from the non-stop transit (vanishing) states where only instantaneous activities can be executed, resulting in a smaller model having only tangible states (those with positive sojourn times) that can be solved directly with less efforts.

Thus, the main contributions of the paper are the following.

- Discrete time SPA with stochastic and deterministic activities dtsdPBC.
- Performance analysis method based on the underlying semi-Markov chain.
- Alternative solutions via the discrete time Markov chain and its reduction.

1.5. **Structure of the paper.** The paper is organized as follows. In Section 2, the syntax of algebra dtsdPBC is proposed. In Section 3, we present the operational semantics of the calculus in terms of labeled probabilistic transition systems. In Section 4, the underlying stochastic process (SMC) is defined and analyzed, after which the alternative solution methods are outlined, based on the corresponding DTMC and RDTMC. Finally, Section 5 summarizes the results obtained and outlines research perspectives in this area. The long proofs are moved to Appendix A.

2. Syntax

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

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

We denote the *set of all finite multisets* over a set X by $\mathbb{N}_{fin}^X$. Let $M, M' \in \mathbb{N}_{fin}^X$. 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 $\forall x \in X,\ M(x) \leq 1,\ M$ can be seen as a proper set $M \subseteq X$. The *set of all subsets (powerset)* of X is denoted by 2^X.

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

A *stochastic multiaction* is a pair (α, ρ), where $\alpha \in \mathcal{L}$ and $\rho \in (0; 1)$ is the *probability* of the multiaction α. This probability is interpreted as that of independent execution of the stochastic multiaction at the next discrete time moment. Such probabilities are used to calculate those to execute (possibly empty) sets of stochastic multiactions after one time unit delay. The probability 1 is left for (implicitly assigned to) waiting multiactions (positively delayed deterministic multiactions, to be defined later), which are delayed for at least one time unit before their execution and have weights to resolve conflicts with other waiting multiactions. There is no sense to allow probability 0 of stochastic multiactions, since they would never be performed in this case. 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 discrete time moment when the corresponding delay has expired. Such weights are used to calculate the probabilities to execute sets of deterministic multiactions after their time 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, to be explained later in more detail) are equal to one time unit, they are executed with probability 1 at the next time moment. Deterministic multiactions have a priority over stochastic ones, and there is also difference in priorities between immediate and waiting multiactions. This means that in a state where all kinds of multiactions can occur, immediate multiactions always occur before waiting ones that, in turn, are always executed before stochastic 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*. Obviously, we have $\mathcal{DL} = \mathcal{IL} \cup \mathcal{WL}$.

Let us note that 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 defined as $\mathcal{A}(\alpha, \kappa) = \mathcal{A}(\alpha)$. The *alphabet* of a multiset of activities $\Upsilon \in \mathbb{N}_{fin}^{\mathcal{SDL}}$ is defined as $\mathcal{A}(\Upsilon) = \cup_{(\alpha,\kappa)\in\Upsilon}\mathcal{A}(\alpha)$. For an activity $(\alpha, \kappa) \in \mathcal{SDL}$, we define its *multiaction part* as $\mathcal{L}(\alpha, \kappa) = \alpha$ and its *probability* or *weight part* as $\Omega(\alpha, \kappa) = \kappa$ if $\kappa \in (0; 1)$; or $\Omega(\alpha, \kappa) = l$ if $\kappa = \natural_l^\theta$, $\theta \in \mathbb{N}$, $l \in \mathbb{R}_{>0}$. The *multiaction part* of a multiset of activities $\Upsilon \in \mathbb{N}_{fin}^{\mathcal{SIL}}$ is defined as $\mathcal{L}(\Upsilon) = \sum_{(\alpha,\kappa)\in\Upsilon} \alpha$.

Activities are combined into formulas (process expressions) by the following operations: *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 process algebras, but parallelism (parallel composition) does not include synchronization, unlike the corresponding operation in CCS [37].

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

Restriction over an elementary action $a \in Act$ means that, for a given 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 $(\alpha \oplus_a \beta)(x) = \begin{cases} \alpha(x) + \beta(x) - 1, & \text{if } x = a \text{ or } x = \hat{a}; \\ \alpha(x) + \beta(x), & \text{otherwise.} \end{cases}$

Activities are synchronized with the use of their multiaction parts, i.e. the synchronization by a of two activities, whose multiaction parts α and β possess the properties mentioned above, results in the activity with the multiaction part $\alpha \oplus_a \beta$. We may synchronize activities of the same type only: either both stochastic multiactions or both deterministic ones *with the same delay*, since stochastic, waiting and immediate multiactions have different priorities, and diverse delays of waiting multiactions contradict their joint timing. Hence, the multiactions of different types cannot be executed together (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 subprocess is executed.

Static expressions specify the structure of processes, i.e. how activities are combined by operations in order to construct the composite process-algebraic formulas. As we shall see, static expressions correspond to unmarked labeled discrete time

stochastic and deterministic Petri nets (LDTSDPNs), which are marked by definition. A marking is the allocation of tokens in the places of a PN and markings are used to describe dynamic behaviour of PNs in terms of transition firings.

We assume that every waiting multiaction has a countdown timer associated, whose value is the discrete time amount left till the moment when the waiting multiaction can be executed. Therefore, besides standard (unstamped) waiting multiactions in the form of $(\alpha, \natural_l^\theta) \in \mathcal{WL}$, a special case of the *stamped* waiting multiactions should be considered in the definition of static expressions. Each (time) stamped waiting multiaction in the form of $(\alpha, \natural_l^\theta)^\delta$ has an extra superscript $\delta \in \{1, \ldots, \theta\}$ assigned that specifies a time stamp indicating the *latest* value of the countdown 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 their operation). The notions of the alphabet, multiaction part, weight part for (the multisets of) stamped waiting multiactions are defined, respectively, like those for (the multisets of) unstamped waiting multiactions.

By reasons of 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 can only be equal to 0.

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

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

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

To avoid technical difficulties with the iteration operator, we should not allow any concurrency at the highest level of the second argument of iteration. This is not a severe restriction, since we can always prefix parallel expressions by an activity with the empty multiaction part. Alternatively, we can use a different, safe, version of the iteration operator, but its net translation has six arguments. See also [10] for discussion on this subject. Remember that a PN is *n-bounded* ($n \in \mathbb{N}$) if for all its reachable (from the initial marking by the sequences of transition firings) markings there are at most n tokens in every place, and a PN is *safe* if it is 1-bounded.

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

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

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

Let E be a regular static expression. The *underlying timer-free regular static expression* $\lfloor E$ of E is obtained by removing from it 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 we shall see, dynamic expressions correspond to LDTSDPNs, which are marked by default. Dynamic expressions are obtained from static ones, by annotating them with upper or lower bars which specify the active components of the system at the current moment of time. The dynamic expression with upper bar (the overlined one) $\overline{E}$ denotes the *initial*, and that with lower bar (the underlined one) $\underline{E}$ denotes the *final* state of the process specified by a static expression E.

For every overlined stamped waiting multiaction in the form of $\overline{(\alpha, \natural_l^\theta)^\delta}$, 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 (equal to the delay of the waiting multiaction) at the moment when the waiting multiaction becomes overlined. Then such a newly overlined stamped waiting multiaction $\overline{(\alpha, \natural_l^\theta)^\theta}$ may be seen similar to the freshly overlined unstamped waiting multiaction $\overline{(\alpha, \natural_l^\theta)}$. Such similarity will be captured by the structural equivalence, to be 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) that follows in the next moment with probability 1, in case the stamped waiting multiaction is still overlined, there are no conflicting with it waiting multiactions, whose RTEs equal to one, and it is not affected by restriction. An activity is affected by restriction, if it is within the scope of a restriction operation with the argument action, such that it or its conjugate is contained in the multiaction part of that activity.

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

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

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

Let G be a dynamic expression. The *underlying static (line-free) expression* $\lfloor G \rfloor$ of G is obtained by removing from it all upper and lower bars. Note that if the underlying static expression of a dynamic one is not regular, the corresponding LDTSDPN can be non-safe (though, it is 2-bounded in the worst case [10]).

Definition 5. *A dynamic expression G is* regular *if its underlying static expression $\lfloor G \rfloor$ 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. As we shall see, 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, hence, its current marking stays unchanged.

Thus, an application of every inaction rule does not require any 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. In this table, $(\alpha, \natural_l^\theta) \in \mathcal{WL}$, $\delta \in \{1, \ldots, \theta\}$, $E, F, K \in$ *RegStatExpr* and $a \in$ *Act*. The first inaction rule suggests that the timer value of each newly overlined waiting multiaction is set to the delay of it.

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

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

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

Table 2. Inaction rules for arbitrary regular dynamic expressions

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

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

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

In the following, we consider regular expressions only and omit the word "regular".

Definition 7. *The relation* $\approx = (\Rightarrow \cup \Leftarrow)^*$ *is a* structural equivalence *of dynamic expressions in dtsdPBC, where* * *is the reflexive and transitive closure operation. 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 X be some set. We denote the Cartesian product $X \times X$ by X^2. Let $\mathcal{E} \subseteq X^2$ be an equivalence relation on X. Then the *equivalence class* (with respect to $\mathcal{E}$) of an element $x \in X$ is defined by $[x]_{\mathcal{E}} = \{y \in X \mid (x,y) \in \mathcal{E}\}$. The equivalence $\mathcal{E}$ partitions X into the *set of equivalence classes* $X/_{\mathcal{E}} = \{[x]_{\mathcal{E}} \mid x \in X\}$.

Let G be a dynamic expression. Then $[G]_{\approx} = \{H \mid G \approx H\}$ is the equivalence class of G with respect to the structural equivalence, 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 $H\}$, i.e. it consists of all stochastic multiactions that, being overlined, are the subexpressions of some operative dynamic expression from the state s. Analogously, 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 $H\}$, i.e. it consists of all waiting multiactions that, being superscribed with the values of their timers and overlined, are the subexpressions of some operative dynamic expression from the state s. 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 $H\}$, i.e. it consists of all waiting multiactions that, being superscribed with the initial values of their timers (delays of those waiting multiactions) and overlined, are the subexpressions of some operative dynamic expression from the state s.

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

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

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

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

Proof. See [49]. □

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

Let G be a saturated operative dynamic expression. Then $\circlearrowleft G$ is written for the *timer decrement* operator $\circlearrowleft$, applied to G. It denotes a saturated operative dynamic expression, obtained from G via decrementing by one time unit all greater than 1 values of the timers associated with all (if any) stamped waiting multiactions from the syntax of G. Thus, each such stamped waiting multiaction changes its timer value from δ in G to $\max\{1, \delta-1\}$ in $\circlearrowleft G$, where $\delta \in \mathbb{N}_{\geq 1}$. More formally, the timer decrement operator affects the (possibly overlined) stamped waiting multiactions being the subexpressions of G as follows. The overlined stamped waiting multiaction $\overline{(\alpha, \natural_l^\theta)^\delta}$ is replaced with $\overline{(\alpha, \natural_l^\theta)^{\max\{1,\delta-1\}}}$ while the stamped waiting multiaction without overline or underline $(\alpha, \natural_l^\theta)^\delta$ is replaced with $(\alpha, \natural_l^\theta)^{\max\{1,\delta-1\}}$.

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.

Observe that the timer decrement operator keeps stamping of the waiting multiactions, since it does not change any overlines or underlines, but it may only decrease their timer values, so that the stamped waiting multiactions stay stamped (with their timer values, possibly decremented by one).

Let G be a dynamic expression. Then $I_G : \mathcal{WL}(G) \to \mathbb{N}_{\geq 1}$ is the *timer valuation function* of the waiting multiactions of G, defined as follows. For $(\alpha, \natural_l^\theta) \in \mathcal{WL}(G)$, let $I_G((\alpha, \natural_l^\theta)) = \delta \in \{1, \dots \theta\}$, if $\exists H \in [G]_{\approx} \cap SatOpRegDynExpr$ $\overline{(\alpha, \natural_l^\theta)^\delta}$ or $(\alpha, \natural_l^\theta)^\delta$ is a subexpression of H. Otherwise, we let $I_G((\alpha, \natural_l^\theta)) = \infty$, where '$\infty$' denotes the undefined value (infinite time till the activity execution). The definition is correct by the argumentation from the proof of Proposition 1. Indeed, for each waiting multiaction of G, its timer value superscript (if any) is the same for every $H \in [G]_{\approx} \cap SatOpRegDynExpr$, in which that waiting multiaction, possibly being superscribed with the value of its timer and overlined or underlined, is a subexpression. Note that we may have $I_G((\alpha, \natural_l^\theta)) < \infty$ for $(\alpha, \natural_l^\theta) \in \mathcal{WL}(G) \setminus EnaWait([G]_{\approx})$, i.e. the non-enabled waiting multiactions of $[G]_{\approx}$ may have finite timer valuations. The latter is allowed only in the "incomplete" specifications by the compositionality reasons. It is assumed that all such non-enabled waiting multiactions have infinite timer values in the "complete" specification, hence, all and only enabled waiting multiactions have finite timer values there.

Let $G \in SatOpRegDynExpr$. For all $(\alpha, \natural_l^\theta) \in \mathcal{WL}(G)$, we have $I_{\circlearrowleft G}((\alpha, \natural_l^\theta)) = \max\{1, I_G((\alpha, \natural_l^\theta)) - 1\}$.

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. As we shall see, the application of an action rule with stochastic (immediate or waiting, respectively) multiactions to a dynamic expression leads in the corresponding LDTSDPN 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, i.e. a single stochastic (immediate or waiting, respectively) multiaction. The reason is the regularity requirement that allows no concurrency at the highest level of the second argument of iteration.

The empty move rule (applicable only when no immediate or waiting multiactions can be executed from the current state) describes dynamic expression transformations in the form of $G \stackrel{\emptyset}{\to} \circlearrowleft G$, called the *empty moves*, due to execution of the empty multiset of activities at a discrete time tick. When no timer values are decremented within G with the empty multiset execution at the next moment (for example, if G contains no stamped waiting multiactions), we have $\circlearrowleft G = G$. In such a case, the empty move from G is in the form of $G \stackrel{\emptyset}{\to} G$, called the *empty loop*. As we shall see, the application of the empty move rule to a dynamic expression leads to a discrete time tick in the corresponding LDTSDPN 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.

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

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

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

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

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

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

Definition 10. *Let $G \in OpRegDynExpr$. We define 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 $\Upsilon \in Can(G \circ E)$, $\Upsilon \in Can(E \circ G)$ $(\circ \in \{;, []\})$, $\Upsilon \in Can(G \| H)$, $\Upsilon \in Can(H \| G)$, $f(\Upsilon) \in Can(G[f])$, $\Upsilon \in Can(G \text{ rs } a)$ (when $a, \hat{a} \notin \mathcal{A}(\Upsilon)$), $\Upsilon \in Can(G \text{ sy } a)$, $\Upsilon \in Can([G * E * F])$, $\Upsilon \in Can([E * G * F])$, $\Upsilon \in Can([E * F * G])$.*
(6) *If $\Upsilon \in Can(G)$ and $\Xi \in Can(H)$ then $\Upsilon + \Xi \in Can(G \| H)$.*
(7) *If $\Upsilon \in Can(G \text{ sy } a)$ and $(\alpha, \kappa), (\beta, \lambda) \in \Upsilon$ are different, $a \in \alpha$, $\hat{a} \in \beta$, then*
   (a) *$(\Upsilon + \{(\alpha \oplus_a \beta, \kappa \cdot \lambda)\} - \{(\alpha, \kappa), (\beta, \lambda)\}) \in Can(G \text{ sy } a)$ if $\kappa, \lambda \in (0; 1)$;*
   (b) *$(\Upsilon + \{(\alpha \oplus_a \beta, \natural_{l+m}^{\theta})\} - \{(\alpha, \kappa), (\beta, \lambda)\}) \in Can(G \text{ sy } a)$ if $\kappa = \natural_l^{\theta}$, $\lambda = \natural_m^{\theta}$, $\theta \in \mathbb{N}$, $l, m \in \mathbb{R}_{>0}$.*

   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 to avoid introducing redundant ones.

   The synchronization of stochastic multiactions $(\alpha, \rho)_1$ and $(\beta, \chi)_2$ in different orders generates the activities $(\alpha \oplus_a \beta, \rho \cdot \chi)_{(1)(2)}$ and $(\beta \oplus_a \alpha, \chi \cdot \rho)_{(2)(1)}$. The synchronization of deterministic multiactions $(\alpha, \natural_l^{\theta})_1$ and $(\beta, \natural_m^{\theta})_2$ in different orders generates the activities $(\alpha \oplus_a \beta, \natural_{l+m}^{\theta})_{(1)(2)}$ and $(\beta \oplus_a \alpha, \natural_{m+l}^{\theta})_{(2)(1)}$. Since $Cont((1)(2)) = \{1, 2\} = Cont((2)(1))$, in both cases, only the first activity (symmetrically, the second one) resulting from synchronization appears in a multiset from $Can(G \text{ sy } a)$.

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

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

multiactions, since deterministic multiactions have a priority over stochastic ones, and should be executed first. Further, if there are no stochastic, but both waiting and immediate multiactions in the multisets from $Can(G)$, then, analogously, no waiting multiactions can be executed from G, since immediate multiactions have a priority over waiting ones (besides that over stochastic ones).

When there are only waiting and, possibly, stochastic multiactions in the multisets from $Can(G)$ then, from above, 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 anymore and they should occur at the next time moment with probability 1. The next definition formalizes these requirements.

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

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

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

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, $E, F \in RegStatExpr$, $G, H \in SatOpRegDynExpr$, $\widetilde{G}, \widetilde{H} \in RegDynExpr, G[]E, E[]G, [E * G * F], [E * F * G] \in SatOpRegDynExpr$ 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 following abbreviations in the names of the rules from the table: "**E**" for "**E**mpty move", "**B**" for "**B**asis case", "**S**" for "**S**equence", "**C**" for "**C**hoice", "**P**" for "**P**arallel", "**L**" for "re**L**abeling", "**R**" for "**R**estriction", "**I**" for "**I**teraton" and "**Sy**" for "**Sy**nchronization". The first rule in the table is the empty move rule **E**. The other rules are the action rules, describing transformations of dynamic expressions, which are built using particular algebraic operations. If we cannot merge the rules with stochastic, immediate ans waiting multiactions in one rule for some operation then we get the coupled action rules. In such cases, the names of the action rules with stochastic multiactions have a suffix '**s**', those with immediate multiactions have a suffix '**i**', and those with waiting multiactions have a suffix '**w**'.

TABLE 3. Action and empty move rules

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

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

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

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

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

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

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

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

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

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

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

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

For explanation of the rules in Table 3, see [49]. We do not have self-synchronization, i.e. synchronization of an activity with itself, since all the (enumerated) activities executed together are considered to be different. This allows us to avoid cumbersome and unexpected behaviour, as well as many technical difficulties [10].

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 adopt the *enabling memory* memory policy [35, 1, 2, 3] 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.

Similar in [24], we are mainly 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 (i.e. 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, to be presented later. Formally, 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 $\Upsilon_1, \ldots, \Upsilon_n \in \mathbb{N}_{fin}^{\mathcal{SDL}}$.

Therefore, we consider a dynamic expression $G = \overline{(\{a\}, \natural_1^2)^1[](\{b\}, \natural_2^3)^1}$ as "illegal" and that $H = \overline{(\{a\}, \natural_1^2)^1[](\{b\}, \natural_2^3)^2}$ as "legal", since the latter is obtained from the overlined static expression without timer value superscripts $\overline{E} = \overline{(\{a\}, \natural_1^2)[](\{b\}, \natural_2^3)}$ after one time tick. On the other hand, G is "illegal" only when it is intended to specify a complete process, but it may become "legal" as a part of some complete specification, like G rs a, since after two time ticks from $\overline{E}$ rs a, the timer values cannot be decreased further when the value 1 is approached. Thus, we should allow the dynamic expressions like G, by assuming that they are incomplete specifications, to be further composed. Further, a dynamic expression $G = \overline{(\{a\}, \frac{1}{2})}; (\{b\}, \natural_1^2)^1$ is "illegal", since the waiting multiaction $(\{b\}, \natural_1^2)$ is not enabled in $[G]_\approx$ and its timer cannot start before the stochastic multiaction $(\{a\}, \frac{1}{2})$ is executed. Enabledness of the stamped waiting multiactions is considered in the next proposition.

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

Proof. See [49]. □

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

Let G be a dynamic expression and $s = [G]_\approx$. The set of *all multisets of activities executable in s* is defined as $Exec(s) = \{\Upsilon \mid \exists H \in s \; \exists \widetilde{H} \; H \stackrel{\Upsilon}{\to} \widetilde{H}\}$. Here $H \stackrel{\Upsilon}{\to} \widetilde{H}$ 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 [49].

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

If $\Upsilon \in Exec(s)$ and $\Upsilon \in \mathbb{N}_{fin}^{\mathcal{SL}} \cup \mathbb{N}_{fin}^{\mathcal{IL}}$ then by rules **P2s**, **P2i**, **Sy2s**, **Sy2i** and definition of $Exec(s)$ $\forall \Xi \subseteq \Upsilon, \; \Xi \neq \emptyset$, we have $\Xi \in Exec(s)$, i.e. $2^\Upsilon \setminus \{\emptyset\} \subseteq Exec(s)$.

Definition 12. *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* $\exists \Upsilon \; H \stackrel{\Upsilon}{\to} \widetilde{H}$ *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 $DR(G) = DR_T(G) \uplus DR_V(G) = DR_{ST}(G) \uplus DR_{WT}(G) \uplus DR_V(G)$ ($\uplus$ denotes disjoint union).

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

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

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

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

If $s \in DR_{ST}(G)$ and $Exec(s) \neq \{\emptyset\}$ then $PF(\Upsilon, s)$ can be interpreted as a *joint* probability of independent events (in a probability sense, i.e. the probability of intersection of these events is equal to the product of their probabilities). Each such an event consists in the positive or the negative decision to be executed of a particular stochastic multiaction. Every executable stochastic multiaction decides probabilistically (using its probabilistic part) and independently (from others), if it wants to be executed in s. If Υ is a multiset of all executable stochastic multiactions which have decided to be executed in s and $\Upsilon \in Exec(s)$ then Υ is ready for execution in s. The multiplication in the definition is used because it reflects the probability of the independent event intersection. Alternatively, when $\Upsilon \neq \emptyset$, $PF(\Upsilon, s)$ can be interpreted as the probability to execute *exclusively* the multiset of stochastic multiactions Υ in s, i.e. the probability of *intersection* of two events calculated using the conditional probability formula in the form of $\mathsf{P}(X \cap Y) = \mathsf{P}(X|Y)\mathsf{P}(Y) = \prod_{(\alpha,\rho)\in\Upsilon} \rho \cdot \prod_{\{\{(\beta,\chi)\}\in Exec(s) \mid (\beta,\chi)\notin\Upsilon\}} (1-\chi)$, as shown in [49]. When $\Upsilon = \emptyset$, $PF(\Upsilon, s)$ can be interpreted as the probability not to execute in s any executable stochastic multiactions, thus, $PF(\emptyset, s) = \prod_{\{(\beta,\chi)\}\in Exec(s)} (1-\chi)$. When only the empty multiset of activities can be executed in s, i.e. $Exec(s) = \{\emptyset\}$, we take $PF(\emptyset, s) = 1$, since nothing more can be executed in s in this case. Since the probabilities of all stochastic multiactions are strictly less than 1, for $s \in DR_{ST}(G)$ we have $PF(\emptyset, s) \in (0; 1]$. Hence, we always execute the empty multiset of activities in s at the next moment with a certain positive probability.

If $s \in DR_{WT}(G) \cup DR_V(G)$ then $PF(\Upsilon, s)$ could be interpreted as the *overall (cumulative)* weight of the deterministic multiactions from Υ, i.e. the sum of all their weights. The summation here is used since the weights can be seen as the rewards which are collected [44]. This means that concurrent execution of the deterministic multiactions has more importance than that of every single one. The weights of deterministic multiactions can also be interpreted as bonus rewards of transitions [7]. The rewards are summed when deterministic multiactions are executed in parallel, because all of them participated in the execution. In particular, since execution of immediate multiactions takes no time, we prefer to collect in a step

(parallel execution of activities) as many parallel immediate multiactions as possible to get more progress in behaviour. As for waiting multiactions, only the maximal multisets of them executable from a state occur in the next moment. Therefore, the steps of waiting multiactions produce maximal overall weights, used to calculate probabilities of alternative maximal steps rather than the cumulative bonuses.

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

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

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

If $s \in DR_{ST}(G)$ then $PT(\Upsilon, s)$ can be interpreted as the *conditional* probability to execute Υ in s, calculated using the conditional probability formula in the form of $\mathsf{P}(Z|W) = \frac{\mathsf{P}(Z)}{\mathsf{P}(W)} = \frac{PF(\Upsilon,s)}{\sum_{\Xi \in Exec(s)} PF(\Xi,s)}$, as shown in [49]. One can also treat $PT(\Upsilon, s)$ and $PF(\Upsilon, s)$ as the *actual* and *potential* probabilities to execute Υ in s, respectively, since we have $PT(\Upsilon, s) = PF(\Upsilon, s)$ only when *all* sets (including the empty one) consisting of the executable stochastic multiactions can be executed in s. In this case, all the mentioned stochastic multiactions can be executed in parallel in s and we have $\sum_{\Xi \in Exec(s)} PF(\Xi, s) = 1$, since this sum collects the products of *all* combinations of the probability parts of the stochastic multiactions and the negations of these parts. But in general, for example, for two stochastic multiactions (α, ρ) and (β, χ) executable in s, it may happen that they cannot be executed in s together, i.e. $\emptyset, \{(\alpha, \rho)\}, \{(\beta, \chi)\} \in Exec(s)$, but $\{(\alpha, \rho), (\beta, \chi)\} \notin Exec(s)$. For $s \in DR_{ST}(G)$ we have $PT(\emptyset, s) \in (0; 1]$, i.e. there is a non-zero probability to execute the empty multiset of activities in s at the next moment.

If $s \in DR_{WT}(G) \cup DR_V(G)$ then $PT(\Upsilon, s)$ can be interpreted as the weight of the set of deterministic multiactions Υ which is ready for execution in s *normalized* by the weights of *all* the sets executable in s. This approach is analogous to that used in the EMPA definition of the probabilities of immediate actions executable from the same process state [8] (inspired by way in which the probabilities of conflicting immediate transitions in GSPNs are calculated [3]). The only difference is that we have a step semantics and, for every set of deterministic multiactions executed in parallel, we should use its cumulative weight. For the analogy with the interleaving semantics of EMPA, we should interpret the weights of immediate actions of EMPA as the cumulative weights of the sets of deterministic multiactions of dtsdPBC.

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

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

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

The summation in the definition above reflects the probability of the mutually exclusive event union, since $\sum_{\{\Upsilon | \exists H \in s,\ \exists \widetilde{H} \in \tilde{s},\ H \xrightarrow{\Upsilon} \widetilde{H}\}} PT(\Upsilon, s) = \frac{1}{\sum_{\Xi \in Exec(s)} PF(\Xi, s)}$.

$\sum_{\{\Upsilon \mid \exists H \in s,\ \exists \widetilde{H} \in \tilde{s},\ H \xrightarrow{\Upsilon} \widetilde{H}\}} PF(\Upsilon, s)$, where for each Υ, $PF(\Upsilon, s)$ is the probability of the exclusive execution of Υ in s. Note that $\forall s \in DR(G)$
$\sum_{\{\tilde{s} \mid \exists H \in s\ \exists \widetilde{H} \in \tilde{s}\ \exists \Upsilon\ H \xrightarrow{\Upsilon} \widetilde{H}\}} PM(s, \tilde{s}) = \sum_{\{\tilde{s} \mid \exists H \in s\ \exists \widetilde{H} \in \tilde{s}\ \exists \Upsilon\ H \xrightarrow{\Upsilon} \widetilde{H}\}}$
$\sum_{\{\Upsilon \mid \exists H \in s\ \exists \widetilde{H} \in \tilde{s}\ H \xrightarrow{\Upsilon} \widetilde{H}\}} PT(\Upsilon, s) = \sum_{\Upsilon \in Exec(s)} PT(\Upsilon, s) = 1$.

Definition 13. *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 $\mathcal{T}_G = \{(s, (\Upsilon, PT(\Upsilon, s)), \tilde{s}) \mid s, \tilde{s} \in DR(G),\ \exists H \in s\ \exists \widetilde{H} \in \tilde{s}\ H \xrightarrow{\Upsilon} \widetilde{H}\}$;*
- *the* initial state *is $s_G = [G]_{\approx}$.*

The definition of $TS(G)$ is correct, i.e. for every state, the sum of the probabilities of all the transitions starting from it is 1, by the note after the definition of $PT(\Upsilon, s)$.

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 (i.e. that consisting of the empty multiset of activities) occurs instantly after one discrete time unit delay. Each step consisting of immediate multiactions occurs instantly without any delay. The step can change the current state to 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 $s \xrightarrow{\Upsilon}_{\mathcal{P}} \tilde{s}$. It is interpreted as follows: the probability to change the state s to $\tilde{s}$ as a result of executing Υ is $\mathcal{P}$.

Note that 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 (i.e. the equivalence class). Then we may have a transition $s \xrightarrow{\emptyset}_{\mathcal{P}} \circlearrowleft s$ from an s-tangible state s to the tangible (i.e. s-tangible or w-tangible) state $\circlearrowleft s = \bigcup\{[\circlearrowleft H]_{\approx} \mid H \in s \cap SatOpRegDynExpr\}$. Thus, $\circlearrowleft s$ is the union of the structural equivalence classes of all saturated operative dynamic expressions from s, whose timer values have been decremented by one, prior to combining them into the equivalence classes. This corresponds to applying the empty move rule to all saturated operative dynamic expressions from s, followed by unifying the structural equivalence classes of all the resulting expressions. We have to keep track of such executions, called the *empty moves*, because they affect the timers and have non-zero probabilities. The latter follows from the definition of $PF(\emptyset, s)$ and the fact that the probabilities of stochastic multiactions cannot be equal to 1 as they belong to the interval $(0; 1)$. When it holds $\forall H \in s \cap SatOpRegDynExpr\ \ \circlearrowleft H = H$, we obtain $\circlearrowleft s = s$ by definition of $\circlearrowleft s$. Then the empty move from s is in the form of $s \xrightarrow{\emptyset}_{\mathcal{P}} s$, called the *empty loop*. For w-tangible and vanishing states Υ cannot be the empty multiset, since we must execute some immediate (waiting, respectively) multiactions from them at the current (next, respectively) time moment.

The step probabilities belong to the interval $(0; 1]$, being 1 in the case when we cannot leave an s-tangible state s and the only transition leaving it is the empty

move one $s \xrightarrow{\emptyset}_1 \circlearrowleft s$, or if there is just a single transition from a w-tangible or a vanishing state to any other one.

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

The first equivalence we are going to introduce is isomorphism which is a coincidence of systems up to renaming of their components or states.

Definition 14. *Let G, G' be dynamic expressions and $TS(G)=(S_G, L_G, \mathcal{T}_G, s_G)$, $TS(G')=(S_{G'}, L_{G'}, \mathcal{T}_{G'}, s_{G'})$ be their transition systems. A mapping $\beta : S_G \to S_{G'}$ is an* isomorphism *between $TS(G)$ and $TS(G')$, 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 15. *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 1. *Let $E = [(\{a\}, \rho) * ((\{b\}, \natural_k^1); ((((\{c\}, \natural_l^0); (\{d\}, \theta))[]((\{e\}, \natural_m^0); (\{f\}, \phi)))) * \mathsf{Stop}]$, where $\rho, \theta, \phi \in (0;1)$ and $k, l, m \in \mathbb{R}_{>0}$. $DR(\overline{E})$ consists of the equivalence classes*

$$
\begin{aligned}
s_1 &= [[\overline{(\{a\}, \rho)} * ((\{b\}, \natural_k^1); ((((\{c\}, \natural_l^0); (\{d\}, \theta))[]((\{e\}, \natural_m^0); (\{f\}, \phi)))) * \mathsf{Stop}]]_{\approx},\\
s_2 &= [[(\{a\}, \rho) * \overline{((\{b\}, \natural_k^1)^1}; ((((\{c\}, \natural_l^0); (\{d\}, \theta))[]((\{e\}, \natural_m^0); (\{f\}, \phi)))) * \mathsf{Stop}]]_{\approx},\\
s_3 &= [[(\{a\}, \rho) * ((\{b\}, \natural_k^1); (\overline{((\{c\}, \natural_l^0)}; (\{d\}, \theta))[]((\{e\}, \natural_m^0); (\{f\}, \phi)))) * \mathsf{Stop}]]_{\approx} =\\
&\quad [[(\{a\}, \rho) * ((\{b\}, \natural_k^1); ((((\{c\}, \natural_l^0); (\{d\}, \theta))[]\overline{((\{e\}, \natural_m^0)}; (\{f\}, \phi)))) * \mathsf{Stop}]]_{\approx},\\
s_4 &= [[(\{a\}, \rho) * ((\{b\}, \natural_k^1); ((((\{c\}, \natural_l^0); \overline{(\{d\}, \theta))}[]((\{e\}, \natural_m^0); (\{f\}, \phi)))) * \mathsf{Stop}]]_{\approx},\\
s_5 &= [[(\{a\}, \rho) * ((\{b\}, \natural_k^1); ((((\{c\}, \natural_l^0); (\{d\}, \theta))[]((\{e\}, \natural_m^0); \overline{(\{f\}, \phi)}))) * \mathsf{Stop}]]_{\approx}.
\end{aligned}
$$

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.

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.** 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

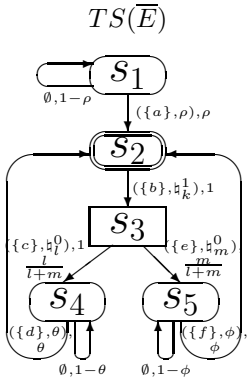

FIG. 1. The transition system of $\overline{E}$ for $E = [(\{a\},\rho) * ((\{b\},\natural_k^1); ((((\{c\},\natural_l^0);(\{d\},\theta))[](\{e\},\natural_m^0);(\{f\},\phi)))) * \mathsf{Stop}]$

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).

Note that 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 $SJ(s)$, $s \in DR(G)$.

The *sojourn time variance in the state s* is

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

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

To evaluate performance of the system specified by a dynamic expression G, we should investigate the stochastic process associated with it. The process is the underlying semi-Markov chain (SMC) [44, 25], 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 [34, 2, 3], and also in the framework of discrete time deterministic and stochastic PNs (DTDSPNs) in [59, 55, 56, 61, 62, 60], as well as within discrete deterministic and stochastic PNs (DDSPNs) [57, 58]. $EDTMC(G)$ 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) [19].

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 exist 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 now have to normalize $PM(s,\tilde{s})$. Thus, we have a two-stage normalization as a result.

Notice that $PM^*(s,\tilde{s})$ 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 [17] taking self-loops instead of silent transitions. Further, we shall not abstract from self-loops with probability 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 16. *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 $1 - PM(s,s)$ while the sojourn time in every $s \in DR_V(G)$ is equal to 0.*

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

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

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 probability less than 1 to get $\mathbf{P}^*$ and then ψ^*, followed by weighting by SJ and normalization. $EDTMC(G)$ has no self-loops with probability 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 probability 1) elements at the main diagonal.

Example 2. *Let E be from Example 1. 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 sojourn time variance vector of $\overline{E}$ is

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

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

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

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

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

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

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

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

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

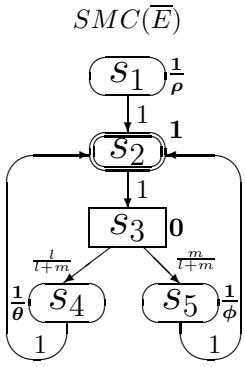

FIG. 2. The underlying SMC of $\overline{E}$ for $E = [(\{a\}, \rho) * ((\{b\}, \natural_k^1); ((((\{c\}, \natural_l^0); (\{d\}, \theta))[](\{e\}, \natural_m^0); (\{f\}, \phi)))) * \mathsf{Stop}]$

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

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

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

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

Let G be a dynamic expression and $s, \tilde{s} \in DR(G),\ S, \widetilde{S} \subseteq DR(G)$. The following standard *performance indices (measures)* can be calculated based on the steady-state PMF φ for $SMC(G)$ and the average sojourn time vector SJ of G [40, 23].

- 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 $ActsProb(\Xi) = \sum_{s \in DR(G)} \varphi(s) \sum_{\{\Upsilon \mid \Xi \subseteq \Upsilon\}} PT(\Upsilon, s)$.
- 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 3. *Let us interpret* E *from Example 1 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 from the syntax of E is as follows. The action a corresponds to the system activation (the travel route has been planned) that takes a time, geometrically distributed with the parameter ρ. The action b represents the completion of looking round the current city and coming to the city station that takes a fixed time equal to 1 *(say, one hour) for every city. The actions c and e correspond to the urgent getting on bus and train, respectively, and thus model the choice between these two transport facilities. The weights of the two corresponding immediate multiactions 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 the 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 meaning of states from $DR(\overline{E})$ is the following. 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 for departure of the transport. The s-tangible states s_4 and s_5 correspond to going by bus and train, respectively.

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

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

As mentioned in [59, 55, 56], if is useful to consider performance measures over only the markings of DTDSPNs, instead of their whole states, whose second components are the remaining firing time vectors. In the context of dtsdPBC, such markings correspond to those of the dtsd-boxes of dynamic expressions, i.e. to the markings of the respective LDTSDPNs [49], obtained from their states by abstracting from the second components, which are the timer valuation functions.

Let G be a dynamic expression. The *underlying timer-free state* of a state $s \in DR(G)$ is defined as $\lfloor s = [\lfloor H]_{\approx}$ for $H \in s$. Since structurally equivalent dynamic expressions obviously remain so after removing their timer value annotations, $\lfloor s$ is unique for each s and the previous definition is correct. The timer-free states (i.e. those from $\lfloor DR(G) = \{\lfloor s \mid s \in DR(G)\}$) correspond to the markings of the LDTSDPN $N = Box_{dtsd}(G)$. Let $s \in DR(G)$ and $\bar{s} = \lfloor s$. The steady-state

PMF for $SMC(G)$ over the timer-free states of G is defined as follows: $\varphi(\bar{s}) = \sum_{\{s \in DR(G) | |s=\bar{s}\}} \varphi(s)$. Then $\varphi(\bar{s})$ can be used to calculate the standard *performance indices over the timer-free states* of G (hence, over the markings of N), by analogy with the standard performance indices, defined over the arbitrary states of G. Then also the performance measures that are specific for LDTSDPNs can be derived, based on the numbers of tokens in the places of N.

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

Definition 17. *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 $\mathcal{P} = PM(s, \tilde{s})$.*

One can see that $EDTMC(G)$ is constructed from $DTMC(G)$ as follows. For each state of $DTMC(G)$, we remove a possible self-loop with probability less than 1, associated with it and then normalize the probabilities of the remaining transitions from the state. Thus, $EDTMC(G)$ and $DTMC(G)$ differ only by existence of self-loops with probability less than 1 and magnitudes of the probabilities of the remaining transitions. Hence, $EDTMC(G)$ and $DTMC(G)$ have the same communication classes of states and $EDTMC(G)$ is irreducible iff $DTMC(G)$ is so. Since both $EDTMC(G)$ and $DTMC(G)$ are finite, they are positive recurrent. Thus, in case of irreducibility, each of them has a single stationary PMF. Note that both $EDTMC(G)$ and $DTMC(G)$ or just one of them may be periodic, thus having a unique stationary distribution, but no steady-state (limiting) one. For example, it may happen that $EDTMC(G)$ is periodic while $DTMC(G)$ is aperiodic due to self-loops associated with some states of the latter. The states of $SMC(G)$ are classified using $EDTMC(G)$, hence, $SMC(G)$ is irreducible (positive recurrent, aperiodic) iff $EDTMC(G)$ is so.

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

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

The steady-state PMF ψ for $DTMC(G)$ is defined like the corresponding notion ψ^* for $EDTMC(G)$.

Let us determine a relationship between steady-state PMFs for $DTMC(G)$ and $EDTMC(G)$. The following theorem proposes the equation that relates the mentioned steady-state PMFs.

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

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

Proposition 3. *Let G be a dynamic expression and SL be its self-loops abstraction vector. Then the steady-state PMFs ψ for $DTMC(G)$ and ψ^* for $EDTMC(G)$ are related as follows: $\forall s \in DR(G)$*

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

Proof. See Appendix A.1. □

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

Proposition 4. *Let G be a dynamic expression, φ be the steady-state PMF for $SMC(G)$ and ψ be the steady-state PMF for $DTMC(G)$. Then $\forall s \in DR(G)$*

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

Proof. Let $s \in DR_T(G)$. Remember that $\forall s \in DR_T(G)\ SL(s) = SJ(s)$ and $\forall s \in DR_V(G)\ SJ(s) = 0$. Then, by Proposition 3, we have

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

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

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

□

Thus, to calculate φ, one can only apply normalization to some elements of ψ (corresponding to the tangible states), instead of abstracting from self-loops with probability less than 1 to get $\mathbf{P}^*$ and then ψ^*, followed by weighting by SJ and normalization. Hence, using $DTMC(G)$ instead of $EDTMC(G)$ allows one to avoid multistage analysis, but the payment for it is more time-consuming numerical and more complex analytical calculation of ψ with respect to ψ^*. The reason is that $DTMC(G)$ may have self-loops with probability less than 1, unlike $EDTMC(G)$, hence, the behaviour of $DTMC(G)$ may stabilize slower than that of $EDTMC(G)$ (if each of them has a single steady state) and $\mathbf{P}$ is potentially more dense matrix than $\mathbf{P}^*$, since $\mathbf{P}$ may have additional non-zero elements at the main diagonal. Nevertheless, Proposition 4 is very important, since the relationship between φ and ψ it discovers will be used in Proposition 5 to relate the steady-state PMFs for $SMC(G)$ and the reduced $DTMC(G)$.

Example 4. *Let E be from Example 1. In Figure 3, the DTMC $DTMC(\overline{E})$ is presented. The TPM for $DTMC(\overline{E})$ is*

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

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

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

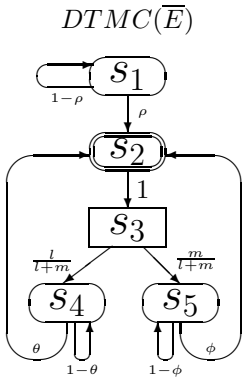

FIG. 3. The DTMC of $\overline{E}$ for $E = [(\{a\}, \rho) * (((\{b\}, \natural_k^1); ((((\{c\}, \natural_l^0); (\{d\}, \theta))[]((\{e\}, \natural_m^0); (\{f\}, \phi)))) * \mathsf{Stop}]$

Remember that $DR_T(\overline{E}) = DR_{ST}(\overline{E}) \cup DR_{WT}(\overline{E}) = \{s_1, s_2, s_4, s_5\}$ *and* $DR_V(\overline{E}) = \{s_3\}$. *Hence,*

$$\sum_{s \in DR_T(\overline{E})} \psi(s) = \psi(s_1) + \psi(s_2) + \psi(s_4) + \psi(s_5) = \frac{\theta\phi(l+m) + \phi l + \theta m}{2\theta\phi(l+m) + \phi l + \theta m}.$$

By Proposition 4, we have

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

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

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

This coincides with the result obtained in Example 2 with the use of ψ^* *and* SJ.

4.3. **Analysis of the reduced DTMC.** Let us now consider the method from [14, 15, 16, 35, 2, 5, 3] 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 [16], 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.

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 [35, 2, 5, 3] 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 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 $\mathbf{P}$ will correspond to the states from $DR_V(G)$ and the last ones will correspond to the states from $DR_T(G)$. Let $|DR(G)| = n$ and $|DR_T(G)| = m$. The resulting matrix can be decomposed as follows:

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

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

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

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

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

If there are no loops among vanishing states then for any vanishing state there exists a value $l \in \mathbb{N}$ such that every sequence of state changes that starts in a vanishing state and is longer than l should reach a tangible state. Thus, $\exists l \in \mathbb{N}\ \forall k > l\ \mathbf{C}^k = \mathbf{0}$ and $\sum_{k=0}^{\infty}\mathbf{C}^k = \sum_{k=0}^{l}\mathbf{C}^k$. If there are loops among vanishing states then all such loops are supposed to be of "transient" rather than "absorbing" type, since the latter is treated as a specification error to be corrected, like in [35, 3]. We have earlier required that $SMC(G)$ has a single closed communication (which is also ergodic) class of states. Remember that 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 [25]. In the latter case, a tangible state will be reached as well, as argued above. Thus, every sequence of vanishing state changes in $SMC(G)$ that starts in a vanishing state will exit the set of all vanishing states in the future. This implies that the probabilities to move from vanishing to vanishing states in $k \in \mathbb{N}$ state changes, without traversal of tangible states, will lead to 0 when k tends to ∞. Then we have $\lim_{k\to\infty} \mathbf{C}^k = \lim_{k\to\infty}(\mathbf{I}-(\mathbf{I}-\mathbf{C}))^k = \mathbf{0}$, hence, $\mathbf{I}-\mathbf{C}$ is a non-singular matrix, i.e. its determinant is not equal to zero. Thus, the inverse matrix of $\mathbf{I}-\mathbf{C}$ exists and may be expressed by a Neumann series as $\sum_{k=0}^{\infty}(\mathbf{I}-(\mathbf{I}-\mathbf{C}))^k = \sum_{k=0}^{\infty}\mathbf{C}^k = (\mathbf{I}-\mathbf{C})^{-1}$. Therefore,

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

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

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

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 the steady-state PMFs for $SMC(G)$ and $RDTMC(G)$. It proves that the steady-state probabilities of the tangible states coincide for them.

Proposition 5. *Let G be a dynamic expression, φ be the steady-state PMF for $SMC(G)$ and $\psi^{\diamond}$ be the steady-state PMF for $RDTMC(G)$. Then $\forall s \in DR(G)$*

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

Proof. See Appendix A.2. □

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 probability less than 1 to get $\mathbf{P}^*$ and then ψ^*, followed by weighting by SJ and normalization. Hence, using $RDTMC(G)$ instead of $EDTMC(G)$ allows one to avoid such a multistage analysis, but constructing $\mathbf{P}^{\diamond}$ also requires some efforts, including calculating matrix powers or inverse matrices. Note that $RDTMC(G)$ may have self-loops with probability 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 5. *Let E be from Example 1. 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

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

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

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

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

$RDTMC(\overline{E})$

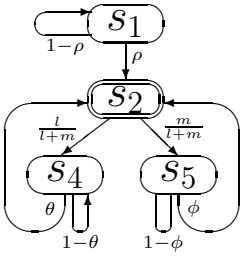

FIG. 4. The reduced DTMC of $\overline{E}$ for $E = [(\{a\}, \rho) * ((\{b\}, \natural_k^1); ((((\{c\}, \natural_l^0); (\{d\}, \theta))[]((\{e\}, \natural_m^0); (\{f\}, \phi)))) * \mathsf{Stop}]$

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

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

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

Note that $\psi^\diamond = (\psi^\diamond(s_1), \psi^\diamond(s_2), \psi^\diamond(s_4), \psi^\diamond(s_5))$. By Proposition 5, we have

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

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

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

This coincides with the result obtained in Example 2 with the use of ψ^* and SJ.

Example 6. *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*

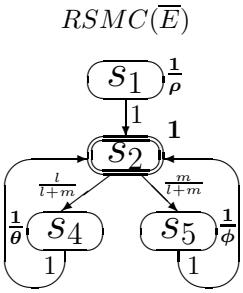

FIG. 5. The reduced SMC of $\overline{E}$ for $E = [(\{a\}, \rho) * ((\{b\}, \natural_k^1); ((((\{c\}, \natural_l^0); (\{d\}, \theta))[]((\{e\}, \natural_m^0); (\{f\}, \phi)))) * \mathsf{Stop}]$

distributed. Hence, the associated parameter of geometrical distribution is uniquely recovered from the average sojourn time in the state.

Our reduction of the underlying SMC by eliminating its vanishing states, resulting in the reduced DTMC, resembles the reduction from [31] by removing instantaneous states of stochastically discontinuous Markov reward chains. The latter are "limits" of continuous time Markov chains with state rewards and fast transitions when the rates (speeds) of these transitions tend to infinity, making them immediate. By analogy with this work, we could consider DTMCs extended with instantaneous states instead of SMCs with geometrically distributed or zero sojourn time in the states. However, within *dtsdPBC*, we have decided to take SMCs as the underlying stochastic process to be able to consider not only geometrically distributed and zero residence time in the states, but arbitrary fixed discrete time delays as well.

5. Conclusion

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 denotational semantics in terms of a subclass of LDTSDPNs. A technique of performance evaluation in the framework of the calculus has been presented that explores the corresponding stochastic process, which is a semi-Markov chain (SMC). It has been proved that the underlying discrete time Markov chain (DTMC) or its reduction (RDTMC) by eliminating vanishing states may alternatively and suitably be studied for that purpose.

The advantage of our framework is twofold. First, one can specify in it concurrent composition and synchronization of (multi)actions, whereas this is not possible in classical Markov chains. Second, algebraic formulas represent processes in a more compact way than PNs and allow one to apply syntactic transformations and comparisons. Process algebras are compositional by definition and their operations naturally correspond to operators of programming languages. Hence, it is much easier to construct a complex model in the algebraic setting than in PNs. The complexity of PNs generated for practical models in the literature shows that it is not straightforward to construct such PNs directly from the system specifications.

dtsdPBC is well suited for the discrete time applications, whose discrete states change with a global time tick, such as business processes, neural and transportation networks, computer and communication systems, timed web services [54], as well

as for those, in which the distributed architecture or the concurrency level should be preserved while modeling and analysis, such as genetic regulatory and cellular signalling networks (featuring maximal parallelism) in biology [13, 4] (remember that, in step semantics, we have additional transitions due to concurrent executions). dtsdPBC is also capable to model and analyze parallel systems with fixed durations of the typical activities (loading, processing, transfer, repair, low-level events, message delivery) and stochastic durations of the randomly occurring activities (arrival, departure, failure, packet loss, message collision), including industrial, manufacturing, queueing, computing and network systems. Thus, the main advantages of dtsdPBC are the flexible multiaction labels, deterministic and stochastic multiactions, powerful operations, as well as its step operational and Petri net denotational semantics, allowing for parallel executions (firings) of activities (transitions), with an ability for analytical performance evaluation.

In the following research, we plan to use step stochastic bisimulation equivalence to reduce behaviour of the algebraic processes by quotienting their transition systems and Markov chains. Such a reduction should simplify the functional (qualitative) and performance (quantitative) analysis. We would like to construct some application examples demonstrating expressiveness of the calculus and application of the behavioural analysis and performance evaluation, both simplified using quotienting by step stochastic bisimulation. Future work could also consist in constructing a congruence relation for dtsdPBC, i.e. the equivalence that withstands application of all operations of the algebra. The first possible candidate is a stronger version of step stochastic bisimulation equivalence, defined via transition systems equipped with two extra transitions skip and redo, like those from sPBC [27]. Moreover, recursion operation could be added to dtsdPBC to increase further specification power of the algebra.

Appendix A. Proofs

A.1. **Proof of Proposition 3.** Let PSL be a vector with the elements

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

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

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

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

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

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

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

Hence, $Diag(SL)Diag(PSL) + \mathbf{I} = Diag(SL)$. Thus,

$$\psi^* Diag(SL)\mathbf{P} = \psi^* Diag(SL).$$

Then, for $v = \psi^* Diag(SL)$, we have

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

In order to calculate ψ on the basis of v, we must normalize it by dividing its elements by their sum, since we should have $\psi\mathbf{1}^T = 1$ as a result:

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

Thus, the elements of ψ are calculated as follows: $\forall s \in DR(G)$

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

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

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

hence, it is indeed the steady-state PMF for $DTMC(G)$. □

A.2. **Proof of Proposition 5.** Let $\mathbf{P}$ be the reordered TPM for $DTMC(G)$ and ψ be the steady-state PMF for $DTMC(G)$, i.e. ψ is a solution of the equation system

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

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

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

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

Then the equation system for ψ is decomposed as follows:

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

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

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

We have

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

By Proposition 4, $\forall s \in DR_T(G)\ \varphi(s) = \frac{\psi(s)}{\sum_{\tilde{s} \in DR_T(G)} \psi(\tilde{s})}$.

Therefore, $\forall s \in DR_T(G)$

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

□

References

[1] W.M.P. van der Aalst, K.M. van Hee, H.A. Reijers, *Analysis of discrete-time stochastic Petri nets*, Statistica Neerlandica, **54**:2 (2000), 237–255. http://tmitwww.tm.tue.nl/staff/hreijers/H.A. Reijers Bestanden/Statistica.pdf. MR1794979

[2] G. Balbo, *Introduction to stochastic Petri nets*, Lecture Notes in Computer Science, **2090** (2001), 84–155. Zbl 0990.68092

[3] G. Balbo, *Introduction to generalized stochastic Petri nets*, Lecture Notes in Computer Science, **4486** (2007), 83–131. Zbl 1323.68400

[4] E. Bartocci, P. Lió, *Computational modeling, formal analysis, and tools for systems biology*, PLoS Computational Biology **12**:1 (2016), e1004591.

[5] F. Bause, P.S. Kritzinger, *Stochastic Petri nets: an introduction to the theory*, Vieweg Verlag, 2002. http://ls4-www.cs.tu-dortmund.de/cms/de/home/bause/bause_kritzinger_spn_book_print.pdf Zbl 1013.60065

[6] J.A. Bergstra, J.W. Klop, *Algebra of communicating processes with abstraction*, Theoretical Computer Science, **37** (1985), 77–121. MR0796314

[7] M. Bernardo, M. Bravetti, *Reward based congruences: can we aggregate more?* Lecture Notes in Computer Science, **2165** (2001), 136–151. MR1904353

[8] M. Bernardo, R. Gorrieri, *A tutorial on EMPA: a theory of concurrent processes with nondeterminism, priorities, probabilities and time*, Theoretical Computer Science, **202** (1998), 1–54. MR1626813

[9] E. Best, R. Devillers, J.G. Hall, *The box calculus: a new causal algebra with multi-label communication*, Lecture Notes in Computer Science, **609** (1992), 21–69. MR1253529

[10] E. Best, R. Devillers, M. Koutny, *Petri net algebra*, EATCS Monographs on Theoretical Computer Science, Springer, 2001. MR1932732

[11] E. Best, M. Koutny, *A refined view of the box algebra*, Lecture Notes in Computer Science, **935** (1995), 1–20. MR1461021

[12] T. Bolognesi, F. Lucidi, S. Trigila, *From timed Petri nets to timed LOTOS*, Proc. IFIP WG 6.1 10^{th} Int. Symposium on Protocol Specification, Testing and Verification 1990, Ottawa, Canada, 1–14, North-Holland, Amsterdam, The Netherlands, 1990.

[13] N. Bonzanni, K.A. Feenstra, W. Fokkink, E. Krepska, *What can formal methods bring to systems biology?* Lecture Notes in Computer Science, **5850** (2009), 16–22.

[14] G. Chiola, *A software package for the analysis of generalized stochastic Petri net models*, Proc. 1^{st} Int. Workshop on Timed Petri Nets 1985, Turin, Italy, IEEE Computer Society Press, July 1985.

[15] G. Ciardo, J.K. Muppala, K.S. Trivedi, *SPNP: stochastic Petri net package*, Proc. 3^{rd} Int. Workshop on Petri Nets and Performance Models (PNPM) 1989, Kyoto, Japan, IEEE Computer Society Press, December 1989, 142–151.

[16] G. Ciardo, J.K. Muppala, K.S. Trivedi, *On the solution of GSPN reward models*, Performance Evaluation, **12**:4 (1991), 237–253. Zbl 0754.60097

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

[18] H.M. Hanish, *Analysis of place/transition nets with timed-arcs and its application to batch process control*, Lecture Notes in Computer Science, **691** (1993), 282–299.

[19] B.R. Haverkort, *Markovian models for performance and dependability evaluation*, Lecture Notes in Computer Science, **2090** (2001), 38–83. Zbl 0990.68020

[20] H. Hermanns, M. Rettelbach, *Syntax, semantics, equivalences and axioms for MTIPP*, Proc. 2^{nd} Int. Workshop on Process Algebras and Performance Modelling (PAPM) 1994 (U. Herzog, M. Rettelbach, eds.), Regensberg / Erlangen, Germany, July 1994, Arbeitsberichte des IMMD, **27**:4 (1994), 71–88. http://ftp.informatik.uni-erlangen.de/local/inf7/papers/Hermanns/syntax_semantics_equivalences_axioms_for_MTIPP.ps.gz

[21] J. Hillston, *A compositional approach to performance modelling*, Cambridge University Press, Cambridge, UK, 1996. http://www.dcs.ed.ac.uk/pepa/book.pdf MR1427945

[22] C.A.R. Hoare, *Communicating sequential processes*, Prentice-Hall, London, UK, 1985. http://www.usingcsp.com/cspbook.pdf MR0805324

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

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

[25] V.G. Kulkarni, *Modeling and analysis of stochastic systems*, Texts in Statistical Science, **84**, Chapman and Hall / CRC Press, 2010. Zbl 1191.60003

[26] H. Macià, V. Valero, D.C. Cazorla, F. Cuartero, *Introducing the iteration in sPBC*, Lecture Notes in Computer Science, **3235** (2004), 292–308. Zbl 1110.68420

[27] H. Macià, V. Valero, F. Cuartero, D. de Frutos, *A congruence relation for sPBC*, Formal Methods in System Design, **32**:2 (2008), 85–128. Zbl 1138.68040

[28] H. Macià, V. Valero, F. Cuartero, M.C. Ruiz, *sPBC: a Markovian extension of Petri box calculus with immediate multiactions*, Fundamenta Informaticae, **87**:3–4 (2008), 367–406. Zbl 1154.68092

[29] H. Macià, V. Valero, F. Cuartero, M.C. Ruiz, I.V. Tarasyuk, *Modelling a video conference system with sPBC*, Applied Mathematics and Information Sciences **10**:2 (2016), 475–493.

[30] H. Macià, V. Valero, D. de Frutos, *sPBC: a Markovian extension of finite Petri box calculus*, Proc. 9^{th} IEEE Int. Workshop on Petri Nets and Performance Models (PNPM) 2001, Aachen, Germany, 207–216, IEEE Computer Society Press, 2001. http://www.info-ab.uclm.es/retics/publications/2001/pnpm01.ps

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

[32] O. Marroquín, D. de Frutos, *TPBC: timed Petri box calculus*, Technical Report, Departamento de Sistemas Infofmáticos y Programación, Universidad Complutense de Madrid, Spain, 2000 (in Spanish).

[33] O. Marroquín, D. de Frutos, *Extending the Petri box calculus with time*, Lecture Notes in Computer Science, **2075** (2001), 303–322. Zbl 0986.68082

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

[35] M.A. Marsan, G. Balbo, G. Conte, S. Donatelli, G. Franceschinis, *Modelling with generalised stochastic Petri nets*, Wiley Series in Parallel Computing, John Wiley and Sons, 1995. http://www.di.unito.it/~greatspn/GSPN-Wiley/ Zbl 0843.68080

[36] Ph.M. Merlin, D.J. Farber, *Recoverability of communication protocols: implications of a theoretical study*, IEEE Transactions on Communications, **24**:9 (1976), 1036–1043. Zbl 0362.68096

[37] R.A.J. Milner, *Communication and concurrency*, Prentice-Hall, Upper Saddle River, NJ, USA, 1989. Zbl 0683.68008

[38] M.K. Molloy, *On the integration of the throughput and delay measures in distributed processing models*, Ph.D. thesis, Report, **CSD-810-921**, 108 p., University of California, Los Angeles, USA, 1981.

[39] M.K. Molloy, *Discrete time stochastic Petri nets*, IEEE Transactions on Software Engineering, **11**:4 (1985), 417–423. MR0788999

[40] T.N. Mudge, H.B. Al-Sadoun, *A semi-Markov model for the performance of multiple-bus systems*, IEEE Transactions on Computers, **C-34(10)** (1985), 934–942.

[41] A. Niaouris, *An algebra of Petri nets with arc-based time restrictions*, Lecture Notes in Computer Science, **3407** (2005), 447–462. Zbl 1109.68076

[42] A. Niaouris, M. Koutny, *An algebra of timed-arc Petri nets*, Technical Report, **CS-TR-895**, 60 p., School of Computer Science, University of Newcastle upon Tyne, UK, 2005. http://www.cs.ncl.ac.uk/publications/trs/papers/895.pdf

[43] C. Ramchandani, *Performance evaluation of asynchronous concurrent systems by timed Petri nets*, Ph.D. thesis, Department of Electrical Engineering, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA, 1973.

[44] S.M. Ross, *Stochastic processes*, John Wiley and Sons, New York, USA, 1996. MR1373653

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

[46] I.V. Tarasyuk, *Iteration in discrete time stochastic Petri box calculus*, Bulletin of the Novosibirsk Computing Center, Series Computer Science, IIS Special Issue, **24** (2006), 129–148. Zbl 1249.68132

[47] I.V. Tarasyuk, *Stochastic Petri box calculus with discrete time*, Fundamenta Informaticae, **76**:1–2 (2007), 189–218. MR2293057

[48] I.V. Tarasyuk, *Equivalence relations for modular performance evaluation in dtsPBC*, Mathematical Structures in Computer Science, **24**:1 (2014), e240103. MR3183269

[49] I.V. Tarasyuk, *Discrete time stochastic and deterministic Petri box calculus dtsdPBC*, Siberian Electronic Mathematical Reports, **17** (2020), 1598–1679. Zbl 1448.68352

[50] I.V. Tarasyuk, H. Macià, V. Valero, *Discrete time stochastic Petri box calculus with immediate multiactions*, Technical Report, **DIAB-10-03-1**, 25 p., Department of Computer Systems, High School of Computer Science Engineering, University of Castilla - La Mancha, Albacete, Spain, 2010. http://www.dsi.uclm.es/descargas/technicalreports/DIAB-10-03-1/dtsipbc.pdf

[51] I.V. Tarasyuk, H. Macià, V. Valero, *Discrete time stochastic Petri box calculus with immediate multiactions dtsiPBC*, Proc. 6^{th} Int. Workshop on Practical Applications of Stochastic Modelling (PASM) 2012 and 11^{th} Int. Workshop on Parallel and Distributed Methods in Verification (PDMC) 2012 (J. Bradley, K. Heljanko, W. Knottenbelt, N. Thomas, eds.), London, UK, 2012, Electronic Notes in Theoretical Computer Science, **296** (2013), 229–252.

[52] I.V. Tarasyuk, H. Macià, V. Valero, *Performance analysis of concurrent systems in algebra dtsiPBC*, Programming and Computer Software, **40**:5 (2014), 229–249.

[53] I.V. Tarasyuk, H. Macià, V. Valero, *Stochastic equivalence for performance analysis of concurrent systems in dtsiPBC*, Siberian Electronic Mathematical Reports, **15** (2018), 1743–1812. Zbl 1414.60062

[54] V. Valero, M.E. Cambronero, *Using unified modelling language to model the publish/subscribe paradigm in the context of timed Web services with distributed resources*, Mathematical and Computer Modelling of Dynamical Systems, **23**:6 (2017), 570–594.

[55] R. Zijal, *Discrete time deterministic and stochastic Petri nets*, Proc. Int. Workshop on Quality of Communication-Based Systems 1994, Technical University of Berlin, Germany, 123–136, Kluwer Academic Publishers, 1995. Zbl 0817.68111

[56] R. Zijal, *Analysis of discrete time deterministic and stochastic Petri nets*, Ph.D. thesis, Technical University of Berlin, Germany, 1997.

[57] R. Zijal, G. Ciardo, *Discrete deterministic and stochastic Petri nets*, ICASE Report, **96-72**, 23 p., Institute for Computer Applications in Science and Engineering (ICASE), NASA, Langley Research Centre, Hampton, VA, USA, 1996. http://www.cs.odu.edu/~mln/ltrs-pdfs/icase-1996-72.pdf, http://www.dtic.mil/dtic/tr/fulltext/u2/a322409.pdf

[58] R. Zijal, G. Ciardo, G. Hommel, *Discrete deterministic and stochastic Petri nets*, Proc. 9^{th} ITG/GI Professional Meeting on Measuring, Modeling and Evaluation of Computer and Communication Systems (MMB) 1997 (K. Irmscher, Ch. Mittasch, K. Richter, eds.), Freiberg, Germany, 1997, Vol. **1**, 103–117, VDE-Verlag, Berlin, Germany, 1997. http://www.cs.ucr.edu/~ciardo/pubs/1997MMB-DDSPN.pdf

[59] R. Zijal, R. German, *A new approach to discrete time stochastic Petri nets*, Proc. 11^{th} Int. Conf. on Analysis and Optimization of Systems, Discrete Event Systems (DES) 1994 (G. Cohen, J.-P. Quadrat, eds.), Sophia-Antipolis, France, 1994, Lecture Notes in Control and Information Sciences, **199** (1994), 198–204.

[60] A. Zimmermann, *Modeling and evaluation of stochastic Petri nets with TimeNET 4.1*, Proc. 6^{th} Int. ICST Conf. on Performance Evaluation Methodologies and Tools (VALUETOOLS)

2012 (B. Gaujal, A. Jean-Marie, E. Jorswieck, A. Seuret, eds.), Cargèse, France, October 2012, 1–10, IEEE Computer Society Press, 2012. https://www.tu-ilmenau.de/fileadmin/public/sse/Veroeffentlichungen/2012/VALUETOOLS2012.pdf

[61] A. Zimmermann, J. Freiheit, R. German, G. Hommel, *Petri net modelling and performability evaluation with TimeNET 3.0*, Lecture Notes in Computer Science, **1786** (2000), 188–202. Zbl 0970.68665

[62] A. Zimmermann, J. Freiheit, G. Hommel, *Discrete time stochastic Petri nets for modeling and evaluation of real-time systems*, Proc. 9^{th} Int. Workshop on Parallel and Distributed Real Time Systems (WPDRTS) 2001, San Francisco, USA, 282–286, 2001. http://pdv.cs.tu-berlin.de/~azi/texte/WPDRTS01.pdf

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