СИБИРСКИЕ ЭЛЕКТРОННЫЕ МАТЕМАТИЧЕСКИЕ ИЗВЕСТИЯ

STOCHASTIC EQUIVALENCE FOR PERFORMANCE ANALYSIS OF CONCURRENT SYSTEMS IN DTSIPBC

I.V. TARASYUK, H. MACIÀ, V. VALERO

Abstract. We propose an extension with immediate multiactions of discrete time stochastic Petri Box Calculus (dtsPBC), presented by I.V. Tarasyuk. The resulting algebra dtsiPBC is a discrete time analogue of stochastic Petri Box Calculus (sPBC) with immediate multiactions, designed by H. Macià, V. Valero et al. within a continuous time domain. The step operational semantics is constructed via labeled probabilistic transition systems. The denotational semantics is based on labeled discrete time stochastic Petri nets with immediate transitions. To evaluate performance, the corresponding semi-Markov chains are analyzed. We define step stochastic bisimulation equivalence of expressions that is applied to reduce their transition systems and underlying semi-Markov chains while preserving the functionality and performance characteristics. We explain how this equivalence can be used to simplify performance analysis of the algebraic processes. In a case study, a method of modeling, performance evaluation and behaviour reduction for concurrent systems is outlined and applied to the shared memory system.

Keywords: stochastic process algebra, Petri box calculus, discrete time, immediate multiaction, operational and denotational semantics, semi-Markov chain, performance evaluation, stochastic equivalence, reduction.

1. Introduction

Algebraic process calculi like CSP [47], ACP [8] and CCS [67] 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 [44], PEPA [46] and EMPA [14, 13, 9]. 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.** PAs specify concurrent systems in a compositional way via an expressive formal syntax. On the other hand, Petri nets (PNs) provide a graphical representation of such systems and capture explicit asynchrony in their behaviour. To combine the advantages of both models, a semantics of algebraic formulas in terms of PNs has been defined. Petri Box Calculus (PBC) [15, 17, 16] 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 some invisible activity. In contrast to CCS, synchronization is separated from parallelism (*concurrent constructs*). Synchronization is a unary multi-way stepwise operation, based on communication of actions and their conjugates. This extends the CCS approach with conjugate matching labels. Synchronization in PBC is asynchronous, unlike that in Synchronous CCS (SCCS) [67]. Other operations are sequence and choice (*sequential constructs*). The calculus includes also restriction and relabeling (*abstraction constructs*). To specify infinite processes, refinement, recursion and iteration operations were added (*hierarchical constructs*). Thus, unlike CCS, PBC has an additional iteration operation to specify infinite behaviour when the semantic interpretation in finite PNs is possible. PBC has a step operational semantics in terms of labeled transition systems, based on the rules of structural operational semantics (SOS) [74]. 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). Note that we do not reason in terms of a big-step (natural) [51] or small-step (structural) [74] operational semantics here, and that PBC (and all its extensions to be mentioned further) have a small-step operational semantics, in that terminology. 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 [15, 16].

1.2. **Stochastic extensions of Petri Box Calculus.** A stochastic extension of PBC, called stochastic Petri Box Calculus (sPBC), was proposed in [61]. 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. Just a finite part of PBC was initially used for the stochastic enrichment, i.e. in its former version sPBC does not have refinement, recursion or iteration operations. 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 [65, 5] and called stochastic Petri boxes (s-boxes). In [58], the iteration operator was added to sPBC. In sPBC with iteration, performance of the processes is evaluated by analyzing their underlying continuous time Markov chains (CTMCs). In [59], a number of new equivalence relations were proposed for regular terms of sPBC with iteration to choose later a suitable candidate for a congruence. sPBC with iteration was enriched with immediate multiactions having zero delay in [60]. We call such an sPBC extension generalized sPBC or 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 [65, 5, 6] and called generalized stochastic Petri boxes (gs-boxes). The performance analysis in gsPBC is based on the semi-Markov chains (SMCs).

PBC has a step operational semantics, whereas sPBC has an interleaving one. Remember that in step semantics, parallel executions of activities (steps) are permitted while in interleaving semantics, we can execute only single activities. Hence, a stochastic extension of PBC with a step semantics is needed to keep the concurrency degree of behavioural analysis at the same level as in PBC. As mentioned in [69, 70], in contrast to continuous time approach (used in sPBC), discrete time approach allows for constructing models of common clock systems and clocked devices. In such models, multiple transition firings (or executions of multiple activities) at time moments (ticks of the central clock) are possible, resulting in a step semantics. Moreover, employment of discrete stochastic time fills the gap between the models with deterministic (fixed) time delays and those with continuous stochastic time delays. As argued in [1], arbitrary delay distributions are much easier to handle in a discrete time domain. In [63, 64, 62], discrete stochastic time was preferred to enable simultaneous expiration of multiple delays. In [79, 81], 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 [69, 70] and called discrete time stochastic Petri boxes (dts-boxes). A variety of stochastic equivalences were proposed to identify stochastic processes with similar behaviour which are differentiated by the semantic equivalence. The interrelations of all the introduced equivalences were studied. In [80, 82], we constructed an enrichment of dtsPBC with the iteration operator used to specify infinite processes. The performance evaluation in dtsPBC with iteration is accomplished via the underlying discrete time Markov chains (DTMCs) of the algebraic processes. Since dtsPBC has a discrete time semantics and geometrically distributed sojourn time in the process states, unlike sPBC with continuous time semantics and exponentially distributed delays, the calculi apply two different approaches to the stochastic extension of PBC, in spite of some similarity of their syntax and semantics inherited from PBC. The main advantage of dtsPBC is

that concurrency is treated like in PBC having step semantics, whereas in sPBC parallelism is simulated by interleaving, obliging one to collect the information on causal independence of activities before constructing the semantics. In [83, 84, 85], we presented the extension dtsiPBC of the latter calculus with immediate multiactions. Immediate multiactions increase the specification capability: they can model logical conditions, probabilistic branching, instantaneous probabilistic choices and activities whose durations are negligible in comparison with those of others. They are also used to specify urgent activities and the ones that are not relevant for performance evaluation. Thus, immediate multiactions can be considered as a kind of instantaneous dynamic state adjustment and, in many cases, they result in a simpler and more clear system representation.

1.3. **Equivalence relations.** A notion of equivalence is important in theory of computing systems. Equivalences are applied both to compare behaviour of systems and reduce their structure. There is a wide diversity of behavioural equivalences, and their interrelations are well explored in the literature. The best-known and widely used one is bisimulation. Typically, the mentioned equivalences take into account only functional (qualitative) but not performance (quantitative) aspects. Additionally, the equivalences are usually interleaving ones, i.e. they interpret concurrency as a sequential nondeterminism. Interleaving equivalences permit to imitate parallel execution of actions via all possible occurrence sequences (interleavings) of them. Step equivalences require instead simulating such a parallel execution by simultaneous occurrence (step) of all the involved actions. To respect quantitative features of behaviour, probabilistic equivalences have additional requirement on execution probabilities. Two equivalent processes must be able to execute the same sequences of actions, and for every such sequence, its execution probabilities within both processes should coincide. In case of probabilistic bisimulation equivalence, the states from which similar future behaviours start are grouped into equivalence classes that form elements of the aggregated state space. From every two bisimilar states, the same actions can be executed, and the subsequent states resulting from execution of an action belong to the same equivalence class. In addition, for both states, the cumulative probabilities to move to the same equivalence class by executing the same action coincide. A different kind of quantitative relations is called Markovian equivalences, which take rate (the parameter of exponential distribution that governs time delays) instead of probability. The probabilistic equivalences can be seen as discrete time analogues of the Markovian ones, since the latter are defined as the continuous time relations.

Interleaving probabilistic weak trace equivalence was introduced in [31] on labeled probabilistic transition systems. Interleaving probabilistic strong bisimulation equivalence was proposed in [57] on the same model. Interleaving probabilistic equivalences were defined for probabilistic processes in [50, 38]. Interleaving Markovian weak bisimulation equivalences were considered in [25] on Markovian process algebras, in [27] on labeled CTSPNs and in [28] on labeled GSPNs. Interleaving Markovian strong bisimulation equivalence was constructed in [44] for MTIPP, in [46] for PEPA and in [14, 13, 9] for EMPA. In [10, 11], interleaving Markovian trace, test, strong and weak bisimulation equivalences were compared on sequential and concurrent Markovian process calculi. However, no appropriate equivalence was defined for concurrent SPAs. The non-interleaving bisimulation equivalence in GSMPA [21, 19] takes ST-semantics for action particles and in Sπ [76] it uses a sophisticated labeling.

1.4. **Our contributions.** We present dtsPBC with iteration extended with immediate multiactions, called *discrete time stochastic and immediate Petri Box Calculus* (dtsiPBC), which is a discrete time analog of sPBC. The latter calculus has iteration and immediate multiactions within the context of a continuous time domain. The step operational semantics is constructed with the use of labeled probabilistic transition systems. The denotational semantics is defined in terms of a subclass of labeled discrete time stochastic and immediate PNs (LDTSPNs with immediate transitions, LDTSIPNs), based on the extension of DTSPNs with transition labeling and immediate transitions, called dtsi-boxes. The consistency of both semantics is demonstrated. The corresponding stochastic process, the underlying SMC, is constructed and investigated, with the purpose of performance evaluation, which is the same for both semantics. In addition, the alternative solution methods are developed, based on the underlying DTMC. Further, we propose step stochastic bisimulation equivalence allowing one to identify algebraic processes with similar behaviour that are however differentiated by the semantics of the calculus. We examine the interrelations of the proposed relation with other equivalences of the algebra. We describe how step stochastic bisimulation equivalence can be used to reduce transition systems of expressions and their underlying SMCs while preserving the qualitative and the quantitative characteristics. We prove that the mentioned equivalence guarantees identity of the stationary behaviour and the residence time properties in the equivalence classes. This implies coincidence of performance indices, based on steady-state probabilities of the modeled stochastic systems. The equivalences possessing the property can be used to reduce the state space of a system and thus simplify its performance evaluation, which is usually a complex problem due to the state space explosion. We present a case study of a system with two processors and a common shared memory explaining how to model concurrent systems within the calculus and analyze their performance, as well as how to reduce the systems behaviour while preserving their performance indices and making easier the performance evaluation. Finally, we consider differences and similarities between dtsiPBC and other SPAs to determine the advantages of our calculus. The salient point of dtsiPBC is a combination of immediate multiactions, discrete stochastic time and step semantics in an SPA.

Concerning differences from our previous papers about dtsiPBC [83, 84, 85], the present text is much more detailed and many new important results have been added. In particular, immediate multiactions now have positive real-valued weights (instead of previously used positive integer weights), all the used notions (such as numbering, functions collecting executable activities, probability functions) are formally defined and completely explained with examples; the operational and denotational semantics are given in full detail (the inaction, action rules, LDTSPNs and dtsi-boxes are extensively described and discussed); compact illustrative examples (of standard and alternative solution methods) are presented; keeping properties of original Markov chains (irreducibility, positive recurrence and aperiodicity) in their embedded and state-aggregated versions is studied. The main new contribution of the paper, step stochastic bisimulation equivalence of the process expressions, is introduced and checked for stationary behaviour and sojourn time features preservation in the equivalence classes; quotienting the transition systems, SMCs and DTMCs by the equivalence, as well as the resulting simplification of performance evaluation, are considered; generalized variant of the shared memory system and quotients of

its behaviour by the equivalence are constructed. In the enhanced related work overview, strong points of dtsiPBC with respect to other SPAs are detected; in the extensive discussion, analytical solution, concurrency interpretation, application area and general advantages of dtsiPBC are explained.

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

- Expressive discrete time SPA with immediate activities called dtsiPBC.
- Step operational semantics via labeled probabilistic transition systems.
- Denotational semantics via discrete time stochastic-immediate Petri nets.
- Performance analysis with the underlying SMCs and DTMCs of expressions.
- Stochastic equivalence for reduction preserving functionality-performance.
- Extended case study showing how to apply the theoretical results in practice.

1.5. **Structure of the paper.** In Section 2, the syntax of the calculus dtsiPBC is presented. In Section 3, we construct the operational semantics of the algebra in terms of labeled probabilistic transition systems. In Section 4, we propose the denotational semantics, based on a subclass of LDTSIPNs. In Section 5, the corresponding stochastic process is derived and analyzed. Step stochastic bisimulation equivalence is defined and investigated in Section 6. In Section 7, we explain how to reduce transition systems and underlying SMCs of process expressions modulo the equivalence. In Section 8, this equivalence is applied to the stationary behaviour comparison in the equivalence classes to verify the performance preservation. In Section 9, the generalized shared memory system is presented as a case study. The difference between dtsiPBC and other well-known SPAs is considered in Section 10. The advantages of dtsiPBC with respect to other SPAs are described in Section 11. Section 12 summarizes the results obtained and outlines the research perspectives.

2. Syntax

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

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

We denote the *set of all finite multisets* over a set X by $\mathbb{N}_{\text{fin}}^X$. Let $M, M' \in \mathbb{N}_{\text{fin}}^X$. 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 $\mathcal{L} = \mathbb{N}_{\text{fin}}^{\mathcal{A}}$ 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 with no visible actions. The *alphabet* of $\alpha \in \mathcal{L}$ is defined as $\mathcal{A}(\alpha) = \{x \in \mathcal{A} \mid \alpha(x) > 0\}$.

A *stochastic multiaction* is a pair (α, ρ), where $\alpha \in \mathcal{L}$ and $\rho \in (0; 1)$ is the *probability* of the multiaction α. This probability is interpreted as that of independent execution of the stochastic multiaction at the next discrete time moment. Such probabilities are used to calculate those to execute (possibly empty) sets of stochastic multiactions after one time unit delay. The probabilities of stochastic

multiactions are required not to be equal to 1 to avoid extra model complexity, since in this case weights would be required to make a choice when several stochastic multiactions with probability 1 can be executed from a state. Furthermore, stochastic multiactions with probability 1 would occur in a step (parallel execution) and all other with the less probabilities do not. In this case, some problems appear with conflicts resolving. See [69, 70] for the discussion on SPNs. On the other hand, there is no sense to allow zero probabilities of stochastic multiactions, since they would never be performed in this case. Let $\mathcal{SL}$ be the set of *all stochastic multiactions*.

An *immediate multiaction* is a pair $(\alpha, \natural_l)$, where $\alpha \in \mathcal{L}$ 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 immediate multiaction at the current discrete time moment. Such weights are used to calculate the probabilities to execute sets of immediate multiactions instantly. Immediate multiactions have a priority over stochastic ones. Thus, in a state where both kinds of multiactions can occur, immediate multiactions always occur before stochastic ones. Stochastic and immediate multiactions cannot participate together in some step (concurrent execution), i.e. the steps consisting only of immediate multiactions or those including only stochastic multiactions are allowed. Let $\mathcal{IL}$ be the set of *all immediate multiactions*.

Note that the same multiaction $\alpha \in \mathcal{L}$ may have different probabilities and weights in the same specification. An *activity* is a stochastic or immediate multiaction. Let $\mathcal{SIL} = \mathcal{SL} \cup \mathcal{IL}$ be the set of *all activities*. The *alphabet* of a multiset of activities $\Upsilon \in \mathbb{N}_{\mathrm{fin}}^{\mathcal{SIL}}$ is defined as $\mathcal{A}(\Upsilon) = \cup_{(\alpha,\kappa)\in\Upsilon}\mathcal{A}(\alpha)$. For an activity $(\alpha, \kappa) \in \mathcal{SIL}$, we define its *multiaction part* as $\mathcal{L}(\alpha, \kappa) = \alpha$ and its *probability* or *weight part* as $\Omega(\alpha, \kappa) = \kappa$ if $\kappa \in (0; 1)$; or $\Omega(\alpha, \kappa) = l$ if $\kappa = \natural_l$, $l \in \mathbb{R}_{>0}$. The *multiaction part* of a multiset of activities $\Upsilon \in \mathbb{N}_{\mathrm{fin}}^{\mathcal{SIL}}$ is defined as $\mathcal{L}(\Upsilon) = \sum_{(\alpha,\kappa)\in\Upsilon} \alpha$.

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

Sequential execution and choice have a standard interpretation, like in other process algebras, but parallelism does not include synchronization, unlike the operation in CCS [67].

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 as usual: for $\alpha \in \mathcal{L}$, we define $f(\alpha) = \sum_{x\in\alpha} f(x)$. Relabeling is extended to the multisets of activities as follows: for $\Upsilon \in \mathbb{N}_{\mathrm{fin}}^{\mathcal{SIL}}$, we define $f(\Upsilon) = \sum_{(\alpha,\kappa)\in\Upsilon}(f(\alpha), \kappa)$.

Restriction over an elementary action $a \in Act$ means that, for a given expression, any process 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 $(\alpha \oplus_a \beta)(x) = \begin{cases} \alpha(x) + \beta(x) - 1, & \text{if } x = a \text{ or } x = \hat{a}; \\ \alpha(x) + \beta(x), & \text{otherwise.} \end{cases}$ In other words, we require that $\alpha \oplus_a \beta = \alpha + \beta - \{a, \hat{a}\}$, since the synchronization of a and $\hat{a}$ produces $\emptyset$. Activities are synchronized by their multiaction parts, i.e. the synchronization on a of two activities, whose multiaction parts α and β possess the above properties, results in the activity with the multiaction part $\alpha \oplus_a \beta$. We may synchronize activities of the same type only: either both stochastic multiactions or

both immediate ones, since immediate multiactions have a priority over stochastic ones, hence, stochastic and immediate multiactions cannot be executed together (note also that the execution of immediate multiactions takes no time, unlike that of stochastic ones). Synchronization on a means that, for a given expression with a process behaviour containing two concurrent activities that can be synchronized on a, there exists also the process behaviour that differs from the former only in that the two activities are replaced by the result of their synchronization.

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

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

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

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

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

To avoid technical difficulties with the iteration operator, we should not allow any concurrency at the highest level of the second argument of iteration. This is not a severe restriction, since we can always prefix parallel expressions by an activity with the empty multiaction part. In [82], we have demonstrated that relaxing the restriction can result in nets which are not safe. Alternatively, we can use a different, safe, version of the iteration operator, but its net translation has six arguments [16].

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

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

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

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

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

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

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

If the underlying static expression of a dynamic one is not regular, the corresponding LDTSIPN can be non-safe (but it is 2-bounded in the worst case [16]).

Definition 5. *A dynamic expression is* regular *if its underlying static one is so.*

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

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 these 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 LDTSIPN, hence, its current marking remains unchanged. An application of every inaction rule does not need a discrete time delay, i.e. the dynamic expression transformation described by the rule is accomplished instantly.

Table 1 defines inaction rules for regular dynamic expressions in the form of overlined and underlined static ones, where $E, F, K \in RegStatExpr$ and $a \in Act$.

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

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

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

TABLE 2. Inaction rules for arbitrary regular dynamic expressions

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

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

Definition 7. *The relation* $\approx\ =\ (\Rightarrow \cup \Leftarrow)^*$ *is a* structural equivalence *of dynamic expressions in dtsiPBC. Thus, two dynamic expressions G and G' are* structurally equivalent*, denoted by $G \approx G'$, if they can be reached from one another by applying the inaction rules in a forward or backward direction.*

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

The action rules with stochastic (or immediate, otherwise) multiactions describe dynamic expression transformations in the form of $G \xrightarrow{\Gamma} \widetilde{G}$ (or $G \xrightarrow{I} \widetilde{G}$) due to execution of non-empty multisets Γ of stochastic (or I of immediate) multiactions. The rules represent possible state changes of the specified processes when some non-empty multisets of stochastic (or immediate) multiactions are executed. As we shall see, the application of an action rule with stochastic (or immediate) multiactions to a dynamic expression leads in the corresponding LDTSIPN to a discrete time tick at which some stochastic transitions fire (or to the instantaneous firing of some immediate transitions) and possible change of the current marking. The current marking remains unchanged only if there is a self-loop produced by the iterative execution of a non-empty multiset, which must be one-element, i.e. the single stochastic (or immediate) multiaction. The reason is the regularity requirement that allows no concurrency at the highest level of the second argument of iteration.

The empty loop rule (applicable only when no immediate multiactions can be executed from the current state) describes dynamic expression transformations in the form of $G \xrightarrow{\emptyset} G$ due to execution of the empty multiset of activities at a discrete time tick. The rule reflects a non-zero probability to stay in the current state at the next moment, which is a feature of discrete time stochastic processes. As we shall see, the application of the empty loop rule to a dynamic expression leads to a discrete time tick in the corresponding LDTSIPN at which no transitions fire and the current marking is not changed. This is a new rule with no prototype among inaction rules of PBC, since it represents a time delay, but PBC has no notion of time. The PBC rule $G \xrightarrow{\emptyset} G$ from [17, 16] in our setting would correspond to a rule $G \Rightarrow G$ that describes staying in the current state when no time elapses. Since we do not need the latter rule to transform dynamic expressions into operative ones and it can destroy the definition of operative expressions, we do not have it.

Thus, an application of every action rule with stochastic multiactions or the empty loop rule requires one discrete time unit delay, i.e. the execution of a (possibly empty) multiset of stochastic multiactions leading to the dynamic expression transformation described by the rule is accomplished after one time unit. However, an application of every action rule with immediate multiactions does not take any
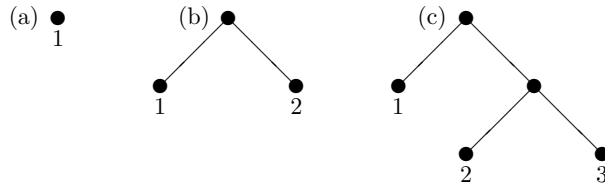
FIG. 1. The binary trees encoded with the numberings 1, (1)(2) and (1)((2)(3))

time, i.e. the execution of a (non-empty) multiset of immediate multiactions is accomplished instantly at the current time.

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

Definition 8. *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.

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

The new activities resulting from synchronizations in different orders should be considered up to 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

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

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

Let X be some set. We denote the Cartesian product $X \times X$ by X^2. Let $\mathcal{E} \subseteq X^2$ be an equivalence relation on X. Then the *equivalence class* (w.r.t. $\mathcal{E}$) 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 w.r.t. the structural equivalence. G is an *initial* dynamic expression, denoted by $init(G)$, if $\exists E \in RegStatExpr$, $G \in [\overline{E}]_{\approx}$. G is a *final* dynamic expression, denoted by $final(G)$, if $\exists E \in RegStatExpr$, $G \in [\underline{E}]_{\approx}$.

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

(1) *If* $final(G)$ *then* $Can(G) = \emptyset$.

(2) *If* $G = \overline{(\alpha, \kappa)}$ *then* $Can(G) = \{\{(\alpha, \kappa)\}\}$.

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

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

(5) *If* $\Upsilon \in Can(G \text{ sy } a)$ *and* $(\alpha, \kappa), (\beta, \lambda) \in \Upsilon$ *are different activities such that* $a \in \alpha$, $\hat{a} \in \beta$, *then*

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

    (b) $(\Upsilon + \{(\alpha \oplus_a \beta, \natural_{l+m})\}) \setminus \{(\alpha, \kappa), (\beta, \lambda)\} \in Can(G \text{ sy } a)$ *if* $\kappa = \natural_l$, $\lambda = \natural_m$, $l, m \in \mathbb{R}_{>0}$.

    When we synchronize the same set of activities in different orders, we obtain several activities with the same multiaction and probability or weight parts, but with different numberings having the same content. Then, we only consider a single one of the resulting activities. For example, the synchronization of stochastic multiactions $(\alpha, \rho)_1$ *and* $(\beta, \chi)_2$ *in different orders generates the activities* $(\alpha \oplus_a \beta, \rho \cdot \chi)_{(1)(2)}$ *and* $(\beta \oplus_a \alpha, \chi \cdot \rho)_{(2)(1)}$. *Similarly, the synchronization of immediate multiactions* $(\alpha, \natural_l)_1$ *and* $(\beta, \natural_m)_2$ *in different orders generates the activities* $(\alpha \oplus_a \beta, \natural_{l+m})_{(1)(2)}$ *and* $(\beta \oplus_a \alpha, \natural_{m+l})_{(2)(1)}$. *Since* $Cont((1)(2)) = \{1, 2\} = Cont((2)(1))$, *in both cases, only the first activity (or the second one) resulting from synchronization will appear in a set from* $Can(G \text{ sy } a)$.

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

Let $G \in OpRegDynExpr$ and $Can(G) \neq \emptyset$. Obviously, if there are only stochastic (or only immediate) multiactions in the sets from $Can(G)$ then these stochastic (or immediate) multiactions can be executed from G. Otherwise, besides stochastic ones, there are also immediate multiactions in the sets from $Can(G)$. By the note above, there are non-empty sets of immediate multiactions in $Can(G)$ as well, i.e. $\exists \Upsilon \in Can(G)$, $\Upsilon \in \mathbb{N}_{\text{fin}}^{\mathcal{IL}} \setminus \{\emptyset\}$. Then no stochastic multiactions can be executed from G, even if $Can(G)$ contains non-empty sets of stochastic multiactions, since immediate multiactions have a priority over stochastic ones.

Definition 10. *Let* $G \in OpRegDynExpr$. *The* set of all non-empty sets of activities which can be executed from G *is*
$$Now(G) = \begin{cases} Can(G), & \text{if } (Can(G) \subseteq \mathbb{N}_{\text{fin}}^{\mathcal{SL}} \setminus \{\emptyset\}) \vee (Can(G) \subseteq \mathbb{N}_{\text{fin}}^{\mathcal{IL}} \setminus \{\emptyset\}); \\ Can(G) \cap \mathbb{N}_{\text{fin}}^{\mathcal{IL}}, & \text{otherwise.} \end{cases}$$

An operative dynamic expression $G \in OpRegDynExpr$ is *tangible*, denoted by $tang(G)$, if $Now(G) \subseteq \mathbb{N}_{\text{fin}}^{\mathcal{SL}} \setminus \{\emptyset\}$. In particular, we have $tang(G)$, if $Now(G) = \emptyset$. Otherwise, the expression G is *vanishing*, denoted by $vanish(G)$, and in this case $\emptyset \neq Now(G) \subseteq \mathbb{N}_{\text{fin}}^{\mathcal{IL}} \setminus \{\emptyset\}$. For a (possibly non-operative) dynamic expression $G \in RegDynExpr$, we write $tang(G)$ $(vanish(G))$, if $\forall H \in [G]_{\approx} \cap OpRegDynExpr$, we have $tang(H)$ $(vanish(H))$. Note that the operative dynamic expressions from $[G]_{\approx}$

may have different types in general. The following example demonstrates operative dynamic expressions H and H' with $H \approx H'$, such that $vanish(H)$, but $tang(H')$.

Example 2. *Let* $G = \overline{(\overline{(\{a\},\natural_1)}[](\{b\},\natural_2))\|\overline{(\{c\},\frac{1}{2})}}$ *and* $G' = ((\{a\},\natural_1)[]\overline{(\{b\},\natural_2)})\|$ $\overline{(\{c\},\frac{1}{2})}$. *Then* $G \approx G'$, *since* $G \Leftarrow G'' \Rightarrow G'$ *for* $G'' = \overline{(\overline{(\{a\},\natural_1)}[](\{b\},\natural_2))\|(\{c\},\frac{1}{2})}$, *but* $Can(G) = \{\{(\{a\},\natural_1)\}, \{(\{c\},\frac{1}{2})\}, \{(\{a\},\natural_1),(\{c\},\frac{1}{2})\}\}$, $Can(G') =$ $\{\{(\{b\},\natural_2)\}, \{(\{c\},\frac{1}{2})\}, \{(\{b\},\natural_2),(\{c\},\frac{1}{2})\}\}$ *and* $Now(G) = \{\{(\{a\},\natural_1)\}\}$, $Now(G') = \{\{(\{b\},\natural_2)\}\}$. *Clearly, we have* $vanish(G)$ *and* $vanish(G')$. *The executions like that of* $\{(\{c\},\frac{1}{2})\}$ *(and all sets including it) from* G *and* G' *must be disabled using preconditions in the action rules, since immediate multiactions have a priority over stochastic ones, hence, the former are always executed first.*

Let $H = \overline{(\{a\},\natural_1)}[](\{b\},\frac{1}{2})$ *and* $H' = (\{a\},\natural_1)[]\overline{(\{b\},\frac{1}{2})}$. *Then* $H \approx H'$, *since* $H \Leftarrow H'' \Rightarrow H'$ *for* $H'' = \overline{(\{a\},\natural_1)}[](\{b\},\frac{1}{2})$, *but* $Can(H) = Now(H) = \{\{(\{a\},\natural_1)\}\}$ *and* $Can(H') = Now(H') = \{\{(\{b\},\frac{1}{2})\}\}$. *We have* $vanish(H)$, *but* $tang(H')$. *To get the action rules correct under structural equivalence, the executions like that of* $\{(\{b\},\frac{1}{2})\}$ *from* H' *must be disabled using preconditions in the action rules, since immediate multiactions have a priority over stochastic ones, hence, the choices are always resolved in favour of the former.*

In Table 3, we define the action and empty loop rules. In this table, $(\alpha,\rho),(\beta,\chi) \in \mathcal{SL}$, $(\alpha,\natural_l),(\beta,\natural_m) \in \mathcal{IL}$ and $(\alpha,\kappa) \in \mathcal{SIL}$. Further, $E, F \in RegStatExpr$, $G, H \in OpRegDynExpr$, $\widetilde{G}, \widetilde{H} \in RegDynExpr$ and $a \in Act$. Moreover, $\Gamma, \Delta \in \mathbb{N}_{\text{fin}}^{\mathcal{SL}} \setminus \{\emptyset\}$, $\Gamma' \in \mathbb{N}_{\text{fin}}^{\mathcal{IL}}$, $I, J \in \mathbb{N}_{\text{fin}}^{\mathcal{IL}} \setminus \{\emptyset\}$, $I' \in \mathbb{N}_{\text{fin}}^{\mathcal{IL}}$ and $\Upsilon \in \mathbb{N}_{\text{fin}}^{\mathcal{SIL}} \setminus \{\emptyset\}$. The first rule is the empty loop rule **El**. The other rules are the action rules, describing transformations of dynamic expressions, which are built using particular algebraic operations. If we cannot merge a rule with stochastic multiactions and a rule with immediate multiactions for some operation then we get the coupled action rules. Then the names of the action rules with immediate multiactions have a suffix "**i**".

Almost all the rules in Table 3 (excepting **El**, **P2**, **P2i**, **Sy2** and **Sy2i**) resemble those of gsPBC [60], but the former correspond to execution of sets of activities, not of single activities, as in the latter, and our rules have simpler preconditions (if any), since all immediate multiactions in dtsiPBC have the same priority level, unlike those of gsPBC. The preconditions in rules **El**, **C**, **P1**, **I2** and **I3** are needed to ensure that (possibly empty) sets of stochastic multiactions are executed only from *tangible* operative dynamic expressions, such that all operative dynamic expressions structurally equivalent to them are tangible as well. For example, if $init(G)$ in rule **C** then $G \approx \overline{F}$ for some static expression F and $G[]E \approx \overline{F}[]E \approx \overline{F[]E} \approx F[]\overline{E}$. Hence, it should be guaranteed that $tang(F[]\overline{E})$, which holds iff $tang(\overline{E})$. The case $E[]G$ is treated similarly. Further, in rule **P1**, assuming that $tang(G)$, it should be guaranteed that $tang(G\|H)$ and $tang(H\|G)$, which holds iff $tang(H)$. The preconditions in rules **I2** and **I3** are analogous to that in rule **C**.

Rule **El** corresponds to one discrete time unit delay while executing no activities and therefore it has no analogues among the rules of gsPBC that adopts the continuous time model. Rules **P2** and **P2i** have no similar rules in gsPBC, since interleaving semantics of the algebra allows no simultaneous execution of activities. **P2** and **P2i** have in PBC the analogous rule **PAR** that is used to construct step semantics of the calculus, but the former two rules correspond to execution of sets of activities, unlike that of multisets of multiactions in the latter rule. Rules **P2** and **P2i** cannot be merged, otherwise simultaneous execution of stochastic and

TABLE 3. Action and empty loop rules

$$\mathbf{El} \; \frac{tang(G)}{G \xrightarrow{\emptyset} G} \qquad \mathbf{B} \; \overline{(\alpha, \kappa)} \xrightarrow{\{(\alpha,\kappa)\}} \underline{(\alpha, \kappa)} \qquad \mathbf{S} \; \frac{G \xrightarrow{\Upsilon} \widetilde{G}}{G; E \xrightarrow{\Upsilon} \widetilde{G}; E, \; E; G \xrightarrow{\Upsilon} E; \widetilde{G}}$$

$$\mathbf{C} \; \frac{G \xrightarrow{\Gamma} \widetilde{G}, \; \neg init(G) \vee (init(G) \wedge tang(\overline{E}))}{G[]E \xrightarrow{\Gamma} \widetilde{G}[]E, \; E[]G \xrightarrow{\Gamma} E[]\widetilde{G}} \qquad \mathbf{Ci} \; \frac{G \xrightarrow{I} \widetilde{G}}{G[]E \xrightarrow{I} \widetilde{G}[]E, \; E[]G \xrightarrow{I} E[]\widetilde{G}}$$

$$\mathbf{P1} \; \frac{G \xrightarrow{\Gamma} \widetilde{G}, \; tang(H)}{G\|H \xrightarrow{\Gamma} \widetilde{G}\|H, \; H\|G \xrightarrow{\Gamma} H\|\widetilde{G}} \qquad \mathbf{P1i} \; \frac{G \xrightarrow{I} \widetilde{G}}{G\|H \xrightarrow{I} \widetilde{G}\|H, \; H\|G \xrightarrow{I} H\|\widetilde{G}}$$

$$\mathbf{P2} \; \frac{G \xrightarrow{\Gamma} \widetilde{G}, \; H \xrightarrow{\Delta} \widetilde{H}}{G\|H \xrightarrow{\Gamma+\Delta} \widetilde{G}\|\widetilde{H}} \qquad \mathbf{P2i} \; \frac{G \xrightarrow{I} \widetilde{G}, \; H \xrightarrow{J} \widetilde{H}}{G\|H \xrightarrow{I+J} \widetilde{G}\|\widetilde{H}}$$

$$\mathbf{L} \; \frac{G \xrightarrow{\Upsilon} \widetilde{G}}{G[f] \xrightarrow{f(\Upsilon)} \widetilde{G}[f]} \qquad \mathbf{Rs} \; \frac{G \xrightarrow{\Upsilon} \widetilde{G}, \; a,\hat{a} \notin \mathcal{A}(\Upsilon)}{G \; \mathsf{rs} \; a \xrightarrow{\Upsilon} \widetilde{G} \; \mathsf{rs} \; a}$$

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

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

$$\mathbf{I3i} \; \frac{G \xrightarrow{I} \widetilde{G}}{[E * F * G] \xrightarrow{I} [E * F * \widetilde{G}]} \qquad \mathbf{Sy1} \; \frac{G \xrightarrow{\Upsilon} \widetilde{G}}{G \; \mathsf{sy} \; a \xrightarrow{\Upsilon} \widetilde{G} \; \mathsf{sy} \; a}$$

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

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

immediate multiactions would be allowed. Rules **Sy2** and **Sy2i** differ from the corresponding synchronization rules in gsPBC, since the probability or the weight of synchronization in the former rules and the rate or the weight of synchronization in the latter rules are calculated in two distinct ways.

Rule **Sy2** establishes that the synchronization of two stochastic multiactions is made by taking the product of their probabilities, since we are considering that both must occur for the synchronization to happen, so this corresponds, in some sense, to the probability of the independent event intersection, but the real situation is more complex, since these stochastic multiactions can also be executed in parallel. Nevertheless, when scoping (the combined operation consisting of synchronization followed by restriction over the same action [16]) is applied over a parallel execution, we get as final result just the simple product of the probabilities, since no normalization is needed there. Multiplication is an associative and commutative binary operation that is distributive over addition, i.e. it fulfills all practical conditions imposed on the synchronization operator in [45]. Further, if

both arguments of multiplication are from $(0; 1)$ then the result belongs to the same interval, hence, multiplication naturally maintains probabilistic compositionality in our model. Our approach is similar to the multiplication of rates of the synchronized actions in MTIPP [44] in the case when the rates are less than 1. Moreover, for the probabilities ρ and χ of two stochastic multiactions to be synchronized we have $\rho\cdot\chi < \min\{\rho, \chi\}$, i.e. multiplication meets the performance requirement stating that the probability of the resulting synchronized stochastic multiaction should be less than the probabilities of the two ones to be synchronized. In terms of performance evaluation, it is usually supposed that the execution of two components together require more system resources and time than the execution of each single one. This resembles the *bounded capacity* assumption from [45]. Thus, multiplication is easy to handle with and it satisfies the algebraic, probabilistic, time and performance requirements. Therefore, we have chosen the product of the probabilities for the synchronization. See also [23, 22] for a discussion about binary operations producing the rates of synchronization in the continuous time setting.

In rule **Sy2i**, we sum the weights of two synchronized immediate multiactions, since the weights can be interpreted as the rewards [78], which we collect. Next, we express that the synchronized execution of immediate multiactions has more importance than that of every single one. The weights of immediate multiactions can also be seen as bonus rewards associated with transitions [12]. The rewards are summed during synchronized execution of immediate multiactions, since in this case all the synchronized activities can be seen as participated in the execution. We prefer to collect more rewards, thus, the transitions providing greater rewards will have a preference and they will be executed with a greater probability. Since execution of immediate multiactions takes no time, we prefer to execute in a step as many synchronized immediate multiactions as possible to get more progress in behaviour. Under behavioural progress we mean an advance in executing activities, which does not always imply a progress in time, as when the activities are immediate multiactions. This aspect will be used later, while evaluating performance via the embedded discrete time Markov chains (EDTMCs) of expressions. Since every state change in EDTMC takes one unit of (local) time, greater advance in operation of the EDTMC allows one to calculate quicker performance indices.

We do not have a self-synchronization, i.e. a synchronization of an activity with itself, since all the (enumerated) activities executed together are considered to be different. This permits to avoid unexpected behaviour and technical difficulties [16].

In Table 4, inaction rules, action rules (with stochastic or immediate multiactions) and empty loop rule are compared according to the three aspects of their application: whether it changes the current state, whether it leads to a time progress, and whether it results in execution of some activities. Positive answers to the questions are denoted by the plus sign while negative ones are specified by the minus sign. If both positive and negative answers can be given to some of the questions in different cases then the plus-minus sign is written. The process states are considered up to structural equivalence of the corresponding expressions, and time progress is not regarded as a state change.

3.3. **Transition systems.** We now construct labeled probabilistic transition systems associated with dynamic expressions to define their operational semantics.

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

TABLE 4. Comparison of inaction, action and empty loop rules

Rules	State change	Time progress	Activities execution
Inaction rules	−	−	−
Action rules with stochastic multiactions	±	+	+
Action rules with immediate multiactions	±	−	+
Empty loop rule	−	+	−

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

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

The set of *all sets of activities executable in s* is defined as $Exec(s) = \{\Upsilon \mid \exists H \in s, \exists \widetilde{H}, H \xrightarrow{\Upsilon} \widetilde{H}\}$. It can be proved by induction on the structure of expressions that $\Upsilon \in Exec(s) \setminus \{\emptyset\}$ implies $\exists H \in s, \Upsilon \in Now(H)$. The reverse statement does not hold in general, as the next example shows.

Example 3. *Let H, H' be from Example 2 and $s = [H]_\approx = [H']_\approx$. We have $Now(H) = \{\{(\{a\}, \natural_1)\}\}$ and $Now(H') = \{\{(\{b\}, \frac{1}{2})\}\}$. Since only rules* **Ci** *and* **B** *can be applied to H, and no action rule can be applied to H', we get $Exec(s) = \{\{(\{a\}, \natural_1)\}\}$. Then, for $H' \in s$ and $\Upsilon = \{(\{b\}, \frac{1}{2})\} \in Now(H')$, we get $\Upsilon \notin Exec(s)$.*

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

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

Since the inaction rules only distribute and move upper and lower bars along the syntax of dynamic expressions, all $H \in s$ have the same underlying static expression F. The action rules **Sy2** and **Sy2i** are the only ones that generate new activities. Since we have a finite number of operators sy in F and all the multiaction parts of the activities are finite multisets, the number of the new synchronized activities is also finite. The action rules contribute to $Exec(s)$ (in addition to the empty set, if rule **El** is applicable) only the sets consisting both of activities from F and the new activities, produced by **Sy2** and **Sy2i**. Since we have a finite number of such activities, the set $Exec(s)$ is finite, hence, summation and multiplication by its elements are well-defined. Similar reasoning can be used to demonstrate that for all dynamic expressions H (not just for those from s), $Now(H)$ is a finite set.

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

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

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

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

If $s \in DR_{\mathrm{T}}(G)$ 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 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 set 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 set of stochastic multiactions Υ in s, i.e. the probability of *intersection* of two events calculated using the conditional probability formula in the form $\mathsf{P}(X \cap Y) = \mathsf{P}(X|Y)\mathsf{P}(Y) = \prod_{(\alpha,\rho) \in \Upsilon} \rho \cdot \prod_{\{\{(\beta,\chi)\} \in Exec(s) | (\beta,\chi) \notin \Upsilon\}} (1 - \chi)$, as shown in [84]. 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 set of activities can be executed in s, i.e. $Exec(s) = \{\emptyset\}$, we take $PF(\emptyset, s) = 1$, since then we stay in s. For $s \in DR_{\mathrm{T}}(G)$ we have $PF(\emptyset, s) \in (0; 1]$, hence, we can stay in s at the next time moment with a certain positive probability.

If $s \in DR_{\mathrm{V}}(G)$ then $PF(\Upsilon, s)$ can be interpreted as the *overall (cumulative)* weight of the immediate multiactions from Υ, i.e. the sum of all their weights. The summation here is used since the weights can be seen as the rewards which are collected [78]. In addition, this means that concurrent execution of the immediate multiactions has more importance than that of every single one. Thus, this reasoning is the same as that used to define the weight of synchronized immediate multiactions in rule **Sy2i**.

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

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

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

If $s \in DR_{\mathrm{T}}(G)$ then $PT(\Upsilon, s)$ can be interpreted as the *conditional* probability to execute Υ in s calculated using the conditional probability formula in the form $\mathsf{P}(Z|W) = \frac{\mathsf{P}(Z \cap W)}{\mathsf{P}(W)} = \frac{PF(\Upsilon,s)}{\sum_{\Xi \in Exec(s)} PF(\Xi,s)}$, as shown in [84]. Note that $PF(\Upsilon, s)$ can be seen as the *potential* probability to execute Υ in s, since we have $PF(\Upsilon, s) = PT(\Upsilon, s)$ only when *all* sets (including the empty one) consisting of the executable stochastic multiactions can be executed in s. In this case, all 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 in parallel, i.e. $\emptyset, \{(\alpha, \rho)\}, \{(\beta, \chi)\} \in Exec(s)$, but $\{(\alpha, \rho), (\beta, \chi)\} \notin Exec(s)$. For $s \in DR_{\mathrm{T}}(G)$ we have $PT(\emptyset, s) \in (0; 1]$, hence, there is a non-zero probability to stay in the state s at the next moment, and the residence time in s is at least one time unit.

If $s \in DR_{\mathrm{V}}(G)$ then $PT(\Upsilon, s)$ can be interpreted as the weight of the set of immediate 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 [14] (inspired by way in which the probabilities of conflicting immediate transitions in GSPNs are calculated [6]). The only difference is that we have a step semantics and, for every set of immediate multiactions executed in parallel, we use its cumulative weight.

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

The *probability to move from s to $\tilde{s}$ by executing any set 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 above reflects the probability of the mutually exclusive event union, since $\sum_{\{\Upsilon | \exists H \in s, \ \exists \widetilde{H} \in \tilde{s}, \ H \xrightarrow{\Upsilon} \widetilde{H}\}} PT(\Upsilon, s) = \frac{1}{\sum_{\Xi \in Exec(s)} PF(\Xi, s)} \cdot \sum_{\{\Upsilon | \exists H \in s, \ \exists \widetilde{H} \in \tilde{s}, \ H \xrightarrow{\Upsilon} \widetilde{H}\}} PF(\Upsilon, s)$, where for each Υ, $PF(\Upsilon, s)$ is the probability of the exclusive execution of Υ in s.

Note that $\forall s \in DR(G)$, $\sum_{\{\tilde{s} | \exists H \in s, \ \exists \widetilde{H} \in \tilde{s}, \ \exists \Upsilon, \ H \xrightarrow{\Upsilon} \widetilde{H}\}} PM(s, \tilde{s}) = \sum_{\{\tilde{s} | \exists H \in s, \ \exists \widetilde{H} \in \tilde{s}, \ \exists \Upsilon, \ H \xrightarrow{\Upsilon} \widetilde{H}\}} \sum_{\{\Upsilon | \exists H \in s, \ \exists \widetilde{H} \in \tilde{s}, \ H \xrightarrow{\Upsilon} \widetilde{H}\}} PT(\Upsilon, s) = \sum_{\Upsilon \in Exec(s)} PT(\Upsilon, s) = 1$.

Example 4. *Let $E = (\{a\}, \rho)[](\{a\}, \chi)$, where $\rho, \chi \in (0; 1)$. $DR(\overline{E})$ consists of the equivalence classes $s_1 = [\overline{E}]_{\approx}$ and $s_2 = [\underline{E}]_{\approx}$. We have $DR_{\mathrm{T}}(\overline{E}) = \{s_1, s_2\}$. The execution probabilities are calculated as follows. Since $Exec(s_1) = \{\emptyset, \{(\{a\}, \rho)\}, \{(\{a\}, \chi)\}\}$, we get $PF(\{(\{a\}, \rho)\}, s_1) = \rho(1 - \chi)$, $PF(\{(\{a\}, \chi)\}, s_1) = \chi(1 - \rho)$ and $PF(\emptyset, s_1) = (1-\rho)(1-\chi)$. Then $\sum_{\Xi \in Exec(s_1)} PF(\Xi, s_1) = \rho(1-\chi) + \chi(1-\rho) + (1-\rho)(1-\chi) = 1 - \rho\chi$. Thus, $PT(\{(\{a\}, \rho)\}, s_1) = \frac{\rho(1-\chi)}{1-\rho\chi}$, $PT(\{(\{a\}, \chi)\}, s_1) = \frac{\chi(1-\rho)}{1-\rho\chi}$ and $PT(\emptyset, s_1) = PM(s_1, s_1) = \frac{(1-\rho)(1-\chi)}{1-\rho\chi}$. Next, $Exec(s_2) = \{\emptyset\}$, hence, $\sum_{\Xi \in Exec(s_2)} PF(\Xi, s_2) = PF(\emptyset, s_2) = 1$ and $PT(\emptyset, s_2) = PM(s_2, s_2) = \frac{1}{1} = 1$. Finally, $PM(s_1, s_2) = PT(\{(\{a\}, \rho)\}, s_1) + PT(\{(\{a\}, \chi)\}, s_1) = \frac{\rho(1-\chi)}{1-\rho\chi} + \frac{\chi(1-\rho)}{1-\rho\chi} = \frac{\rho+\chi-2\rho\chi}{1-\rho\chi}$.*

Let $E' = (\{a\}, \natural_l)[](\{a\}, \natural_m)$, where $l, m \in \mathbb{R}_{>0}$. $DR(\overline{E'})$ consists of the equivalence classes $s_1' = [\overline{E'}]_{\approx}$ and $s_2' = [\underline{E'}]_{\approx}$. We have $DR_{\mathrm{T}}(\overline{E'}) = \{s_2'\}$ and $DR_{\mathrm{V}}(\overline{E'}) = \{s_1'\}$. The execution probabilities are calculated as follows. Since $Exec(s_1') = \{\{(\{a\}, \natural_l)\}, \{(\{a\}, \natural_m)\}\}$, we get $PF(\{(\{a\}, \natural_l)\}, s_1') = l$ and $PF(\{(\{a\}, \natural_m)\}, s_1') = m$. Then $\sum_{\Xi \in Exec(s_1')} PF(\Xi, s_1') = l + m$. Thus, $PT(\{(\{a\}, \natural_l)\}, s_1') = \frac{l}{l+m}$ and $PT(\{(\{a\}, \natural_m)\}, s_1') = \frac{m}{l+m}$. Next, $Exec(s_2') = \{\emptyset\}$, hence, $\sum_{\Xi \in Exec(s_2')} PF(\Xi, s_2') =$

$PF(\emptyset, s_2') = 1$ and $PT(\emptyset, s_2') = PM(s_2', s_2') = \frac{1}{1} = 1$.

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

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

- *the set of* states *is $S_G = DR(G)$;*
- *the set of* labels *is $L_G = 2^{\mathcal{SIL}} \times (0; 1]$;*
- *the set of* transitions *is $\mathcal{T}_G = \{(s, (\Upsilon, PT(\Upsilon, s)), \tilde{s}) \mid s, \tilde{s} \in DR(G), \ \exists H \in s,$*
  $\exists \widetilde{H} \in \tilde{s}, \ H \xrightarrow{\Upsilon} \widetilde{H}\}$;
- *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. This is guaranteed by the note after the definition of $PT(\Upsilon, s)$. Thus, we have defined a *generative* model of probabilistic processes [38]. The reason is that the sum of the probabilities of the transitions with all possible labels should be equal to 1, not only of those with the same labels (up to enumeration of activities they include) as in the *reactive* models, and we do not have a nested probabilistic choice as in the *stratified* models.

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

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

The step probabilities belong to the interval $(0; 1]$, being 1 in the case when we cannot leave a tangible state s and the only transition leaving it is the empty loop one $s \xrightarrow{\emptyset}_1 s$, or if there is just a single transition from a vanishing state to any other one. We write $s \xrightarrow{\Upsilon} \tilde{s}$ if $\exists \mathcal{P}, \ s \xrightarrow{\Upsilon}_{\mathcal{P}} \tilde{s}$ and $s \rightarrow \tilde{s}$ 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 13. *Let $TS(G) = (S_G, L_G, \mathcal{T}_G, s_G)$ and $TS(G') = (S_{G'}, L_{G'}, \mathcal{T}_{G'}, s_{G'})$ be the transition systems of dynamic expressions G and G', respectively. A mapping $\beta : S_G \rightarrow S_{G'}$ 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 14. *Two dynamic expressions G and G' are* equivalent w.r.t. transition systems, *denoted by $G =_{\mathrm{ts}} G'$, if $TS(G) \simeq TS(G')$.*

Example 5. *Consider the expression* $\mathsf{Stop} = (\{g\}, \frac{1}{2}) \ \mathsf{rs} \ g$ *specifying the non-terminating process that performs only empty loops with probability 1. Then, for $\rho, \chi, \theta, \phi \in (0; 1)$ and $l, m \in \mathbb{R}_{>0}$, let*
$E = [(\{a\}, \rho) * ((\{b\}, \chi); ((((\{c\}, \natural_l); (\{d\}, \theta)) [] ((\{e\}, \natural_m); (\{f\}, \phi)))) * \mathsf{Stop}].$
  $DR(\overline{E})$ consists of the equivalence classes

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

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

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

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

$s_5 = [[(\{a\}, \rho) * ((\{b\}, \chi); ((((\{c\}, \natural_l); (\{d\}, \theta)) [] ((\{e\}, \natural_m); \overline{(\{f\}, \phi)}))) * \mathsf{Stop}]]_{\approx}.$

  We have $DR_{\mathrm{T}}(\overline{E}) = \{s_1, s_2, s_4, s_5\}$ and $DR_{\mathrm{V}}(\overline{E}) = \{s_3\}$. In the first part of Figure 3, the transition system $TS(\overline{E})$ is presented. The tangible states are depicted in ovals and the vanishing ones in boxes. For simplicity of the graphical representation, the singleton sets of activities are written without outer braces.

4. Denotational semantics

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

4.1. **Labeled DTSIPNs.** Let us introduce a class of labeled discrete time stochastic and immediate Petri nets (LDTSIPNs), a subclass of DTSPNs [69, 70] (we do not allow the transition probabilities to be equal to 1) extended with transition labeling and immediate transitions. LDTSIPNs resemble in part discrete time deterministic and stochastic PNs (DTDSPNs) [89], as well as discrete deterministic and stochastic PNs (DDSPNs) [88]. DTDSPNs and DDSPNs are the extensions of DTSPNs with deterministic transitions (having fixed delay that can be zero), inhibitor arcs, priorities and guards. Next, while stochastic transitions of DTDSPNs, like those of DTSPNs, have geometrically distributed delays, stochastic transitions of DDSPNs have discrete time phase-type distributed delays. Nevertheless, LDTSIPNs are not subsumed by DTDSPNs or DDSPNs, since LDTSIPNs have a step semantics while DTDSPNs and DDSPNs have interleaving one. LDTSIPNs are somewhat similar to labeled weighted DTSPNs from [29], but in the latter there are no immediate transitions, all (stochastic) transitions have weights, the transition probabilities may be equal to 1 and only maximal fireable subsets of the enabled transitions are fired.

Stochastic preemptive time Petri nets (spTPNs) [24] is a discrete time model with a maximal step semantics, where both time ticks and instantaneous parallel firings of maximal transition sets are possible, but the transition steps in LDTSIPNs are not obliged to be maximal. The transition delays in spTPNs are governed by static general discrete distributions, associated with the transitions, while the transitions of LDTSIPNs are only associated with probabilities, used later to calculate the

step probabilities after one unit (from tangible markings) or zero (from vanishing markings) delay. Further, LDTSIPNs have just geometrically distributed or deterministic zero delays in the markings. Moreover, the discrete time tick and concurrent transition firing are treated in spTPNs as different events while firing every (possibly empty) set of stochastic transitions in LDTSIPNs requires one unit time delay. spTPNs are essentially a modification and extension of unlabeled LWDTSPNs with additional facilities, such as inhibitor arcs, priorities, resources, preemptions, schedulers etc. However, the price of such an expressiveness of spTPNs is that the model is rather intricate and difficult to analyze.

Note that guards in DTDSPNs and DDSPNs, inhibitor arcs and priorities in DTDSPNs, DDSPNs and spTPNs, the maximal step semantics of LWDTSPNs and spTPNs make these models Turing powerful, resulting in undecidability of important behavioural properties.

Definition 15. *A* labeled discrete time stochastic and immediate Petri net (LDTSIPN) *is a tuple* $N = (P_N, T_N, W_N, \Omega_N, \mathcal{L}_N, M_N)$, *where*

- P_N *and* $T_N = Ts_N \uplus Ti_N$ *are finite sets of* places *and* stochastic and immediate transitions, *respectively, such that* $P_N \cup T_N \neq \emptyset$ *and* $P_N \cap T_N = \emptyset$;
- $W_N : (P_N \times T_N) \cup (T_N \times P_N) \to \mathbb{N}$ *is a function giving the* weights of arcs *between places and transitions;*
- Ω_N *is the* transition probability and weight *function such that*
  - $\Omega_N|_{Ts_N} : Ts \to (0; 1)$ *(it provides stochastic transitions with probabilities);*
  - $\Omega_N|_{Ti_N} : Ti \to \mathbb{R}_{>0}$ *(it associates immediate transitions with weights);*
- $\mathcal{L}_N : T_N \to \mathcal{L}$ *is the* transition labeling *function assigning multiactions to transitions;*
- $M_N \in \mathbb{N}_{\text{fin}}^{P_N}$ *is the* initial marking.

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

Let N be an LDTSIPN and $t \in T_N$, $U \in \mathbb{N}_{\text{fin}}^{T_N}$. The *precondition* $^\bullet t$ and the *postcondition* $t^\bullet$ of t are the multisets of places $(^\bullet t)(p) = W_N(p,t)$ and $(t^\bullet)(p) = W_N(t,p)$. The *precondition* $^\bullet U$ and the *postcondition* $U^\bullet$ of U are the multisets of places $^\bullet U = \sum_{t \in U} {}^\bullet t$ and $U^\bullet = \sum_{t \in U} t^\bullet$. Note that for $U = \emptyset$ we have $^\bullet\emptyset = \emptyset = \emptyset^\bullet$.

Let N be an LDTSIPN and $M, \widetilde{M} \in \mathbb{N}_{\text{fin}}^{P_N}$. Immediate transitions have a priority over stochastic ones, thus, immediate transitions always fire first if they can. A transition $t \in T_N$ is *enabled* at M if $^\bullet t \subseteq M$ and one of the following holds: (1) $t \in Ti_N$ or (2) $\forall u \in T_N$, $^\bullet u \subseteq M \Rightarrow u \in Ts_N$.

Thus, a transition is enabled at a marking if it has enough tokens in its input places (i.e. in the places from its precondition) and it is immediate one; otherwise, when it is stochastic, there exists no immediate transition with enough tokens in its input places. Let $Ena(M)$ be the set of *all transitions enabled at* M. By definition, it follows that $Ena(M) \subseteq Ti_N$ or $Ena(M) \subseteq Ts_N$. A set of transitions $U \subseteq Ena(M)$ is *enabled* at a marking M if $^\bullet U \subseteq M$. Firings of transitions

are atomic operations, and transitions may fire concurrently in steps. We assume that all transitions participating in a step should differ, hence, only the sets (not multisets) of transitions may fire. Thus, we do not allow self-concurrency, i.e. firing of transitions in parallel to themselves. This restriction is introduced to avoid some technical difficulties while calculating probabilities for multisets of transitions as we shall see after the following formal definitions. Moreover, we do not need to consider self-concurrency, since denotational semantics of expressions will be defined via dtsi-boxes which are safe LDTSIPNs (hence, no self-concurrency is possible).

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

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

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

In the case $U = \emptyset$ and $tang(M)$ we define

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

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

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

Observe that in the case $U = \emptyset$ and $tang(M)$ we have $M = \widetilde{M}$. Note that for all markings of an LDTSIPN N, the sum of outgoing probabilities is equal to 1, i.e. $\forall M \in \mathbb{N}_{\text{fin}}^{P_N}$, $\sum_{\{U \subseteq Ena(M) \mid {}^\bullet U \subseteq M\}} PT(U, M) = 1$. This follows from the definition of $PT(U, M)$ and guarantees that it defines a probability distribution.

We write $M \xrightarrow{U} \widetilde{M}$ if $\exists \mathcal{P}$, $M \xrightarrow{U}_{\mathcal{P}} \widetilde{M}$ and $M \to \widetilde{M}$ if $\exists U$, $M \xrightarrow{U} \widetilde{M}$.

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

$$
PM(M, \widetilde{M}) = \sum_{\{U \mid M \xrightarrow{U} \widetilde{M}\}} PT(U, M).
$$

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

Definition 16. *Let N be an LDTSIPN. The* reachability set *of N, denoted by $RS(N)$, is the minimal set of markings such that*

- $M_N \in RS(N)$;
- *if $M \in RS(N)$ and $M \to \widetilde{M}$ then $\widetilde{M} \in RS(N)$.*

Definition 17. *Let N be an LDTSIPN. The* reachability graph *of N is a (labeled probabilistic) transition system $RG(N) = (S_N, L_N, \mathcal{T}_N, s_N)$, where*

- *the set of* states *is $S_N = RS(N)$;*
- *the set of* labels *is $L_N = 2^{T_N} \times (0;1]$;*
- *the set of* transitions *is $\mathcal{T}_N = \{(M, (U, \mathcal{P}), \widetilde{M}) \mid M, \widetilde{M} \in RS(N), \ M \xrightarrow{U}_{\mathcal{P}} \widetilde{M}\}$;*
- *the* initial state *is $s_N = M_N$.*

Let $RS_{\mathrm{T}}(N)$ be the set of *all tangible markings from $RS(N)$* and $RS_{\mathrm{V}}(N)$ be the set of *all vanishing markings from $RS(N)$*. Obviously, $RS(N) = RS_{\mathrm{T}}(N) \uplus RS_{\mathrm{V}}(N)$.

4.2. **Algebra of dtsi-boxes.** We now present discrete time stochastic and immediate Petri boxes and algebraic operations to define the net representation of dtsiPBC expressions.

Definition 18. *A discrete time stochastic and immediate Petri box (dtsi-box) is a tuple $N = (P_N, T_N, W_N, \Lambda_N)$, where*

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

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

A dtsi-box is *plain* if $\forall t \in T_N$, $\exists (\alpha, \kappa) \in \mathcal{SIL}$, $\Lambda_N(t) = \varrho_{(\alpha, \kappa)}$, where $\varrho_{(\alpha, \kappa)} = \{(\emptyset, (\alpha, \kappa))\}$ is a *constant relabeling*, identified with the activity (α, κ). A *marked plain dtsi-box* is a pair (N, M_N), where N is a plain dtsi-box and $M_N \in \mathbb{N}_{\mathrm{fin}}^{P_N}$ is its marking. We denote $\overline{N} = (N, {}^{\circ}N)$ and $\underline{N} = (N, N^{\circ})$. Note that a marked plain dtsi-box $(P_N, T_N, W_N, \Lambda_N, M_N)$ could be interpreted as the LDTSIPN $(P_N, T_N, W_N, \Omega_N, \mathcal{L}_N, M_N)$, where functions Ω_N and $\mathcal{L}_N$ are defined as follows: $\forall t \in T_N$, $\Omega_N(t) = \kappa$ if $\kappa \in (0;1)$; or $\Omega_N(t) = l$ if $\kappa = \natural_l$, $l \in \mathbb{R}_{>0}$; and $\mathcal{L}_N(t) = \alpha$, where $\Lambda_N(t) = \varrho_{(\alpha, \kappa)}$. Behaviour of the marked dtsi-boxes follows from the firing rule of LDTSIPNs. A plain dtsi-box N is *n-bounded* ($n \in \mathbb{N}$) if $\overline{N}$ is so, i.e. $\forall M \in RS(\overline{N})$, $\forall p \in P_N$, $M(p) \leq n$, and it is *safe* if it is 1-bounded. A plain dtsi-box N is *clean* if $\forall M \in RS(\overline{N})$, ${}^{\circ}N \subseteq M \Rightarrow M = {}^{\circ}N$ and $N^{\circ} \subseteq M \Rightarrow M = N^{\circ}$, i.e. if there are tokens in all its entry (exit) places then no other places have tokens.
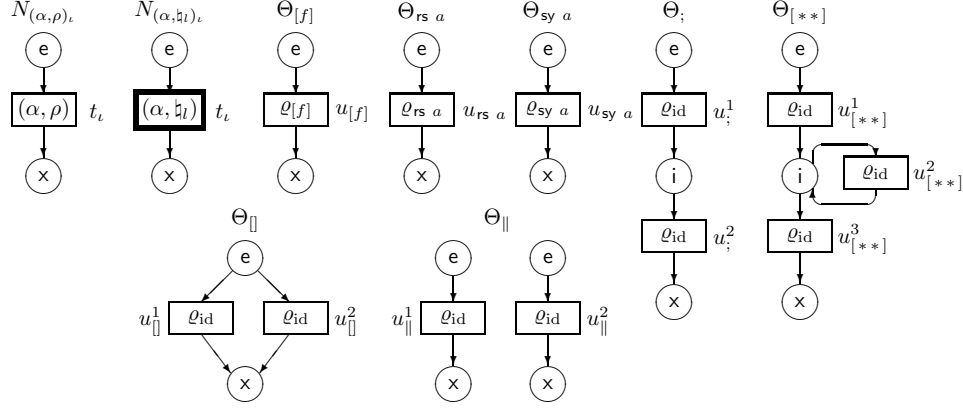
FIG. 2. The plain and operator dtsi-boxes

The structure of the plain dtsi-box corresponding to a static expression is constructed like in PBC [17, 16], i.e. we use simultaneous refinement and relabeling meta-operator (net refinement) in addition to the *operator dtsi-boxes* corresponding to the algebraic operations of dtsiPBC and featuring transformational transition relabelings. Operator dtsi-boxes specify n-ary functions from plain dtsi-boxes to plain dtsi-boxes (we have $1 \leq n \leq 3$ in dtsiPBC). Thus, as we shall see in Theorem 1, the resulting plain dtsi-boxes are safe and clean. In the definition of the denotational semantics, we shall apply standard constructions used for PBC. Let Θ denote *operator box* and u denote *transition name* from PBC setting.

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

- $\varrho_{\mathsf{id}} = \{(\{(\alpha, \kappa)\}, (\alpha, \kappa)) \mid (\alpha, \kappa) \in \mathcal{SIL}\}$ is the *identity relabeling*;
- $\varrho_{(\alpha, \kappa)} = \{(\emptyset, (\alpha, \kappa))\}$ is the *constant relabeling*, identified with $(\alpha, \kappa) \in \mathcal{SIL}$;
- $\varrho_{[f]} = \{(\{(\alpha, \kappa)\}, (f(\alpha), \kappa)) \mid (\alpha, \kappa) \in \mathcal{SIL}\}$;
- $\varrho_{\mathsf{rs}\ a} = \{(\{(\alpha, \kappa)\}, (\alpha, \kappa)) \mid (\alpha, \kappa) \in \mathcal{SIL},\ a, \hat{a} \notin \alpha\}$;
- $\varrho_{\mathsf{sy}\ a}$ is the least relabeling containing ϱ_{id} such that if $(\Upsilon, (\alpha, \kappa)), (\Xi, (\beta, \lambda)) \in \varrho_{\mathsf{sy}\ a},\ a \in \alpha,\ \hat{a} \in \beta$, then
  - $(\Upsilon + \Xi, (\alpha \oplus_a \beta, \kappa \cdot \lambda)) \in \varrho_{\mathsf{sy}\ a}$ if $\kappa, \lambda \in (0; 1)$;
  - $(\Upsilon + \Xi, (\alpha \oplus_a \beta, \natural_{l+m})) \in \varrho_{\mathsf{sy}\ a}$ if $\kappa = \natural_l,\ \lambda = \natural_m,\ l, m \in \mathbb{R}_{>0}$.

The plain dtsi-boxes $N_{(\alpha, \rho)_\iota}$, $N_{(\alpha, \natural_l)_\iota}$, where $\rho \in (0; 1)$, $l \in \mathbb{R}_{>0}$, and operator dtsi-boxes are presented in Figure 2. The label i of internal places is usually omitted.

In the case of the iteration, a decision that we must take is the selection of the operator box that we shall use for it, since we have two proposals in plain PBC for that purpose [16]. One of them provides us with a safe version with six transitions in the operator box, but there is also a simpler version, which has only three transitions. In general, in PBC, with the latter version we may generate 2-bounded nets, which only occurs when a parallel behaviour appears at the highest level of the body of the iteration. Nevertheless, in our case, and due to the syntactical restriction introduced for regular terms, this particular situation cannot occur, so that the net obtained will be always safe.

To construct a semantic function assigning a plain dtsi-box to every static expression of dtsiPBC, we define the *enumeration* function $Enu : T \to Num$, which associates the numberings with transitions of a plain dtsi-box $N = (P, T, W, \Lambda)$ according to those of activities. For synchronization, the function associates with

the resulting new transition the concatenation of the parenthesized numberings of the transitions it comes from.

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

- $Box_{\mathrm{dtsi}}((\alpha, \kappa)_\iota) = N_{(\alpha, \kappa)_\iota}$. Since a single transition t_ι corresponds to the activity $(\alpha, \kappa)_\iota \in \mathcal{SIL}$, their numberings coincide: $Enu(t_\iota) = \iota$.
- $Box_{\mathrm{dtsi}}(E \circ F) = \Theta_\circ(Box_{\mathrm{dtsi}}(E), Box_{\mathrm{dtsi}}(F))$, $\circ \in \{; , [], \|\}$. Since we do not introduce new transitions, we preserve the initial numbering:
  $$Enu(t) = \begin{cases} Enu_E(t), & \text{if } t \in T_E; \\ Enu_F(t), & \text{if } t \in T_F. \end{cases}$$
- $Box_{\mathrm{dtsi}}(E[f]) = \Theta_{[f]}(Box_{\mathrm{dtsi}}(E))$. Since we only replace the labels of some multiactions by a bijection, we preserve the initial numbering:
  $$Enu(t) = Enu_E(t), \ t \in T_E.$$
- $Box_{\mathrm{dtsi}}(E \ \mathsf{rs} \ a) = \Theta_{\mathsf{rs} \ a}(Box_{\mathrm{dtsi}}(E))$. Since we remove all transitions labeled with multiactions containing a or $\hat{a}$, the remaining transitions numbering is not changed: $Enu(t) = Enu_E(t), \ t \in T_E, \ a, \hat{a} \notin \alpha, \ \Lambda_E(t) = \varrho_{(\alpha, \kappa)}$.
- $Box_{\mathrm{dtsi}}(E \ \mathsf{sy} \ a) = \Theta_{\mathsf{sy} \ a}(Box_{\mathrm{dtsi}}(E))$. Note that $\forall v, w \in T_E$, such that $\Lambda_E(v) = \varrho_{(\alpha, \kappa)}$, $\Lambda_E(w) = \varrho_{(\beta, \lambda)}$ and $a \in \alpha$, $\hat{a} \in \beta$, the new transition t resulting from synchronization of v and w has the label $\Lambda(t) = \varrho_{(\alpha \oplus_a \beta, \kappa \cdot \lambda)}$ if t is a stochastic transition ($\kappa, \lambda \in (0; 1)$); or $\Lambda(t) = \varrho_{(\alpha \oplus_a \beta, \natural_{l+m})}$ if t is an immediate one ($\kappa = \natural_l, \ \lambda = \natural_m, \ l, m \in \mathbb{R}_{>0}$); and the numbering $Enu(t) = (Enu_E(v))(Enu_E(w))$. Thus, the enumeration function is defined as
  $$Enu(t) = \begin{cases} Enu_E(t), & \text{if } t \in T_E; \\ (Enu_E(v))(Enu_E(w)), & \text{if } t \text{ results} \\ \quad \text{from synchronization of } v \text{ and } w. \end{cases}$$
  According to the definition of $\varrho_{\mathsf{sy} \ a}$, the synchronization is only possible when all the transitions in the set are stochastic or all of them are immediate. If we synchronize the same set of transitions in different orders, we obtain several resulting transitions with the same label and probability or weight, but with the different numberings having the same content. Then we only consider a single transition from the resulting ones in the plain dtsi-box to avoid introducing redundant transitions.

  For example, if the transitions t and u are generated by synchronizing v and w in different orders, we have $\Lambda(t) = \varrho_{(\alpha \oplus_a \beta, \kappa \cdot \lambda)} = \Lambda(u)$ for stochastic transitions ($\kappa, \lambda \in (0; 1)$) or $\Lambda(t) = \varrho_{(\alpha \oplus_a \beta, \natural_{l+m})} = \Lambda(u)$ for immediate ones ($\kappa = \natural_l, \ \lambda = \natural_m, \ l, m \in \mathbb{R}_{>0}$), but $Enu(t) = (Enu_E(v))(Enu_E(w)) \neq (Enu_E(w))(Enu_E(v)) = Enu(u)$ while $Cont(Enu(t)) = Cont(Enu(v)) \cup Cont(Enu(w)) = Cont(Enu(u))$. Then only one transition t (or, symmetrically, u) will appear in $Box_{\mathrm{dtsi}}(E \ \mathsf{sy} \ a)$.
- $Box_{\mathrm{dtsi}}([E * F * K]) = \Theta_{[**]}(Box_{\mathrm{dtsi}}(E), Box_{\mathrm{dtsi}}(F), Box_{\mathrm{dtsi}}(K))$. Since we do not introduce new transitions, we preserve the initial numbering:
  $$Enu(t) = \begin{cases} Enu_E(t), & \text{if } t \in T_E; \\ Enu_F(t), & \text{if } t \in T_F; \\ Enu_K(t), & \text{if } t \in T_K. \end{cases}$$

We now can formally define the denotational semantics as a homomorphism.

Definition 19. *Let* $(\alpha, \kappa) \in \mathcal{SIL}$, $a \in Act$ *and* $E, F, K \in RegStatExpr$. *The denotational semantics of dtsiPBC is a mapping* Box_{dtsi} *from* $RegStatExpr$ *into the domain of plain dtsi-boxes defined as follows:*

(1) $Box_{\mathrm{dtsi}}((\alpha, \kappa)_\iota) = N_{(\alpha, \kappa)_\iota}$;

(2) $Box_{\mathrm{dtsi}}(E \circ F) = \Theta_\circ(Box_{\mathrm{dtsi}}(E), Box_{\mathrm{dtsi}}(F)), \ \circ \in \{; , [], \|\}$;

(3) $Box_{\mathrm{dtsi}}(E[f]) = \Theta_{[f]}(Box_{\mathrm{dtsi}}(E))$;

(4) $Box_{\mathrm{dtsi}}(E \circ a) = \Theta_{\circ a}(Box_{\mathrm{dtsi}}(E)), \ \circ \in \{\mathsf{rs}, \mathsf{sy}\}$;

(5) $Box_{\mathrm{dtsi}}([E * F * K]) = \Theta_{[**]}(Box_{\mathrm{dtsi}}(E), Box_{\mathrm{dtsi}}(F), Box_{\mathrm{dtsi}}(K))$.

The dtsi-boxes of dynamic expressions can be defined. For $E \in RegStatExpr$, let $Box_{\mathrm{dtsi}}(\overline{E}) = \overline{Box_{\mathrm{dtsi}}(E)}$ and $Box_{\mathrm{dtsi}}(\underline{E}) = \underline{Box_{\mathrm{dtsi}}(E)}$. This definition is compositional in the sense that, for any arbitrary dynamic expression, we may decompose it in some inner dynamic and static expressions, for which we may apply the definition, thus obtaining the corresponding plain dtsi-boxes, which can be joined according to the term structure (by definition of Box_{dtsi}), the resulting plain box being marked in the places that were marked in the argument nets.

Theorem 1. *For any static expression* E, $Box_{\mathrm{dtsi}}(\overline{E})$ *is safe and clean.*

Proof. The structure of the net is obtained as in PBC [17, 16], combining both refinement and relabeling. Hence, the dtsi-boxes thus obtained will be safe and clean. $\qquad\square$

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

Theorem 2. *For any static expression* E, $TS(\overline{E}) \simeq RG(Box_{\mathrm{dtsi}}(\overline{E}))$.

Proof. As for the qualitative behaviour, we have the same isomorphism as in PBC [17, 16]. The quantitative behaviour is the same by the following reasons. First, the activities of an expression have the probability or weight parts coinciding with the probabilities or weights of the transitions belonging to the corresponding dtsi-box. Second, we use analogous probability or weight functions to construct the corresponding transition systems and reachability graphs. $\qquad\square$

Example 6. *Let* E *be from Example 5. In Figure 3, the marked dtsi-box* $N = Box_{\mathrm{dtsi}}(\overline{E})$ *and its reachability graph* $RG(N)$ *are presented. It is easy to see that* $TS(\overline{E})$ *and* $RG(N)$ *are isomorphic.*

5. PERFORMANCE EVALUATION

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

5.1. **Analysis of the underlying SMC.** For a dynamic expression G, a discrete random variable is associated with every tangible state $s \in DR_{\mathrm{T}}(G)$. The variable captures a residence time in the state. One can interpret staying in a state at the next discrete time moment as a failure and leaving it as a success of some trial series. It is easy to see that the random variables are geometrically distributed with the parameter $1 - PM(s, s)$, since the probability to stay in s for $k - 1$ time moments and leave it at the moment $k \geq 1$ is $PM(s, s)^{k-1}(1 - PM(s, s))$ (the residence time is k in this case, and this formula defines the probability mass function (PMF) of
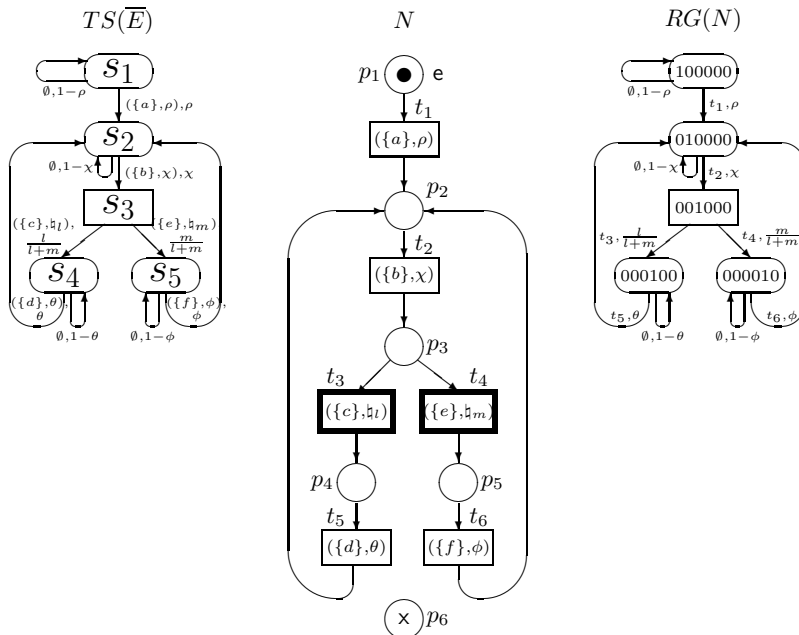
FIG. 3. *The transition system of* $\overline{E}$, *marked dtsi-box*
$N = Box_{\text{dtsi}}(\overline{E})$ *and its reachability graph for* $E = [(\{a\}, \rho) *$
$((\{b\}, \chi); ((((\{c\}, \natural_l); (\{d\}, \theta))[]((\{e\}, \natural_m); (\{f\}, \phi)))) * \mathsf{Stop}]$

residence time in s). Hence, the probability distribution function (PDF) of residence time in s is $1 - PM(s, s)^k$ ($k \geq 0$) (the probability that the residence time in s is less than or equal to k). The mean value formula for the geometrical distribution allows us to calculate the average sojourn time in s as $\frac{1}{1 - PM(s,s)}$. Clearly, the average sojourn time in a vanishing state is zero. Let $s \in DR(G)$.

The *average sojourn time in the state* s is

$$SJ(s) = \begin{cases} \frac{1}{1 - PM(s,s)}, & \text{if } s \in DR_{\text{T}}(G); \\ 0, & \text{if } s \in DR_{\text{V}}(G). \end{cases}$$

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

The *sojourn time variance in the state* s is

$$VAR(s) = \begin{cases} \frac{PM(s,s)}{(1 - PM(s,s))^2}, & \text{if } s \in DR_{\text{T}}(G); \\ 0, & \text{if } s \in DR_{\text{V}}(G). \end{cases}$$

The *sojourn time variance vector* VAR of G 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 SMC [78, 55], 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 GSPNs [65, 5, 6], DTDSPNs [89] and DDSPNs [88]. $EDTMC(G)$ 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 equal to one discrete time unit. Each 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) [42].

Let G be a dynamic expression and $s, \tilde{s} \in DR(G)$. The transition system $TS(G)$ can have self-loops from a state to itself with 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 $s \neq \tilde{s}$. The *probability to move from s to $\tilde{s}$ by executing any set of activities after possible self-loops* is

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

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

Here $SL(s)$ is the *self-loops abstraction factor in the state s*. The *self-loops abstraction vector* of G, denoted by SL, has the elements $SL(s)$, $s \in DR(G)$. The value $k = 0$ in the summation above corresponds to the case when no self-loops occur. Note that $\forall s \in DR_{\mathrm{T}}(G)$, $SL(s) = \frac{1}{1-PM(s,s)} = SJ(s)$, hence, $\forall s \in DR_{\mathrm{T}}(G)$, $PM^*(s, \tilde{s}) = SJ(s)PM(s, \tilde{s})$, since we always have the empty loop (the self-loop) $s \xrightarrow{\emptyset} s$ from every tangible state s. Empty loops are not possible from vanishing states, hence, $\forall s \in DR_{\mathrm{V}}(G)$, $PM^*(s, \tilde{s}) = \frac{PM(s,\tilde{s})}{1-PM(s,s)}$, when there are non-empty self-loops (due to iteration) from s, or $PM^*(s, \tilde{s}) = PM(s, \tilde{s})$, when there are no self-loops from s.

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

Definition 20. *Let G be a dynamic expression. The* embedded (absorbing) discrete time Markov chain (EDTMC) *of G, denoted by $EDTMC(G)$, has the state space $DR(G)$, the initial state $[G]_\approx$ and the transitions $s \twoheadrightarrow_{\mathcal{P}} \tilde{s}$ if $s \to \tilde{s}$ and $s \neq \tilde{s}$, where $\mathcal{P} = PM^*(s, \tilde{s})$. The* underlying SMC *of G, denoted by $SMC(G)$, has the EDTMC $EDTMC(G)$ and the sojourn time in every $s \in DR_{\mathrm{T}}(G)$ is geometrically distributed with the parameter $1-PM(s, s)$ while the sojourn time in every $s \in DR_{\mathrm{V}}(G)$ is zero.*

Let G be a dynamic expression. The elements $\mathcal{P}^*_{ij}$ ($1 \leq i, j \leq n = |DR(G)|$) of the (one-step) transition probability matrix (TPM) $\mathbf{P}^*$ for $EDTMC(G)$ are
$$
\mathcal{P}^*_{ij} = \left\{ \begin{array}{ll} PM^*(s_i, s_j), & \text{if } s_i \to s_j, \ s_i \neq s_j; \\ 0, & \text{otherwise}. \end{array} \right.
$$
The transient (k-step, $k \in \mathbb{N}$) PMF $\psi^*[k] = (\psi^*[k](s_1), \dots, \psi^*[k](s_n))$ for $EDTMC(G)$ is calculated as

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

where $\psi^*[0] = (\psi^*[0](s_1), \dots, \psi^*[0](s_n))$ is the initial PMF defined as
$$
\psi^*[0](s_i) = \left\{ \begin{array}{ll} 1, & \text{if } s_i = [G]_\approx; \\ 0, & \text{otherwise}. \end{array} \right. \quad \text{Note also that } \psi^*[k+1] = \psi^*[k]\mathbf{P}^* \ (k \in \mathbb{N}).
$$

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

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

where $\mathbf{I}$ is the identity matrix of order n and $\mathbf{0}$ ($\mathbf{1}$) is a row vector of n values 0 (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 $\psi^*(s_i)$ ($1 \le i \le n$) by the average sojourn time $SJ(s_i)$ in the state s_i, after which we normalize the resulting values. Remember that for a vanishing state $s \in DR_{\mathrm{V}}(G)$ we have $SJ(s) = 0$.

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

$$\varphi(s_i) = \begin{cases} \dfrac{\psi^*(s_i)SJ(s_i)}{\displaystyle\sum_{j=1}^{n} \psi^*(s_j)SJ(s_j)}, & \text{if } s_i \in DR_{\mathrm{T}}(G); \\[2mm] 0, & \text{if } s_i \in DR_{\mathrm{V}}(G). \end{cases}$$

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

Example 7. *Let E be from Example 5. In Figure 4, 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 $SJ = \left(\frac{1}{\rho}, \frac{1}{\chi}, 0, \frac{1}{\theta}, \frac{1}{\phi}\right)$.

The sojourn time variance vector of $\overline{E}$ is $VAR = \left(\frac{1-\rho}{\rho^2}, \frac{1-\chi}{\chi^2}, 0, \frac{1-\theta}{\theta^2}, \frac{1-\phi}{\phi^2}\right)$.

The TPM for $EDTMC(\overline{E})$ 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 $\left(0, \frac{1}{3\chi}, 0, \frac{l}{3\theta(l+m)}, \frac{m}{3\phi(l+m)}\right)$.

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

The steady-state PMF for $SMC(\overline{E})$ is $\varphi = \frac{1}{\theta\phi(l+m)+\chi(\phi l+\theta m)}(0, \theta\phi(l+m), 0, \chi\phi l, \chi\theta m)$.

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 [72, 52].

- The *average recurrence (return) time in the state s* is $\frac{1}{\varphi(s)}$.
- The *fraction of residence time in the state s* is $\varphi(s)$.

- The *fraction of residence time in the set of states S* or the *probability of the event determined by a condition that is true for all states from S* is $\sum_{s \in S} \varphi(s)$.
- The *relative fraction of residence time in S w.r.t. that in $\widetilde{S}$* is $\frac{\sum_{s \in S} \varphi(s)}{\sum_{\tilde{s} \in \tilde{S}} \varphi(\tilde{s})}$.
- The *rate of leaving the state s* is $\frac{\varphi(s)}{SJ(s)}$.
- The *steady-state probability to perform a step with a set of activities Ξ* is $\sum_{s \in DR(G)} \varphi(s) \sum_{\{\Upsilon | \Xi \subseteq \Upsilon\}} PT(\Upsilon, s)$.
- The *probability of the event determined by a reward function r on the states* is $\sum_{s \in DR(G)} \varphi(s) r(s)$, where $\forall s \in DR(G)$, $0 \leq r(s) \leq 1$.

Let $N = (P_N, T_N, W_N, \Omega_N, \mathcal{L}_N, M_N)$ be a LDTSIPN and $M, \widetilde{M} \in \mathbb{N}_{\mathrm{fin}}^{P_N}$. Then the average sojourn time $SJ(M)$, the sojourn time variance $VAR(M)$, the probabilities $PM^*(M, \widetilde{M})$, the transition relation $M \twoheadrightarrow_{\mathcal{P}} \widetilde{M}$, the EDTMC $EDTMC(N)$, the underlying SMC $SMC(N)$ and the steady-state PMF for it are defined like the corresponding notions for dynamic expressions. Since every marked plain dtsi-box could be interpreted as the LDTSIPN, we can evaluate performance with the LDTSIPNs corresponding to dtsi-boxes and then transfer the results to the latter.

Let $\simeq$ denote isomorphism between SMCs that binds their initial states, where two SMCs are isomorphic if their EDTMCs are so and the sojourn times in the isomorphic states are identically distributed.

Proposition 1. *For any static expression E, $SMC(\overline{E}) \simeq SMC(Box_{\mathrm{dtsi}}(\overline{E}))$.*

Proof. By Theorem 2, definitions of underlying SMCs for dynamic expressions and LDTSIPNs, and by the following. First, for the associated SMCs, the average sojourn time in the states is the same, since it is defined via the analogous probability functions. Second, the transition probabilities of the associated SMCs are the sums of those belonging to transition systems or reachability graphs. $\qquad\square$

Example 8. *Let E be from Example 5. In Figure 4, the underlying SMC $SMC(N)$ is presented. Clearly, $SMC(\overline{E})$ and $SMC(N)$ are isomorphic.*

5.2. **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 21. *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 $\mathcal{P} = PM(s, \tilde{s})$.*

Note that $EDTMC(G)$ is constructed from $DTMC(G)$ as follows. For each state of $DTMC(G)$, we remove a possible self-loop 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 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 $EDTMC(G)$ and/or $DTMC(G)$ 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 $\mathcal{P}_{ij}$ $(1 \leq i,j \leq n = |DR(G)|)$ of (one-step) transition probability matrix (TPM) $\mathbf{P}$ for $DTMC(G)$ are defined as

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

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 theorem below proposes the required equation.

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

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

Proposition 2. *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: $\forall s \in DR(G)$,*

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

Proof. See [84, 85]. $\qquad\square$

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

Proposition 3. *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 $\forall s \in DR(G)$,*

$$\varphi(s) = \begin{cases} \dfrac{\psi(s)}{\displaystyle\sum_{\tilde{s} \in DR_{\mathrm{T}}(G)} \psi(\tilde{s})}, & \text{if } s \in DR_{\mathrm{T}}(G); \\ 0, & \text{if } s \in DR_{\mathrm{V}}(G). \end{cases}$$

Proof. See [84, 85]. $\qquad\square$

Thus, to calculate φ, one can only apply normalization to some elements of ψ (corresponding to the tangible states), instead of abstracting from self-loops to get $\mathbf{P}^*$ and then ψ^*, followed by weighting by SJ and normalization. Hence, using $DTMC(G)$ instead of $EDTMC(G)$ allows one to avoid multistage analysis, but the payment for it is more time-consuming numerical and more complex analytical calculation of ψ w.r.t. ψ^*. The reason is that $DTMC(G)$ has self-loops, unlike $EDTMC(G)$, hence, the behaviour of $DTMC(G)$ stabilizes slower than that of $EDTMC(G)$ (if each of them has a single steady state) and $\mathbf{P}$ is denser matrix than $\mathbf{P}^*$, since $\mathbf{P}$ may additionally have non-zero elements at the main diagonal. Nevertheless, Proposition 3 is very important, since the relationship between φ and ψ it discovers will be used in Section 8 to prove preservation of the stationary behaviour by a stochastic equivalence.

Example 9. *Let E be from Example 5. In Figure 4, the DTMC $DTMC(\overline{E})$ is shown.*

The TPM for $DTMC(\overline{E})$ is $\mathbf{P} = \begin{pmatrix} 1-\rho & \rho & 0 & 0 & 0 \\ 0 & 1-\chi & \chi & 0 & 0 \\ 0 & 0 & 0 & \frac{l}{l+m} & \frac{m}{l+m} \\ 0 & \theta & 0 & 1-\theta & 0 \\ 0 & \phi & 0 & 0 & 1-\phi \end{pmatrix}$.
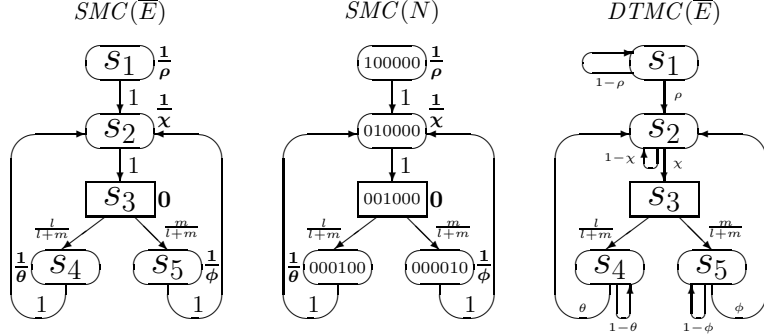
FIG. 4. The underlying SMCs of $\overline{E}$ and $N = Box_{\mathrm{dtsi}}(\overline{E})$ and DTMC of $\overline{E}$ for $E = [(\{a\}, \rho) * ((\{b\}, \chi); ((((\{c\}, \natural_l); (\{d\}, \theta))[]$
$(((\{e\}, \natural_m); (\{f\}, \phi)))) * \mathsf{Stop}]$

The steady-state PMF for $DTMC(\overline{E})$ is
$\psi = \frac{1}{\theta\phi(1+\chi)(l+m)+\chi(\phi l+\theta m)}(0, \theta\phi(l+m), \chi\theta\phi(l+m), \chi\phi l, \chi\theta m)$.
Since $DR_\mathrm{T}(\overline{E}) = \{s_1, s_2, s_4, s_5\}$ and $DR_\mathrm{V}(\overline{E}) = \{s_3\}$, we have $\sum_{\tilde{s} \in DR_\mathrm{T}(\overline{E})} \psi(\tilde{s}) =$
$\psi(s_1) + \psi(s_2) + \psi(s_4) + \psi(s_5) = \frac{\theta\phi(l+m)+\chi(\phi l+\theta m)}{\theta\phi(1+\chi)(l+m)+\chi(\phi l+\theta m)}$. By Proposition 3,

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

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

$\varphi(s_3) = 0,$

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

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

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

$\varphi = \frac{1}{\theta\phi(l+m)+\chi(\phi l+\theta m)}(0, \theta\phi(l+m), 0, \chi\phi l, \chi\theta m)$. This coincides with the result obtained in Example 7 with the use of ψ^* and SJ.

6. Stochastic equivalences

Consider the expressions $E = (\{a\}, \frac{1}{2})$ and $E' = (\{a\}, \frac{1}{3})_1 [] (\{a\}, \frac{1}{3})_2$, for which $\overline{E} \neq_\mathrm{ts} \overline{E'}$, since $TS(\overline{E})$ has only one transition from the initial to the final state (with probability $\frac{1}{2}$) while $TS(\overline{E'})$ has two such ones (with probabilities $\frac{1}{4}$). On the other hand, all the mentioned transitions are labeled by activities with the same multiaction part $\{a\}$. Next, the overall probabilities of the mentioned transitions of $TS(\overline{E})$ and $TS(\overline{E'})$ coincide: $\frac{1}{2} = \frac{1}{4} + \frac{1}{4}$. Further, $TS(\overline{E})$ (as well as $TS(\overline{E'})$) has one empty loop transition from the initial state to itself with probability $\frac{1}{2}$ and one empty loop transition from the final state to itself with probability 1. The empty loop transitions are labeled by the empty set of activities. For calculating the transition probabilities of $TS(\overline{E'})$, take $\rho = \chi = \frac{1}{3}$ in Example 4. Then you will see that the probability parts $\frac{1}{3}$ and $\frac{1}{3}$ of the activities $(\{a\}, \frac{1}{3})_1$ and $(\{a\}, \frac{1}{3})_2$ are "splitted" among probabilities $\frac{1}{4}$ and $\frac{1}{4}$ of the corresponding transitions and the probability $\frac{1}{2}$ of the empty loop transition. Unlike $=_\mathrm{ts}$, most of the probabilistic and stochastic equivalences proposed in the literature do not differentiate between the processes such as those specified by E and E'. In Figure 5(a), the marked

dtsi-boxes corresponding to the dynamic expressions $\overline{E}$ and $\overline{E'}$ are presented, i.e. $N = Box_{dtsi}(\overline{E})$ and $N' = Box_{dtsi}(\overline{E'})$.

Since the semantic equivalence $=_{\mathrm{ts}}$ is too discriminating in many cases, we need weaker equivalence notions. These equivalences should possess the following necessary properties. First, any two equivalent processes must have the same sequences of multisets of multiactions, which are the multiaction parts of the activities executed in steps starting from the initial states of the processes. Second, for every such sequence, its execution probabilities within both processes must coincide. Third, the desired equivalence should preserve the branching structure of computations, i.e. the points of choice of an external observer between several extensions of a particular computation should be taken into account. In this section, we define one such notion: step stochastic bisimulation equivalence.

6.1. **Step stochastic bisimulation equivalence.** Bisimulation equivalences respect the particular points of choice in the behaviour of a system. To define stochastic bisimulation equivalences, we consider a bisimulation as an *equivalence* relation that partitions the states of the *union* of the transition systems $TS(G)$ and $TS(G')$ of two dynamic expressions G and G' to be compared. For G and G' to be bisimulation equivalent, the initial states $[G]_{\approx}$ and $[G']_{\approx}$ of their transition systems should be related by a bisimulation having the following transfer property: if two states are related then in each of them the same multisets of multiactions can occur, leading with the identical overall probability from each of the two states to *the same equivalence class* for every such multiset.

Thus, we follow the approaches of [50, 57, 44, 46, 14, 10, 11], but we implement step semantics instead of interleaving one considered in these papers. We use the generative probabilistic transition systems, like in [50], in contrast to the reactive model, treated in [57], and we take transition probabilities instead of transition rates from [44, 46, 14, 10, 11]. Hence, step stochastic bisimulation equivalence that define further is (in a probability sense) comparable only with interleaving probabilistic bisimulation one from [50], and our equivalence is obviously stronger.

In the definition below, we consider $\mathcal{L}(\Upsilon) \in \mathbb{N}_{\mathrm{fin}}^{\mathcal{L}}$ for $\Upsilon \in \mathbb{N}_{\mathrm{fin}}^{\mathcal{SIL}}$, i.e. (possibly empty) multisets of multiactions. The multiactions can be empty as well. In this case, $\mathcal{L}(\Upsilon)$ contains the elements $\emptyset$, but it is not empty itself.

Let G be a dynamic expression and $\mathcal{H} \subseteq DR(G)$. For any $s \in DR(G)$ and $A \in \mathbb{N}_{\mathrm{fin}}^{\mathcal{L}}$, we write $s \xrightarrow{A}_{\mathcal{P}} \mathcal{H}$, where $\mathcal{P} = PM_A(s, \mathcal{H})$ is the *overall probability to move from s into the set of states $\mathcal{H}$ via steps with the multiaction part A* defined as

$$PM_A(s, \mathcal{H}) = \sum_{\{\Upsilon | \exists \tilde{s} \in \mathcal{H}, \ s \xrightarrow{\Upsilon} \tilde{s}, \ \mathcal{L}(\Upsilon)=A\}} PT(\Upsilon, s).$$

We write $s \xrightarrow{A} \mathcal{H}$ if $\exists \mathcal{P}, \ s \xrightarrow{A}_{\mathcal{P}} \mathcal{H}$. Further, we write $s \rightarrow_{\mathcal{P}} \mathcal{H}$ if $\exists A, \ s \xrightarrow{A} \mathcal{H}$, where $\mathcal{P} = PM(s, \mathcal{H})$ is the *overall probability to move from s into the set of states $\mathcal{H}$ via any steps* defined as

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

For $\tilde{s} \in DR(G)$, we write $s \xrightarrow{A}_{\mathcal{P}} \tilde{s}$ if $s \xrightarrow{A}_{\mathcal{P}} \{\tilde{s}\}$ and $s \xrightarrow{A} \tilde{s}$ if $\exists \mathcal{P} \ s \xrightarrow{A}_{\mathcal{P}} \tilde{s}$.

To introduce a stochastic bisimulation between dynamic expressions G and G', we should consider the "composite" set of states $DR(G) \cup DR(G')$, since we have to identify the probabilities to come from any two equivalent states into the same "composite" equivalence class (w.r.t. the stochastic bisimulation). For $G \neq G'$, transitions starting from the states of $DR(G)$ (or $DR(G')$) always lead to those from the same set, since $DR(G) \cap DR(G') = \emptyset$, allowing us to "mix" the sets of states in the definition of stochastic bisimulation.

Definition 22. *Let G and G' be dynamic expressions. An* equivalence *relation* $\mathcal{R} \subseteq (DR(G) \cup DR(G'))^2$ *is a* step stochastic bisimulation *between G and G', denoted by $\mathcal{R} : G \underline{\leftrightarrow}_{ss} G'$, if:*

(1) $([G]_{\approx}, [G']_{\approx}) \in \mathcal{R}$.
(2) $(s_1, s_2) \in \mathcal{R} \Rightarrow \forall \mathcal{H} \in (DR(G) \cup DR(G'))/_{\mathcal{R}}, \ \forall A \in \mathbb{N}_{\text{fin}}^{\mathcal{L}}$,
$\quad s_1 \xrightarrow{A}_{\mathcal{P}} \mathcal{H} \Leftrightarrow s_2 \xrightarrow{A}_{\mathcal{P}} \mathcal{H}$.

Two dynamic expressions G and G' are step stochastic bisimulation equivalent, *denoted by $G \underline{\leftrightarrow}_{ss} G'$, if $\exists \mathcal{R} : G \underline{\leftrightarrow}_{ss} G'$.*

We now define the multiaction transition systems, whose transitions are labeled with the multisets of multiactions, extracted from the corresponding activities.

Definition 23. *Let G be a dynamic expression. The* (labeled probabilistic) multiaction transition system *of G is a quadruple $TS_{\mathcal{L}}(G) = (S_{\mathcal{L}}, L_{\mathcal{L}}, \mathcal{T}_{\mathcal{L}}, s_{\mathcal{L}})$, where*

- $S_{\mathcal{L}} = DR(G)$;
- $L_{\mathcal{L}} = \mathbb{N}_{fin}^{\mathcal{L}} \times (0; 1]$;
- $\mathcal{T}_{\mathcal{L}} = \{(s, (A, PM_A(s, \{\tilde{s}\})), \tilde{s}) \mid s, \tilde{s} \in DR(G), \ s \xrightarrow{A} \tilde{s}\}$;
- $s_{\mathcal{L}} = [G]_{\approx}$.

The transition $(s, (A, \mathcal{P}), \tilde{s}) \in \mathcal{T}_{\mathcal{L}}$ will be written as $s \xrightarrow{A}_{\mathcal{P}} \tilde{s}$.

The multiaction transition systems of static expressions can be defined as well. For $E \in RegStatExpr$ let $TS_{\mathcal{L}}(E) = TS_{\mathcal{L}}(\overline{E})$.

Let G and G' be dynamic expressions and $\mathcal{R} : G \underline{\leftrightarrow}_{ss} G'$. Then the relation $\mathcal{R}$ can be interpreted as a step stochastic bisimulation between the transition systems $TS_{\mathcal{L}}(G)$ and $TS_{\mathcal{L}}(G')$, denoted by $\mathcal{R} : TS_{\mathcal{L}}(G) \underline{\leftrightarrow}_{ss} TS_{\mathcal{L}}(G')$, which is defined by analogy (excepting step semantics) with interleaving probabilistic bisimulation on generative probabilistic transition systems from [50].

Example 10. *Let F be an abstraction of the static expression E from Example 5, with $c = e$, $d = f$, $\theta = \phi$, i.e. $F = [(\{a\}, \rho) * ((\{b\}, \chi); (((\{c\}, \natural_l); (\{d\}, \theta)_1)[](\{c\}, \natural_m); (\{d\}, \theta)_2))) * \mathsf{Stop}]$. Then $DR(\overline{F}) = \{s_1, s_2, s_3, s_4, s_5\}$ is obtained from $DR(\overline{E})$ via substitution of the symbols e, f, ϕ by c, d, θ, respectively, in the specifications of the corresponding states from the latter set. We have $DR_T(\overline{F}) = \{s_1, s_2, s_4, s_5\}$ and $DR_V(\overline{F}) = \{s_3\}$. In Figure 6, the multiaction transition system $TS_{\mathcal{L}}(\overline{F})$ is presented. The singleton multisets of multiactions are written without outer braces.*

The following proposition states that every step stochastic bisimulation binds tangible states only with tangible ones and the same is valid for vanishing states.

Proposition 4. *Let G and G' be dynamic expressions and $\mathcal{R} : G \underline{\leftrightarrow}_{ss} G'$. Then*

$$\mathcal{R} \subseteq (DR_T(G) \cup DR_T(G'))^2 \uplus (DR_V(G) \cup DR_V(G'))^2.$$

Proof. By definition of transition systems of expressions, for every tangible state, there is an empty loop from it, and no empty loop transitions are possible from vanishing states. Further, $\mathcal{R}$ preserves empty loops. To verify this, first take $A = \emptyset$ in its definition to get $\forall (s_1, s_2) \in \mathcal{R}$, $\forall \mathcal{H} \in (DR(G) \cup DR(G'))/_{\mathcal{R}}$, $s_1 \xrightarrow{\emptyset}_{\mathcal{P}} \mathcal{H} \Leftrightarrow s_2 \xrightarrow{\emptyset}_{\mathcal{P}} \mathcal{H}$, and then observe that the empty loop transition from a state leads only to the same state. $\square$

Let $\mathcal{R}_{\mathrm{ss}}(G, G') = \bigcup \{ \mathcal{R} \mid \mathcal{R} : G \underleftrightarrow{}_{\mathrm{ss}} G' \}$ be the *union of all step stochastic bisimulations* between G and G'. The next proposition proves that $\mathcal{R}_{\mathrm{ss}}(G, G')$ is also an *equivalence* and $\mathcal{R}_{\mathrm{ss}}(G, G') : G \underleftrightarrow{}_{\mathrm{ss}} G'$.

Proposition 5. *Let G and G' be dynamic expressions and $G \underleftrightarrow{}_{\mathrm{ss}} G'$. Then $\mathcal{R}_{\mathrm{ss}}(G, G')$ is the largest step stochastic bisimulation between G and G'.*

Proof. See Appendix A.1. $\square$

In [3], an algorithm for strong probabilistic bisimulation on labeled probabilistic transition systems (a reformulation of probabilistic automata) was proposed with time complexity $O(n^2 m)$, where n is the number of states and m is the number of transitions. In [4], a decision algorithm for strong probabilistic bisimulation on generative labeled probabilistic transition systems was constructed with time complexity $O(m \log n)$ and space complexity $O(m + n)$. In [30], a polynomial algorithm for strong probabilistic bisimulation on probabilistic automata was presented. The mentioned algorithms for interleaving probabilistic bisimulation equivalence can be adapted for $\underleftrightarrow{}_{\mathrm{ss}}$ using the method from [49], applied to get the decidability results for step bisimulation equivalence. The method respects that transition systems in interleaving and step semantics differ only by availability of the additional transitions corresponding to parallel execution of activities in the latter (our case).

6.2. **Interrelations of the stochastic equivalences.** We now compare the discrimination power of the stochastic equivalences.

Theorem 3. *For dynamic expressions G, G' the next* strict *implications hold:*

$$G \approx G' \;\Rightarrow\; G =_{\mathrm{ts}} G' \;\Rightarrow\; G \underleftrightarrow{}_{\mathrm{ss}} G'.$$

Proof. Let us check the validity of the implications.

- The implication $=_{\mathrm{ts}} \Rightarrow \underleftrightarrow{}_{\mathrm{ss}}$ is proved as follows. Let $\beta : G =_{\mathrm{ts}} G'$. Then it is easy to see that $\mathcal{R} : G \underleftrightarrow{}_{\mathrm{ss}} G'$, where $\mathcal{R} = \{(s, \beta(s)) \mid s \in DR(G)\}$.
- The implication $\approx \Rightarrow =_{\mathrm{ts}}$ is valid, since the transition system of a dynamic formula is defined based on its structural equivalence class.

Let us see that that the implications are strict, by the following counterexamples.

(a) Let $E = (\{a\}, \frac{1}{2})$ and $E' = (\{a\}, \frac{1}{3})_1[](\{a\}, \frac{1}{3})_2$. Then $\overline{E} \underleftrightarrow{}_{\mathrm{ss}} \overline{E'}$, but $\overline{E} \neq_{\mathrm{ts}} \overline{E'}$, since $TS(\overline{E})$ has only one transition from the initial to the final state while $TS(\overline{E'})$ has two such ones.

(b) Let $E = (\{a\}, \frac{1}{2}); (\{\hat{a}\}, \frac{1}{2})$ and $E' = ((\{a\}, \frac{1}{2}); (\{\hat{a}\}, \frac{1}{2}))$ sy a. Then $\overline{E} =_{\mathrm{ts}} \overline{E'}$, but $\overline{E} \not\approx \overline{E'}$, since $\overline{E}$ and $\overline{E'}$ cannot be reached from each other by inaction rules. $\square$

Example 11. *In Figure 5, the marked dtsi-boxes corresponding to the dynamic expressions from examples of Theorem 3 are presented, i.e. $N = Box_{\mathrm{dtsi}}(\overline{E})$ and $N' = Box_{\mathrm{dtsi}}(\overline{E'})$ for each picture (a)–(b).*
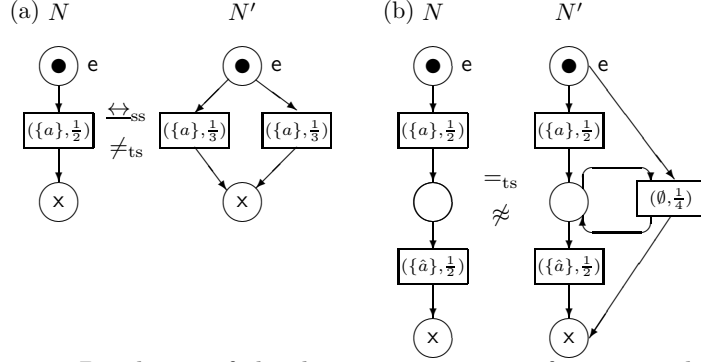
Fig. 5. Dtsi-boxes of the dynamic expressions from equivalence examples of Theorem 3

7. Reduction modulo equivalences

The proposed equivalences can be used to reduce transition systems and SMCs of expressions (reachability graphs and SMCs of dtsi-boxes). Reductions of graph-based models, like transition systems, reachability graphs and SMCs, result in those with less states (the graph nodes). The goal of the reduction is to decrease the number of states in the semantic representation of the modeled system while preserving its important qualitative and quantitative properties. Thus, the reduction allows one to simplify the behavioural and performance analysis of systems.

An *autobisimulation* is a bisimulation between an expression and itself. For a dynamic expression G and a step stochastic autobisimulation on it $\mathcal{R} : G\underline{\leftrightarrow}_{\mathrm{ss}}G$, let $\mathcal{K} \in DR(G)/_{\mathcal{R}}$ and $s_1, s_2 \in \mathcal{K}$. We have $\forall \widetilde{\mathcal{K}} \in DR(G)/_{\mathcal{R}}$, $\forall A \in \mathbb{N}_{\mathrm{fin}}^{\mathcal{L}}$, $s_1 \xrightarrow{A}_{\mathcal{P}} \widetilde{\mathcal{K}} \Leftrightarrow s_2 \xrightarrow{A}_{\mathcal{P}} \widetilde{\mathcal{K}}$. The previous equality is valid for all $s_1, s_2 \in \mathcal{K}$, hence, we can rewrite it as $\mathcal{K} \xrightarrow{A}_{\mathcal{P}} \widetilde{\mathcal{K}}$, where $\mathcal{P} = PM_A(\mathcal{K}, \widetilde{\mathcal{K}}) = PM_A(s_1, \widetilde{\mathcal{K}}) = PM_A(s_2, \widetilde{\mathcal{K}})$. We write $\mathcal{K} \xrightarrow{A} \widetilde{\mathcal{K}}$ if $\exists \mathcal{P}$, $\mathcal{K} \xrightarrow{A}_{\mathcal{P}} \widetilde{\mathcal{K}}$ and $\mathcal{K} \to \widetilde{\mathcal{K}}$ if $\exists A$, $\mathcal{K} \xrightarrow{A} \widetilde{\mathcal{K}}$. The similar arguments allow us to write $\mathcal{K} \to_{\mathcal{P}} \widetilde{\mathcal{K}}$, where $\mathcal{P} = PM(\mathcal{K}, \widetilde{\mathcal{K}}) = PM(s_1, \widetilde{\mathcal{K}}) = PM(s_2, \widetilde{\mathcal{K}})$.

By Proposition 4, $\mathcal{R} \subseteq (DR_{\mathrm{T}}(G))^2 \uplus (DR_{\mathrm{V}}(G))^2$. Hence, $\forall \mathcal{K} \in DR(G)/_{\mathcal{R}}$, all states from $\mathcal{K}$ are tangible if $\mathcal{K} \in DR_{\mathrm{T}}(G)/_{\mathcal{R}}$, or vanishing if $\mathcal{K} \in DR_{\mathrm{V}}(G)/_{\mathcal{R}}$.

The *average sojourn time in the equivalence class (w.r.t. $\mathcal{R}$) of states $\mathcal{K}$* is

$$SJ_{\mathcal{R}}(\mathcal{K}) = \begin{cases} \frac{1}{1-PM(\mathcal{K},\mathcal{K})}, & \text{if } \mathcal{K} \in DR_{\mathrm{T}}(G)/_{\mathcal{R}}; \\ 0, & \text{if } \mathcal{K} \in DR_{\mathrm{V}}(G)/_{\mathcal{R}}. \end{cases}$$

The *average sojourn time vector for the equivalence classes (w.r.t. $\mathcal{R}$) of states $SJ_{\mathcal{R}}$* of G has the elements $SJ_{\mathcal{R}}(\mathcal{K})$, $\mathcal{K} \in DR(G)/_{\mathcal{R}}$.

The *sojourn time variance in the equivalence class (w.r.t. $\mathcal{R}$) of states $\mathcal{K}$* is

$$VAR_{\mathcal{R}}(\mathcal{K}) = \begin{cases} \frac{PM(\mathcal{K},\mathcal{K})}{(1-PM(\mathcal{K},\mathcal{K}))^2}, & \text{if } \mathcal{K} \in DR_{\mathrm{T}}(G)/_{\mathcal{R}}; \\ 0, & \text{if } \mathcal{K} \in DR_{\mathrm{V}}(G)/_{\mathcal{R}}. \end{cases}$$

The *sojourn time variance vector for the equivalence classes (w.r.t. $\mathcal{R}$) of states $VAR_{\mathcal{R}}$* of G has the elements $VAR_{\mathcal{R}}(\mathcal{K})$, $\mathcal{K} \in DR(G)/_{\mathcal{R}}$.

Let $\mathcal{R}_{\mathrm{ss}}(G) = \bigcup\{\mathcal{R} \mid \mathcal{R} : G\underline{\leftrightarrow}_{\mathrm{ss}}G\}$ be the *union of all step stochastic autobisimulations* on G. By Proposition 5, $\mathcal{R}_{\mathrm{ss}}(G)$ is the largest step stochastic autobisimulation on G. Based on the equivalence classes w.r.t. $\mathcal{R}_{\mathrm{ss}}(G)$, the quotient (by $\underline{\leftrightarrow}_{\mathrm{ss}}$) transition

systems and the quotient (by $\underline{\leftrightarrow}_{ss}$) underlying SMCs of expressions can be defined. The mentioned equivalence classes become the quotient states. The average sojourn time in a quotient state is that in the corresponding equivalence class. Every quotient transition between two such composite states represents all steps (having the same multiaction part in case of the transition system quotient) from the first state to the second one.

Definition 24. *Let G be a dynamic expression. The* quotient (by $\underline{\leftrightarrow}_{ss}$) (labeled probabilistic) transition system *of G is a quadruple*
$TS_{\underline{\leftrightarrow}_{ss}}(G) = (S_{\underline{\leftrightarrow}_{ss}}, L_{\underline{\leftrightarrow}_{ss}}, \mathcal{T}_{\underline{\leftrightarrow}_{ss}}, s_{\underline{\leftrightarrow}_{ss}})$, *where*

- $S_{\underline{\leftrightarrow}_{ss}} = DR(G)/_{\mathcal{R}_{ss}(G)}$;
- $L_{\underline{\leftrightarrow}_{ss}} = \mathbb{N}^{\mathcal{L}}_{\text{fin}} \times (0;1]$;
- $\mathcal{T}_{\underline{\leftrightarrow}_{ss}} = \{(\mathcal{K}, (A, PM_A(\mathcal{K}, \widetilde{\mathcal{K}})), \widetilde{\mathcal{K}}) \mid \mathcal{K}, \widetilde{\mathcal{K}} \in DR(G)/_{\mathcal{R}_{ss}(G)}, \ \mathcal{K} \xrightarrow{A} \widetilde{\mathcal{K}}\}$;
- $s_{\underline{\leftrightarrow}_{ss}} = [[G]_{\approx}]_{\mathcal{R}_{ss}(G)}$.

The transition $(\mathcal{K}, (A, \mathcal{P}), \widetilde{\mathcal{K}}) \in \mathcal{T}_{\underline{\leftrightarrow}_{ss}}$ will be written as $\mathcal{K} \xrightarrow{A}_{\mathcal{P}} \widetilde{\mathcal{K}}$.

Let G be a dynamic expression. We define the relation $\mathcal{R}_{\mathcal{L}ss}(G) = \{(s, \mathcal{K}), (\mathcal{K}, s) \mid s \in \mathcal{K} \in DR(G)/_{\mathcal{R}_{ss}(G)}\}^+$, where $^+$ is the transitive closure operation. One can see that $\mathcal{R}_{\mathcal{L}ss}(G) \subseteq (DR(G) \cup DR(G)/_{\mathcal{R}_{ss}(G)})^2$ is an equivalence relation that partitions the set $DR(G) \cup DR(G)/_{\mathcal{R}_{ss}(G)}$ to the equivalence classes $\mathcal{L}_1, \ldots, \mathcal{L}_n$, defined as $\mathcal{L}_i = \mathcal{K}_i \cup \{\mathcal{K}_i\}$ ($1 \le i \le n$), where $DR(G)/_{\mathcal{R}_{ss}(G)} = \{\mathcal{K}_1, \ldots, \mathcal{K}_n\}$. The relation $\mathcal{R}_{\mathcal{L}ss}(G)$ can be interpreted as a step stochastic bisimulation between the transition systems $TS_{\mathcal{L}}(G)$ and $TS_{\underline{\leftrightarrow}_{ss}}(G)$, denoted by $\mathcal{R}_{\mathcal{L}ss}(G) : TS_{\mathcal{L}}(G) \underline{\leftrightarrow}_{ss} TS_{\underline{\leftrightarrow}_{ss}}(G)$, which is defined by analogy (excepting step semantics) with interleaving probabilistic bisimulation on generative probabilistic transition systems from [50]. It is clear that from this viewpoint, $\mathcal{R}_{\mathcal{L}ss}(G)$ is also the union of all step stochastic bisimulations and largest step stochastic bisimulation between $TS_{\mathcal{L}}(G)$ and $TS_{\underline{\leftrightarrow}_{ss}}(G)$.

Example 12. *Let F be from Example 10. Then $DR(\overline{F})/_{\mathcal{R}_{ss}(\overline{F})} = \{\mathcal{K}_1, \mathcal{K}_2, \mathcal{K}_3, \mathcal{K}_4\}$, where $\mathcal{K}_1 = \{s_1\}$, $\mathcal{K}_2 = \{s_2\}$, $\mathcal{K}_3 = \{s_3\}$, $\mathcal{K}_4 = \{s_4, s_5\}$. We also have $DR_T(\overline{F})/_{\mathcal{R}_{ss}(\overline{F})} = \{\mathcal{K}_1, \mathcal{K}_2, \mathcal{K}_4\}$ and $DR_V(\overline{F})/_{\mathcal{R}_{ss}(\overline{F})} = \{\mathcal{K}_3\}$. In Figure 6, the quotient transition system $TS_{\underline{\leftrightarrow}_{ss}}(\overline{F})$ is presented.*

The *quotient (by $\underline{\leftrightarrow}_{ss}$)* average sojourn time vector *of G is defined as $SJ_{\underline{\leftrightarrow}_{ss}} = SJ_{\mathcal{R}_{ss}(G)}$.* The *quotient (by $\underline{\leftrightarrow}_{ss}$)* sojourn time variance vector *of G is defined as $VAR_{\underline{\leftrightarrow}_{ss}} = VAR_{\mathcal{R}_{ss}(G)}$.*

Let $\mathcal{K} \to \widetilde{\mathcal{K}}$ and $\mathcal{K} \ne \widetilde{\mathcal{K}}$. The *probability to move from $\mathcal{K}$ to $\widetilde{\mathcal{K}}$ by executing any set of activities after possible self-loops* is

$$PM^*(\mathcal{K}, \widetilde{\mathcal{K}}) = \begin{cases} PM(\mathcal{K}, \widetilde{\mathcal{K}}) \sum_{k=0}^{\infty} PM(\mathcal{K}, \mathcal{K})^k = \frac{PM(\mathcal{K}, \widetilde{\mathcal{K}})}{1 - PM(\mathcal{K}, \mathcal{K})}, & \text{if } \mathcal{K} \to \mathcal{K}; \\ PM(\mathcal{K}, \widetilde{\mathcal{K}}), & \text{otherwise.} \end{cases}$$

The value $k = 0$ in the summation above corresponds to the case with no self-loops. Note that $\forall \mathcal{K} \in DR_T(G)/_{\mathcal{R}_{ss}(G)}$, $PM^*(\mathcal{K}, \widetilde{\mathcal{K}}) = SJ_{\underline{\leftrightarrow}_{ss}}(\mathcal{K})PM(\mathcal{K}, \widetilde{\mathcal{K}})$, since we always have the empty loop (self-loop) $\mathcal{K} \xrightarrow{\emptyset} \mathcal{K}$ from every equivalence class of tangible states $\mathcal{K}$. Empty loops are not possible from equivalence classes of vanishing states, hence, $\forall \mathcal{K} \in DR_V(G)/_{\mathcal{R}_{ss}(G)}$, $PM^*(\mathcal{K}, \widetilde{\mathcal{K}}) = \frac{PM(\mathcal{K}, \widetilde{\mathcal{K}})}{1 - PM(\mathcal{K}, \mathcal{K})}$, when there are

non-empty self-loops (produced by iteration) from $\mathcal{K}$, or $PM^*(\mathcal{K}, \widetilde{\mathcal{K}}) = PM(\mathcal{K}, \widetilde{\mathcal{K}})$, when there are no self-loops from $\mathcal{K}$.

Definition 25. *Let G be a dynamic expression. The* quotient (by $\underline{\leftrightarrow}_{ss}$) EDTMC *of G, denoted by $EDTMC_{\underline{\leftrightarrow}_{ss}}(G)$, has the state space $DR(G)/_{\mathcal{R}_{ss}(G)}$, the initial state $[[G]_\approx]_{\mathcal{R}_{ss}(G)}$ and the transitions $\mathcal{K} \twoheadrightarrow_\mathcal{P} \widetilde{\mathcal{K}}$ if $\mathcal{K} \rightarrow \widetilde{\mathcal{K}}$ and $\mathcal{K} \neq \widetilde{\mathcal{K}}$, where $\mathcal{P} = PM^*(\mathcal{K}, \widetilde{\mathcal{K}})$. The* quotient (by $\underline{\leftrightarrow}_{ss}$) underlying SMC *of G, denoted by $SMC_{\underline{\leftrightarrow}_{ss}}(G)$, has the EDTMC $EDTMC_{\underline{\leftrightarrow}_{ss}}(G)$ and the sojourn time in every $\mathcal{K} \in DR_T(G)/_{\mathcal{R}_{ss}(G)}$ is geometrically distributed with the parameter $1 - PM(\mathcal{K}, \mathcal{K})$ while that in every $\mathcal{K} \in DR_V(G)/_{\mathcal{R}_{ss}(G)}$ is zero.*

The steady-state PMFs $\psi^*_{\underline{\leftrightarrow}_{ss}}$ for $EDTMC_{\underline{\leftrightarrow}_{ss}}(G)$ and $\varphi_{\underline{\leftrightarrow}_{ss}}$ for $SMC_{\underline{\leftrightarrow}_{ss}}(G)$ are defined like the corresponding notions ψ^* for $EDTMC(G)$ and φ for $SMC(G)$.

Example 13. *Let F be from Example 12. In Figure 6, the quotient underlying SMC $SMC_{\underline{\leftrightarrow}_{ss}}(\overline{F})$ is presented.*

The quotients of both transition systems and underlying SMCs are their minimal reductions modulo step stochastic bisimulations. The quotients simplify analysis of system properties, preserved by $\underline{\leftrightarrow}_{ss}$, since less states should be examined for it. Such reduction method resembles that from [2], based on place bisimulation equivalence for PNs, but the former method merges states while the latter merges places.

Moreover, there exist algorithms to construct the quotients of transition systems by an equivalence (like bisimulation one) [73] and those of (discrete or continuous time) Markov chains by ordinary lumping [34]. These algorithms have time complexity $O(m \log n)$ and space complexity $O(m + n)$, where n is the number of states and m is the number of transitions. As mentioned in [87], the algorithm from [34] can be easily adjusted to produce quotients of labeled probabilistic transition systems by the probabilistic bisimulation equivalence. In [87], the symbolic partition refinement algorithm on the state space of CTMCs was proposed. The algorithm can be applied to DTMCs and labeled probabilistic transition systems. Such a symbolic lumping is memory efficient due to compact representation of the state space partition. It is time efficient, since fast algorithm of the partition representation and refinement is applied. In [35], a polynomial-time algorithm for minimizing behaviour of probabilistic automata by probabilistic bisimulation equivalence was outlined that results in the canonical quotient structures. One could adapt the above algorithms for our framework.

Let us define the quotient (by $\underline{\leftrightarrow}_{ss}$) DTMCs of expressions, based on the probabilities $PM(\mathcal{K}, \widetilde{\mathcal{K}})$.

Definition 26. *Let G be a dynamic expression. The* quotient (by $\underline{\leftrightarrow}_{ss}$) DTMC *of G, denoted by $DTMC_{\underline{\leftrightarrow}_{ss}}(G)$, has the state space $DR(G)/_{\mathcal{R}_{ss}(G)}$, the initial state $[[G]_\approx]_{\mathcal{R}_{ss}(G)}$ and the transitions $\mathcal{K} \rightarrow_\mathcal{P} \widetilde{\mathcal{K}}$, where $\mathcal{P} = PM(\mathcal{K}, \widetilde{\mathcal{K}})$.*

The steady-state PMF $\psi_{\underline{\leftrightarrow}_{ss}}$ for $DTMC_{\underline{\leftrightarrow}_{ss}}(G)$ is defined like the corresponding notion ψ for $DTMC(G)$.

Example 14. *Let F be from Example 12. In Figure 6, the quotient DTMC $DTMC_{\underline{\leftrightarrow}_{ss}}(\overline{F})$ is presented.*

Clearly, the relationships between the steady-state PMFs $\psi_{\underline{\leftrightarrow}_{ss}}$ and $\psi^*_{\underline{\leftrightarrow}_{ss}}$, as well as $\varphi_{\underline{\leftrightarrow}_{ss}}$ and $\psi_{\underline{\leftrightarrow}_{ss}}$, are the same as those between their "non-quotient" versions in Proposition 2 and Proposition 3.
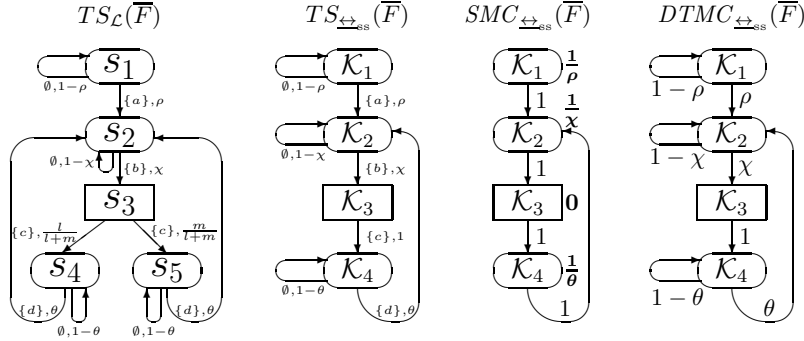
FIG. 6. The multiaction transition system, quotient transition system, quotient underlying SMC and quotient DTMC of $\overline{F}$ for $F = [(\{a\}, \rho) * ((\{b\}, \chi); (((\{c\}, \natural_l); (\{d\}, \theta)_1)[]((\{c\}, \natural_m); (\{d\}, \theta)_2))) * \mathsf{Stop}]$

The detailed illustrative quotient example will be presented in Section 9.

In [26], it is proved that irreducibility is preserved by aggregation w.r.t. any partition (or equivalence relation) on the states of finite DTMCs (so they are also positive recurrent). Aggregation decreases the number of states, hence, the aggregated DTMCs are also finite and positive recurrence is preserved by every aggregation. It is known [78, 55] that irreducible and positive recurrent DTMCs have a single stationary PMF. Note that the original and/or aggregated DTMCs may be periodic, thus having a unique stationary distribution, but no steady-state (limiting) one. For example, it may happen that the original DTMC is aperiodic while the aggregated DTMC is periodic due to merging some states of the former. Thus, both finite irreducible DTMCs and their arbitrary aggregates have a single stationary PMF. It is also shown in [26] that for every DTMC aggregated by ordinary lumpability, the stationary probability of each aggregate state is a sum of the stationary probabilities of all its constituent states from the original DTMC. The information about individual stationary probabilities of the original DTMC is lost after such a summation, but in many cases, the stationary probabilities of the aggregated DTMC are enough to calculate performance measures of the high-level model, from which the original DTMC is extracted. As mentioned in [26], in some applications, the aggregated DTMC can be extracted directly from the high-level model. Thus, the aggregation techniques, based on lumping, are of practical importance, since they allow one to reduce the state space of the modeled systems, hence, the computational costs for evaluating their performance.

Let G be a dynamic expression. By definition of $\underleftrightarrow{}_{ss}$, the relation $\mathcal{R}_{ss}(G)$ on $TS(G)$ induces ordinary lumping on $SMC(G)$, i.e. if the states of $TS(G)$ are related by $\mathcal{R}_{ss}(G)$ then the same states in $SMC(G)$ are related by ordinary lumping. The quotient (maximal aggregate) of $SMC(G)$ by such an induced ordinary lumping is $SMC_{\underleftrightarrow{}_{ss}}(G)$. Since we consider only finite SMCs, irreducibility of $SMC(G)$ will imply irreducibility of $SMC_{\underleftrightarrow{}_{ss}}(G)$ and they are positive recurrent. Then a unique quotient stationary PMF of $SMC_{\underleftrightarrow{}_{ss}}(G)$ can be calculated from a unique original stationary PMF of $SMC(G)$ by summing some elements of the latter, as described in [26]. Similar arguments show that the same holds for $DTMC(G)$ and $DTMC_{\underleftrightarrow{}_{ss}}(G)$.

8. STATIONARY BEHAVIOUR

Let us examine how the proposed equivalences can be used to compare the behaviour of stochastic processes in their steady states. We shall consider only formulas specifying stochastic processes with infinite behaviour, i.e. expressions with the iteration operator. Note that the iteration operator does not guarantee infiniteness of behaviour, since there can exist a deadlock (blocking) within the body (the second argument) of iteration when the corresponding subprocess does not reach its final state by some reasons. In particular, if the body of iteration contains the Stop expression then the iteration will be "broken". On the other hand, the iteration body can be left after a finite number of repeated executions and perform the iteration termination. To avoid executing activities after the iteration body, we take Stop as the termination argument of iteration.

Like in the framework of SMCs, in LDTSIPNs the most common systems for performance analysis are *ergodic* (irreducible, positive recurrent and aperiodic) ones. For ergodic LDTSIPNs, the steady-state marking probabilities exist and can be determined. In [69, 70], the following sufficient (but not necessary) conditions for ergodicity of DTSPNs are stated: *liveness* (for each transition and any reachable marking there exist a sequence of markings from it leading to the marking enabling that transition), *boundedness* (for any reachable marking the number of tokens in every place is not greater than some fixed number) and *nondeterminism* (the transition probabilities are strictly less than 1).

Consider dtsi-box of a dynamic expression $G = \overline{[E * F * \mathsf{Stop}]}$ specifying a process, which we assume has no deadlocks while performing F. If, starting in $[[E*\overline{F}*\mathsf{Stop}]]_{\approx}$ and ending in $[[E*\underline{F}*\mathsf{Stop}]]_{\approx}$, only tangible states are passed through, then the three ergodicity conditions are satisfied: the subnet corresponding to the looping of the iteration body F is live, safe (1-bounded) and nondeterministic (since all markings of the subnet are tangible and non-terminal, the probabilities of transitions from them are strictly less than 1). Hence, according to [69, 70], for the dtsi-box, its underlying SMC, restricted to the markings of the mentioned subnet, is ergodic. The isomorphism between SMCs of expressions and those of the corresponding dtsi-boxes, which is stated by Proposition 1, guarantees that $SMC(G)$ is ergodic, if restricted to the states between $[[E * \overline{F} * \mathsf{Stop}]]_{\approx}$ and $[[E * \underline{F} * \mathsf{Stop}]]_{\approx}$.

The ergodicity conditions above are not necessary, i.e. there exist dynamic expressions with vanishing states traversed while executing their iteration bodies, such that the properly restricted underlying SMCs are nevertheless ergodic, as Example 7 demonstrated. However, it has been shown in [7] that even live, safe and nondeterministic DTSPNs (as well as live and safe CTSPNs and GSPNs) may be non-ergodic.

In this section, we consider only the process expressions such that their underlying SMCs contain exactly one closed communication class of states, and this class should also be ergodic to ensure uniqueness of the stationary distribution, which is also the limiting one. The states not belonging to that class do not disturb the uniqueness, since the closed communication class is single, hence, they all are transient. Then, for each transient state, the steady-state probability to be in it is zero while the steady-state probability to enter into the ergodic class starting from that state is equal to one. 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.

8.1. **Steady state, residence time and equivalences.** The following proposition demonstrates that, for two dynamic expressions related by $\underline{\leftrightarrow}_{ss}$, the steady-state probabilities to enter into an equivalence class coincide, or the mean recurrence time for an equivalence class is the same for both expressions.

Proposition 6. *Let* G, G' *be dynamic expressions with* $\mathcal{R} : G\underline{\leftrightarrow}_{ss}G'$, φ *be the steady-state PMF for* $SMC(G)$ *and* φ' *be the steady-state PMF for* $SMC(G')$. *Then* $\forall \mathcal{H} \in (DR(G) \cup DR(G'))/_{\mathcal{R}}$,

$$\sum_{s \in \mathcal{H} \cap DR(G)} \varphi(s) = \sum_{s' \in \mathcal{H} \cap DR(G')} \varphi'(s').$$

Proof. See Appendix A.2. □

Let G be a dynamic expression, φ be the steady-state PMF for $SMC(G)$ and $\varphi_{\underline{\leftrightarrow}_{ss}}$ be the steady-state PMF for $SMC_{\underline{\leftrightarrow}_{ss}}(G)$. By Proposition 6 (modified for $\mathcal{R}_{\mathcal{L}ss}(G)$), we have $\forall \mathcal{K} \in DR(G)/_{\mathcal{R}_{ss}(G)}$, $\varphi_{\underline{\leftrightarrow}_{ss}}(\mathcal{K}) = \sum_{s \in \mathcal{K}} \varphi(s)$. Hence, using $SMC_{\underline{\leftrightarrow}_{ss}}(G)$ instead of $SMC(G)$ simplifies the analytical solution, since we have less states, but constructing the TPM for $EDTMC_{\underline{\leftrightarrow}_{ss}}(G)$, denoted by $\mathbf{P}^*_{\underline{\leftrightarrow}_{ss}}$, also requires some efforts, including determining $\mathcal{R}_{ss}(G)$ and calculating the probabilities to move from one equivalence class to other. The behaviour of $EDTMC_{\underline{\leftrightarrow}_{ss}}(G)$ stabilizes quicker than that of $EDTMC(G)$ (if each of them has a single steady state), since $\mathbf{P}^*_{\underline{\leftrightarrow}_{ss}}$ is denser matrix than $\mathbf{P}^*$ (the TPM for $EDTMC(G)$) due to the fact that the former matrix is smaller and the transitions between the equivalence classes "include" all the transitions between the states belonging to these equivalence classes.

By Proposition 6, $\underline{\leftrightarrow}_{ss}$ preserves the quantitative properties of the stationary behaviour (the level of SMCs). We now demonstrate that the qualitative properties, based on the multiaction labels, are preserved as well (the transition systems level).

Definition 27. *A derived step trace of a dynamic expression* G *is a chain* $\Sigma = A_1 \cdots A_n \in (\mathbb{N}_{fin}^{\mathcal{L}})^*$, *where* $\exists s \in DR(G)$, $s \xrightarrow{\Upsilon_1} s_1 \xrightarrow{\Upsilon_2} \cdots \xrightarrow{\Upsilon_n} s_n$, $\mathcal{L}(\Upsilon_i) = A_i$ $(1 \le i \le n)$. *Then the* probability to execute the derived step trace Σ in s is

$$PT(\Sigma, s) = \sum_{\{\Upsilon_1, \ldots, \Upsilon_n | s = s_0 \xrightarrow{\Upsilon_1} s_1 \xrightarrow{\Upsilon_2} \cdots \xrightarrow{\Upsilon_n} s_n, \; \mathcal{L}(\Upsilon_i) = A_i \; (1 \le i \le n)\}} \prod_{i=1}^{n} PT(\Upsilon_i, s_{i-1}).$$

The following theorem demonstrates that, for two dynamic expressions related by $\underline{\leftrightarrow}_{ss}$, the steady-state probabilities to enter into an equivalence class and start a derived step trace from it coincide.

Theorem 4. *Let* G, G' *be dynamic expressions with* $\mathcal{R} : G\underline{\leftrightarrow}_{ss}G'$, φ *be the steady-state PMF for* $SMC(G)$, φ' *be the steady-state PMF for* $SMC(G')$ *and* Σ *be a derived step trace of* G *and* G'. *Then* $\forall \mathcal{H} \in (DR(G) \cup DR(G'))/_{\mathcal{R}}$,

$$\sum_{s \in \mathcal{H} \cap DR(G)} \varphi(s)PT(\Sigma, s) = \sum_{s' \in \mathcal{H} \cap DR(G')} \varphi'(s')PT(\Sigma, s').$$

Proof. See Appendix A.3. □

Let G be a dynamic expression, φ be the steady-state PMF for $SMC(G)$, $\varphi_{\underline{\leftrightarrow}_{ss}}$ be the steady-state PMF for $SMC_{\underline{\leftrightarrow}_{ss}}(G)$ and Σ be a derived step trace of G. By Theorem 4 (modified for $\mathcal{R}_{\mathcal{L}ss}(G)$), we have $\forall \mathcal{K} \in DR(G)/_{\mathcal{R}_{ss}(G)}$, $\varphi_{\underline{\leftrightarrow}_{ss}}(\mathcal{K})PT(\Sigma, \mathcal{K}) = \sum_{s \in \mathcal{K}} \varphi(s)PT(\Sigma, s)$, where $\forall s \in \mathcal{K}$, $PT(\Sigma, \mathcal{K}) = PT(\Sigma, s)$.

We now present a result not concerning the steady-state probabilities, but revealing important properties of residence time in the equivalence classes. The next proposition demonstrates that, for two dynamic expressions related by $\underleftrightarrow{}_{ss}$, the sojourn time averages (and variances) in an equivalence class coincide.

Proposition 7. *Let G, G' be dynamic expressions with $\mathcal{R} : G\underleftrightarrow{}_{ss}G'$. Then $\forall \mathcal{H} \in (DR(G) \cup DR(G'))/_{\mathcal{R}}$,*

$$SJ_{\mathcal{R}\cap(DR(G))^2}(\mathcal{H} \cap DR(G)) = SJ_{\mathcal{R}\cap(DR(G'))^2}(\mathcal{H} \cap DR(G')),$$
$$VAR_{\mathcal{R}\cap(DR(G))^2}(\mathcal{H} \cap DR(G)) = VAR_{\mathcal{R}\cap(DR(G'))^2}(\mathcal{H} \cap DR(G')).$$

Proof. See Appendix A.4. $\square$

Example 15. *Let $E = [(\{a\}, \frac{1}{2}) * ((\{b\}, \frac{1}{2}); ((\{c\}, \frac{1}{3})_1[](\{c\}, \frac{1}{3})_2)) * \mathsf{Stop}]$,*
$E' = [(\{a\}, \frac{1}{2})*(((\{b\}, \frac{1}{3})_1; (\{c\}, \frac{1}{2})_1)[]((\{b\}, \frac{1}{3})_2; (\{c\}, \frac{1}{2})_2))*\mathsf{Stop}]$. *It holds $\overline{E}\underleftrightarrow{}_{ss}\overline{E'}$.*
$DR(\overline{E})$ *consists of the equivalence classes*

$s_1 = [[\overline{(\{a\}, \frac{1}{2})} * ((\{b\}, \frac{1}{2}); ((\{c\}, \frac{1}{3})_1[](\{c\}, \frac{1}{3})_2)) * \mathsf{Stop}]]_{\approx}$,

$s_2 = [[(\{a\}, \frac{1}{2}) * (\overline{(\{b\}, \frac{1}{2})}; ((\{c\}, \frac{1}{3})_1[](\{c\}, \frac{1}{3})_2)) * \mathsf{Stop}]]_{\approx}$,

$s_3 = [[(\{a\}, \frac{1}{2}) * ((\{b\}, \frac{1}{2}); \overline{((\{c\}, \frac{1}{3})_1[](\{c\}, \frac{1}{3})_2)}) * \mathsf{Stop}]]_{\approx}$.

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

$s_1' = [[\overline{(\{a\}, \frac{1}{2})} * (((\{b\}, \frac{1}{3})_1; (\{c\}, \frac{1}{2})_1)[]((\{b\}, \frac{1}{3})_2; (\{c\}, \frac{1}{2})_2)) * \mathsf{Stop}]]_{\approx}$,

$s_2' = [[(\{a\}, \frac{1}{2}) * \overline{(((\{b\}, \frac{1}{3})_1; (\{c\}, \frac{1}{2})_1)[]((\{b\}, \frac{1}{3})_2; (\{c\}, \frac{1}{2})_2))} * \mathsf{Stop}]]_{\approx}$,

$s_3' = [[(\{a\}, \frac{1}{2}) * (((\{b\}, \frac{1}{3})_1; \overline{(\{c\}, \frac{1}{2})_1})[]((\{b\}, \frac{1}{3})_2; (\{c\}, \frac{1}{2})_2)) * \mathsf{Stop}]]_{\approx}$,

$s_4' = [[(\{a\}, \frac{1}{2}) * (((\{b\}, \frac{1}{3})_1; (\{c\}, \frac{1}{2})_1)[]((\{b\}, \frac{1}{3})_2; \overline{(\{c\}, \frac{1}{2})_2})) * \mathsf{Stop}]]_{\approx}$.

The steady-state PMFs φ for $SMC(\overline{E})$ and φ' for $SMC(\overline{E'})$ are $\varphi = \left(0, \frac{1}{2}, \frac{1}{2}\right)$,
$\varphi' = \left(0, \frac{1}{2}, \frac{1}{4}, \frac{1}{4}\right)$.

Consider the equivalence class (w.r.t. $\mathcal{R}_{ss}(\overline{E}, \overline{E'})$) $\mathcal{H} = \{s_3, s_3', s_4'\}$. One can see that the steady-state probabilities for $\mathcal{H}$ coincide: $\sum_{s \in \mathcal{H} \cap DR(\overline{E})} \varphi(s) = \varphi(s_3) = \frac{1}{2} = \frac{1}{4} + \frac{1}{4} = \varphi'(s_3') + \varphi'(s_4') = \sum_{s' \in \mathcal{H} \cap DR(\overline{E'})} \varphi'(s')$. Let $\Sigma = \{\{c\}\}$. The steady-state probabilities to enter into the equivalence class $\mathcal{H}$ and start the derived step trace Σ from it coincide as well: $\varphi(s_3)(PT(\{(\{c\}, \frac{1}{3})_1\}, s_3) + PT(\{(\{c\}, \frac{1}{3})_2\}, s_3)) = \frac{1}{2}\left(\frac{1}{4} + \frac{1}{4}\right) = \frac{1}{4} = \frac{1}{4} \cdot \frac{1}{2} + \frac{1}{4} \cdot \frac{1}{2} = \varphi'(s_3')PT(\{(\{c\}, \frac{1}{2})_1\}, s_3') + \varphi'(s_4')PT(\{(\{c\}, \frac{1}{2})_2\}, s_4')$.

Further, the sojourn time averages in the equivalence class $\mathcal{H}$ coincide:
$SJ_{\mathcal{R}_{ss}(\overline{E},\overline{E'})\cap(DR(\overline{E}))^2}(\mathcal{H}\cap DR(G)) = SJ_{\mathcal{R}_{ss}(\overline{E},\overline{E'})\cap(DR(\overline{E}))^2}(\{s_3\}) = \frac{1}{1-PM(\{s_3\},\{s_3\})} = \frac{1}{1-PM(s_3,s_3)} = \frac{1}{1-\frac{1}{2}} = 2 = \frac{1}{1-\frac{1}{2}} = \frac{1}{1-PM(s_3',s_3')} = \frac{1}{1-PM(s_4',s_4')} = \frac{1}{1-PM(\{s_3',s_4'\},\{s_3',s_4'\})} = SJ_{\mathcal{R}_{ss}(\overline{E},\overline{E'})\cap(DR(\overline{E'}))^2}(\{s_3',s_4'\}) = SJ_{\mathcal{R}_{ss}(\overline{E},\overline{E'})\cap(DR(\overline{E'}))^2}(\mathcal{H} \cap DR(G'))$.

Finally, the sojourn time variances in the equivalence class $\mathcal{H}$ coincide:
$VAR_{\mathcal{R}_{ss}(\overline{E},\overline{E'})\cap(DR(\overline{E}))^2}(\mathcal{H} \cap DR(G)) = VAR_{\mathcal{R}_{ss}(\overline{E},\overline{E'})\cap(DR(\overline{E}))^2}(\{s_3\}) = \frac{PM(\{s_3\},\{s_3\})}{(1-PM(\{s_3\},\{s_3\}))^2} = \frac{PM(s_3,s_3)}{(1-PM(s_3,s_3))^2} = \frac{\frac{1}{2}}{\left(1-\frac{1}{2}\right)^2} = 2 = \frac{\frac{1}{2}}{\left(1-\frac{1}{2}\right)^2} = \frac{PM(s_3',s_3')}{(1-PM(s_3',s_3'))^2} = \frac{PM(s_4',s_4')}{(1-PM(s_4',s_4'))^2} = \frac{PM(\{s_3',s_4'\},\{s_3',s_4'\})}{(1-PM(\{s_3',s_4'\},\{s_3',s_4'\}))^2} = VAR_{\mathcal{R}_{ss}(\overline{E},\overline{E'})\cap(DR(\overline{E'}))^2}(\{s_3',s_4'\}) = VAR_{\mathcal{R}_{ss}(\overline{E},\overline{E'})\cap(DR(\overline{E'}))^2}(\mathcal{H} \cap DR(G'))$.

In Figure 7, the marked dtsi-boxes corresponding to the dynamic expressions above are presented, i.e. $N = Box_{dtsi}(\overline{E})$ and $N' = Box_{dtsi}(\overline{E'})$.
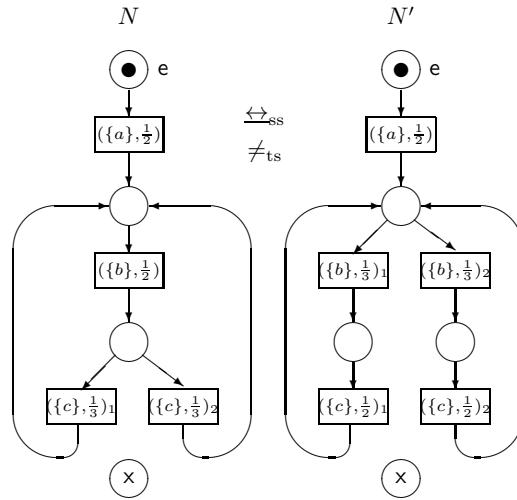
FIG. 7. $\underline{\leftrightarrow}_{ss}$ preserves steady-state behaviour and sojourn time properties in the equivalence classes

8.2. **Preservation of performance and simplification of its analysis.** Many performance indices are based on the steady-state probabilities to enter into a set of similar states or, after coming in it, to start a derived step trace from this set. The similarity of states is usually captured by an equivalence relation, hence, the sets are often the equivalence classes. Proposition 6, Theorem 4 and Proposition 7 guarantee coincidence of the mentioned indices for the expressions related by $\underline{\leftrightarrow}_{ss}$. Thus, $\underline{\leftrightarrow}_{ss}$ (hence, all the stronger equivalences considered) preserves performance of stochastic systems modeled by expressions of dtsiPBC.

It is also easier to evaluate performance using an SMC with less states, since in this case the size of the transition probability matrix is smaller, and we solve systems of less equations to calculate the steady-state probabilities. The reasoning above validates the following method of performance analysis simplification.

(1) The investigated system is specified by a static expression of dtsiPBC.
(2) The transition system of the expression is constructed.
(3) After treating the transition system for self-similarity, a step stochastic autobisimulation equivalence for the expression is determined.
(4) The quotient underlying SMC is derived from the quotient transition system.
(5) Stationary probabilities and performance indices are obtained from the SMC.

The limitation of the method above is its applicability only to the expressions such that their underlying SMCs contain exactly one closed communication class of states, and this class should also be ergodic to ensure uniqueness of the stationary distribution. If an SMC contains several closed communication classes of states that are all ergodic then several stationary distributions may exist, which depend on the initial PMF. There is an analytical method to determine stationary probabilities for SMCs of this kind as well [55]. Note that the underlying SMC of every process expression has only one initial PMF (that at the time moment 0), hence, the stationary distribution will be unique in this case too. The general steady-state probabilities are then calculated as the sum of the stationary probabilities of all the ergodic subsets of states, weighted by the probabilities to enter into these subsets,
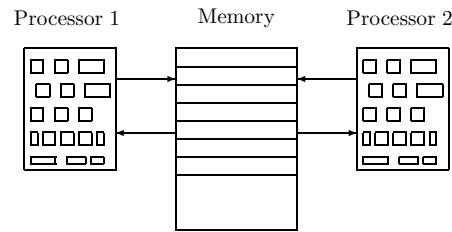
FIG. 8. The diagram of the shared memory system

starting from the initial state and passing through some transient states. It is worth applying the method only to the systems with similar subprocesses.

Before calculating stationary probabilities, we can further reduce the quotient underlying SMC, using the algorithm from [66, 5, 6] (employed within dtsiPBC in [85]) that eliminates vanishing states from the corresponding EDTMC and thereby decreases the size of its TPM. For SMCs reduction we can also apply an analogue of the deterministic barrier partitioning method from [40] for semi-Markov processes (SMPs), which allows one to perform quicker the first passage-time analysis. Another option is the method of stochastic state classes [48] for generalized SMPs (GSMPs) reduction that simplifies the transient performance analysis.

9. Generalized shared memory system

Let us consider a model of two processors accessing a common shared memory described in [66, 5, 6] in the continuous time setting on GSPNs. We shall analyze this shared memory system in the discrete time stochastic setting of dtsiPBC, where concurrent execution of activities is possible, while no two transitions of a GSPN may fire simultaneously (in parallel). Our model parameterizes that from [83]. The model behaves as follows. After activation of the system (turning the computer on), two processors are active, and the common memory is available. Each processor can request an access to the memory after which the instantaneous decision is made. When the decision is made in favour of a processor, it starts acquisition of the memory and the other processor should wait until the former one ends its memory operations, and the system returns to the state with both active processors and available common memory. The diagram of the system is depicted in Figure 8.

9.1. **The concrete system.** The meaning of actions from the dtsiPBC expressions which will specify the system modules is as follows. The action a corresponds to the system activation. The actions r_i $(1 \leq i \leq 2)$ represent the common memory request of processor i. The actions d_i correspond to the instantaneous decision on the memory allocation in favour of the processor i. The actions m_i represent the common memory access of processor i. The other actions are used for communication purposes only via synchronization, and we abstract from them later using restriction. For $a_1, \ldots, a_n \in Act$ $(n \in \mathbb{N})$, we shall abbreviate sy $a_1 \cdots$ sy a_n rs $a_1 \cdots$ rs a_n to sr $(a_1, \ldots, a_n)$.

We take general values for all multiaction probabilities and weights in the specification. Let all stochastic multiactions have the same generalized probability $\rho \in (0; 1)$ and all immediate ones have the same generalized weight $l \in \mathbb{R}_{>0}$. The resulting specification K of the generalized shared memory system is as follows.
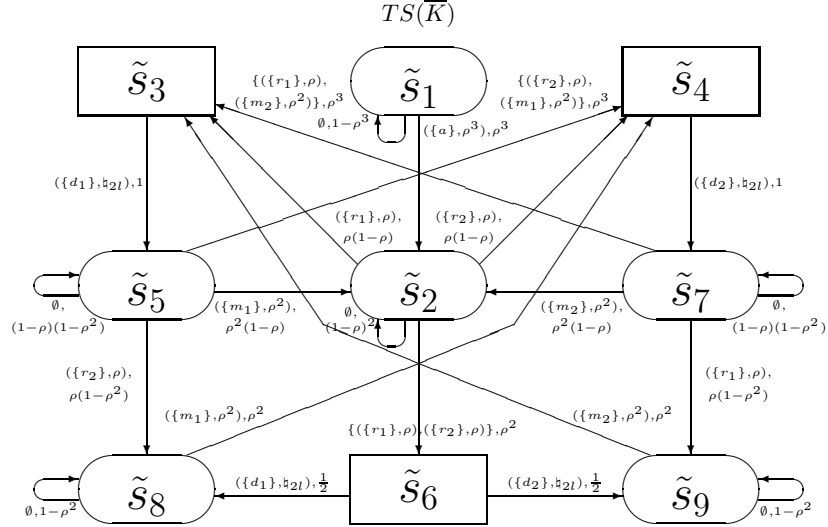
$$TS(\overline{K})$$

FIG. 9. The transition system of the generalized shared memory system

The static expression of the first processor is
$$K_1 = [(\{x_1\}, \rho) * ((\{r_1\}, \rho); (\{d_1, y_1\}, \natural_l); (\{m_1, z_1\}, \rho)) * \mathsf{Stop}].$$

The static expression of the second processor is
$$K_2 = [(\{x_2\}, \rho) * ((\{r_2\}, \rho); (\{d_2, y_2\}, \natural_l); (\{m_2, z_2\}, \rho)) * \mathsf{Stop}].$$

The static expression of the shared memory is
$$K_3 = [(\{a, \widehat{x_1}, \widehat{x_2}\}, \rho) * (((\{\widehat{y_1}\}, \natural_l); (\{\widehat{z_1}\}, \rho)) [] ((\{\widehat{y_2}\}, \natural_l); (\{\widehat{z_2}\}, \rho))) * \mathsf{Stop}].$$

The static expression of the generalized shared memory system is
$$K = (K_1 \| K_2 \| K_3) \text{ sr } (x_1, x_2, y_1, y_2, z_1, z_2).$$

As a result of the synchronization of immediate multiactions $(\{d_i, y_i\}, \natural_l)$ and $(\{\widehat{y_i}\}, \natural_l)$ we get $(\{d_i\}, \natural_{2l})$ $(1 \le i \le 2)$. The synchronization of stochastic multiactions $(\{m_i, z_i\}, \rho)$ and $(\{\widehat{z_i}\}, \rho)$ produces $(\{m_i\}, \rho^2)$ $(1 \le i \le 2)$. The result of synchronization of $(\{a, \widehat{x_1}, \widehat{x_2}\}, \rho)$ with $(\{x_1\}, \rho)$ is $(\{a, \widehat{x_2}\}, \rho^2)$, and that of synchronization of $(\{a, \widehat{x_1}, \widehat{x_2}\}, \rho)$ with $(\{x_2\}, \rho)$ is $(\{a, \widehat{x_1}\}, \rho^2)$. After applying synchronization to $(\{a, \widehat{x_2}\}, \rho^2)$ and $(\{x_2\}, \rho)$, as well as to $(\{a, \widehat{x_1}\}, \rho^2)$ and $(\{x_1\}, \rho)$, we get the same activity $(\{a\}, \rho^3)$.

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

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

In Figure 9, the transition system $TS(\overline{K})$ is presented. In Figure 10, the underlying SMC $SMC(\overline{K})$ is depicted. Note that, in step semantics, we may execute the following activities in parallel: $(\{r_1\}, \rho), (\{r_2\}, \rho)$, as well as $(\{r_1\}, \rho), (\{m_2\}, \rho^2)$, and $(\{r_2\}, \rho), (\{m_1\}, \rho^2)$. Therefore, the state $\tilde{s}_6$ only exists in step semantics, since it is reachable exclusively by executing $(\{r_1\}, \rho)$ and $(\{r_2\}, \rho)$ in parallel.
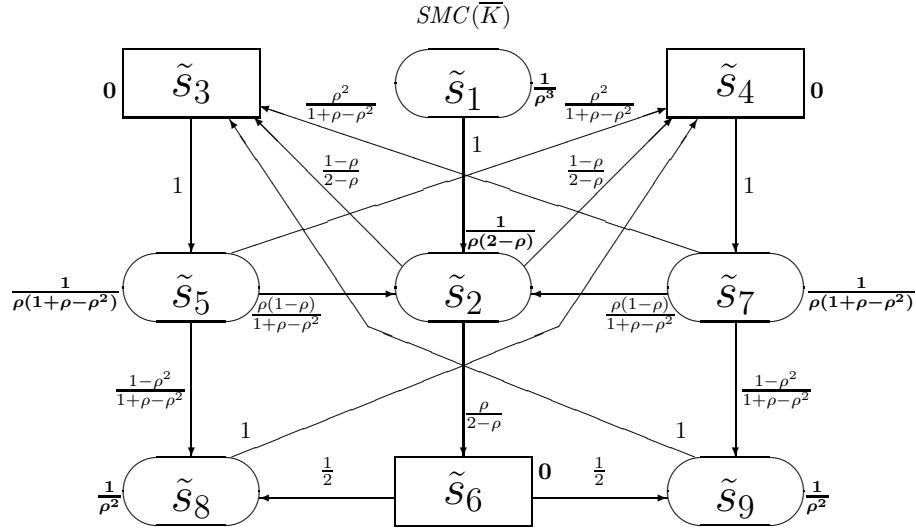
FIG. 10. The underlying SMC of the generalized shared memory system

The average sojourn time vector of $\overline{K}$ is
$$\widetilde{SJ} = \left(\frac{1}{\rho^3}, \frac{1}{\rho(2-\rho)}, 0, 0, \frac{1}{\rho(1+\rho-\rho^2)}, 0, \frac{1}{\rho(1+\rho-\rho^2)}, \frac{1}{\rho^2}, \frac{1}{\rho^2} \right).$$

The sojourn time variance vector of $\overline{K}$ is
$$\widetilde{VAR} = \left(\frac{1-\rho^3}{\rho^6}, \frac{(1-\rho)^2}{\rho^2(2-\rho)^2}, 0, 0, \frac{(1-\rho)^2(1+\rho)}{\rho^2(1+\rho-\rho^2)^2}, 0, \frac{(1-\rho)^2(1+\rho)}{\rho^2(1+\rho-\rho^2)^2}, \frac{1-\rho^2}{\rho^4}, \frac{1-\rho^2}{\rho^4} \right).$$

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

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

The steady-state PMF for $EDTMC(\overline{K})$ is
$$\tilde{\psi}^* = \frac{1}{2(6+3\rho-9\rho^2+2\rho^3)}(0, 2\rho(2-3\rho-\rho^2), 2+\rho-3\rho^2+\rho^3, 2+\rho-3\rho^2+\rho^3,$$
$$2+\rho-3\rho^2+\rho^3, 2\rho^2(1-\rho), 2+\rho-3\rho^2+\rho^3, 2-\rho-\rho^2, 2-\rho-\rho^2).$$

The steady-state PMF $\tilde{\psi}^*$ weighted by $\widetilde{SJ}$ is
$$\frac{1}{2\rho^2(6+3\rho-9\rho^2+2\rho^3)}(0, 2\rho^2(1-\rho), 0, 0, \rho(2-\rho), 0, \rho(2-\rho), 2-\rho-\rho^2, 2-\rho-\rho^2).$$

We normalize the steady-state weighted PMF, dividing it by the sum of its components $\tilde{\psi}^* \widetilde{SJ}^{\mathrm{T}} = \frac{2+\rho-\rho^2-\rho^3}{\rho^2(6+3\rho-9\rho^2+2\rho^3)}$. The steady-state PMF for $SMC(\overline{K})$ is
$$\tilde{\varphi} = \frac{1}{2(2+\rho-\rho^2-\rho^3)}(0, 2\rho^2(1-\rho), 0, 0, \rho(2-\rho), 0, \rho(2-\rho), 2-\rho-\rho^2, 2-\rho-\rho^2).$$

We can now calculate the main performance indices.

- The average recurrence time in the state $\tilde{s}_2$, where no processor requests the memory, called the *average system run-through*, is $\frac{1}{\tilde{\varphi}_2} = \frac{2+\rho-\rho^2-\rho^3}{\rho^2(1-\rho)}$.

- The common memory is available only in the states $\tilde{s}_2, \tilde{s}_3, \tilde{s}_4, \tilde{s}_6$. The steady-state probability that the memory is available is $\tilde{\varphi}_2 + \tilde{\varphi}_3 + \tilde{\varphi}_4 + \tilde{\varphi}_6 = \frac{\rho^2(1-\rho)}{2+\rho-\rho^2-\rho^3} + 0 + 0 + 0 = \frac{\rho^2(1-\rho)}{2+\rho-\rho^2-\rho^3}$. The steady-state probability that the memory is used (i.e. not available), called the *shared memory utilization*, is $1 - \frac{\rho^2(1-\rho)}{2+\rho-\rho^2-\rho^3} = \frac{2+\rho-2\rho^2}{2+\rho-\rho^2-\rho^3}$.

- After activation of the system, we leave the state $\tilde{s}_1$ for ever, and the common memory is either requested or allocated in every remaining state, with exception of $\tilde{s}_2$. The *rate with which the necessity of shared memory emerges* coincides with the rate of leaving $\tilde{s}_2$, $\frac{\tilde{\varphi}_2}{SJ_2} = \frac{\rho^2(1-\rho)}{2+\rho-\rho^2-\rho^3} \cdot \frac{\rho(2-\rho)}{1} = \frac{\rho^3(1-\rho)(2-\rho)}{2+\rho-\rho^2-\rho^3}$.

- The parallel common memory request of two processors $\{(\{r_1\},\rho),(\{r_2\},\rho)\}$ is only possible from the state $\tilde{s}_2$. In this state, the request probability is the sum of the execution probabilities for all multisets of activities containing both $(\{r_1\},\rho)$ and $(\{r_2\},\rho)$. The *steady-state probability of the shared memory request from two processors* is $\tilde{\varphi}_2 \sum_{\{\Upsilon|(\{(\{r_1\},\rho),(\{r_2\},\rho)\}\subseteq\Upsilon\}} PT(\Upsilon,\tilde{s}_2) = \frac{\rho^2(1-\rho)}{2+\rho-\rho^2-\rho^3}\rho^2 = \frac{\rho^4(1-\rho)}{2+\rho-\rho^2-\rho^3}$.

- The common memory request of the first processor $(\{r_1\},\rho)$ is only possible from the states $\tilde{s}_2, \tilde{s}_7$. In each of the states, the request probability is the sum of the execution probabilities for all sets of activities containing $(\{r_1\},\rho)$. The *steady-state probability of the shared memory request from the first processor* is $\tilde{\varphi}_2\sum_{\{\Upsilon|(\{r_1\},\rho)\in\Upsilon\}}PT(\Upsilon,\tilde{s}_2)+\tilde{\varphi}_7\sum_{\{\Upsilon|(\{r_1\},\rho)\in\Upsilon\}}PT(\Upsilon,\tilde{s}_7)= \frac{\rho^2(1-\rho)}{2+\rho-\rho^2-\rho^3}(\rho(1-\rho)+\rho^2)+\frac{\rho(2-\rho)}{2(2+\rho-\rho^2-\rho^3)}(\rho(1-\rho^2)+\rho^3) = \frac{\rho^2(2+\rho-2\rho^2)}{2(2+\rho-\rho^2-\rho^3)}$.

In Figure 11, the marked dtsi-boxes corresponding to the dynamic expressions of two processors, shared memory and the generalized shared memory system are presented, i.e. $N_i = Box_{\text{dtsi}}(\overline{K_i})$ $(1 \le i \le 3)$ and $N = Box_{\text{dtsi}}(\overline{K})$.

9.2. **The abstract system.** Let us consider a modification of the generalized shared memory system with abstraction from identifiers of the processors, i.e. such that the processors are indistinguishable. To implement the abstraction, we replace the actions r_i, d_i, m_i $(1 \le i \le 2)$ in the system specification by r, d, m, respectively.

The static expression of the first processor is
$L_1 = [(\{x_1\},\rho) * ((\{r\},\rho); (\{d,y_1\},\natural_l); (\{m,z_1\},\rho)) * \mathsf{Stop}]$.

The static expression of the second processor is
$L_2 = [(\{x_2\},\rho) * ((\{r\},\rho); (\{d,y_2\},\natural_l); (\{m,z_2\},\rho)) * \mathsf{Stop}]$.

The static expression of the shared memory is
$L_3 = [(\{a,\widehat{x_1},\widehat{x_2}\},\rho) * (((\{\widehat{y_1}\},\natural_l); (\{\widehat{z_1}\},\rho))[](\{\widehat{y_2}\},\natural_l); (\{\widehat{z_2}\},\rho))) * \mathsf{Stop}]$.

The static expression of the abstract generalized shared memory system is
$L = (L_1\|L_2\|L_3) \text{ sr } (x_1,x_2,y_1,y_2,z_1,z_2)$.

$DR(\overline{L})$ resembles $DR(\overline{K})$, and $TS(\overline{L})$ is similar to $TS(\overline{K})$. We have $SMC(\overline{L}) \simeq SMC(\overline{K})$. Thus, the average sojourn time vectors of $\overline{L}$ and $\overline{K}$, as well as the TPMs and the steady-state PMFs for $EDTMC(\overline{L})$ and $EDTMC(\overline{K})$, coincide.

The first, second and third performance indices are the same for the generalized system and its abstraction. The next index is specific to the abstract system.

- The common memory request of a processor $(\{r\},\rho)$ is only possible from the states $\tilde{s}_2, \tilde{s}_5, \tilde{s}_7$. In each of the states, the request probability is the sum of the execution probabilities for all sets of activities containing $(\{r\},\rho)$.
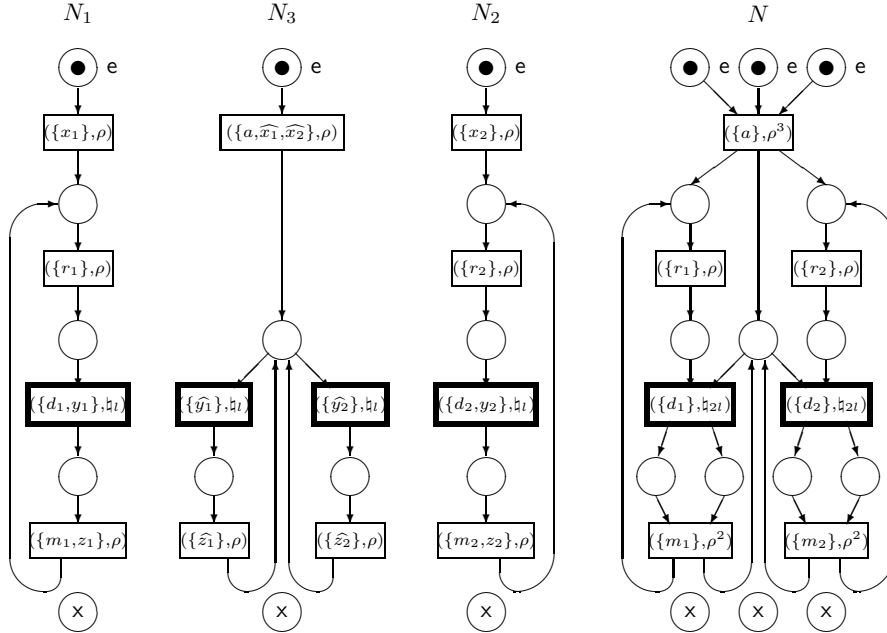
FIG. 11. The marked dtsi-boxes of two processors, shared memory and the generalized shared memory system

The *steady-state probability of the shared memory request from a processor* is $\tilde{\varphi}_2 \sum_{\{\Upsilon | (\{r\}, \rho) \in \Upsilon\}} PT(\Upsilon, \tilde{s}_2) + \tilde{\varphi}_5 \sum_{\{\Upsilon | (\{r\}, \rho) \in \Upsilon\}} PT(\Upsilon, \tilde{s}_5) +$

$\tilde{\varphi}_7 \sum_{\{\Upsilon | (\{r\}, \rho) \in \Upsilon\}} PT(\Upsilon, \tilde{s}_7) = \frac{\rho^2 (1-\rho)}{2+\rho-\rho^2-\rho^3} (\rho(1-\rho) + \rho(1-\rho) + \rho^2) +$

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

We have $DR(\overline{L})/_{\mathcal{R}_{ss}(\overline{L})} = \{\widetilde{\mathcal{K}}_1, \widetilde{\mathcal{K}}_2, \widetilde{\mathcal{K}}_3, \widetilde{\mathcal{K}}_4, \widetilde{\mathcal{K}}_5, \widetilde{\mathcal{K}}_6\}$, where $\widetilde{\mathcal{K}}_1 = \{\tilde{s}_1\}$ (the initial state), $\widetilde{\mathcal{K}}_2 = \{\tilde{s}_2\}$ (the system is activated and the memory is not requested), $\widetilde{\mathcal{K}}_3 = \{\tilde{s}_3, \tilde{s}_4\}$ (the memory is requested by one processor), $\widetilde{\mathcal{K}}_4 = \{\tilde{s}_5, \tilde{s}_7\}$ (the memory is allocated to a processor), $\widetilde{\mathcal{K}}_5 = \{\tilde{s}_6\}$ (the memory is requested by two processors), $\widetilde{\mathcal{K}}_6 = \{\tilde{s}_8, \tilde{s}_9\}$ (the memory is allocated to a processor and the memory is requested by another processor).

Further, $DR_{\mathrm{T}}(\overline{L})/_{\mathcal{R}_{ss}(\overline{L})} = \{\widetilde{\mathcal{K}}_1, \widetilde{\mathcal{K}}_2, \widetilde{\mathcal{K}}_4, \widetilde{\mathcal{K}}_6\}$ and $DR_{\mathrm{V}}(\overline{L})/_{\mathcal{R}_{ss}(\overline{L})} = \{\widetilde{\mathcal{K}}_3, \widetilde{\mathcal{K}}_5\}$.

In Figure 12, the quotient transition system $TS_{\underleftrightarrow{ss}}(\overline{L})$ is presented. In Figure 13, the quotient underlying SMC $SMC_{\underleftrightarrow{ss}}(\overline{L})$ is depicted. Note that, in step semantics, we may execute the following multiactions in parallel: $\{r\}, \{r\}$, as well as $\{r\}, \{m\}$. Again, the state $\widetilde{\mathcal{K}}_5$ only exists in step semantics, since it is reachable exclusively by executing $\{r\}$ and $\{r\}$ in parallel.

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

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

The quotient sojourn time variance vector of $\overline{F}$ is

$\widetilde{VAR}' = \left(\frac{1-\rho^3}{\rho^6}, \frac{(1-\rho)^2}{\rho^2(2-\rho)^2}, 0, \frac{(1-\rho)^2(1+\rho)}{\rho^2(1+\rho-\rho^2)^2}, 0, \frac{1-\rho^2}{\rho^4} \right).$
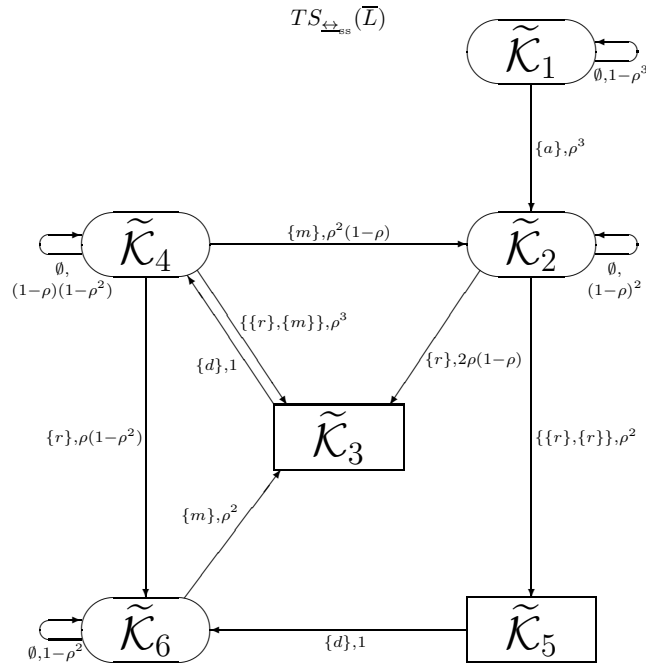
$TS_{\underline{\leftrightarrow}_{ss}}(\overline{L})$

FIG. 12. The quotient transition system of the abstract generalized shared memory system

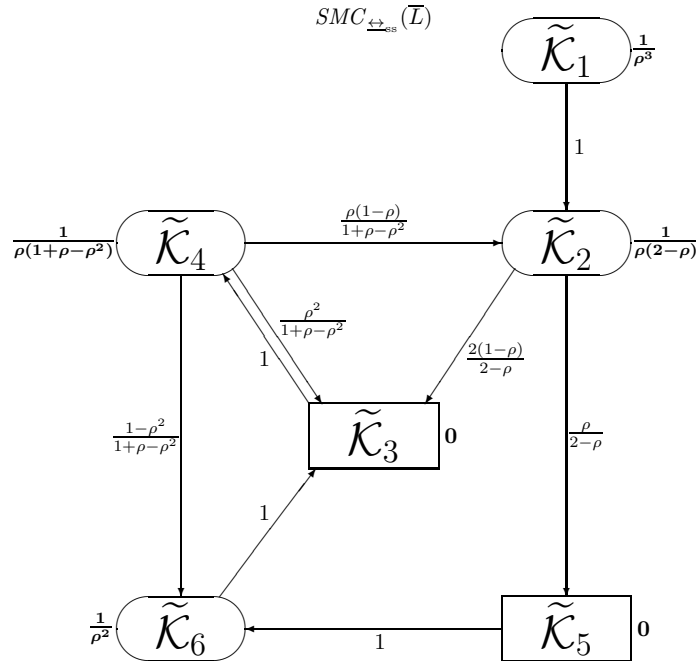$SMC_{\underline{\leftrightarrow}_{ss}}(\overline{L})$

FIG. 13. The quotient underlying SMC of the abstract generalized shared memory system

The TPM for $EDTMC_{\underline{\leftrightarrow}_{ss}}(\overline{L})$ is $\widetilde{\mathbf{P}}'^* = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{2(1-\rho)}{2-\rho} & 0 & \frac{\rho}{2-\rho} & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & \frac{\rho(1-\rho)}{1+\rho-\rho^2} & \frac{\rho^2}{1+\rho-\rho^2} & 0 & 0 & \frac{1-\rho^2}{1+\rho-\rho^2} \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}.$

The steady-state PMF for $EDTMC_{\underline{\leftrightarrow}_{ss}}(\overline{L})$ is $\tilde{\psi}'^* = \frac{1}{6+3\rho-9\rho^2+2\rho^3}(0, \rho(2-3\rho+\rho^2),$ $2+\rho-3\rho^2+\rho^3, 2+\rho-3\rho^2+\rho^3, \rho^2(1-\rho), 2-\rho-\rho^2)$. The steady-state PMF $\tilde{\psi}'^*$ weighted by $\widetilde{SJ}'$ is $\frac{1}{\rho^2(6+3\rho-9\rho^2+2\rho^3)}(0, \rho^2(1-\rho), 0, \rho(2-\rho), 0, 2-\rho-\rho^2)$.

We normalize the steady-state weighted PMF, dividing it by the sum of its components $\tilde{\psi}'^*\widetilde{SJ}'^{\mathrm{T}} = \frac{2+\rho-\rho^2-\rho^3}{\rho^2(6+3\rho-9\rho^2+2\rho^3)}$. The steady-state PMF for $SMC_{\underline{\leftrightarrow}_{ss}}(\overline{L})$ is $\tilde{\varphi}' = \frac{1}{2+\rho-\rho^2-\rho^3}(0, \rho^2(1-\rho), 0, \rho(2-\rho), 0, 2-\rho-\rho^2)$.

We can now calculate the main performance indices.

- The average recurrence time in the state $\widetilde{\mathcal{K}}_2$, where no processor requests the memory, called the *average system run-through*, is $\frac{1}{\tilde{\varphi}'_2} = \frac{2+\rho-\rho^2-\rho^3}{\rho^2(1-\rho)}$.

- The common memory is available only in the states $\widetilde{\mathcal{K}}_2, \widetilde{\mathcal{K}}_3, \widetilde{\mathcal{K}}_5$. The steady-state probability that the memory is available is $\tilde{\varphi}'_2 + \tilde{\varphi}'_3 + \tilde{\varphi}'_5 = \frac{\rho^2(1-\rho)}{2+\rho-\rho^2-\rho^3} + 0 + 0 = \frac{\rho^2(1-\rho)}{2+\rho-\rho^2-\rho^3}$. The steady-state probability that the memory is used (i.e. not available), called the *shared memory utilization*, is $1 - \frac{\rho^2(1-\rho)}{2+\rho-\rho^2-\rho^3} = \frac{2+\rho-2\rho^2}{2+\rho-\rho^2-\rho^3}$.

- After activation of the system, we leave the state $\widetilde{\mathcal{K}}_1$ for ever, and the common memory is either requested or allocated in every remaining state, with exception of $\widetilde{\mathcal{K}}_2$. The *rate with which the necessity of shared memory emerges* coincides with the rate of leaving $\widetilde{\mathcal{K}}_2$, $\frac{\tilde{\varphi}'_2}{\widetilde{SJ}_2} = \frac{\rho^2(1-\rho)}{2+\rho-\rho^2-\rho^3} \cdot \frac{\rho(2-\rho)}{1} = \frac{\rho^3(1-\rho)(2-\rho)}{2+\rho-\rho^2-\rho^3}$.

- The parallel common memory request of two processors $\{\{r\}, \{r\}\}$ is only possible from the state $\widetilde{\mathcal{K}}_2$. In this state, the request probability is the sum of the execution probabilities for all multisets of multiactions containing $\{r\}$ twice. The *steady-state probability of the shared memory request from two processors* is $\tilde{\varphi}'_2 \sum_{\{A,\widetilde{\mathcal{K}}|\{\{r\},\{r\}\}\subseteq A, \ \widetilde{\mathcal{K}}_2 \xrightarrow{A} \widetilde{\mathcal{K}}\}} PM_A(\widetilde{\mathcal{K}}_2, \widetilde{\mathcal{K}}) = \frac{\rho^2(1-\rho)}{2+\rho-\rho^2-\rho^3}\rho^2 = \frac{\rho^4(1-\rho)}{2+\rho-\rho^2-\rho^3}$.

- The common memory request of a processor $\{r\}$ is only possible from the states $\widetilde{\mathcal{K}}_2, \widetilde{\mathcal{K}}_4$. In each of the states, the request probability is the sum of the execution probabilities for all multisets of multiactions containing $\{r\}$. The *steady-state probability of the shared memory request from a processor* is $\tilde{\varphi}'_2 \sum_{\{A,\widetilde{\mathcal{K}}|\{r\}\in A, \ \widetilde{\mathcal{K}}_2 \xrightarrow{A} \widetilde{\mathcal{K}}\}} PM_A(\widetilde{\mathcal{K}}_2, \widetilde{\mathcal{K}}) + \tilde{\varphi}'_4 \sum_{\{A,\widetilde{\mathcal{K}}|\{r\}\in A, \ \widetilde{\mathcal{K}}_4 \xrightarrow{A} \widetilde{\mathcal{K}}\}} PM_A(\widetilde{\mathcal{K}}_4, \widetilde{\mathcal{K}}) = \frac{\rho^2(1-\rho)}{2+\rho-\rho^2-\rho^3}(2\rho(1-\rho)+\rho^2) + \frac{\rho(2-\rho)}{2+\rho-\rho^2-\rho^3}(\rho(1-\rho^2)+\rho^3) = \frac{\rho^2(2-\rho)(1+\rho-\rho^2)}{2+\rho-\rho^2-\rho^3}$.

The performance indices are the same for the complete and the quotient abstract generalized shared memory systems. The coincidence of the first, second and third performance indices obviously illustrates the results of Propositions 6 and 7 (both modified for $\mathcal{R}_{\mathcal{L}ss}(\overline{L})$). The coincidence of the fourth performance index is due to
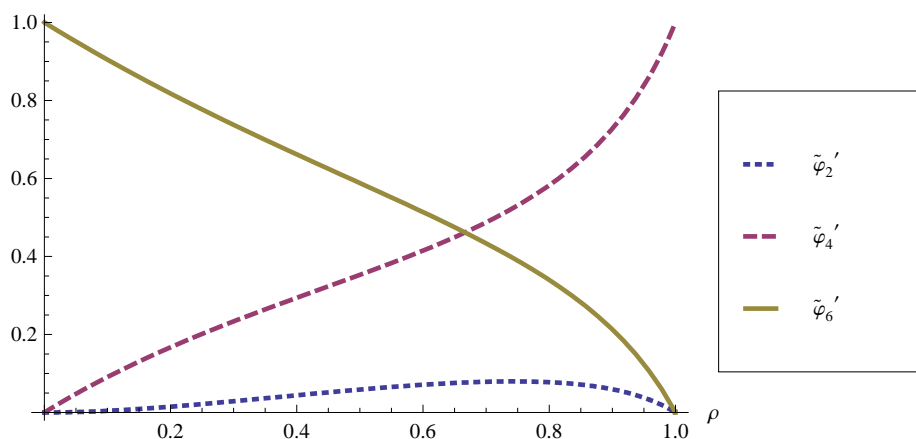
FIG. 14. Steady-state probabilities $\tilde{\varphi}'_2$, $\tilde{\varphi}'_4$, $\tilde{\varphi}'_6$ as functions of the parameter ρ

Theorem 4 (modified for $\mathcal{R}_{\mathcal{L}ss}(\overline{L})$): one should just apply this result to the derived step trace $\{\{r\}, \{r\}\}$ of the expression $\overline{L}$ and itself. The coincidence of the fifth performance index is due to Theorem 4 (modified for $\mathcal{R}_{\mathcal{L}ss}(\overline{L})$): one should just apply this result to the derived step traces $\{\{r\}\}$, $\{\{r\}, \{r\}\}$, $\{\{r\}, \{m\}\}$ of the expression $\overline{L}$ and itself, and then sum the left and right parts of the three resulting equalities.

Consider the effect of quantitative changes of the parameter ρ upon performance of the quotient abstract generalized shared memory system in its steady state. Remember that $\rho \in (0; 1)$ is the probability of every stochastic multiaction in the specification of the system. The closer is ρ to 0, the less is the probability to execute some activities at every discrete time tick, hence, the system will most probably *stand idle*. The closer is ρ to 1, the greater is the probability to execute some activities at every discrete time tick, hence, the system will most probably *operate*.

Since $\tilde{\varphi}'_1 = \tilde{\varphi}'_3 = \tilde{\varphi}'_5 = 0$, only $\tilde{\varphi}'_2 = \frac{\rho^2(1-\rho)}{2+\rho-\rho^2-\rho^3}$, $\tilde{\varphi}'_4 = \frac{\rho(2-\rho)}{2+\rho-\rho^2-\rho^3}$, $\tilde{\varphi}'_6 = \frac{2-\rho-\rho^2}{2+\rho-\rho^2-\rho^3}$ depend on ρ. In Figure 14, the plots of $\tilde{\varphi}'_2$, $\tilde{\varphi}'_4$, $\tilde{\varphi}'_6$ as functions of ρ are depicted. Remember that we do not allow $\rho = 0$ or $\rho = 1$.

One can see that $\tilde{\varphi}'_2$, $\tilde{\varphi}'_4$ tend to 0 and $\tilde{\varphi}'_6$ tends to 1 when ρ approaches 0. Thus, when ρ is closer to 0, the probability that the memory is allocated to a processor and the memory is requested by another processor increases, hence, we have *more unsatisfied memory requests*.

Next, $\tilde{\varphi}'_2$, $\tilde{\varphi}'_6$ tend to 0 and $\tilde{\varphi}'_4$ tends to 1 when ρ approaches 1. When ρ is closer to 1, the probability that the memory is allocated to a processor (and not requested by another one) increases, hence, we have *less unsatisfied memory requests*.

The maximal value 0.0797 of $\tilde{\varphi}'_2$ is reached when $\rho \approx 0.7433$. The probability that the system is activated and the memory is not requested is maximal, the *maximal shared memory availability*, is about 8%.

In Figure 15, the plot of the average system run-through, calculated as $\frac{1}{\tilde{\varphi}'_2}$, as a function of ρ is depicted. The run-through tends to ∞ when ρ approaches 0 or 1. Its minimal value 12.5516 is reached when $\rho \approx 0.7433$. To speed up operation of the system, one should take the parameter ρ closer to 0.7433.
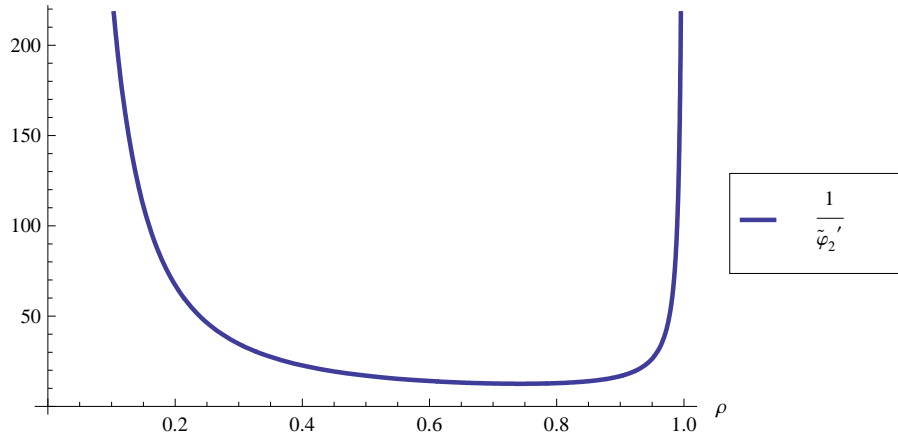
FIG. 15. Average system run-through $\frac{1}{\tilde{\varphi}_2'}$ as a function of the parameter ρ

The first curve in Figure 16 represents the shared memory utilization, calculated as $1 - \tilde{\varphi}_2' - \tilde{\varphi}_3' - \tilde{\varphi}_5'$, as a function of ρ. The utilization tends to 1 both when ρ approaches 0 and when ρ approaches 1. The minimal value 0.9203 of the utilization is reached when $\rho \approx 0.7433$. Thus, the *minimal shared memory utilization* is about 92%. To increase the utilization, one should take the parameter ρ closer to 0 or 1.

The second curve in Figure 16 represents the rate with which the necessity of shared memory emerges, calculated as $\frac{\tilde{\varphi}_2'}{\widetilde{SJ}_2}$, as a function of ρ. The rate tends to 0 both when ρ approaches 0 and when ρ approaches 1. The maximal value 0.0751 of the rate is reached when $\rho \approx 0.7743$. The *maximal rate with which the necessity of shared memory emerges* is about $\frac{1}{13}$. To decrease the rate, one must take the parameter ρ closer to 0 or 1.

The third curve in Figure 16 represents the steady-state probability of the shared memory request from two processors, calculated as $\tilde{\varphi}_2'\widetilde{\mathcal{P}}_{25}'$, where $\widetilde{\mathcal{P}}_{25}' = \sum_{\{A,\widetilde{\mathcal{K}}|\{\{r\},\{r\}\}\subseteq A,\ \widetilde{\mathcal{K}}_2\overset{A}{\to}\widetilde{\mathcal{K}}\}} PM_A(\widetilde{\mathcal{K}}_2,\widetilde{\mathcal{K}}) = PM(\widetilde{\mathcal{K}}_2,\widetilde{\mathcal{K}}_5)$, as function of ρ. One can see that the probability tends to 0 both when ρ approaches 0 and when ρ approaches 1. The maximal value 0.0517 of the probability is reached when $\rho \approx 0.8484$. To decrease the mentioned probability, one should take the parameter ρ closer to 0 or 1.

The fourth curve in Figure 16 represents the steady-state probability of the shared memory request from a processor, calculated as $\tilde{\varphi}_2'\widetilde{\Sigma}_2' + \tilde{\varphi}_4'\widetilde{\Sigma}_4'$, as a function of ρ, where $\widetilde{\Sigma}_i' = \sum_{\{A,\widetilde{\mathcal{K}}|\{r\}\in A,\ \widetilde{\mathcal{K}}_i\overset{A}{\to}\widetilde{\mathcal{K}}\}} PM_A(\widetilde{\mathcal{K}}_i,\widetilde{\mathcal{K}})$, $i \in \{2,4\}$. One can see that the probability tends to 0 when ρ approaches 0 and it tends to 1 when ρ approaches 1. To increase the probability, one should take the parameter ρ closer to 1.

10. RELATED WORK

Consider differences and similarities between dtsiPBC and other well-known SPAs.

10.1. Continuous time and interleaving semantics. Let us compare dtsiPBC with the classical interleaving SPAs.

Markovian Timed Processes for Performance Evaluation (MTIPP) [44] specifies every activity as a pair consisting of the action name (including the symbol τ for
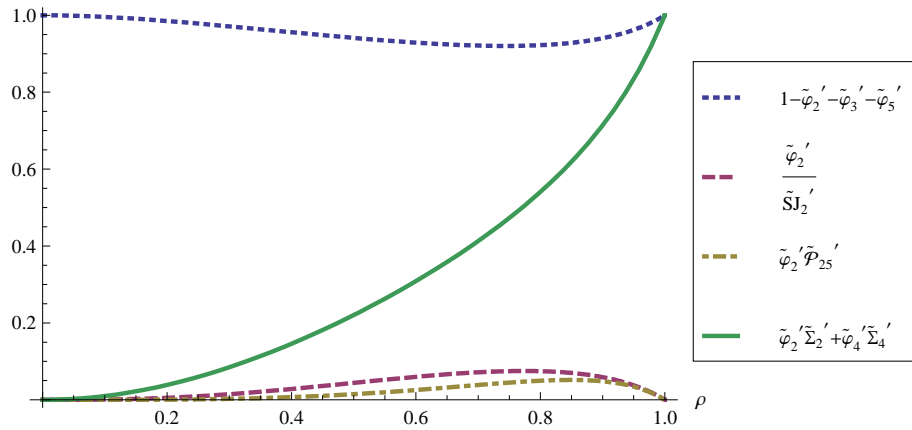
FIG. 16. Some performance indices as functions of the parameter ρ

the *internal*, invisible action) and the parameter of exponential distribution of the action delay (the *rate*). The interleaving operational semantics is defined on the basis of Markovian (i.e. extended with the specification of rates) labeled transition systems. The interleaving behaviour is here because the exponential PDF is a continuous one and simultaneous execution of any two activities has zero probability according to the properties of continuous distributions. CTMCs can be derived from the transition systems to analyze performance.

Performance Evaluation Process Algebra (PEPA) [46] treats the activities as pairs consisting of action types (including the *unknown* type τ) and activity rates. The rate is either the parameter of exponential distribution of the activity duration or it is *unspecified*. An activity with unspecified rate is *passive* by its action type. The operational semantics is interleaving, it is defined via the extension of labeled transition systems with a possibility to specify activity rates. Based on the transition systems, the continuous time Markov processes (CTMPs) are generated which are used for performance evaluation with the help of the embedded continuous time Markov chains (ECTMCs). In [37], a denotational semantics of PEPA has been proposed via PEPA nets that are high-level CTSPNs with coloured tokens (coloured CTSPNs), from which the underlying CTMCs can be retrieved.

Extended Markovian Process Algebra (EMPA) [14] interprets each action as a pair consisting of its type and rate. Actions can be *external* or *internal* (denoted by τ) according to types. There are three kinds of actions according to rates: *timed* ones with exponentially distributed durations (essentially, the actions from MTIPP and PEPA), *immediate* ones with priorities and weights (the actions analogous to immediate transitions of GSPNs) and *passive* ones (similar to passive actions of PEPA). The operational semantics is interleaving and based on the labeled transition systems enriched with the information about action rates. For the exponentially timed kernel of the algebra (the sublanguage including only exponentially timed and passive actions), it is possible to construct CTMCs from the transition systems of the process terms to analyze the performance. In [13, 9], a denotational semantics of EMPA, based on GSPNs, has been defined, from which one can also extract the underlying SMCs and CTMCs (when both immediate and timed transitions are present) or DTMCs (but when there are only immediate transitions).

dtsiPBC considers every activity as a pair consisting of the multiaction (not just an action, as in the classical SPAs) as a first element. The second element is either the probability (not the rate, as in the classical SPAs) to execute the multiaction independently (the activity is called a stochastic multiaction in this case) or the weight expressing how important is the execution of this multiaction (then the activity is called an immediate multiaction). Immediate multiactions in dtsiPBC are similar to immediate actions in EMPA, but all the immediate multiactions in dtsiPBC have the same high priority (with the goal to execute them always before stochastic multiactions, all having the same low priority), whereas the immediate actions in EMPA can have different priorities. Associating the same priority with all immediate multiactions in dtsiPBC results in the simplified specification and analysis, and such a decision is appropriate to the calculus, since weights (assigned also to immediate actions in EMPA) are enough to denote preferences among immediate multiactions and to produce the conformable probabilistic behaviours. There are no immediate actions in MTIPP and PEPA. Immediate actions are available only in iPEPA [43], where they are analogous to immediate multiactions in dtsiPBC, and in a variant of TIPP [39] discussed while constructing the calculus PM-TIPP [77], but there immediate activities are used just to specify probabilistic branching and they cannot be synchronized. dtsiPBC has a discrete time semantics, and residence time in the tangible states is geometrically distributed, unlike the classical SPAs with continuous time semantics and exponentially distributed activity delays. As a consequence, dtsiPBC has a step operational semantics in contrast to interleaving operational semantics of the classical SPAs. The performance in dtsiPBC is analyzed via the underlying SMCs and (reduced) DTMCs [85] extracted from the labeled probabilistic transition systems associated with the expressions. In the classical SPAs, CTMCs are usually used for performance evaluation. dtsiPBC has a denotational semantics, based on LDTSIPNs, from which the underlying SMCs and (reduced) DTMCs are derived, unlike (reduced) CTMCs in PEPA and EMPA. MTIPP has no denotational semantics.

10.2. **Continuous time and non-interleaving semantics.** A few non-interleaving SPAs were considered among non-Markovian ones [53, 20].

Generalized Stochastic Process Algebra (GSPA) [23] is a stochastic extension of Simple Process Algebra [23]. GSPA has no operational semantics. GSPA has a true-concurrent denotational semantics via generalized stochastic event structures (GSESs) with non-Markovian stochastic delays of events. In [54], generalized semi-Markov processes (GSMPs) were extracted from GSESs to analyze performance.

Generalized Stochastic π-calculus (Sπ) [75, 76] extends π-calculus [68]. Sπ allows for general continuous distributions of activity delays. It has a proved operational semantics with transitions labeled by encodings of their deduction trees. The transition labels encode the action causality information and allow one to derive the enabling relations and the firing distributions of concurrent transitions from the transition sequences. Nevertheless, abstracting from stochastic delays leads to the classical early interleaving semantics of π-calculus. No well-established underlying performance model for this version of Sπ exists.

Generalized Semi-Markovian Process Algebra (GSMPA) [21, 19] is an enrichment of EMPA. GSMPA has an ST-operational semantics and non-Markovian action delays. The ST-operational semantics of GSMPA is based on decorated transition systems governed by transition rules with rather complex preconditions. There are

two types of transitions: the choice (action beginning) and the termination (action ending) ones. The choice transitions are labeled by weights of single actions chosen for execution while the termination transitions have no labels. Only single actions can begin, but several actions can end in parallel. Thus, the choice transitions happen just sequentially while the termination transitions can happen simultaneously. As a result, the decorated interleaving / step transition systems are obtained. The performance analysis in GSMPA is accomplished via GSMPs.

dtsiPBC has immediate multiactions while GSPA, Sπ and GSMPA do not specify instantaneous events or activities. Geometrically distributed or zero delays are associated with process states in dtsiPBC, unlike generally distributed delays assigned to events in GSPA or to activities in Sπ and GSMPA. dtsiPBC has a discrete time operational semantics allowing for concurrent execution of activities in steps. GSPA has no operational semantics while Sπ and GSMPA have continuous time ones. In continuous time semantics, concurrency is simulated by interleaving, since simultaneous occurrence of any two events has zero probability according to the properties of continuous probability distributions. Therefore, interleaving transitions should be annotated with an additional information to maintain the concurrency. dtsiPBC has an SPN-based denotational semantics. In comparison with event structures, PNs are more expressive and visually tractable formalism, capable of finitely specifying an infinite behaviour. Recursion in GSPA produces infinite GSESs while dtsiPBC has iteration operation with a finite SPN semantics. Identification of infinite GSESs that can be finitely represented in GSPA was left for a future research.

10.3. **Discrete time.** Much fewer SPAs with discrete time semantics were designed.

Dts-nets [1] are a class of compositional DTSPNs with generally distributed discrete time transition delays. The denotational semantics of a stochastic extension (we call it stochastic ACP or sACP) of a subset of Algebra of Communicating Processes (ACP) [8] can be constructed via dts-nets. There are two types of transitions: immediate (timeless) ones, with zero delays, and time ones, whose delays are random variables with general discrete distributions. The top-down synthesis of dts-nets consists in the substitution of their transitions by blocks (dts-subnets) corresponding to some composition operators. It was explained how to calculate the throughput time of dts-nets using the service time (holding time or delay) of their transitions. For this, the notions of service distribution for the transitions and throughput distribution for the building blocks were defined. Since the throughput time of the parallelism block was calculated as the maximal service time for its two constituting transitions, the analogue of the step semantics was implemented.

Theory of Communicating Processes with discrete stochastic time (TCP^{dst}) [63, 64], later called Theory of Communicating Processes with discrete real and stochastic time (TCP^{drst}) [62], is another stochastic extension of ACP. TCP^{dst} has discrete real time (deterministic) delays (including zero time delays) and discrete stochastic time delays. The algebra generalizes real time processes to discrete stochastic time ones by applying real time properties to stochastic time and imposing race condition to real time semantics. TCP^{dst} has an interleaving operational semantics in terms of stochastic transition systems. The performance is analyzed via discrete time probabilistic reward graphs which are essentially the reward transition systems with probabilistic states having finite number of outgoing probabilistic

TABLE 5. Classification of stochastic process algebras

Time	Immediate (multi)actions	Interleaving semantics	Non-interleaving semantics
Continuous	No	MTIPP (CTMC), **PEPA** (CTMP), **sPBC** (CTMC)	GSPA (GSMP), Sπ, GSMPA (GSMP)
	Yes	**EMPA** (SMC, CTMC), **gsPBC** (SMC)	—
Discrete	No	—	**dtsPBC** (DTMC)
	Yes	TCP^{dst} (DTMRC)	**sACP**, **dtsiPBC** (SMC, DTMC)

transitions and timed states having a single outgoing timed transition. The mentioned graphs can be transformed by unfolding or geometrization into discrete time Markov reward chains (DTMRCs) appropriate for transient or stationary analysis.

dtsiPBC, sACP and TCP^{dst}, all have zero delays. However, discrete time delays in dtsiPBC are zeros or geometrically distributed and associated with process states. The zero delays are possible just in vanishing states while geometrically distributed delays are possible only in tangible states. For each tangible state, the parameter of geometric distribution governing the delay in the state is completely determined by the probabilities of all stochastic multiactions executable from it. In sACP and TCP^{dst}, delays are generally distributed, but they are assigned to transitions in sACP and separated from actions (excepting zero delays) in TCP^{dst}. Moreover, a special attention is given to zero delays in sACP and deterministic delays in TCP^{dst}. In sACP, immediate (timeless) transitions with zero delays serve as source and sink transitions of the dts-subnets corresponding to the choice, parallelism and iteration operators. In TCP^{dst}, zero delays of actions are specified by undelayable action prefixes while positive deterministic delays of processes are specified with timed delay prefixes. Neither formal syntax nor operational semantics for sACP were defined and it was not explained how to derive Markov chains from the algebraic expressions or the corresponding dts-nets to analyze performance. It was not stated explicitly, which type of semantics (interleaving or step) is accommodated in sACP. In spite of the discrete time approach, operational semantics of TCP^{dst} is still interleaving, unlike that of dtsiPBC. TCP^{dst} has no denotational semantics.

Table 5 summarizes the SPAs comparison above and that from Section 1, by classifying the SPAs according to the concept of time, the presence of immediate (multi)actions and the type of operational semantics. The names of SPAs, whose denotational semantics is based on SPNs, are printed in bold font. The underlying stochastic process (if defined) is specified near the name of the corresponding SPA.

11. DISCUSSION

Let us now discuss which advantages has dtsiPBC in comparison with the SPAs described in Section 10.

11.1. **Analytical solution.** An important aspect is the analytical tractability of the underlying stochastic process, used for performance evaluation in SPAs. The underlying CTMCs in MTIPP and PEPA, as well as SMCs in EMPA, are treated analytically, but these continuous time SPAs have interleaving semantics. GSPA, Sπ and GSMPA are the continuous time models, for which a non-interleaving

semantics is constructed, but for the underlying GSMPs in GSPA and GSMPA, only simulation and numerical methods are applied, whereas no performance model for $S\pi$ is defined. sACP and TCP^{dst} are the discrete time models with the associated analytical methods for the throughput calculation in sACP or for the performance evaluation, based on the underlying DTMRCs in TCP^{dst}, but both models have interleaving semantics. dtsiPBC is a discrete time model with a non-interleaving semantics, where analytical methods are applied to the underlying SMCs. Hence, if an interleaving model is appropriate as a framework for the analytical solution towards performance evaluation then one has a choice between the continuous time SPAs MTIPP, PEPA, EMPA and the discrete time ones sACP, TCP^{dst}. Otherwise, if one needs a non-interleaving model with the associated analytical methods for performance evaluation and the discrete time approach is feasible then dtsiPBC is the right choice.

The existence of an analytical solution also permits to interpret quantitative values (rates, probabilities, weights etc.) from the system specifications as parameters, which can be adjusted to optimize the system performance, like in dtsPBC, dtsiPBC and parametric probabilistic transition systems (i.e. DTMCs whose transition probabilities may be real-value parameters) [56]. Note that DTMCs whose transition probabilities are parameters were introduced in [32]. Parametric CTMCs with the transition rates treated as parameters were investigated in [41]. On the other hand, no parameters in formulas of SPAs were considered in the literature so far. In dtsiPBC we can easily construct examples with more parameters than we did in our case study. The performance indices will be then interpreted as functions of several variables. The advantage of our approach is that, unlike of the method from [56], we should not impose to the parameters any special conditions needed to guarantee that the real values, interpreted as the transition probabilities, always lie in the interval $[0; 1]$. To be convinced of this fact, just remember that, as we have demonstrated, the positive probability functions PF, PT, PM, PM^* define probability distributions, hence, they always return values belonging to $(0; 1]$ for any probability parameters from $(0; 1)$ and weight parameters from $\mathbb{R}_{>0}$. In addition, the transition constraints (their probabilities, rates and guards), calculated using the parameters, in our case should not always be polynomials over variables-parameters, as often required in the mentioned papers, but they may also be fractions of polynomials, like in our case study.

11.2. **Concurrency interpretation.** The stochastic process calculi proposed in the literature are based on interleaving, as a rule, and parallelism is simulated by synchronous or asynchronous execution. As a semantic domain, the interleaving formalism of transition systems is often used. However, to properly support intuition of the behaviour of concurrent and distributed systems, their semantics should treat parallelism as a primitive concept that cannot be reduced to nondeterminism. Moreover, in interleaving semantics, some important properties of these systems cannot be expressed, such as simultaneous occurrence of concurrent transitions [33] or local deadlock in the spatially distributed processes [71]. Therefore, investigation of stochastic extensions for more expressive and powerful algebraic calculi is an important issue. The development of step or "true concurrency" (such that parallelism is considered as a causal independence) SPAs is an interesting and nontrivial problem that has attracted special attention in the last years. Nevertheless, not so many

formal stochastic models of parallel systems were defined whose underlying stochastic processes were based on DTMCs. As mentioned in [36], such models are more difficult to analyze, since several events can occur simultaneously in discrete time systems (the models have a step semantics) and the probability of a set of events cannot be easily related to the probability of the single ones. Therefore, parallel executions of actions are often not considered also in the discrete time SPAs, such as TCP^{dst}, whose underlying stochastic process is DTMCs with rewards (DTMRCs). As observed in [48], even for stochastic models with generally distributed time delays, some restrictions on the concurrency degree were imposed to simplify their analysis techniques. In particular, the enabling restriction requires that no two generally distributed transitions are enabled in any reachable marking. Hence, their activity periods do not intersect and no two such transitions can fire simultaneously, this results in interleaving semantics of the model.

Stochastic models with discrete time and step semantics have the following important advantage over those having just an interleaving semantics. The underlying Markov chains of parallel stochastically timed processes have additional transitions corresponding to the simultaneous execution of concurrent (i.e. non-synchronized) activities. These additional transitions allow us one to bypass a lot of intermediate states, which otherwise should be visited when interleaving semantics is accommodated. When step semantics is used, the intermediate states can also be visited with some probability (this is an advantage, since some alternative system's behaviour may start from these states), but this probability is not greater than the corresponding one in case of interleaving semantics. While in interleaving semantics, only the empty or singleton (multi)sets of activities can be executed, in step semantics, generally, the (multi)sets of activities with more than one element can be executed as well. Hence, in step semantics, there are more variants of execution from each state than in the interleaving case and the executions probabilities, whose sum must be equal to 1, are distributed among more possibilities. Therefore, the systems with parallel stochastic processes usually have smaller average run-through. Thus, when the underlying Markov chains of the processes are ergodic, they will take less discrete time units to stabilize the behaviour, since their TPMs will be denser because of additional non-zero elements outside the main diagonal. Hence, both the first passage-time performance indices, based on the transient probabilities, and the steady-state performance indices, based on the stationary probabilities, can be computed quicker, resulting in faster quantitative analysis of the systems. On the other hand, step semantics, induced by simultaneous firing several transitions at each step, is natural for Petri nets and allows one to exploit full power of the model. Hence, it is important to respect the probabilities of parallel executions of activities in discrete time SPAs, especially in those with a Petri net denotational semantics.

Example 16. *In Figure 17, the interleaving transition system of the generalized shared memory system $ts(\overline{K})$ from Section 9 is presented. The transition system $ts(\overline{K})$ is constructed from the (step) one $TS(\overline{K})$ in Figure 9 as follows. First, all the transitions due to executing more than one activity are removed. Second, the states that become non-reachable (from the initial state) in the absence of such "parallel" transitions are deleted, together with all the transitions from these states. Third, for each of the states left, the remaining outgoing transition probabilities are normalized. As a result, one can see many serious differences between $ts(\overline{K})$ and $TS(\overline{K})$, i.e. between the system's behaviour in the interleaving and step semantics. One can*

$ts(\overline{K})$

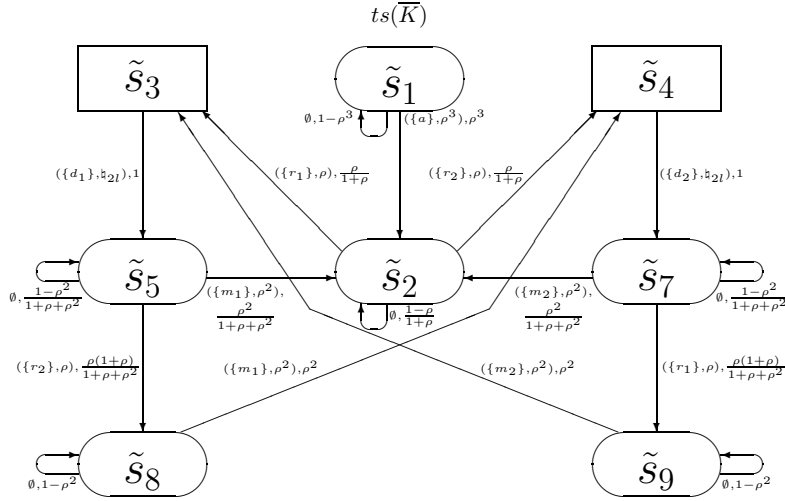
FIG. 17. The interleaving transition system of the generalized shared memory system

define interleaving stochastic bisimulation equivalence $\underleftrightarrow{}_{\mathrm{is}}$ analogously to $\underleftrightarrow{}_{\mathrm{ss}}$, but using the interleaving transition systems of expressions instead of the standard (step) ones. Then, from $ts(\overline{K})$, the interleaving quotient (by $\underleftrightarrow{}_{\mathrm{is}}$) transition system of the abstract generalized shared memory system $ts_{\underleftrightarrow{}_{\mathrm{is}}}(\overline{L})$ can be constructed, depicted in Figure 18. Again, there exist substantial differences between $ts_{\underleftrightarrow{}_{\mathrm{is}}}(\overline{L})$ and $TS_{\underleftrightarrow{}_{\mathrm{ss}}}(\overline{L})$ in Figure 12. From $ts_{\underleftrightarrow{}_{\mathrm{is}}}(\overline{L})$, the interleaving quotient SMC of the abstract generalized shared memory system $smc_{\underleftrightarrow{}_{\mathrm{is}}}(\overline{L})$ can also be extracted, depicted in Figure 19. There are serious differences between $smc_{\underleftrightarrow{}_{\mathrm{is}}}(\overline{L})$ and $SMC_{\underleftrightarrow{}_{\mathrm{ss}}}(\overline{L})$ in Figure 13. Then the steady-state PMF for $smc_{\underleftrightarrow{}_{\mathrm{is}}}(\overline{L})$ is $\tilde{\tilde{\phi}}' = \frac{1}{2+4\rho+3\rho^2+3\rho^3}(0, \rho^2(1+\rho), 0, 2\rho(1+\rho+\rho^2), 0, 2(1+\rho))$, whereas the steady-state PMF for $SMC_{\underleftrightarrow{}_{\mathrm{ss}}}(\overline{L})$ is $\tilde{\varphi}' = \frac{1}{2+\rho-\rho^2-\rho^3}(0, \rho^2(1-\rho), 0, \rho(2-\rho), 0, 2-\rho-\rho^2)$. In Figure 20, the plots of $\tilde{\phi}'_2$, $\tilde{\phi}'_4$, $\tilde{\phi}'_6$ as functions of ρ are depicted. One cane see substantial differences between plots in Figure 20 and those in Figure 14. The differences indicate that when ρ tends to 1, the increase of performance (treated as the time fraction when the memory is allocated to a processor and not required by another one) is much quicker in step semantics (the function $\tilde{\varphi}'_4$) than in the interleaving semantics ($\tilde{\phi}'_4$). Moreover, when $\rho = 1$, the memory is allocated and not requested ($\tilde{\varphi}'_4$) with probability 1 in the step case while all the three variants exist with positive probabilities in the interleaving case: the memory is not allocated and not requested ($\tilde{\phi}'_2$) with probability $\frac{1}{6}$ or the memory is allocated and not requested ($\tilde{\phi}'_4$) with probability $\frac{1}{2}$ or the memory is allocated and requested ($\tilde{\phi}'_6$) with probability $\frac{1}{3}$. On the other hand, the dependence of the steady-state PMF on ρ is more complex and interesting in step semantics than in interleaving one, since the functions $\tilde{\varphi}'_2$, $\tilde{\varphi}'_4$, $\tilde{\varphi}'_6$ have local extremes and more inflections than $\tilde{\phi}'_2$, $\tilde{\phi}'_4$, $\tilde{\phi}'_6$.

11.3. **Application area.** From the application viewpoint, MTIPP and PEPA are well-suited for interleaving continuous time systems, in which the activity rates or the average sojourn time in the states are known in advance and exponential
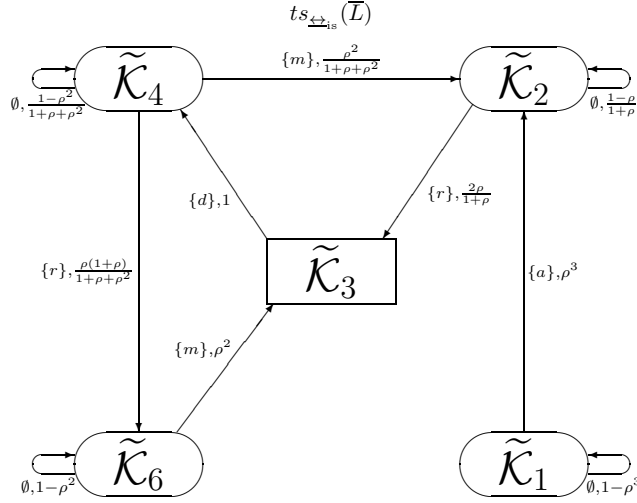
$$ts_{\underleftrightarrow{}_{\mathrm{is}}}(\overline{L})$$

FIG. 18. The interleaving quotient transition system of the abstract generalized shared memory system

$$smc_{\underleftrightarrow{}_{\mathrm{is}}}(\overline{L})$$
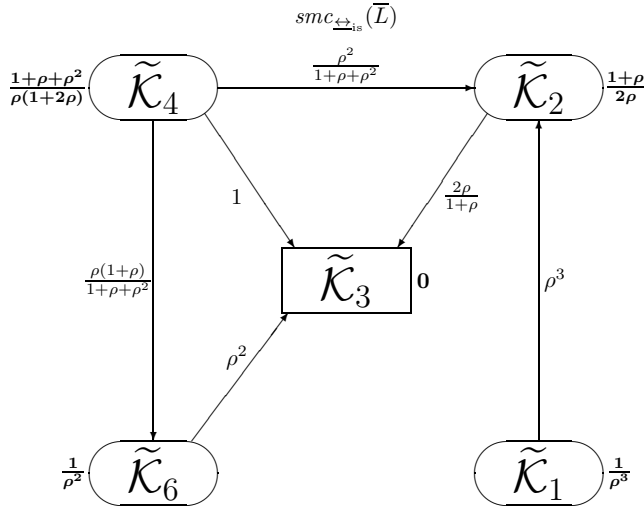

FIG. 19. The interleaving quotient underlying SMC of the abstract generalized shared memory system

distribution approximates well the activity delay distributions. EMPA, however, can be used to model the mentioned systems with the activity delays of different duration order or the extended systems, in which purely probabilistic choices or urgent activities must be implemented. GSPA and GSMPA fit well for modeling continuous time systems with a capability to keep the activity causality information, and with known activity delay distributions, which cannot be approximated accurately by exponential distributions. $S\pi$ can additionally model mobility in such systems. TCP^{dst} is a good choice for interleaving discrete time systems with deterministic (fixed) and generalized stochastic delays, whereas sACP is capable to model non-interleaving systems as well, but it offers not enough performance analysis
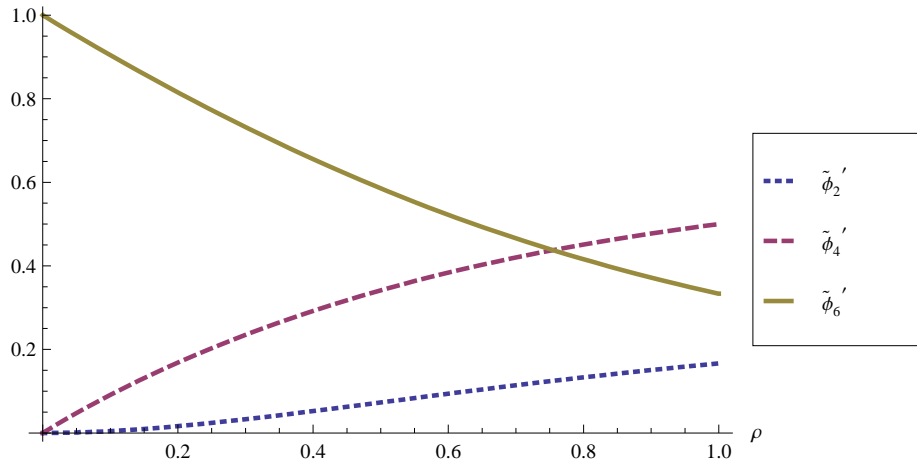
FIG. 20. Interleaving steady-state probabilities $\tilde{\phi}_2'$, $\tilde{\phi}_4'$, $\tilde{\phi}_6'$ as functions of the parameter ρ

methods. dtsiPBC is consistent for the step discrete time systems such that the independent execution probabilities of activities are known and geometrical distribution approximates well the state residence time distributions. In addition, dtsiPBC can model these systems featuring very scattered activity delays or even more complex systems with instantaneous probabilistic choice or urgency, hence, dtsiPBC can be taken as a non-interleaving discrete time counterpart of EMPA.

11.4. **Advantages of dtsiPBC.** The advantages of dtsiPBC are the flexible multi-action labels, immediate multiactions, powerful operations, as well as a step operational and a Petri net denotational semantics allowing for concurrent execution of activities (transitions), together with an ability for analytical and parametric performance evaluation.

12. Conclusion

In this paper, we have proposed a discrete time stochastic extension dtsiPBC of a finite part of PBC enriched with iteration and immediate multiactions. The calculus has a parallel step operational semantics, based on labeled probabilistic transition systems, and a denotational semantics in terms of a subclass of LDTSIPNs. A method of performance evaluation in the framework of the calculus has been presented. Step stochastic bisimulation equivalence of process expressions has been defined and its interrelations with other equivalences of the calculus have been investigated. We have explained how to reduce transition systems and underlying SMCs of expressions w.r.t. the introduced equivalence. We have proved that the mentioned equivalence guarantees identity of the stationary behaviour and the sojourn time properties, and thus preserves performance measures. A case study of a generalization of the shared memory system by allowing for variable probabilities in its specification has been presented. The case study is an example of modeling, performance evaluation and performance preserving reduction within the calculus.

The advantage of our framework is twofold. First, one can specify in it concurrent composition and synchronization of (multi)actions, whereas this is not possible in

classical Markov chains. Second, algebraic formulas represent processes in a more compact way than Petri nets and allow one to apply syntactic transformations and comparisons. Process algebras are compositional by definition and their operations naturally correspond to operators of programming languages. Hence, it is much easier to construct a complex model in the algebraic setting than in PNs. The complexity of PNs generated for practical models in the literature demonstrates that it is not straightforward to construct such PNs directly from the system specifications. dtsiPBC is well suited for the discrete time applications, whose discrete states change with a global time tick, such as business processes, neural and transportation networks, computer and communication systems, timed web services [86], as well as for those, in which the distributed architecture or the concurrency level should be preserved (remember that in step semantics, we have additional transitions due to concurrent executions).

Future work will consist in constructing a congruence for dtsiPBC, i.e. the equivalence that withstands application of all its operations. A possible candidate is a stronger version of $\underline{\leftrightarrow}_{\mathrm{ss}}$ defined via transition systems equipped with two extra transitions skip and redo, like those from [59]. We also plan to extend the calculus with deterministically timed multiactions having a fixed discrete time delay (including the zero one which is the case of immediate multiactions) to enhance expressiveness of the calculus and extend application area of the associated analysis techniques. Hopefully, the resulting SPA will be a concurrent discrete time analogue of SM-PEPA [18], whose underlying stochastic model is a semi-Markov chain. Finally, recursion could be added to dtsiPBC to increase its specification power.

APPENDIX A. PROOFS

A.1. **Proof of Proposition 5.** Like for strong equivalence in Proposition 8.2.1 from [46], we shall prove the following fact about step stochastic bisimulation. Let us have $\forall j \in \mathcal{J}$, $\mathcal{R}_j : G\underline{\leftrightarrow}_{\mathrm{ss}}G'$ for some index set $\mathcal{J}$. Then the transitive closure of the union of all relations $\mathcal{R} = (\cup_{j\in\mathcal{J}}\mathcal{R}_j)^+$ is also an equivalence and $\mathcal{R} : G\underline{\leftrightarrow}_{\mathrm{ss}}G'$.

Since $\forall j \in \mathcal{J}$, $\mathcal{R}_j$ is an equivalence, by definition of $\mathcal{R}$, we get that $\mathcal{R}$ is also an equivalence. Let $j \in \mathcal{J}$, then, by definition of $\mathcal{R}$, $(s_1, s_2) \in \mathcal{R}_j$ implies $(s_1, s_2) \in \mathcal{R}$. Hence, $\forall\mathcal{H}_{jk} \in (DR(G) \cup DR(G'))/_{\mathcal{R}_j}$, $\exists\mathcal{H} \in (DR(G) \cup DR(G'))/_{\mathcal{R}}$, $\mathcal{H}_{jk} \subseteq \mathcal{H}$. Moreover, $\exists\mathcal{J}'$, $\mathcal{H} = \cup_{k\in\mathcal{J}'}\mathcal{H}_{jk}$.

We denote $\mathcal{R}(n) = (\cup_{j\in\mathcal{J}}\mathcal{R}_j)^n$. Let $(s_1, s_2) \in \mathcal{R}$, then, by definition of $\mathcal{R}$, $\exists n > 0$, $(s_1, s_2) \in \mathcal{R}(n)$. We shall prove that $\mathcal{R} : G\underline{\leftrightarrow}_{\mathrm{ss}}G'$ by induction on n. It is clear that $\forall j \in \mathcal{J}$, $\mathcal{R}_j : G\underline{\leftrightarrow}_{\mathrm{ss}}G'$ implies $\forall j \in \mathcal{J}$, $([G]_{\approx}, [G']_{\approx}) \in \mathcal{R}_j$ and we have $([G]_{\approx}, [G']_{\approx}) \in \mathcal{R}$ by definition of $\mathcal{R}$. It remains to prove that $(s_1, s_2) \in \mathcal{R}$ implies $\forall\mathcal{H} \in (DR(G) \cup DR(G'))/_{\mathcal{R}}$, $\forall A \in \mathbb{N}_{\mathrm{fin}}^{\mathcal{L}}$, $PM_A(s_1, \mathcal{H}) = PM_A(s_2, \mathcal{H})$.

- $n = 1$
  In this case, $(s_1, s_2) \in \mathcal{R}$ implies $\exists j \in \mathcal{J}$, $(s_1, s_2) \in \mathcal{R}_j$. Since $\mathcal{R}_j : G\underline{\leftrightarrow}_{\mathrm{ss}}G'$, we get $\forall\mathcal{H} \in (DR(G) \cup DR(G'))/_{\mathcal{R}}$, $\forall A \in \mathbb{N}_{\mathrm{fin}}^{\mathcal{L}}$,

$$PM_A(s_1, \mathcal{H}) = \sum_{k\in\mathcal{J}'} PM_A(s_1, \mathcal{H}_{jk}) = \sum_{k\in\mathcal{J}'} PM_A(s_2, \mathcal{H}_{jk}) = PM_A(s_2, \mathcal{H}).$$

- $n \to n+1$
  Suppose that $\forall m \leq n$, $(s_1, s_2) \in \mathcal{R}(m)$ implies $\forall\mathcal{H} \in (DR(G)\cup DR(G'))/_{\mathcal{R}}$, $\forall A \in \mathbb{N}_{\mathrm{fin}}^{\mathcal{L}}$, $PM_A(s_1, \mathcal{H}) = PM_A(s_2, \mathcal{H})$. Then $(s_1, s_2) \in \mathcal{R}(n+1)$ implies

$\exists j \in \mathcal{J}$, $(s_1, s_2) \in \mathcal{R}_j \circ \mathcal{R}(n)$, i.e. $\exists s_3 \in (DR(G) \cup DR(G'))$, such that $(s_1, s_3) \in \mathcal{R}_j$ and $(s_3, s_2) \in \mathcal{R}(n)$. Then, like for the case $n = 1$, we get $PM_A(s_1, \mathcal{H}) = PM_A(s_3, \mathcal{H})$. By the induction hypothesis, we get $PM_A(s_3, \mathcal{H}) = PM_A(s_2, \mathcal{H})$. Thus, $\forall \mathcal{H} \in (DR(G) \cup DR(G'))/_{\mathcal{R}}$, $\forall A \in \mathbb{N}_{\mathrm{fin}}^{\mathcal{L}}$,

$$PM_A(s_1, \mathcal{H}) = PM_A(s_3, \mathcal{H}) = PM_A(s_2, \mathcal{H}).$$

By definition, $\mathcal{R}_{\mathrm{ss}}(G, G')$ is at least as large as the largest step stochastic bisimulation between G and G'. It follows from above that $\mathcal{R}_{\mathrm{ss}}(G, G')$ is an equivalence and $\mathcal{R}_{\mathrm{ss}}(G, G') : G\underline{\leftrightarrow}_{\mathrm{ss}}G'$, hence, it is the largest step stochastic bisimulation between G and G'. $\qquad\square$

A.2. **Proof of Proposition 6.** By Proposition 4, $(DR(G) \cup DR(G'))/_{\mathcal{R}} = ((DR_{\mathrm{T}}(G) \cup DR_{\mathrm{T}}(G'))/_{\mathcal{R}}) \uplus ((DR_{\mathrm{V}}(G) \cup DR_{\mathrm{V}}(G'))/_{\mathcal{R}})$. Hence, $\forall \mathcal{H} \in (DR(G) \cup DR(G'))/_{\mathcal{R}}$, all states from $\mathcal{H}$ are tangible, when $\mathcal{H} \in (DR_{\mathrm{T}}(G) \cup DR_{\mathrm{T}}(G'))/_{\mathcal{R}}$, or all of them are vanishing, when $\mathcal{H} \in (DR_{\mathrm{V}}(G) \cup DR_{\mathrm{V}}(G'))/_{\mathcal{R}}$.

By definition of the steady-state PMFs for SMCs, $\forall s \in DR_{\mathrm{V}}(G)$, $\varphi(s) = 0$ and $\forall s' \in DR_{\mathrm{V}}(G')$, $\varphi'(s') = 0$. Thus, $\forall \mathcal{H} \in (DR_{\mathrm{V}}(G) \cup DR_{\mathrm{V}}(G'))/_{\mathcal{R}}$, $\sum_{s \in \mathcal{H} \cap DR(G)} \varphi(s) = \sum_{s \in \mathcal{H} \cap DR_{\mathrm{V}}(G)} \varphi(s) = 0 = \sum_{s' \in \mathcal{H} \cap DR_{\mathrm{V}}(G')} \varphi'(s') = \sum_{s' \in \mathcal{H} \cap DR(G')} \varphi'(s')$.

By Proposition 3, $\forall s \in DR_{\mathrm{T}}(G)$, $\varphi(s) = \frac{\psi(s)}{\sum_{\tilde{s} \in DR_{\mathrm{T}}(G)} \psi(\tilde{s})}$ and $\forall s' \in DR_{\mathrm{T}}(G')$, $\varphi'(s') = \frac{\psi'(s')}{\sum_{\tilde{s}' \in DR_{\mathrm{T}}(G')} \psi'(\tilde{s}')}$, where ψ and ψ' are the steady-state PMFs for $DTMC(G)$ and $DTMC(G')$, respectively. Thus, $\forall \mathcal{H}, \widetilde{\mathcal{H}} \in (DR_{\mathrm{T}}(G) \cup DR_{\mathrm{T}}(G'))/_{\mathcal{R}}$,

$\sum_{s \in \mathcal{H} \cap DR(G)} \varphi(s) = \sum_{s \in \mathcal{H} \cap DR_{\mathrm{T}}(G)} \varphi(s) = \sum_{s \in \mathcal{H} \cap DR_{\mathrm{T}}(G)} \left(\frac{\psi(s)}{\sum_{\tilde{s} \in DR_{\mathrm{T}}(G)} \psi(\tilde{s})} \right) = \frac{\sum_{s \in \mathcal{H} \cap DR_{\mathrm{T}}(G)} \psi(s)}{\sum_{\tilde{s} \in DR_{\mathrm{T}}(G)} \psi(\tilde{s})} = \frac{\sum_{s \in \mathcal{H} \cap DR_{\mathrm{T}}(G)} \psi(s)}{\sum_{\widetilde{\mathcal{H}}} \sum_{\tilde{s} \in \widetilde{\mathcal{H}} \cap DR_{\mathrm{T}}(G)} \psi(\tilde{s})}$ and $\sum_{s' \in \mathcal{H} \cap DR(G')} \varphi'(s') = \sum_{s' \in \mathcal{H} \cap DR_{\mathrm{T}}(G')} \varphi'(s') = \sum_{s' \in \mathcal{H} \cap DR_{\mathrm{T}}(G')} \left(\frac{\psi'(s')}{\sum_{\tilde{s}' \in DR_{\mathrm{T}}(G')} \psi'(\tilde{s}')} \right) = \frac{\sum_{s' \in \mathcal{H} \cap DR_{\mathrm{T}}(G')} \psi'(s')}{\sum_{\tilde{s}' \in DR_{\mathrm{T}}(G')} \psi'(\tilde{s}')} = \frac{\sum_{s' \in \mathcal{H} \cap DR_{\mathrm{T}}(G')} \psi'(s')}{\sum_{\widetilde{\mathcal{H}}} \sum_{\tilde{s}' \in \widetilde{\mathcal{H}} \cap DR_{\mathrm{T}}(G')} \psi'(\tilde{s}')}$.

It remains to prove that $\forall \mathcal{H} \in (DR_{\mathrm{T}}(G) \cup DR_{\mathrm{T}}(G'))/_{\mathcal{R}}$, $\sum_{s \in \mathcal{H} \cap DR_{\mathrm{T}}(G)} \psi(s) = \sum_{s' \in \mathcal{H} \cap DR_{\mathrm{T}}(G')} \psi'(s')$. Since $(DR(G) \cup DR(G'))/_{\mathcal{R}} = ((DR_{\mathrm{T}}(G) \cup DR_{\mathrm{T}}(G'))/_{\mathcal{R}}) \uplus ((DR_{\mathrm{V}}(G) \cup DR_{\mathrm{V}}(G'))/_{\mathcal{R}})$, the previous equality is a consequence of the following: $\forall \mathcal{H} \in (DR(G) \cup DR(G'))/_{\mathcal{R}}$, $\sum_{s \in \mathcal{H} \cap DR(G)} \psi(s) = \sum_{s' \in \mathcal{H} \cap DR(G')} \psi'(s')$.

It is sufficient to prove the previous statement for transient PMFs only, since $\psi = \lim_{k \to \infty} \psi[k]$ and $\psi' = \lim_{k \to \infty} \psi'[k]$. We proceed by induction on k.

- $k = 0$

  The only nonzero values of the initial PMFs of $DTMC(G)$ and $DTMC(G')$ are $\psi[0]([G]_{\approx})$ and $\psi[0]([G']_{\approx})$. Let $\mathcal{H}_0$ be the equivalence class containing $[G]_{\approx}$ and $[G']_{\approx}$. Then $\sum_{s \in \mathcal{H}_0 \cap DR(G)} \psi[0](s) = \psi[0]([G]_{\approx}) = 1 = \psi'[0]([G']_{\approx}) = \sum_{s' \in \mathcal{H}_0 \cap DR(G')} \psi'[0](s')$. As for other equivalence classes, $\forall \mathcal{H} \in ((DR(G) \cup DR(G'))/_{\mathcal{R}}) \setminus \mathcal{H}_0$, we have $\sum_{s \in \mathcal{H} \cap DR(G)} \psi[0](s) = 0 = \sum_{s' \in \mathcal{H} \cap DR(G')} \psi'[0](s')$.

- $k \to k + 1$

  Let $\mathcal{H} \in (DR(G) \cup DR(G'))/_{\mathcal{R}}$ and $s_1, s_2 \in \mathcal{H}$. We have $\forall \widetilde{\mathcal{H}} \in (DR(G) \cup DR(G'))/_{\mathcal{R}}$, $\forall A \in \mathbb{N}_{\mathrm{fin}}^{\mathcal{L}}$, $s_1 \xrightarrow{A}_{\mathcal{P}} \widetilde{\mathcal{H}} \Leftrightarrow s_2 \xrightarrow{A}_{\mathcal{P}} \widetilde{\mathcal{H}}$.

  Therefore, $PM(s_1, \widetilde{\mathcal{H}}) = \sum_{\{\Upsilon | \exists \tilde{s}_1 \in \widetilde{\mathcal{H}}, \ s_1 \xrightarrow{\Upsilon} \tilde{s}_1\}} PT(\Upsilon, s_1) = \sum_{A \in \mathbb{N}_{\mathrm{fin}}^{\mathcal{L}}} \sum_{\{\Upsilon | \exists \tilde{s}_1 \in \widetilde{\mathcal{H}}, \ s_1 \xrightarrow{\Upsilon} \tilde{s}_1, \ \mathcal{L}(\Upsilon) = A\}} PT(\Upsilon, s_1) = \sum_{A \in \mathbb{N}_{\mathrm{fin}}^{\mathcal{L}}} PM_A(s_1, \widetilde{\mathcal{H}}) =$

$\sum_{A\in\mathbb{N}^{\mathcal{L}}_{\mathrm{fin}}}PM_A(s_2,\widetilde{\mathcal{H}})=\sum_{A\in\mathbb{N}^{\mathcal{L}}_{\mathrm{fin}}}\sum_{\{\Upsilon|\exists\tilde{s}_2\in\widetilde{\mathcal{H}},\ s_2\xrightarrow{\Upsilon}\tilde{s}_2,\ \mathcal{L}(\Upsilon)=A\}}PT(\Upsilon,s_2)=$

$\sum_{\{\Upsilon|\exists\tilde{s}_2\in\widetilde{\mathcal{H}},\ s_2\xrightarrow{\Upsilon}\tilde{s}_2\}}PT(\Upsilon,s_2)=PM(s_2,\widetilde{\mathcal{H}})$. Since this equality is valid for all $s_1,s_2\in\mathcal{H}$, we can denote $PM(\mathcal{H},\widetilde{\mathcal{H}})=PM(s_1,\widetilde{\mathcal{H}})=PM(s_2,\widetilde{\mathcal{H}})$. Transitions from the states of $DR(G)$ always lead to those from the same set, hence, $\forall s\in DR(G),\ PM(s,\widetilde{\mathcal{H}})=PM(s,\widetilde{\mathcal{H}}\cap DR(G))$. The same holds for $DR(G')$.

By induction hypothesis, $\sum_{s\in\mathcal{H}\cap DR(G)}\psi[k](s)=\sum_{s'\in\mathcal{H}\cap DR(G')}\psi'[k](s')$. Then $\sum_{\tilde{s}\in\widetilde{\mathcal{H}}\cap DR(G)}\psi[k+1](\tilde{s})=\sum_{\tilde{s}\in\widetilde{\mathcal{H}}\cap DR(G)}\sum_{s\in DR(G)}\psi[k](s)PM(s,\tilde{s})=$

$\sum_{s\in DR(G)}\sum_{\tilde{s}\in\widetilde{\mathcal{H}}\cap DR(G)}\psi[k](s)PM(s,\tilde{s})=$

$\sum_{s\in DR(G)}\psi[k](s)\sum_{\tilde{s}\in\widetilde{\mathcal{H}}\cap DR(G)}PM(s,\tilde{s})=$

$\sum_{\mathcal{H}}\sum_{s\in\mathcal{H}\cap DR(G)}\psi[k](s)\sum_{\tilde{s}\in\widetilde{\mathcal{H}}\cap DR(G)}PM(s,\tilde{s})=$

$\sum_{\mathcal{H}}\sum_{s\in\mathcal{H}\cap DR(G)}\psi[k](s)\sum_{\tilde{s}\in\widetilde{\mathcal{H}}\cap DR(G)}\sum_{\{\Upsilon|s\xrightarrow{\Upsilon}\tilde{s}\}}PT(\Upsilon,s)=$

$\sum_{\mathcal{H}}\sum_{s\in\mathcal{H}\cap DR(G)}\psi[k](s)\sum_{\{\Upsilon|\exists\tilde{s}\in\widetilde{\mathcal{H}}\cap DR(G),\ s\xrightarrow{\Upsilon}\tilde{s}\}}PT(\Upsilon,s)=$

$\sum_{\mathcal{H}}\sum_{s\in\mathcal{H}\cap DR(G)}\psi[k](s)PM(s,\widetilde{\mathcal{H}})=\sum_{\mathcal{H}}\sum_{s\in\mathcal{H}\cap DR(G)}\psi[k](s)PM(\mathcal{H},\widetilde{\mathcal{H}})=$

$\sum_{\mathcal{H}}PM(\mathcal{H},\widetilde{\mathcal{H}})\sum_{s\in\mathcal{H}\cap DR(G)}\psi[k](s)=$

$\sum_{\mathcal{H}}PM(\mathcal{H},\widetilde{\mathcal{H}})\sum_{s'\in\mathcal{H}\cap DR(G')}\psi'[k](s')=$

$\sum_{\mathcal{H}}\sum_{s'\in\mathcal{H}\cap DR(G')}\psi'[k](s')PM(\mathcal{H},\widetilde{\mathcal{H}})=$

$\sum_{\mathcal{H}}\sum_{s'\in\mathcal{H}'\cap DR(G')}\psi'[k](s')PM(s',\widetilde{\mathcal{H}})=$

$\sum_{\mathcal{H}}\sum_{s'\in\mathcal{H}\cap DR(G')}\psi'[k](s')\sum_{\{\Upsilon|\exists\tilde{s}'\in\widetilde{\mathcal{H}}\cap DR(G'),\ s'\xrightarrow{\Upsilon}\tilde{s}'\}}PT(\Upsilon,s')=$

$\sum_{\mathcal{H}}\sum_{s'\in\mathcal{H}\cap DR(G')}\psi'[k](s')\sum_{\tilde{s}'\in\widetilde{\mathcal{H}}\cap DR(G')}\sum_{\{\Upsilon|\exists\tilde{s}',\ s'\xrightarrow{\Upsilon}\tilde{s}'\}}PT(\Upsilon,s')=$

$\sum_{\mathcal{H}}\sum_{s'\in\mathcal{H}\cap DR(G')}\psi'[k](s')\sum_{\tilde{s}'\in\widetilde{\mathcal{H}}\cap DR(G')}PM(s',\tilde{s}')=$

$\sum_{s'\in DR(G')}\psi'[k](s')\sum_{\tilde{s}'\in\widetilde{\mathcal{H}}\cap DR(G')}PM(s',\tilde{s}')=$

$\sum_{s'\in DR(G')}\sum_{\tilde{s}'\in\widetilde{\mathcal{H}}\cap DR(G')}\psi'[k](s')PM(s',\tilde{s}')=$

$\sum_{\tilde{s}'\in\widetilde{\mathcal{H}}\cap DR(G')}\sum_{s'\in DR(G')}\psi'[k](s')PM(s',\tilde{s}')=\sum_{\tilde{s}'\in\widetilde{\mathcal{H}}\cap DR(G')}\psi'[k+1](\tilde{s}')$.
$\square$

A.3. **Proof of Theorem 4.** Let $\mathcal{H}\in(DR(G)\cup DR(G'))/_{\mathcal{R}}$ and $s,\bar{s}\in\mathcal{H}$. We have $\forall\widetilde{\mathcal{H}}\in(DR(G)\cup DR(G'))/_{\mathcal{R}},\ \forall A\in\mathbb{N}^{\mathcal{L}}_{\mathrm{fin}},\ s\xrightarrow{A}_{\mathcal{P}}\widetilde{\mathcal{H}}\Leftrightarrow\bar{s}\xrightarrow{A}_{\mathcal{P}}\widetilde{\mathcal{H}}$. Since this equality is valid for all $s,\bar{s}\in\mathcal{H}$, we can rewrite it as $\mathcal{H}\xrightarrow{A}_{\mathcal{P}}\widetilde{\mathcal{H}}$ and denote $PM_A(\mathcal{H},\widetilde{\mathcal{H}})=PM_A(s,\widetilde{\mathcal{H}})=PM_A(\bar{s},\widetilde{\mathcal{H}})$. The transitions from the states of $DR(G)$ always lead to those from the same set, hence, $\forall s\in DR(G),\ PM_A(s,\widetilde{\mathcal{H}})=PM_A(s,\widetilde{\mathcal{H}}\cap DR(G))$. The same holds for $DR(G')$.

Let $\Sigma=A_1\cdots A_n$ be a derived step trace of G and G'. Then $\exists\mathcal{H}_0,\ldots,\mathcal{H}_n\in(DR(G)\cup DR(G'))/_{\mathcal{R}},\ \mathcal{H}_0\xrightarrow{A_1}_{\mathcal{P}_1}\mathcal{H}_1\xrightarrow{A_2}_{\mathcal{P}_2}\cdots\xrightarrow{A_n}_{\mathcal{P}_n}\mathcal{H}_n$. Let us prove that the sum of probabilities of all the paths starting in every $s_0\in\mathcal{H}_0$ and going through the states from $\mathcal{H}_1,\ldots,\mathcal{H}_n$ is equal to the product of $\mathcal{P}_1,\ldots,\mathcal{P}_n$:

$$\sum_{\{\Upsilon_1,\ldots,\Upsilon_n|s_0\xrightarrow{\Upsilon_1}\cdots\xrightarrow{\Upsilon_n}s_n,\ \mathcal{L}(\Upsilon_i)=A_i,\ s_i\in\mathcal{H}_i\ (1\leq i\leq n)\}}\prod_{i=1}^{n}PT(\Upsilon_i,s_{i-1})=\prod_{i=1}^{n}PM_{A_i}(\mathcal{H}_{i-1},\mathcal{H}_i).$$

We prove this equality by induction on the derived step trace length n.

- $n=1$

$\sum_{\{\Upsilon_1|s_0\xrightarrow{\Upsilon_1}s_1,\ \mathcal{L}(\Upsilon_1)=A_1,\ s_1\in\mathcal{H}_1\}}PT(\Upsilon_1,s_0)=PM_{A_1}(s_0,\mathcal{H}_1)=PM_{A_1}(\mathcal{H}_0,\mathcal{H}_1)$.

- $n \to n+1$

$$\sum_{\{\Upsilon_1,\ldots,\Upsilon_n,\Upsilon_{n+1}|s_0 \overset{\Upsilon_1}{\to}\ldots\overset{\Upsilon_n}{\to}s_n \overset{\Upsilon_{n+1}}{\to}s_{n+1},\ \mathcal{L}(\Upsilon_i)=A_i,\ s_i\in\mathcal{H}_i\ (1\le i\le n+1)\}}$$
$$\prod_{i=1}^{n+1} PT(\Upsilon_i, s_{i-1}) = \sum_{\{\Upsilon_1,\ldots,\Upsilon_n|s_0 \overset{\Upsilon_1}{\to}\ldots\overset{\Upsilon_n}{\to}s_n,\ \mathcal{L}(\Upsilon_i)=A_i,\ s_i\in\mathcal{H}_i\ (1\le i\le n)\}}$$
$$\sum_{\{\Upsilon_{n+1}|s_n \overset{\Upsilon_{n+1}}{\to}s_{n+1},\ \mathcal{L}(\Upsilon_{n+1})=A_{n+1},\ s_n\in\mathcal{H}_n,\ s_{n+1}\in\mathcal{H}_{n+1}\}}$$
$$\prod_{i=1}^{n} PT(\Upsilon_i, s_{i-1})PT(\Upsilon_{n+1}, s_n) =$$
$$\sum_{\{\Upsilon_1,\ldots,\Upsilon_n|s_0 \overset{\Upsilon_1}{\to}\ldots\overset{\Upsilon_n}{\to}s_n,\ \mathcal{L}(\Upsilon_i)=A_i,\ s_i\in\mathcal{H}_i\ (1\le i\le n)\}} \left[\prod_{i=1}^{n} PT(\Upsilon_i, s_{i-1})\right.$$
$$\left.\sum_{\{\Upsilon_{n+1}|s_n \overset{\Upsilon_{n+1}}{\to}s_{n+1},\ \mathcal{L}(\Upsilon_{n+1})=A_{n+1},\ s_n\in\mathcal{H}_n,\ s_{n+1}\in\mathcal{H}_{n+1}\}} PT(\Upsilon_{n+1}, s_n)\right] =$$
$$\sum_{\{\Upsilon_1,\ldots,\Upsilon_n|s_0 \overset{\Upsilon_1}{\to}\ldots\overset{\Upsilon_n}{\to}s_n,\ \mathcal{L}(\Upsilon_i)=A_i,\ s_i\in\mathcal{H}_i\ (1\le i\le n)\}}$$
$$\prod_{i=1}^{n} PT(\Upsilon_i, s_{i-1})PM_{A_{n+1}}(s_n, \mathcal{H}_{n+1}) =$$
$$\sum_{\{\Upsilon_1,\ldots,\Upsilon_n|s_0 \overset{\Upsilon_1}{\to}\ldots\overset{\Upsilon_n}{\to}s_n,\ \mathcal{L}(\Upsilon_i)=A_i,\ s_i\in\mathcal{H}_i\ (1\le i\le n)\}}$$
$$\prod_{i=1}^{n} PT(\Upsilon_i, s_{i-1})PM_{A_{n+1}}(\mathcal{H}_n, \mathcal{H}_{n+1}) =$$
$$PM_{A_{n+1}}(\mathcal{H}_n, \mathcal{H}_{n+1}) \sum_{\{\Upsilon_1,\ldots,\Upsilon_n|s_0 \overset{\Upsilon_1}{\to}\ldots\overset{\Upsilon_n}{\to}s_n,\ \mathcal{L}(\Upsilon_i)=A_i,\ s_i\in\mathcal{H}_i\ (1\le i\le n)\}}$$
$$\prod_{i=1}^{n} PT(\Upsilon_i, s_{i-1}) = PM_{A_{n+1}}(\mathcal{H}_n, \mathcal{H}_{n+1})\prod_{i=1}^{n} PM_{A_i}(\mathcal{H}_{i-1}, \mathcal{H}_i) =$$
$$\prod_{i=1}^{n+1} PM_{A_i}(\mathcal{H}_{i-1}, \mathcal{H}_i).$$

Let $s_0, \bar{s}_0 \in \mathcal{H}_0$. We have $PT(A_1 \cdots A_n, s_0) =$
$$\sum_{\{\Upsilon_1,\ldots,\Upsilon_n|s_0 \overset{\Upsilon_1}{\to}\ldots\overset{\Upsilon_n}{\to}s_n,\ \mathcal{L}(\Upsilon_i)=A_i,\ (1\le i\le n)\}} \prod_{i=1}^{n} PT(\Upsilon_i, s_{i-1}) =$$
$$\sum_{\mathcal{H}_1,\ldots,\mathcal{H}_n} \sum_{\{\Upsilon_1,\ldots,\Upsilon_n|s_0 \overset{\Upsilon_1}{\to}\ldots\overset{\Upsilon_n}{\to}s_n,\ \mathcal{L}(\Upsilon_i)=A_i,\ s_i\in\mathcal{H}_i\ (1\le i\le n)\}} \prod_{i=1}^{n} PT(\Upsilon_i, s_{i-1}) =$$
$$\sum_{\mathcal{H}_1,\ldots,\mathcal{H}_n} \prod_{i=1}^{n} PM_{A_i}(\mathcal{H}_{i-1}, \mathcal{H}_i) =$$
$$\sum_{\mathcal{H}_1,\ldots,\mathcal{H}_n} \sum_{\{\overline{\Upsilon}_1,\ldots,\overline{\Upsilon}_n|\bar{s}_0 \overset{\overline{\Upsilon}_1}{\to}\ldots\overset{\overline{\Upsilon}_n}{\to}\bar{s}_n,\ \mathcal{L}(\overline{\Upsilon}_i)=A_i,\ \bar{s}_i\in\mathcal{H}_i\ (1\le i\le n)\}} \prod_{i=1}^{n} PT(\overline{\Upsilon}_i, \bar{s}_{i-1}) =$$
$$\sum_{\{\overline{\Upsilon}_1,\ldots,\overline{\Upsilon}_n|\bar{s}_0 \overset{\overline{\Upsilon}_1}{\to}\ldots\overset{\overline{\Upsilon}_n}{\to}\bar{s}_n,\ \mathcal{L}(\overline{\Upsilon}_i)=A_i,\ (1\le i\le n)\}} \prod_{i=1}^{n} PT(\overline{\Upsilon}_i, \bar{s}_{i-1}) = PT(A_1 \cdots A_n, \bar{s}_0).$$
Since we have the previous equality for all $s_0, \bar{s}_0 \in \mathcal{H}_0$, we can denote
$PT(A_1 \cdots A_n, \mathcal{H}_0) = PT(A_1 \cdots A_n, s_0) = PT(A_1 \cdots A_n, \bar{s}_0).$

By Proposition 6, $\sum_{s\in\mathcal{H}\cap DR(G)} \varphi(s) = \sum_{s'\in\mathcal{H}\cap DR(G')} \varphi'(s')$. We now can complete the proof: $\sum_{s\in\mathcal{H}\cap DR(G)} \varphi(s)PT(\Sigma, s) = \sum_{s\in\mathcal{H}\cap DR(G)} \varphi(s)PT(\Sigma, \mathcal{H}) =$
$PT(\Sigma, \mathcal{H}) \sum_{s\in\mathcal{H}\cap DR(G)} \varphi(s) = PT(\Sigma, \mathcal{H}) \sum_{s'\in\mathcal{H}\cap DR(G')} \varphi'(s') =$
$\sum_{s'\in\mathcal{H}\cap DR(G')} \varphi'(s')PT(\Sigma, \mathcal{H}) = \sum_{s'\in\mathcal{H}\cap DR(G')} \varphi'(s')PT(\Sigma, s').$ $\qquad\square$

A.4. **Proof of Proposition 7.** We present two facts that will be used in the proof.

(1) By Proposition 4, $(DR(G) \cup DR(G'))/_{\mathcal{R}} = ((DR_{\mathrm{T}}(G) \cup DR_{\mathrm{T}}(G'))/_{\mathcal{R}}) \uplus$
$((DR_{\mathrm{V}}(G) \cup DR_{\mathrm{V}}(G'))/_{\mathcal{R}})$. Hence, $\forall\mathcal{H} \in (DR(G) \cup DR(G'))/_{\mathcal{R}}$, all states from $\mathcal{H}$ are tangible, when $\mathcal{H} \in (DR_{\mathrm{T}}(G) \cup DR_{\mathrm{T}}(G'))/_{\mathcal{R}}$, or all of them are vanishing, when $\mathcal{H} \in (DR_{\mathrm{V}}(G) \cup DR_{\mathrm{V}}(G'))/_{\mathcal{R}}$.

(2) Let $\mathcal{H} \in (DR(G) \cup DR(G'))/_{\mathcal{R}}$ and $s_1, s_2 \in \mathcal{H}$. We have $\forall\widetilde{\mathcal{H}} \in (DR(G) \cup DR(G'))/_{\mathcal{R}}$, $\forall A \in \mathbb{N}_{\mathrm{fin}}^{\mathcal{L}}$, $s_1 \overset{A}{\to}_{\mathcal{P}} \widetilde{\mathcal{H}} \Leftrightarrow s_2 \overset{A}{\to}_{\mathcal{P}} \widetilde{\mathcal{H}}$.
Therefore, $PM(s_1, \widetilde{\mathcal{H}}) = \sum_{\{\Upsilon|\exists\tilde{s}_1\in\widetilde{\mathcal{H}},\ s_1\overset{\Upsilon}{\to}\tilde{s}_1\}} PT(\Upsilon, s_1) =$
$\sum_{A\in\mathbb{N}_{\mathrm{fin}}^{\mathcal{L}}} \sum_{\{\Upsilon|\exists\tilde{s}_1\in\widetilde{\mathcal{H}},\ s_1\overset{\Upsilon}{\to}\tilde{s}_1,\ \mathcal{L}(\Upsilon)=A\}} PT(\Upsilon, s_1) = \sum_{A\in\mathbb{N}_{\mathrm{fin}}^{\mathcal{L}}} PM_A(s_1, \widetilde{\mathcal{H}}) =$
$\sum_{A\in\mathbb{N}_{\mathrm{fin}}^{\mathcal{L}}} PM_A(s_2, \widetilde{\mathcal{H}}) = \sum_{A\in\mathbb{N}_{\mathrm{fin}}^{\mathcal{L}}} \sum_{\{\Upsilon|\exists\tilde{s}_2\in\widetilde{\mathcal{H}},\ s_2\overset{\Upsilon}{\to}\tilde{s}_2,\ \mathcal{L}(\Upsilon)=A\}} PT(\Upsilon, s_2) =$
$\sum_{\{\Upsilon|\exists\tilde{s}_2\in\widetilde{\mathcal{H}},\ s_2\overset{\Upsilon}{\to}\tilde{s}_2\}} PT(\Upsilon, s_2) = PM(s_2, \widetilde{\mathcal{H}})$. Since we have the previous equality for all $s_1, s_2 \in \mathcal{H}$, we can denote $PM(\mathcal{H}, \widetilde{\mathcal{H}}) = PM(s_1, \widetilde{\mathcal{H}}) =$

$PM(s_2, \widetilde{\mathcal{H}})$. The transitions from the states of $DR(G)$ always lead to those from the same set, hence, $\forall s \in DR(G)$, $PM(s, \widetilde{\mathcal{H}}) = PM(s, \widetilde{\mathcal{H}} \cap DR(G))$. The same is true for $DR(G')$. Hence, for all $s \in \mathcal{H} \cap DR(G)$, we obtain $PM(\mathcal{H}, \widetilde{\mathcal{H}}) = PM(s, \widetilde{\mathcal{H}}) = PM(s, \widetilde{\mathcal{H}} \cap DR(G)) = PM(\mathcal{H} \cap DR(G), \widetilde{\mathcal{H}} \cap DR(G))$. The same is true for $DR(G')$. Finally, $PM(\mathcal{H} \cap DR(G), \widetilde{\mathcal{H}} \cap DR(G)) = PM(\mathcal{H}, \widetilde{\mathcal{H}}) = PM(\mathcal{H} \cap DR(G'), \widetilde{\mathcal{H}} \cap DR(G'))$.

Let us now prove the proposition statement for the sojourn time averages.

- Let $\mathcal{H} \in (DR_V(G) \cup DR_V(G'))/_{\mathcal{R}}$.
  We have $\mathcal{H} \cap DR(G) = \mathcal{H} \cap DR_V(G) \in DR_V(G)/_{\mathcal{R}}$ and $\mathcal{H} \cap DR(G') = \mathcal{H} \cap DR_V(G') \in DR_V(G')/_{\mathcal{R}}$. By definition of the average sojourn time in an equivalence class of states, we get $SJ_{\mathcal{R} \cap (DR(G))^2}(\mathcal{H} \cap DR(G)) = SJ_{\mathcal{R} \cap (DR(G))^2}(\mathcal{H} \cap DR_V(G)) = 0 = SJ_{\mathcal{R} \cap (DR(G'))^2}(\mathcal{H} \cap DR_V(G')) = SJ_{\mathcal{R} \cap (DR(G'))^2}(\mathcal{H} \cap DR(G'))$.
- Let $\mathcal{H} \in (DR_T(G) \cup DR_T(G'))/_{\mathcal{R}}$.
  We have $\mathcal{H} \cap DR(G) = \mathcal{H} \cap DR_T(G) \in DR_T(G)/_{\mathcal{R}}$ and $\mathcal{H} \cap DR(G') = \mathcal{H} \cap DR_T(G') \in DR_T(G')/_{\mathcal{R}}$. By definition of the average sojourn time in an equivalence class of states, we get $SJ_{\mathcal{R} \cap (DR(G))^2}(\mathcal{H} \cap DR(G)) = SJ_{\mathcal{R} \cap (DR(G))^2}(\mathcal{H} \cap DR_T(G)) = \frac{1}{1 - PM(\mathcal{H} \cap DR_T(G), \mathcal{H} \cap DR_T(G))} = \frac{1}{1 - PM(\mathcal{H} \cap DR(G), \mathcal{H} \cap DR(G))} = \frac{1}{1 - PM(\mathcal{H}, \mathcal{H})} = \frac{1}{1 - PM(\mathcal{H} \cap DR(G'), \mathcal{H} \cap DR(G'))} = \frac{1}{1 - PM(\mathcal{H} \cap DR_T(G'), \mathcal{H} \cap DR_T(G'))} = SJ_{\mathcal{R} \cap (DR(G'))^2}(\mathcal{H} \cap DR_T(G')) = SJ_{\mathcal{R} \cap (DR(G'))^2}(\mathcal{H} \cap DR(G'))$.

Thus, $\forall \mathcal{H} \in (DR(G) \cup DR(G'))/_{\mathcal{R}}$ we have $SJ_{\mathcal{R} \cap (DR(G))^2}(\mathcal{H} \cap DR(G)) = SJ_{\mathcal{R} \cap (DR(G'))^2}(\mathcal{H} \cap DR(G'))$.

The proposition statement for the sojourn time variances is proved similarly. $\square$

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

[2] C. Autant, Ph. Schnoebelen, *Place bisimulations in Petri nets*, Lecture Notes in Computer Science, **616** (1992), 45–61.

[3] C. Baier, *Polynomial time algorithms for testing probabilistic bisimulation and simulation*, Lecture Notes in Computer Science, **1102** (1996), 50–61.

[4] C. Baier, B. Engelen, M. Majster-Cederbaum, *Deciding bisimilarity and similarity for probabilistic processes*, Journal of Computer and System Sciences, **60**:1 (2000), 187–231.

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

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

[7] 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

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

[9] M. Bernardo, *Theory and application of extended Markovian process algebra*, Ph.D. thesis, 276 p., University of Bologna, Italy, 1999. http://www.sti.uniurb.it/bernardo/documents/ phdthesis.pdf

[10] M. Bernardo, *A survey of Markovian behavioral equivalences*, Lecture Notes in Computer Science, **4486** (2007), 180–219.

[11] M. Bernardo, *On the tradeoff between compositionality and exactness in weak bisimilarity for integrated-time Markovian process calculi*, Theoretical Computer Science, **563** (2015), 99–143.

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

[13] M. Bernardo, L. Donatiello, R. Gorrieri, *A formal approach to the integration of performance aspects in the modeling and analysis of concurrent systems*, Information and Computation, **144**:2 (1998), 83–154.

[14] 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.

[15] 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.

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

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

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

[19] M. Bravetti, *Specification and analysis of stochastic real-time systems*, Ph.D. thesis, 432 p., University of Bologna, Italy, 2002. http://www.cs.unibo.it/~bravetti/papers/phdthesis.ps.gz

[20] M. Bravetti, P.R. D'Argenio, *Tutte le algebre insieme: concepts, discussions and relations of stochastic process algebras with general distributions*, Lecture Notes in Computer Science, **2925** (2004), 44–88.

[21] M. Bravetti, M. Bernardo, R. Gorrieri, *Towards performance evaluation with general distributions in process algebras*, Lecture Notes in Computer Science, **1466** (1998), 405–422.

[22] E. Brinksma, H. Hermanns, *Process algebra and Markov chains*, Lecture Notes in Computer Science, **2090** (2001), 183–231.

[23] E. Brinksma, J.-P. Katoen, R. Langerak, D. Latella, *A stochastic causality-based process algebra*, The Computer Journal, **38**:7 (1995), 552–565.

[24] G. Bucci, L. Sassoli, E. Vicario, *Correctness verification and performance analysis of real-time systems using stochastic preemptive time Petri nets*, IEEE Transactions on Software Engineering, **31**:11 (2005), 913–927.

[25] P. Buchholz, *Markovian process algebra: composition and equivalence*, Proc. 2nd 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), 11–30.

[26] P. Buchholz, *Exact and ordinary lumpability in finite Markov chains*, Journal of Applied Probability, **31**:1 (1994), 59–75.

[27] P. Buchholz, *A notion of equivalence for stochastic Petri nets*, Lecture Notes in Computer Science, **935** (1995), 161–180.

[28] P. Buchholz, *Iterative decomposition and aggregation of labeled GSPNs*, Lecture Notes in Computer Science, **1420** (1998), 226–245.

[29] P. Buchholz, I.V. Tarasyuk, *Net and algebraic approaches to probabilistic modeling*, Joint Novosibirsk Computing Center and Institute of Informatics Systems Bulletin, Series Computer Science, **15** (2001), 31–64.

[30] S. Cattani, R. Segala, *Decision algorithms for probabilistic bisimulation*, Lecture Notes in Computer Science, **2421** (2002), 371–385.

[31] I. Christoff, *Testing equivalence and fully abstract models of probabilistic processes*, Lecture Notes in Computer Science, **458** (1990), 126–140.

[32] C. Daws, *Symbolic and parametric model checking of discrete-time Markov chains*, Lecture Notes in Computer Science, **3407** (2005), 280–294.

[33] P. Degano, C. Priami, *Non-interleaving semantics for mobile processes*, Theoretical Computer Science, **216**:1–2 (1999), 237–270.

[34] S. Derisavi, H. Hermanns, W.H. Sanders, *Optimal state-space lumping of Markov chains*, Information Processing Letters, **87**:6 (2003), 309–315.

[35] Ch. Eisentraut, H. Hermanns, J. Schuster, A. Turrini, L. Zhang, *The quest for minimal quotients for probabilistic automata*, Lecture Notes in Computer Science, **7795** (2013), 16–31.

[36] J.M. Fourneau, *Collaboration of discrete-time Markov chains: tensor and product form*, Performance Evaluation, **67** (2010), 779–796.

[37] S. Gilmore, J. Hillston, L. Kloul, M. Ribaudo, *PEPA nets: a structured performance modelling formalism*, Performance Evaluation, **54** (2003), 79–104.

[38] R.J. van Glabbeek, S.A. Smolka, B. Steffen, *Reactive, generative, and stratified models of probabilistic processes*, Information and Computation, **121**:1 (1995), 59–80.

[39] N. Götz, U. Herzog, M. Rettelbach, *Multiprocessor and distributed system design: the integration of functional specification and performance analysis using stochastic process algebras*, Lecture Notes in Computer Science, **729** (1993), 121–146.

[40] M.C. Guenther, N.J. Dingle, J.T. Bradley, W.J. Knottenbelt, *Passage-time computation and aggregation strategies for large semi-Markov processes*, Performance Evaluation, **68** (2011), 221–236.

[41] T. Han, J.-P. Katoen, A. Mereacre, *Approximate parameter synthesis for probabilistic time-bounded reachability*, Proc. 29th IEEE Real-Time Systems Symposium (RTSS) 2008, New York, USA, 173–182, IEEE Computer Society Press, 2008.

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

[43] R.A. Hayden, J.T. Bradley, A. Clark, *Performance specification and evaluation with unified stochastic probes and fluid analysis*, IEEE Transactions on Software Engineering, **39**:1 (2013), 97–118.

[44] H. Hermanns, M. Rettelbach, *Syntax, semantics, equivalences and axioms for MTIPP*, Proc. 2nd 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

[45] J. Hillston, *The nature of synchronisation*, Proc. 2nd Int. Workshop on Process Algebra and Performance Modelling (PAPM) 1994 (U. Herzog, M. Rettelbach, eds.), Regensberg / Erlangen, Germany, July 1994, Arbeitsberichte des IMMD, **27**:4 (1994), 51–70. http://www.dcs.ed.ac.uk/pepa/synchronisation.pdf

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

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

[48] A. Horváth, M., Paolieri, L. Ridi, E. Vicario, *Transient analysis of non-Markovian models using stochastic state classes*, Performance Evaluation, **69**:7–8 (2012), 315–335.

[49] L. Jategaonkar, A.R. Meyer, *Deciding true concurrency equivalences on safe, finite nets*, Theoretical Computer Science, **154**:1 (1996), 107–143.

[50] C.-C. Jou, S.A. Smolka, *Equivalences, congruences and complete axiomatizations for probabilistic processes*, Lecture Notes in Computer Science, **458** (1990), 367–383.

[51] G. Kahn, *Natural semantics*, Proc. 4th Annual Symposium on Theoretical Aspects of Computer Science (STACS) 1987, 22–39, Springer, 1987.

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

[53] J.-P. Katoen P.R., D'Argenio, *General distributions in process algebra*, Lecture Notes in Computer Science, **2090** (2001), 375–429.

[54] J.-P. Katoen, E. Brinksma, D. Latella, R. Langerak, *Stochastic simulation of event structures*, Proc. 4th Int. Workshop on Process Algebra and Performance Modelling (PAPM) 1996 (M. Ribaudo, ed.), 21–40, CLUT Press, Torino, Italy, 1996. http://eprints.eemcs.utwente.nl/6487/ 01/263_KLLB96b.pdf

[55] V.G. Kulkarni, *Modeling and analysis of stochastic systems. Second edition*, Texts in Statistical Science, **84**, Chapman and Hall / CRC Press, 2009.

[56] R. Lanotte, A. Maggiolo-Schettini, A. Troina, *Parametric probabilistic transition systems for system design and analysis*, Formal Aspects of Computing, **19**:1 (2007), 93–109.

[57] K.G. Larsen, A. Skou, *Bisimulation through probabilistic testing*, Information and Computation, **94**:1 (1991), 1–28.

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

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

[60] 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.

[61] H. Macià, V. Valero, D. de Frutos, *sPBC: a Markovian extension of finite Petri box calculus*, Proc. 9th 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

[62] J. Markovski, P.R. D'Argenio, J.C.M. Baeten, E.P. de Vink, *Reconciling real and stochastic time: the need for probabilistic refinement*, Formal Aspects of Computing, **24**:4–6 (2012), 497–518.

[63] J. Markovski, E.P. de Vink, *Extending timed process algebra with discrete stochastic time*, Lecture Notes of Computer Science, **5140** (2008), 268–283.

[64] J. Markovski, E.P. de Vink, *Performance evaluation of distributed systems based on a discrete real- and stochastic-time process algebra*, Fundamenta Informaticae, **95**:1 (2009), 157–186.

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

[66] 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/

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

[68] R.A.J. Milner, J.G. Parrow, D.J. Walker, *A calculus of mobile processes (I and II)*, Information and Computation, **100**:1 (1992), 1–77.

[69] 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.

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

[71] U. Montanari, M. Pistore, D. Yankelevich, *Efficient minimization up to location equivalence*, Lecture Notes in Computer Science, **1058** (1996), 265–279.

[72] 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.

[73] R. Paige, R.E. Tarjan, *Three partition refinement algorithms*, SIAM Journal of Computing, **16**:6 (1987), 973–989.

[74] G.D. Plotkin, *A structural approach to operational semantics*, Technical Report, **DAIMI FN-19**, Computer Science Department, Aarhus University, Aarhus, Denmark, 1981.

[75] C. Priami, *Stochastic π-calculus with general distributions*, Proc. 4th Int. Workshop on Process Algebra and Performance Modelling (PAPM) 1996 (M. Ribaudo, ed.), 41–57, CLUT Press, Torino, Italy, 1996.

[76] C. Priami, *Language-based performance prediction for distributed and mobile systems*, Information and Computation, **175**:2 (2002), 119–145.

[77] M. Rettelbach, *Probabilistic branching in Markovian process algebras*, The Computer Journal, **38**:7 (1995), 590–599.

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

[79] 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

[80] 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.

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

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

[83] I.V. Tarasyuk, H. Macià, V. Valero, *Discrete time stochastic Petri box calculus with immediate multiactions dtsiPBC*, Proc. 6th Int. Workshop on Practical Applications of Stochastic

Modelling (PASM) 2012 and 11th 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.

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

[85] I.V. Tarasyuk, H. Macià, V. Valero, *Stochastic process reduction for performance evaluation in dtsiPBC*, Siberian Electronic Mathematical Reports, **12** (2015), 513–551.

[86] 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.

[87] R. Wimmer, S. Derisavi, H. Hermanns, *Symbolic partition refinement with automatic balancing of time and space*, Performance Evaluation, **67** (2010), 816–836.

[88] R. Zijal, G. Ciardo, G. Hommel, *Discrete deterministic and stochastic Petri nets*, Proc. 9th 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

[89] A. Zimmermann, J. Freiheit, G. Hommel, *Discrete time stochastic Petri nets for modeling and evaluation of real-time systems*, Proc. 9th 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 Valerievich Tarasyuk
A.P. Ershov Institute of Informatics Systems,
Siberian Branch of the Russian Academy of Sciences,
6, Acad. Lavrentiev pr.,
630090 Novosibirsk, Russian Federation
Email address: itar@iis.nsk.su

Hermenegilda Macià Soler, Valentín Valero Ruiz
High School of Informatics Engineering,
University of Castilla - La Mancha,
Avda. de España s/n,
02071 Albacete, Spain
Email address: Hermenegilda.Macia@uclm.es, Valentin.Valero@uclm.es