Stochastic equivalence for performance evaluation of concurrent systems in dtsiPBC

Igor V. Tarasyuk[a,1,*], Hermenegilda Macià[b,1], Valentín Valero[b,1]

[a]A.P. Ershov Institute of Informatics Systems, Siberian Branch of the Russian Academy of Sciences, 6, Acad. Lavrentiev ave., 630090 Novosibirsk, Russian Federation
[b]High School of Information Engineering, University of Castilla - La Mancha, Avda. de España s/n, 02071 Albacete, Spain

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, proposed by H. Macià, V. Valero and others within a continuous time domain. The step operational semantics is constructed via labeled probabilistic transition systems. The denotational semantics is defined on the basis of a subclass of labeled discrete time stochastic Petri nets with immediate transitions. A consistency of the both semantics is demonstrated. In order to evaluate performance, the corresponding semi-Markov chains and (reduced) discrete time Markov chains are analyzed. We define step stochastic bisimulation equivalence of expressions and prove that it can be applied to reduce their transition systems and underlying semi-Markov chains while preserving the functionality and performance characteristics. We explain how this equivalence may help to simplify performance analysis of the algebraic processes. In a case study, a method of modeling, performance evaluation and behaviour preserving reduction of concurrent systems is outlined and applied to the shared memory system.

Keywords: stochastic process algebra, Petri box calculus, discrete time, immediate multiaction, performance evaluation, stochastic equivalence
2000 MSC: 60J20, 68Q60, 68Q85, 68Q55

1. Introduction

Algebraic process calculi like CSP [37], ACP [5] and CCS [58] are a well-known formal model for the specification of computing systems and analysis of their behaviour. In such process algebras (PAs), systems and processes are specified by formulas, and verification of their properties is accomplished at a syntactic level via equivalences, axioms and inference rules. In the last decades, stochastic extensions of PAs were proposed such as MTIPP [34], PEPA [36] and EMPA [10, 9, 6]. Stochastic process algebras (SPAs) do not just specify actions which can occur as usual process algebras (qualitative features), but they associate some quantitative parameters with actions (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 advantages of both models, a semantics of algebraic formulas in terms of PNs is defined. Petri box

*Corresponding author. Tel.: +7 3833306360; fax: +7 3833323494.
Email addresses: `itar@iis.nsk.su` (Igor V. Tarasyuk), `Hermenegilda.Macia@uclm.es` (Hermenegilda Macià), `Valentin.Valero@uclm.es` (Valentín Valero)
URL: `http://itar.iis.nsk.su` (Igor V. Tarasyuk), `http://www.dsi.uclm.es/personal/ValentinValero` (Valentín Valero)

[1]Partially supported by Spanish government within the project “Modeling and formal analysis of contracts and Web services with distributed resources”, grant TIN2012-36812-C02-02.

calculus (PBC) [11, 13, 12] 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) [58]. 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 construction to specify infiniteness when the semantic interpretation in finite PNs is possible. PBC has a step operational semantics in terms of labeled transition systems. 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 see [11, 12]. In the last years, several extensions of PBC with a deterministic, a nondeterministic or a stochastic model of time were presented.

1.2. Time extensions of Petri box calculus

To specify systems with time constraints, such as real time systems, deterministic (fixed) or nondeterministic (interval) time delays are used. A deterministic time model was considered in timed Petri box calculus (TPBC) [54], whereas a nondeterministic one was accommodated in time Petri box calculus (tPBC) [43] and in arc time Petri box calculus (atPBC) [63]. In tPBC each *action* has a time interval associated (the earliest and the latest firing time), and a step operational semantics is defined. The denotational semantics is then defined in terms of a subclass of labeled time PNs (LtPNs), based on tPNs [57], and called time Petri boxes (ct-boxes). In contrast to tPBC, multiactions of TPBC are not instantaneous, but have time durations. For the latter model a step operational semantics is also considered, and a denotational semantics, using a subclass of labeled timed PNs (LTPNs), based on TPNs [66], and called timed Petri boxes (T-boxes). In atPBC multiactions are associated with time delay intervals, and a step operational semantics is defined. The denotational semantics is defined on a subclass of labeled arc time PNs (atPNs), where time restrictions are associated with the arcs, called arc time Petri boxes (at-boxes).

1.3. Stochastic extensions of Petri box calculus

The set of states for the systems with deterministic or nondeterministic delays often differs drastically from that for the timeless systems, hence, the analysis results for untimed systems may be not valid for the time ones. To solve this problem, stochastic delays are considered, which are the random variables with a (discrete or continuous) probability distribution. If the random variables governing delays have an infinite support then the corresponding SPA can exhibit all the same behaviour as its underlying untimed PA. A stochastic extension of PBC, called stochastic Petri box calculus (sPBC), was proposed in [50]. In sPBC, multiactions have stochastic delays that follow negative exponential distribution. Each multiaction is instantaneous and equipped with a rate that is a parameter of the corresponding exponential distribution. The execution of a multiaction is possible only after the corresponding stochastic time delay. Only a finite part of PBC was initially used for the stochastic enrichment, i.e. in its former version sPBC has neither refinement nor recursion nor iteration operations. The calculus has an interleaving operational semantics 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 (LCTSPNs), based on CTSPNs [55, 3], and called stochastic Petri boxes (s-boxes). In [47], 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 [48], 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 further with immediate multiactions in [49]. A denotational semantics of such an sPBC extension (we call it generalized sPBC or gsPBC) was defined via a subclass of labeled generalized SPNs (LGSPNs), based on GSPNs [55, 3, 4], and called generalized stochastic Petri boxes (gs-boxes). The performance analysis in gsPBC is based on the underlying semi-Markov chains (SMCs).

PBC has a step operational semantics, whereas sPBC has interleaving one, 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.

In [70], a discrete time stochastic extension dtsPBC of finite PBC was presented. A step operational semantics of dtsPBC was constructed via labeled probabilistic transition systems. Its denotational semantics was defined in terms of a subclass of labeled discrete time stochastic PNs (LDTSPNs), based on DTSPNs [60, 61], 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 [69, 71], 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 delays 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 [72], we presented the extension dtsiPBC of the latter calculus with immediate multiactions. Immediate multiactions increase the specification capability: they can model logical conditions, instant 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 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.

1.4. 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 were well explored in the literature. The most well-known and widely used one is bisimulation. Standardly, 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 sequential nondeterminism. 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 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 are called Markovian equivalences, which take rate (the parameter of exponential distribution that governs time delays) instead of probability. Note that the probabilistic equivalences can be seen as discrete time analogues of the Markovian ones, since the latter are defined as the continuous time equivalences.

Interleaving probabilistic weak trace equivalence was introduced in [22] on labeled probabilistic transition systems. Interleaving probabilistic strong bisimulation equivalence was proposed in [46] on the same model. Interleaving Markovian strong bisimulation equivalence was constructed in [34] for MTIPP, in [36] for PEPA and in [10, 9, 6] for EMPA. Interleaving probabilistic equivalences were defined for probabilistic processes in [39, 28]. Some variants of interleaving Markovian weak bisimulation equivalence were considered in [18] on Markovian process algebras, in [19] on labeled CTSPNs and in [20] on GSPNs. In [7], a comparison of interleaving Markovian trace, test and bisimulation equivalences was carried out on sequential and concurrent Markovian process calculi. Nevertheless, no appropriate equivalence notion was defined for concurrent SPAs.

1.5. Our contributions

In this paper, 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. LDTSIPNs possess some features of discrete time deterministic and stochastic PNs (DTDSPNs) [76] and discrete deterministic and stochastic PNs (DDSPNs) [75], but in LDTSIPNs simultaneous transition firings are possible while in DTDSPNs and DDSPNs only firings of single transitions are allowed. A consistency of both semantics is demonstrated. The corresponding stochastic process, the underlying semi-Markov chain (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 discrete time Markov chain (DTMC) and its reduction (RDTMC) by eliminating vanishing states. 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 behaviour. We prove that the mentioned equivalence guarantees identity of the stationary behaviour and the residence time properties. 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, what 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 with the calculus and analyze their performance, as well as in which way to reduce the systems while preserving their performance indices and making simpler the performance evaluation. At last, we consider differences and similarities between dtsiPBC and other SPAs to determine the advantages of our calculus.

The first results on this subject can be found in [72]. What concerns differences from our previous paper about dtsiPBC [73], the present text is much more detailed and many new results have been added. In particular, all the used notions (such as numbering and enumeration function) are formally defined, enhanced action rules (some preconditions are excluded) are proposed, alternative performance analysis methods (based on the DTMCs and RDTMCs) are developed, compact illustrative examples (of performance evaluation) are presented, step stochastic bisimulation equivalence is introduced and checked for stationary behaviour preservation, reduction of the transition systems and SMCs by the equivalence and the resulting simplification of performance evaluation are considered, generalized variant and reductions of the shared memory system are constructed, strong points of dtsiPBC with respect to other SPAs are detected.

If we compare dtsiPBC with the classical SPAs MTIPP, PEPA and EMPA, the first main difference between them comes from PBC, since dtsiPBC is based on this calculus: all algebraic operations and a notion of multiaction are inherited from PBC. The second main difference is discrete probabilities of activities induced by the discrete time approach, whereas action rates are used in the standard SPAs with continuous time. As a consequence, dtsiPBC has a non-interleaving step operational semantics. This is in contrast to the classical SPAs, where concurrency is modeled by interleaving because of the continuous probability distributions of action delays and the race condition applied when several actions can be executed in a state. The third main difference is immediate multiactions. The salient point of dtsiPBC is a combination of immediate multiactions, discrete stochastic time and step semantics in an SPA. Thus, the main contributions of the paper are the following.

- Powerful and expressive discrete time SPA with immediate activities dtsiPBC in its final form.
- Step operational semantics of dtsiPBC in terms of labeled probabilistic transition systems.
- Petri net denotational semantics of dtsiPBC based on discrete time stochastic and immediate Petri nets.
- Performance analysis via underlying semi-Markov chains and (reduced) discrete time Markov chains.
- Stochastic equivalence used for behaviour-preserving reduction that simplifies the performance evaluation.
- Extended case study illustrating how to apply the obtained theoretical results in practice.

1.6. Structure of the paper

The paper is organized as follows. In Section 2, the syntax of the extended 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 defined 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, the introduced equivalence is applied to the stationary behaviour comparison to verify the performance preservation. In Section 9, a shared memory system is presented as a case study. The difference between dtsiPBC and other well-known or similar SPAs is considered in Section 10. The advantages of dtsiPBC are explained in Section 11. Finally, 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 2.1. Let X be a set. A finite *multiset (bag)* M over X is a mapping $M : X \to \mathbb{N}$ such that $|\{x \in X \mid M(x) > 0\}| < \infty$, i.e. it can contain a finite number of elements only.

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

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

A *stochastic multiaction* is a pair (α, ρ), where $\alpha \in \mathcal{L}$ and $\rho \in (0; 1)$ is the *probability* of the multiaction α. This probability is interpreted as that of independent execution of the stochastic multiaction at the next discrete time moment. Such probabilities are used to calculate those to execute (possibly empty) sets of stochastic multiactions after one time unit delay. The probabilities of stochastic multiactions are required not to be equal to 1 to avoid extra model complexity due to assigning with them weights needed to make a choice when several stochastic multiactions with probability 1 can be executed from a state. In this case, some problems appear with conflicts resolving. See [60, 61] for the discussion on SPNs. This decision also allows us to avoid technical difficulties related to conditioning events with probability 0. Another reason is that not allowing probability 1 for stochastic multiactions excludes a potential source of periodicity (hence, non-ergodicity) in the underlying SMCs of the algebraic expressions. On the other hand, there is no sense to allow zero probabilities of multiactions, since they would never be performed in this case. Let $\mathcal{SL}$ be the set of *all stochastic multiactions*.

An *immediate multiaction* is a pair (α, l), where $\alpha \in \mathcal{L}$ and $l \in \mathbb{N}_{\geq 1} = \{1, 2, \ldots\}$ is the non-zero *weight* of the multiaction α. This weight is interpreted as a measure of importance (urgency, interest) or a bonus reward associated with execution of the immediate multiaction at the current discrete time moment. Such weights are used to calculate the probabilities to execute sets of immediate multiactions instantly. Immediate multiactions have a priority over stochastic ones. One can assume that all immediate multiactions have priority 1, whereas all stochastic ones have priority 0. This means that in a state where both kinds of multiactions can occur, immediate multiactions always occur before stochastic ones. Stochastic and immediate multiactions cannot be executed 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. It is easy to differentiate between probabilities and weights, hence, between stochastic and immediate multiactions, since the probabilities of stochastic multiactions belong to the interval $(0; 1)$, and the weights of immediate multiactions are non-zero (positive) natural numbers from $\mathbb{N}_{\geq 1}$. An *activity* is a stochastic or an immediate multiaction. Let $\mathcal{SIL} = \mathcal{SL} \cup \mathcal{IL}$ be the set of *all activities*. The *alphabet* of a multiset of activities $\Upsilon \in \mathbb{N}^{\mathcal{SIL}}_{fin}$ 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 $\Omega(\alpha, \kappa) = \kappa$. The *multiaction part* of a multiset of activities $\Upsilon \in \mathbb{N}^{\mathcal{SIL}}_{fin}$ is defined as $\mathcal{L}(\Upsilon) = \sum_{(\alpha,\kappa) \in \Upsilon} \alpha$.

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

Sequential execution and choice have a standard interpretation, like in other process algebras, but parallelism does not include synchronization, unlike the corresponding operation in CCS [58].

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

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 $\alpha \oplus_a \beta = \gamma$, where

$$\gamma(x) = \begin{cases} \alpha(x) + \beta(x) - 1, & x = a \text{ or } x = \hat{a}; \\ \alpha(x) + \beta(x), & \text{otherwise.} \end{cases}$$

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

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

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

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

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

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

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

To avoid technical difficulties with the iteration operator, we should not allow any concurrency at the highest level of the second argument of iteration. This is not a severe restriction though, since we can always prefix parallel expressions by an activity with the empty multiaction part. Later on, in Example 4.2, we shall demonstrate 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. See also [12] for discussion on this subject.

Definition 2.3. Let $(\alpha, \kappa) \in \mathcal{SIL}$ and $a \in Act$. A *regular static expression* of dtsiPBC is defined as

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

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

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

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

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

Note that if the underlying static expression of a dynamic one is not regular, the corresponding LDTSIPN can be non-safe (though, it is 2-bounded in the worst case [12]).

Definition 2.5. A dynamic expression is *regular* if its underlying static expression is regular.

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

3. Operational semantics

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

3.1. Inaction rules

The inaction rules for dynamic expressions describe their structural transformations which do not change the states of the specified processes. The goal of these syntactic transformations is to obtain the well-structured terminal 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 step or any transition firing in the corresponding LDTSIPN, hence, its current marking remains unchanged.

Thus, an application of every inaction rule does not require any discrete time delay, i.e. the dynamic expression transformation described by the rule is accomplished instantly.

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

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

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

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

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

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

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

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

3.2. Action and empty loop rules

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

The empty loop rule $G \stackrel{\emptyset}{\rightarrow} G$ with a pre-condition (rule **El** in Table 3) describes dynamic expression transformations due to execution of the empty multiset of activities at a discrete time step. The rule reflects a non-zero probability to stay in the current state at the next time moment, which is an essential feature of discrete time stochastic processes. As we shall see, the application of the empty loop rule to a dynamic expression leads to a discrete time step in the corresponding LDTSIPN at which no transitions fire and the current marking is not changed. This is a new rule that has no prototype among inaction rules of PBC, since it represents a time delay, but no notion of time exists in PBC. The PBC rule $G \stackrel{\emptyset}{\rightarrow} G$ from [13, 12] in our setting would correspond to the rule $G \Rightarrow G$ describing the stay in the current state when no time elapses. Since we do not need the latter rule to transform dynamic expressions into operative ones and it can even destroy the definition of operative expressions, we do not introduce it in dtsiPBC.

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

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

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

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

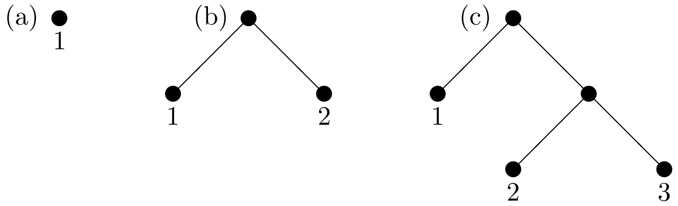

Figure 1: The binary trees encoded with the numberings 1, (1)(2) and (1)((2)(3)).

Example 3.1. *The numbering* 1 *encodes the binary tree depicted in Figure 1(a) with the root labeled by* 1*. The numbering* (1)(2) *corresponds to the binary tree depicted 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 depicted 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 be able to identify the mentioned instances.

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

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

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

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

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

Definition 3.4. 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 $(\alpha, \kappa) \in \mathcal{SIL},\ E, F \in RegStatExpr,\ G, H \in OpRegDynExpr$ and $a \in Act$.

1. If $final(G)$ then $Can(G) = \emptyset$.
2. If $G = \overline{(\alpha, \kappa)}$ then $Can(G) = \{\{(\alpha, \kappa)\}\}$.
3. If $\Upsilon \in Can(G)$ then $\Upsilon \in Can(G \circ E),\ \Upsilon \in Can(E \circ G)\ (\circ \in \{;, []\}),\ \Upsilon \in Can(G\|H),\ \Upsilon \in Can(H\|G)$, $f(\Upsilon) \in Can(G[f]),\ \Upsilon \in Can(G \text{ rs } a)$ (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) $(\Upsilon + \{(\alpha \oplus_a \beta, \kappa + \lambda)\}) \setminus \{(\alpha, \kappa), (\beta, \lambda)\} \in Can(G \text{ sy } a)$, if $\kappa, \lambda \in \mathbb{N}_{\geq 1}$.

   When we synchronize the same 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 to avoid introducing redundant ones.

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

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

Let $G \in OpRegDynExpr$ and $Can(G) \neq \emptyset$. Obviously, if there are only stochastic (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. $\exists \Upsilon \in Can(G)\ \Upsilon \in \boldsymbol{N}_{fin}^{\mathcal{IL}} \setminus \{\emptyset\}$. In this case, no stochastic multiactions can be executed from G, even if $Can(G)$ contains non-empty sets of stochastic multiactions, since immediate multiactions have a priority over stochastic ones, and should be executed first.

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

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

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

Example 3.2. *Let* $G = (\overline{(\{a\},1)}[](\{b\},2))\|\overline{(\{c\},\frac{1}{2})}$ *and* $G' = ((\{a\},1)[]\overline{(\{b\},2)})\|\overline{(\{c\},\frac{1}{2})}$. *Then* $G \approx G'$, *but* $Can(G) = \{\{(\{a\},1)\},\{(\{c\},\frac{1}{2})\},\{(\{a\},1),(\{c\},\frac{1}{2})\}\}$, $Can(G') = \{\{(\{b\},2)\},\{(\{c\},\frac{1}{2})\},\{(\{b\},2),(\{c\},\frac{1}{2})\}\}$ *and* $Now(G) = \{\{(\{a\},1)\}\}$, $Now(G') = \{\{(\{b\},2)\}\}$. *Clearly, we have* $vanish(G)$ *and* $vanish(G')$. *The executions like that of* $\{(\{c\},\frac{1}{2})\}$ *(and all sets including it) from H and H′ 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 $H = \overline{(\{a\},1)}[](\{b\},\frac{1}{2})$ *and* $H' = (\{a\},1)[]\overline{(\{b\},\frac{1}{2})}$. *Then* $H \approx H'$, *but* $Can(H) = Now(H) = \{\{(\{a\},1)\}\}$ *and* $Can(H') = Now(H') = \{\{(\{b\},\frac{1}{2})\}\}$. *We have* $vanish(H)$, *but* $tang(H')$. *To make 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 between them are always resolved in favour of the former.*

Now, in Table 3, we define the action and empty loop rules. In this table, $(\alpha,\rho),(\beta,\chi) \in \mathcal{SL}$, $(\alpha,l),(\beta,m) \in \mathcal{IL}$ and $(\alpha,\kappa) \in \mathcal{SIL}$. Further, $E,F \in RegStatExpr$, $G,H \in OpRegDynExpr$, $\widetilde{G},\widetilde{H} \in RegDynExpr$ and $a \in Act$. Moreover, $\Gamma,\Delta \in \boldsymbol{N}_{fin}^{\mathcal{SL}} \setminus \{\emptyset\}$, $\Gamma' \in \boldsymbol{N}_{fin}^{\mathcal{SL}}$, $I,J \in \boldsymbol{N}_{fin}^{\mathcal{IL}} \setminus \{\emptyset\}$, $I' \in \boldsymbol{N}_{fin}^{\mathcal{IL}}$ and $\Upsilon \in \boldsymbol{N}_{fin}^{\mathcal{SIL}} \setminus \{\emptyset\}$. The first rule in the table is the empty loop rule **El**. The other rules are the action rules, describing transformations of dynamic expressions, which are built using particular algebraic operations. If we cannot merge a rule with stochastic multiactions and a rule with immediate multiactions for some operation then we get the coupled action rules. In such cases, 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 [49], 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 structurally equivalent to them operative dynamic expressions 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. On the other hand, **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.

Table 3: Action and empty loop rules.

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

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 [12]) 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 [35]. 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 [34] 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. While 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 [35]. 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 [17, 16] 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 [68], thus, we collect the rewards. 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 [8]. The rewards are summed during synchronized execution of immediate multiactions, since in this case all the synchronized activities can be seen as "operated". We prefer to collect more rewards, thus, the transitions providing greater rewards will have a preference and they will be executed with a greater probability. 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 significant progress in behaviour. Under behavioural progress we understand an advance in executing activities, which does not always imply a progress in time, as in the case when the activities are immediate multiactions. This aspect will be used later, while evaluating performance via analysis of the embedded discrete time Markov chains (EDTMCs) of expressions. Since every state change in EDTMC takes one unit of (local) time, greater advance in operation of the EDTMC allows one to calculate quicker performance indices.

Example 3.3. *In the following cases, the weights of immediate multiactions are interpreted as bonus rewards to be summed while synchronous or parallel execution of the immediate multiactions specifying instant probabilistic choice.*

- *A customer in a shop considers which products to purchase. He will get a bonus (pay points) k when he decides* $(\{a\}, k)$ *to buy the product A and, for the deciding* $(\{\hat{a}\}, l)$ *to buy the product B, he will have the bonus l. Thus, on every decision to buy both products A and B (first A, and next time B; or first B, then A; or on the decision* $\{(\{a\}, k), \{\hat{a}\}, l)\}$ *to buy A and B together, in one visit to the shop, i.e. in parallel; or on the decision* $(\emptyset, k + l)$ *to buy a kit with A and B, which corresponds to their synchronized composition), the customer will get a bonus* $k + l$*, this is a standard and well-accepted practice.*

- *A cook in a fast-food restaurant plans his everyday work (cooking a two-component dinner dish of vegetables and meat), that consists in the decision* $(\{a\}, k)$ *to perform work A (boil vegetables), for which he will get a payment k, and the decision* $(\{\hat{a}\}, l)$ *to perform work B (fry meat), with the payment l. The works A and B are independent, and they can be even done together, since there are several (at least, two) free rings on the electric cooker in the kitchen. Then, on every decision to perform both works A and B (first A, then B; or first B, then A; or on the decision* $\{(\{a\}, k), \{\hat{a}\}, l)\}$ *to perform A and B in parallel; or on the decision* $(\emptyset, k + l)$ *to do a work including both A and B, for example, to warm up a frozen combined (two-in-one) product (consisting of vegetables and meat), prepared by the cook ahead of time, which corresponds to the synchronized composition of works A and B), the cook will get a payment* $k + l$*, this is logical and fair.*

In the both situations above, more successful customer or cook spends less resources (strength, electricity, water, etc.) to get his bonus or payment $k+l$*. Thus, the preferred and encouraged way of doing (the ideal behaviour or work) consists in the parallel or the synchronized executing of actions. Since we prefer to collect more bonus rewards, clearly, the decisions providing more rewards must have a preference and should be executed with a greater probability.*

The standard approach while system modeling within dtsiPBC is to split the system operations into the probabilistic decision, specified by an immediate multiaction, and the time-consuming work followed, that is specified by one or more stochastic multiactions. It is more natural to interpret weights of immediate multiactions as bonus rewards, subsequently used to determine the decision probabilities, since probabilities of stochastic multiactions are intended to calculate the duration of work.

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

In Table 4, inaction rules, action rules (with stochastic or immediate multiactions) and empty loop rule are compared according to the three questions about 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

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

Table 4: Comparison of inaction, action and empty loop rules.

Rules	State change	Time progress	Activities execution
Inaction rules	$-$	$-$	$-$
Action rules (stochastic multiactions)	$\pm$	$+$	$+$
Action rules (immediate multiactions)	$\pm$	$-$	$+$
Empty loop rule	$-$	$+$	$-$

Definition 3.6. The *derivation set* of a dynamic expression G, denoted by $DR(G)$, is the minimal set such that

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

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

The set of *all sets of activities executable in s* is defined as $Exec(s) = \{\Upsilon \mid \exists H \in s,\ \exists\widetilde{H},\ H \xrightarrow{\Upsilon} \widetilde{H}\}$. Note that if $\Upsilon \in Exec(s) \setminus \{\emptyset\}$ then $\exists H \in s,\ \Upsilon \in Now(H)$. The state s is *tangible*, if $Exec(s) \subseteq \boldsymbol{N}_{fin}^{\mathcal{SL}}$. For tangible states we may have $Exec(s) = \{\emptyset\}$. Otherwise, the state s is *vanishing*, and in this case $Exec(s) \subseteq \boldsymbol{N}_{fin}^{\mathcal{IL}} \setminus \{\emptyset\}$. The set of *all tangible states from $DR(G)$* is denoted by $DR_T(G)$, and the set of *all vanishing states from $DR(G)$* is denoted by $DR_V(G)$. 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 $Exec(s)$ $\forall\Xi \subseteq \Upsilon,\ \Xi \neq \emptyset$ we have $\Xi \in Exec(s)$.

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

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

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

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

If $s \in DR_T(G)$ and $Exec(s) \neq \{\emptyset\}$ then $PF(\Upsilon, s)$ can be interpreted as a *joint* probability of independent events (in 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 $\mathsf{P}(X \cap Y) = \mathsf{P}(X|Y)\mathsf{P}(Y)$. The event X consists in the execution of Υ in s. The event Y consists in the non-execution in s of all the executable stochastic multiactions not belonging to Υ. Since the mentioned non-executions are obviously independent events, the probability of Y is a product of the probabilities of the non-executions: $\mathsf{P}(Y) = \prod_{\{\{(\beta,\chi)\}\in Exec(s)\mid(\beta,\chi)\notin\Upsilon\}}(1-\chi)$. The conditioning of X by Y makes the executions of the stochastic multiactions from Υ independent, since all of them can be executed in parallel in s by definition of $Exec(s)$. Hence, the probability to execute Υ *under condition* that no executable stochastic multiactions not belonging to Υ are executed in s is a product of probabilities of these stochastic multiactions: $\mathsf{P}(X|Y) = \prod_{(\alpha,\rho)\in\Upsilon} \rho$. Thus, the probability that Υ is executed *and* no executable stochastic multiactions not belonging to Υ are executed in s is the probability of X conditioned by Y multiplied by the probability of Y: $\mathsf{P}(X \cap Y) = \mathsf{P}(X|Y)\mathsf{P}(Y) = \prod_{(\alpha,\rho)\in\Upsilon} \rho \cdot \prod_{\{\{(\beta,\chi)\}\in Exec(s)|(\beta,\chi)\notin\Upsilon\}}(1-\chi)$. 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 we stay in s in this case. Note that for $s \in DR_T(G)$ 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 $s \in DR_V(G)$ 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 [68]. In addition, this means that concurrent execution of the immediate multiactions has more importance than that of every single one. The weights of immediate multiactions can also be interpreted as bonus rewards of transitions [8]. The rewards are summed when immediate multiactions are executed in parallel, because all of them “work” thereby. Since execution of immediate multiactions takes no time, we prefer to execute in a step as many parallel immediate multiactions as possible to get more progress in behaviour. This aspect will be used later, while evaluating performance on the basis of the EDTMCs of expressions. Note that this reasoning is the same as that used to define the probability of synchronized immediate multiactions in the rule **Sy2i**. Another reason is that our approach is analogous to the definition of the probability of conflicting immediate transitions in GSPNs [4]. 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. To get the analogy with GSPNs possessing interleaving semantics, we interpret the weights of immediate transitions of GSPNs as the cumulative weights of the sets of immediate multiactions of dtsiPBC.

Note that the definition of $PF(\Upsilon, s)$ (as well as the definitions of other probability functions which we shall present) 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

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

If $s \in DR_T(G)$ then $PT(\Upsilon, s)$ can be interpreted as the *conditional* probability to execute Υ in s calculated using the conditional probability formula in the form $\mathsf{P}(Z|W) = \frac{\mathsf{P}(Z\cap W)}{\mathsf{P}(W)}$. The event Z consists in the exclusive execution of Υ in s, hence, $\mathsf{P}(Z) = PF(\Upsilon, s)$. The event W consists in the exclusive execution of any set (including the empty one) $\Xi \in Exec(s)$ in s. Thus, $W = \cup_j Z_j$, where $\forall j,\ Z_j$ are mutually exclusive events and $\exists i,\ Z = Z_i$ (in a probability sense, i.e. intersection of these events is the empty event). We have $\mathsf{P}(W) = \sum_j \mathsf{P}(Z_j) = \sum_{\Xi\in Exec(s)} PF(\Xi, s)$, because summation reflects the probability of the mutually exclusive event union. Since $Z \cap W = Z_i \cap (\cup_j Z_j) = Z_i = Z$, we have $\mathsf{P}(Z|W) = \frac{\mathsf{P}(Z)}{\mathsf{P}(W)} = \frac{PF(\Upsilon,s)}{\sum_{\Xi\in Exec(s)} PF(\Xi,s)}$. $PF(\Upsilon, s)$ can also 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 together, in parallel, i.e. $\emptyset, \{(\alpha, \rho)\}, \{(\beta, \chi)\} \in Exec(s)$, but $\{(\alpha, \rho), (\beta, \chi)\} \notin Exec(s)$. Note that for $s \in DR_T(G)$ we have $PT(\emptyset, s) \in (0; 1]$, hence, there is a non-zero probability to stay in the state s at the next time moment, and the residence time in s is at least 1 discrete time unit.

If $s \in DR_V(G)$ 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.

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

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

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

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

Since $PM(s,\tilde{s})$ is the probability to move from s to $\tilde{s}$ by executing any set of activities (including the empty one), we use summation in the definition. Note that $\forall s \in DR(G),\ \sum_{\{\tilde{s}\mid \exists H\in s,\ \exists \widetilde{H}\in\tilde{s},\ \exists\Upsilon,\ H\xrightarrow{\Upsilon}\widetilde{H}\}} PM(s,\tilde{s}) = \sum_{\{\tilde{s}\mid \exists H\in s,\ \exists \widetilde{H}\in\tilde{s},\ \exists\Upsilon,\ H\xrightarrow{\Upsilon}\widetilde{H}\}}$ $\sum_{\{\Upsilon\mid \exists H\in s,\ \exists \widetilde{H}\in\tilde{s},\ H\xrightarrow{\Upsilon}\widetilde{H}\}} PT(\Upsilon,s) = \sum_{\Upsilon\in Exec(s)} PT(\Upsilon,s) = 1$.

Example 3.4. *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 $DR_T(\overline{E}) = \{s_1,s_2\}$. 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 $PT(\emptyset,s_1) = PM(s_1,s_1) = \frac{(1-\rho)(1-\chi)}{1-\rho\chi}$. Further, $Exec(s_2) = \{\emptyset\}$, hence, $\sum_{\Xi\in Exec(s_2)} PF(\Xi,s_2) = PF(\emptyset,s_2) = 1$ and $PT(\emptyset,s_2) = PM(s_2,s_2) = \frac{1}{1} = 1$. At last, $PM(s_1,s_2) = PT(\{(\{a\},\rho)\},s_1) + PT(\{(\{a\},\chi)\},s_1) = \frac{\rho(1-\chi)}{1-\rho\chi} + \frac{\chi(1-\rho)}{1-\rho\chi} = \frac{\rho+\chi-2\rho\chi}{1-\rho\chi}$.*

Let $E' = (\{a\},l)[](\{a\},m)$, where $l,m \in \mathbb{N}_{\geq 1}$. $DR(\overline{E'})$ consists of the equivalence classes $s_1' = [\overline{E'}]_\approx$ and $s_2' = [\underline{E'}]_\approx$. We have $DR_T(\overline{E'}) = \{s_2'\}$ and $DR_V(\overline{E'}) = \{s_1'\}$. The execution probabilities are calculated as follows. Since $Exec(s_1') = \{\{(\{a\},l)\},\{(\{a\},m)\}\}$, we get $PF(\{(\{a\},l)\},s_1') = l$ and $PF(\{(\{a\},m)\},s_1') = m$. Then $\sum_{\Xi\in Exec(s_1')} PF(\Xi,s_1') = l+m$. Thus, $PT(\{(\{a\},l)\},s_1') = \frac{l}{l+m}$ and $PT(\{(\{a\},m)\},s_1') = \frac{m}{l+m}$. Further, $Exec(s_2') = \{\emptyset\}$, hence, $\sum_{\Xi\in Exec(s_2')} PF(\Xi,s_2') = PF(\emptyset,s_2') = 1$ and $PT(\emptyset,s_2') = PM(s_2',s_2') = \frac{1}{1} = 1$. At last, $PM(s_1',s_2') = PT(\{(\{a\},l)\},s_1') + PT(\{(\{a\},m)\},s_1') = \frac{l}{l+m} + \frac{m}{l+m} = 1$.

Definition 3.7. Let G be a dynamic expression. The *(labeled probabilistic) transition system* of G is a quadruple $TS(G) = (S_G, L_G, \mathcal{T}_G, s_G)$, where

- 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 [28]. 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 $s \xrightarrow{\Upsilon}_{\mathcal{P}} \tilde{s}$. It is interpreted as follows: the probability to change s to $\tilde{s}$ as a result of executing Υ is $\mathcal{P}$.

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

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

Definition 3.8. 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 $\beta : S_G \to S_{G'}$ is an *isomorphism* between $TS(G)$ and $TS(G')$, denoted by $\beta : TS(G) \simeq TS(G')$, if

1. β is a bijection such that $\beta(s_G) = s_{G'}$;
2. $\forall s, \tilde{s} \in S_G,\ \forall\Upsilon,\ s \xrightarrow{\Upsilon}_{\mathcal{P}} \tilde{s} \ \Leftrightarrow\ \beta(s) \xrightarrow{\Upsilon}_{\mathcal{P}} \beta(\tilde{s})$.

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

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

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

Example 3.5. *Consider the expression* $\mathsf{Stop} = (\{g\}, \frac{1}{2})$ rs g *specifying the special process that is only able to perform empty loops with probability* 1 *and never terminates. We could actually use any arbitrary action from* $\mathcal{A}$ *and any probability belonging to the interval* $(0;1)$ *in the definition of* Stop*. Note that* Stop *is analogous to the one used in the examples of [48]. Then, for* $\rho, \chi, \theta, \phi \in (0;1)$ *and* $l, m \in \mathbb{N}_{\geq 1}$*, let*

$$E = [(\{a\}, \rho) * ((\{b\}, \chi); (((\{c\}, l); (\{d\}, \theta))[]((\{e\}, m); (\{f\}, \phi)))) * \mathsf{Stop}],$$

where $\rho, \chi, \theta, \phi \in (0;1)$ *and* $l, m \in \mathbb{N}_{\geq 1}$.

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

$$\begin{aligned}
s_1 &= [[\overline{(\{a\}, \rho)} * ((\{b\}, \chi); (((\{c\}, l); (\{d\}, \theta))[]((\{e\}, m); (\{f\}, \phi)))) * \mathsf{Stop}]]_{\approx},\\
s_2 &= [[(\{a\}, \rho) * (\overline{(\{b\}, \chi)}; (((\{c\}, l); (\{d\}, \theta))[]((\{e\}, m); (\{f\}, \phi)))) * \mathsf{Stop}]]_{\approx},\\
s_3 &= [[(\{a\}, \rho) * ((\{b\}, \chi); \overline{(((\{c\}, l); (\{d\}, \theta))[]((\{e\}, m); (\{f\}, \phi)))}) * \mathsf{Stop}]]_{\approx},\\
s_4 &= [[(\{a\}, \rho) * ((\{b\}, \chi); (((\{c\}, l); \overline{(\{d\}, \theta)})[]((\{e\}, m); (\{f\}, \phi)))) * \mathsf{Stop}]]_{\approx},\\
s_5 &= [[(\{a\}, \rho) * ((\{b\}, \chi); (((\{c\}, l); (\{d\}, \theta))[]((\{e\}, m); \overline{(\{f\}, \phi)}))) * \mathsf{Stop}]]_{\approx}.
\end{aligned}$$

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

In Figure 2, the transition system $TS(\overline{E})$ *is presented. The tangible states are depicted in ovals and the vanishing ones are depicted in boxes. For simplicity of the graphical representation, the singleton sets of activities are written without braces.*

4. Denotational semantics

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

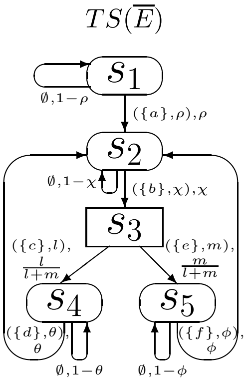

Figure 2: The transition system of $\overline{E}$ for $E = [(\{a\},\rho) * ((\{b\},\chi); (((\{c\},l); (\{d\},\theta))[]((\{e\},m); (\{f\},\phi)))) * \mathsf{Stop}]$.

4.1. Labeled DTSIPNs

Let us introduce a class of labeled discrete time stochastic and immediate Petri nets (LDTSIPNs), a subclass of DTSPNs [60, 61] (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) [76], as well as discrete deterministic and stochastic PNs (DDSPNs) [75]. DTDSPNs and DTSPNs are the extensions of DTSPNs with deterministic transitions (having fixed delay that can be zero), inhibitor arcs priorites and guards. In addition, while stochastic transitions of DTDSPNs, like those of DTSPNs, have geometrically distributed delays, stochastic transitions of DTSPNs have discrete time phase distributed delays. At the same time, LDTSIPNs are not subsumed by DTDSPNs or DTSPNs, since LDTSIPNs have a step semantics while DTDSPNs and DDSPNs have interleaving one. LDTSIPNs are somewhat similar to labeled weighted DTSPNs (LWDTSPNs) from [21], but in LWDTSPNs there are no immediate transitions, all (stochastic) transitions have weights, the transition probabilities may be equal to 1 and only maximal fireable subsets of the enabled transitions are fired.

First, we present a formal definition of LDTSIPNs.

Definition 4.1. A *labeled discrete time stochastic and immediate Petri net (LDTSIPN)* is a tuple $N = (P_N, T_N, W_N, \Omega_N, L_N, M_N)$, where

- P_N and $T_N = Ts_N \uplus Ti_N$ 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 providing the *weights of arcs* between places and transitions;
- $\Omega_N : T_N \to (0; 1) \cup (\mathbb{N}_{\geq 1})$ is the *transition probability and weight* function associating stochastic transitions with probabilities and immediate ones with weights;
- $L_N : T_N \to \mathcal{L}$ is the *transition labeling* function assigning multiactions to transitions;
- $M_N \in \mathbb{N}_{fin}^{P_N}$ 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. In the case 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 used to describe the stationary behaviour. The weights of arcs are depicted with them. The names of places and transitions are depicted near them when needed. If the names are omitted but used, it is supposed that the places and transitions are numbered from left to right and from top to down.

Now we consider the semantics of LDTSIPNs.

Let N be an LDTSIPN and $t \in T_N,\ U \in \mathbb{N}_{fin}^{T_N}$. The *precondition* ${}^\bullet t$ and the *postcondition* $t^\bullet$ of t are the multisets of places defined as $({}^\bullet t)(p) = W_N(p,t)$ and $(t^\bullet)(p) = W_N(t,p)$. The *precondition* ${}^\bullet U$ and the *postcondition* $U^\bullet$ of U are the multisets of places defined as ${}^\bullet U = \sum_{t\in U} {}^\bullet t$ 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 $M, \widetilde{M} \in \mathbb{N}_{fin}^{P_N}$.

Immediate transitions have a priority over stochastic ones, thus, immediate transitions always fire first, if they can. Suppose that all stochastic transitions have priority 0 and all immediate ones have priority 1. A transition $t \in T_N$ is *enabled* in M if ${}^\bullet t \subseteq M$ and one of the following holds:

1. $t \in Ti_N$ or
2. $\forall u \in T_N,\ {}^\bullet u \subseteq M \ \Rightarrow\ u \in Ts_N$.

In other words, a transition is enabled in 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 one, there exists no immediate transition with enough tokens in its input places. Let $Ena(M)$ be the set of *all transitions enabled in* M. By definition, it follows that $Ena(M) \subseteq Ti_N$ or $Ena(M) \subseteq Ts_N$. A set of transitions $U \subseteq Ena(M)$ is *enabled* in 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 concurrently 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

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

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

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

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

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

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

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

Note that for all markings of an LDTSIPN N, the sum of outgoing probabilities is equal to 1. More formally, $\forall M \in \mathbb{N}_{fin}^{P_N},\ \sum_{\{U\subseteq Ena(M) \mid {}^\bullet U\subseteq M\}} PT(U,M) = 1$. This obviously 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 $M \rightarrow \widetilde{M}$ 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 $\forall M \in \mathbb{N}_{fin}^{P_N},\ \sum_{\{\widetilde{M} \mid M \rightarrow \widetilde{M}\}} PM(M,\widetilde{M}) = \sum_{\{\widetilde{M} \mid M \rightarrow \widetilde{M}\}} \sum_{\{U \mid M \xrightarrow{U} \widetilde{M}\}} PT(U,M) = \sum_{\{U \subseteq Ena(M) \mid {}^\bullet U \subseteq M\}} PT(U,M) = 1$.

Definition 4.2. 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 $M \rightarrow \widetilde{M}$ then $\widetilde{M} \in RS(N)$.
- The *reachability graph* of N, denoted by $RG(N)$, is a directed labeled graph with the set of nodes $RS(N)$ and the arcs labeled with $(U,\mathcal{P})$ between nodes M and $\widetilde{M}$ iff $M \xrightarrow{U}_{\mathcal{P}} \widetilde{M}$.

The set of *all tangible markings from* $RS(N)$ is denoted by $RS_T(N)$, and the set of *all vanishing markings from* $RS(N)$ is denoted by $RS_V(N)$. Obviously, $RS(N) = RS_T(N) \uplus RS_V(N)$.

4.2. Algebra of dtsi-boxes

Now we introduce discrete time stochastic and immediate Petri boxes and the algebraic operations to define a net representation of dtsiPBC expressions.

Definition 4.3. A *discrete time stochastic and immediate Petri box (dtsi-box)* is a tuple $N = (P_N, T_N, W_N, \Lambda_N)$, where

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

Moreover, $\forall t \in T_N,\ {}^\bullet t \neq \emptyset \neq t^\bullet$. In addition, 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 condition holds: ${}^\circ N \neq \emptyset \neq N^\circ,\ {}^\bullet({}^\circ N) = \emptyset = (N^\circ)^\bullet$.

A dtsi-box is *plain* if $\forall t \in T_N,\ \Lambda_N(t) \in \mathcal{SIL}$, i.e. $\Lambda_N(t)$ is a constant relabeling that will be defined later. In case of the constant relabeling, the shorthand notation (by an activity) for $\Lambda_N(t)$ will be used. A *marked plain dtsi-box* is a pair (N, M_N), where N is a plain dtsi-box and $M_N \in \mathbb{N}_{fin}^{P_N}$ is its marking. We shall use the following notation: $\overline{N} = (N, {}^\circ N)$ and $\underline{N} = (N, N^\circ)$. Note that a marked plain dtsi-box $(P_N, T_N, W_N, \Lambda_N, M_N)$ could be interpreted as the LDTSIPN $(P_N, T_N, W_N, \Omega_N, L_N, M_N)$, where functions Ω_N and L_N are defined as follows: $\forall t \in T_N,\ \Omega_N(t) = \Omega(\Lambda_N(t))$ and $L_N(t) = \mathcal{L}(\Lambda_N(t))$. 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.

The structure of the plain dtsi-box corresponding to a static expression is constructed like in PBC [13, 12], 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

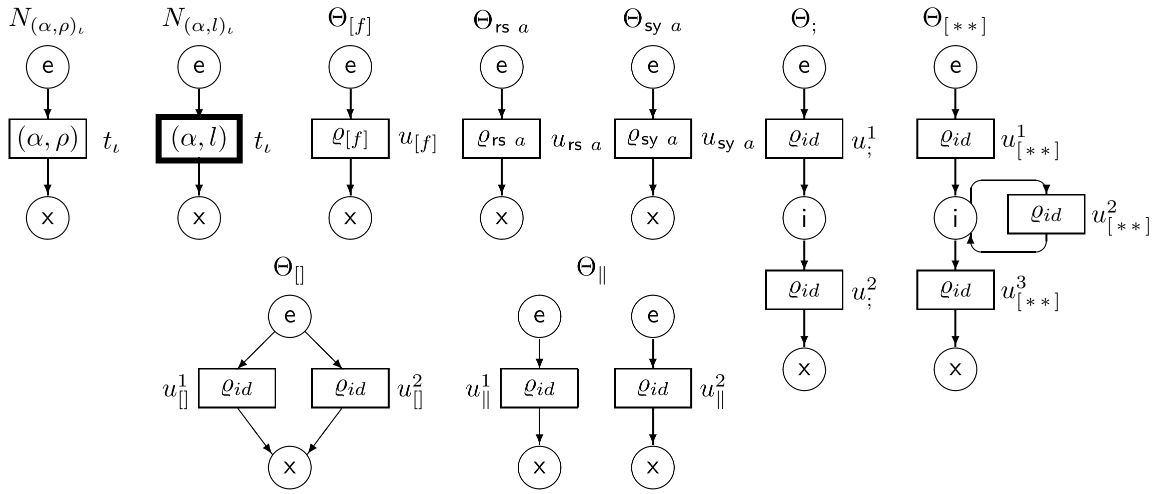

Figure 3: The plain and operator dtsi-boxes.

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 4.1, 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:

- $\varrho_{id} = \{(\{(\alpha,\kappa)\},(\alpha,\kappa)) \mid (\alpha,\kappa) \in \mathcal{SIL}\}$ is the *identity relabeling* keeping the interface as it is;
- $\varrho_{(\alpha,\kappa)} = \{(\emptyset,(\alpha,\kappa))\}$ is the *constant relabeling* that can be identified with $(\alpha,\kappa) \in \mathcal{SIL}$ itself;
- $\varrho_{[f]} = \{(\{(\alpha,\kappa)\},(f(\alpha),\kappa)) \mid (\alpha,\kappa) \in \mathcal{SIL}\}$;
- $\varrho_{\mathsf{rs}\ a} = \{(\{(\alpha,\kappa)\},(\alpha,\kappa)) \mid (\alpha,\kappa) \in \mathcal{SIL},\ a,\hat{a} \notin \alpha\}$;
- $\varrho_{\mathsf{sy}\ a}$ is the least relabeling relation 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 $\kappa,\lambda \in (0;1)$;
  - $(\Upsilon + \Xi,(\alpha \oplus_a \beta,\kappa + \lambda)) \in \varrho_{\mathsf{sy}\ a}$, if $\kappa,\lambda \in \mathbb{N}_{\geq 1}$.

The plain dtsi-boxes $N_{(\alpha,\rho)_\iota},\ N_{(\alpha,l)_\iota}$, where $\rho \in (0;1)$ and $l \in \mathbb{N}_{\geq 1}$, and operator dtsi-boxes are presented in Figure 3. 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 [12]. One of them provides us with a safe version with six transitions in the operator box, but there is also a simpler version, which has only three transitions. In general, in PBC, with the latter version we may generate 2-bounded nets, which only occurs when a parallel behavior appears at the highest level of the body of the iteration. Nevertheless, in our case, 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 the semantic function that associates a plain dtsi-box with every static expression of dtsiPBC, we introduce the *enumeration* function $Enu : T \to Num$, which associates the numberings with transitions of a plain dtsi-box $N = (P,T,W,\Lambda)$ in accordance with those of activities. In the case of synchronization, the function associates with the resulting new transition the concatenation of the parenthesized numberings of the transitions it comes from.

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

- $Box_{dtsi}((\alpha,\kappa)_\iota) = N_{(\alpha,\kappa)_\iota}$. Since a single transition t_ι corresponds to the activity $(\alpha,\kappa)_\iota \in \mathcal{SIL}$, their numberings coincide:

$$Enu(t_\iota) = \iota.$$

- $Box_{dtsi}(E \circ F) = \Theta_\circ(Box_{dtsi}(E), Box_{dtsi}(F)),\ \circ \in \{;, [], \|\}$. Since we do not introduce new transitions, we preserve the initial numbering:

$$Enu(t) = \begin{cases} Enu_E(t), & t \in T_E; \\ Enu_F(t), & t \in T_F. \end{cases}$$

- $Box_{dtsi}(E[f]) = \Theta_{[f]}(Box_{dtsi}(E))$. Since we only replace the labels of some multiactions by a bijection, we preserve the initial numbering:

$$Enu(t) = Enu_E(t),\ t \in T_E.$$

- $Box_{dtsi}(E \text{ rs } a) = \Theta_{\text{rs } a}(Box_{dtsi}(E))$. Since we remove all transitions labeled with multiactions containing a or $\hat{a}$, this does not change the numbering of the remaining transitions:

$$Enu(t) = Enu_E(t),\ t \in T_E,\ a, \hat{a} \notin \mathcal{L}(\Lambda_E(t)).$$

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

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

  According to the definition of $\varrho_{\text{sy } a}$, the synchronization is only possible when all the transitions in the set are stochastic or when all of them are immediate. If we synchronize the same set of transitions in different orders, we obtain several resulting transitions with the same label and probability or weight, but with the different numberings having the same content. Then, we only consider a single transition from the resulting ones in the plain dtsi-box to avoid introducing redundant transitions.

  For example, if the transitions t and u are generated by synchronizing v and w in different orders, we have $\Lambda(t) = (\alpha \oplus_a \beta, \kappa \cdot \lambda) = \Lambda(u)$ for stochastic transitions or $\Lambda(t) = (\alpha \oplus_a \beta, \kappa + \lambda) = \Lambda(u)$ for immediate ones, but $Enu(t) = (Enu_E(v))(Enu_E(w)) \neq (Enu_E(w))(Enu_E(v)) = Enu(u)$, whereas $Cont(Enu(t)) = Cont(Enu(v)) \cup Cont(Enu(w)) = Cont(Enu(u))$. Then only one transition t (or, symmetrically, u) will appear in $Box_{dtsi}(E \text{ sy } a)$.

- $Box_{dtsi}([E * F * K]) = \Theta_{[\,*\,*\,]}(Box_{dtsi}(E), Box_{dtsi}(F), Box_{dtsi}(K))$. Since we do not introduce new transitions, we preserve the initial numbering:

$$Enu(t) = \begin{cases} Enu_E(t), & t \in T_E; \\ Enu_F(t), & t \in T_F; \\ Enu_K(t), & t \in T_K. \end{cases}$$

Now we can formally define the denotational semantics as a homomorphism.

Definition 4.4. Let $(\alpha,\kappa) \in \mathcal{SIL},\ a \in Act$ and $E, F, K \in RegStatExpr$. The *denotational semantics* of dtsiPBC is a mapping Box_{dtsi} from $RegStatExpr$ into the domain of plain dtsi-boxes defined as follows:

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

2. $Box_{dtsi}(E \circ F) = \Theta_{\circ}(Box_{dtsi}(E), Box_{dtsi}(F)),\ \circ \in \{;\,, [], \|\}$;
3. $Box_{dtsi}(E[f]) = \Theta_{[f]}(Box_{dtsi}(E))$;
4. $Box_{dtsi}(E \circ a) = \Theta_{\circ a}(Box_{dtsi}(E)),\ \circ \in \{\mathsf{rs}, \mathsf{sy}\}$;
5. $Box_{dtsi}([E * F * K]) = \Theta_{[**]}(Box_{dtsi}(E), Box_{dtsi}(F), Box_{dtsi}(K))$.

The dtsi-boxes of dynamic expressions can be defined as well. For $E \in RegStatExpr$, let $Box_{dtsi}(\overline{E}) = \overline{Box_{dtsi}(E)}$ and $Box_{dtsi}(\underline{E}) = \underline{Box_{dtsi}(E)}$.

Note that 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 4.1. *For any static expression E, $Box_{dtsi}(\overline{E})$ is safe and clean.*

Proof. The structure of the net is obtained as in PBC, combining both refinement and relabeling. Consequently, the dtsi-boxes thus obtained will be safe and clean. □

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

Theorem 4.2. *For any static expression E,*

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

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

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. □

Example 4.1. *Let E be from Example 3.5. In Figure 4, the marked dtsi-box $N = Box_{dtsi}(\overline{E})$ and its reachability graph $RG(N)$ are presented. It is easy to see that $TS(\overline{E})$ and $RG(N)$ are isomorphic.*

The following example demonstrates that without the syntactic restriction on regularity of expressions the corresponding marked dtsi-boxes may be not safe.

Example 4.2. *Let $E = [((\{a\}, \frac{1}{2}) * ((\{b\}, \frac{1}{2}) \| (\{c\}, \frac{1}{2})) * (\{d\}, \frac{1}{2})]$. In Figure 5, the marked dtsi-box $N = Box_{dtsi}(\overline{E})$ and its reachability graph $RG(N)$ are presented. In the marking $(0, 1, 1, 2, 0, 0)$ there are 2 tokens in the place p_4. Symmetrically, in the marking $(0, 1, 1, 0, 2, 0)$ there are 2 tokens in the place p_5. Thus, allowing concurrency in the second argument of iteration in the expression $\overline{E}$ can lead to non-safeness of the corresponding marked dtsi-box N, though, it is 2-bounded in the worst case [12]. The origin of the problem is that N has a self-loop with two subnets which can function independently. Therefore, we have decided to consider regular expressions only, since the alternative, which is a safe version of the iteration operator with six arguments in the corresponding dtsi-box, like that from [12], is rather cumbersome and has too intricate Petri net interpretation. Our motivation was to keep the algebraic and Petri net specifications as simple as possible.*

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.

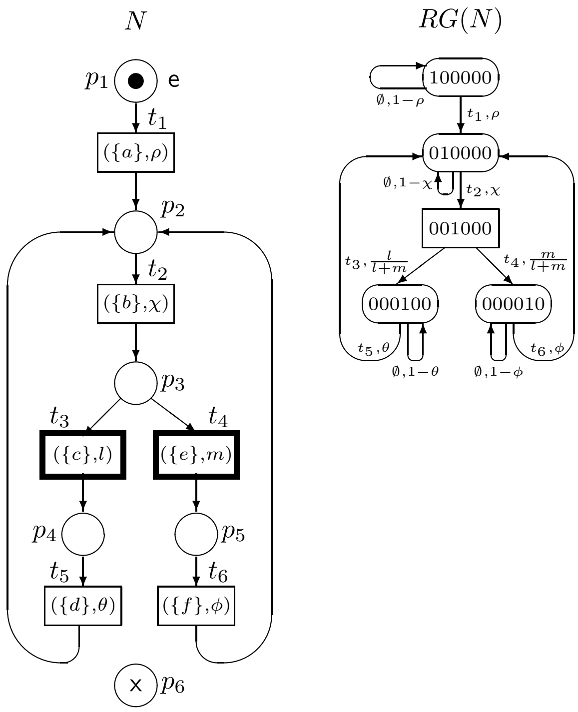

Figure 4: The marked dtsi-box $N = Box_{dtsi}(\overline{E})$ for $E = [(\{a\},\rho) * ((\{b\},\chi); (((\{c\}, l); (\{d\},\theta))[]((\{e\}, m); (\{f\},\phi)))) * \mathsf{Stop}]$ and its reachability graph.

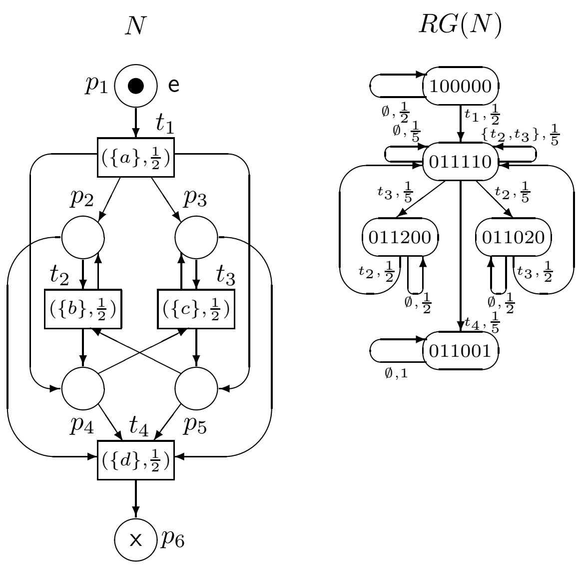

Figure 5: The marked dtsi-box $N = Box_{dtsi}(\overline{E})$ for $E = [(((\{a\}, \frac{1}{2}) * ((\{b\}, \frac{1}{2})\|(\{c\}, \frac{1}{2})) * (\{d\}, \frac{1}{2})]$ and its reachability graph.

5.1. Analysis of the underlying stochastic process

For a dynamic expression G, a discrete random variable is associated with every tangible state $s \in DR_T(G)$. The variable captures a residence time in the state. One can interpret staying in a state at the next discrete time moment as a failure and leaving it as a success of some trial series. It is easy to see that the random variables are geometrically distributed with the parameter $1 - PM(s,s)$, since the probability to stay in s for $k-1$ time moments and leave it at the moment $k \geq 1$ is $PM(s,s)^{k-1}(1-PM(s,s))$ (the residence time is k in this case, and this formula defines the probability mass function (PMF) of residence time in s). Hence, the probability distribution function (PDF) of residence time in s is $1 - PM(s,s)^{k-1}$ $(k \geq 1)$ (the probability that the residence time in s is less than k). The mean value formula for the geometrical distribution allows us to calculate the average sojourn time in s as $\frac{1}{1-PM(s,s)}$. 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

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

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

The *sojourn time variance in the state s* is

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

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

To evaluate performance of the system specified by a dynamic expression G, we should investigate the stochastic process associated with it. The process is the underlying semi-Markov chain (SMC) [68], denoted by $SMC(G)$, which can be analyzed by extracting from it the embedded (absorbing) discrete time Markov chain (EDTMC) corresponding to G, denoted by $EDTMC(G)$. The construction of the latter is analogous to that applied in the context of generalized stochastic PNs (GSPNs) in [55, 3, 4], and also in the framework of discrete time deterministic and stochastic PNs (DTDSPNs) in [76], as well as within discrete deterministic and stochastic PNs (DDSPNs) [75]. $EDTMC(G)$ 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. It is well-known that every SMC is fully described by the EDTMC and the state sojourn time distributions (the latter can be specified by the vector of PDFs of residence time in the states) [32].

Let G be a dynamic expression and $s, \tilde{s} \in DR(G)$. The transition system $TS(G)$ can have self-loops going from a state to itself which have a non-zero probability. Obviously, the current state remains unchanged in this case.
Let $s \to s$. The *probability to stay in s due to k* $(k \geq 1)$ *self-loops* is

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

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

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

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

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

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

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

We decided to consider self-loops followed only by a state-changing step just for convenience. Alternatively, we could take a state-changing step followed by self-loops or a state-changing step preceded and followed by self-loops. In all these three cases our sequence begins or/and ends with the loops which do not change states. At the same time, the overall probabilities of the evolutions can differ, since self-loops have positive probabilities. To avoid inconsistency of definitions and too complex description, we consider sequences ending with a state-changing step. It resembles in some sense a construction of branching bisimulation [27] taking self-loops instead of silent transitions.

Definition 5.1. Let G be a dynamic expression. The *embedded (absorbing) discrete time Markov chain (EDTMC)* of G, denoted by $EDTMC(G)$, has the state space $DR(G)$, the initial state $[G]_\approx$ and the transitions $s \twoheadrightarrow_{\mathcal{P}} \tilde{s}$, if $s \to \tilde{s}$ and $s \neq \tilde{s}$, where $\mathcal{P} = PM^*(s, \tilde{s})$.

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

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

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

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

The transient (k-step, $k \in \mathbb{N}$) PMF $\psi^*[k] = (\psi^*[k](s_1), \dots, \psi^*[k](s_n))$ 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

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

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

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

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

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

When $EDTMC(G)$ has a single steady state, we have $\psi^* = \lim_{k\to\infty} \psi^*[k]$.

The steady-state PMF for the underlying semi-Markov chain $SMC(G)$ is calculated via multiplication of every $\psi^*(s_i)$ $(1 \le i \le n)$ 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 $s \in DR_V(G)$ we have $SJ(s) = 0$.

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

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

Thus, to calculate φ, we apply 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 5.1. *Let E be from Example 3.5. In Figure 6, the underlying SMC $SMC(\overline{E})$ is presented. The average sojourn time in the states of the underlying SMC is written next to them in bold font.*

The average sojourn time vector of $\overline{E}$ 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 by dividing it by the sum of its components

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

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

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

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

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

Let G be a dynamic expression and $s, \tilde{s} \in DR(G)$, $S, \widetilde{S} \subseteq DR(G)$. The following standard *performance indices (measures)* can be calculated based on the steady-state PMF for $SMC(G)$ [62, 40].

- The *average recurrence (return) time in the state s* (the number of discrete time units required for this) is $\frac{1}{\varphi(s)}$.
- The *fraction of residence time in the state s* is $\varphi(s)$.
- The *fraction of residence time in the set of states S* or the *probability of the event determined by a condition that is true for all states from S* is $\sum_{s \in S} \varphi(s)$.

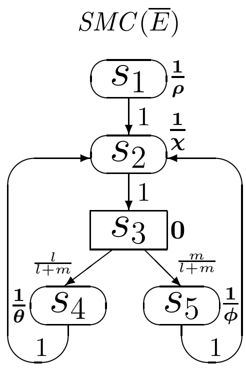

Figure 6: The underlying SMC of $\overline{E}$ for $E = [(\{a\},\rho) * ((\{b\},\chi); (((\{c\}, l); (\{d\},\theta))[]((\{e\}, m); (\{f\},\phi)))) * \mathsf{Stop}]$.

- The *relative fraction of residence time in the set of states S with respect to that in $\widetilde{S}$* is $\frac{\sum_{s\in S}\varphi(s)}{\sum_{\tilde{s}\in\widetilde{S}}\varphi(\tilde{s})}$.
- The *rate of leaving the state s* is $\frac{\varphi(s)}{SJ(s)}$.
- The *steady-state probability to perform a step with an activity (α,κ)* is $\sum_{s\in DR(G)}\varphi(s)\sum_{\{\Upsilon|(\alpha,\kappa)\in\Upsilon\}} PT(\Upsilon, s)$.
- The *probability of the event determined by a reward function r on the states* is $\sum_{s\in DR(G)}\varphi(s)r(s)$.

Let $N = (P_N, T_N, W_N, \Omega_N, L_N, M_N)$ be a LDTSIPN and $M, \widetilde{M} \in \mathbb{N}_{fin}^{P_N}$. 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.

As we have mentioned earlier, every marked plain dtsi-box could be interpreted as the LDTSIPN. Therefore, we can evaluate performance with the LDTSIPNs corresponding to dtsi-boxes and then transfer the results to the latter.

Let $\simeq$ denote isomorphism between SMCs that binds their initial states.

Proposition 5.1. *For any static expression E,*

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

Proof. By Theorem 4.2 and definitions of underlying SMCs for dynamic expressions and LDTSIPNs taking into account 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. □

Example 5.2. *Let E be from Example 3.5. In Figure 7, the underlying SMC $SMC(N)$ is presented. Clearly, $SMC(\overline{E})$ and $SMC(N)$ are isomorphic. Thus, both the transient and steady-state PMFs for $SMC(N)$ and $SMC(\overline{E})$ coincide.*

5.2. Alternative solution methods

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

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

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

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

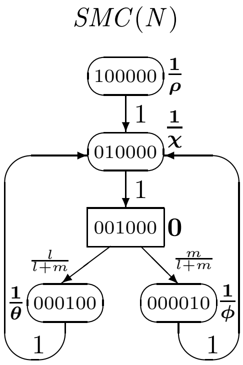

Figure 7: The underlying SMC of $N = Box_{dtsi}(\overline{E})$ for $E = [(\{a\},\rho) * ((\{b\},\chi); (((\{c\},l); (\{d\},\theta))[]((\{e\},m); (\{f\},\phi)))) * \mathsf{Stop}]$.

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

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

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

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

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

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

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

Proof. Let PSL be a vector with the elements

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Proof. Let $s \in DR_T(G)$. Remember that $\forall s \in DR_T(G)$, $SL(s) = SJ(s)$ and $\forall s \in DR_V(G)$, $SJ(s) = 0$. Then, by Theorem 5.1, we have $\frac{\psi(s)}{\sum_{\tilde{s} \in DR_T(G)} \psi(\tilde{s})} = \frac{\frac{\psi^*(s)SL(s)}{\sum_{\tilde{s} \in DR(G)} \psi^*(\tilde{s})SL(\tilde{s})}}{\sum_{\tilde{s} \in DR_T(G)} \left(\frac{\psi^*(\tilde{s})SL(\tilde{s})}{\sum_{\tilde{\tilde{s}} \in DR(G)} \psi^*(\tilde{\tilde{s}})SL(\tilde{\tilde{s}})} \right)} = \frac{\psi^*(s)SL(s)}{\sum_{\tilde{s} \in DR(G)} \psi^*(\tilde{s})SL(\tilde{s})} \cdot \frac{\sum_{\tilde{\tilde{s}} \in DR(G)} \psi^*(\tilde{\tilde{s}})SL(\tilde{\tilde{s}})}{\sum_{\tilde{s} \in DR_T(G)} \psi^*(\tilde{s})SL(\tilde{s})} = \frac{\psi^*(s)SL(s)}{\sum_{\tilde{s} \in DR_T(G)} \psi^*(\tilde{s})SL(\tilde{s})} = \frac{\psi^*(s)SJ(s)}{\sum_{\tilde{s} \in DR_T(G)} \psi^*(\tilde{s})SJ(\tilde{s})} = \frac{\psi^*(s)SJ(s)}{\sum_{\tilde{s} \in DR(G)} \psi^*(\tilde{s})SJ(\tilde{s})} = \varphi(s)$. ◻

Thus, to calculate φ, one can only apply normalization to some elements of ψ (corresponding to the tangible states), instead of abstracting from self-loops to get $\mathbf{P}^*$ and then ψ^*, followed by weighting by SJ and normalization. Hence, using $DTMC(G)$ instead of $EDTMC(G)$ allows one to avoid multistage analysis, but the payment for it is more time-consuming numerical and more complex analytical calculation of ψ with respect to ψ^*. The reason is that $DTMC(G)$ has self-loops, unlike $EDTMC(G)$, hence, the behaviour of $DTMC(G)$ stabilizes slower than that of $EDTMC(G)$ (if each of them has a single steady state) and $\mathbf{P}$ is more dense matrix than $\mathbf{P}^*$, since $\mathbf{P}$ may additionally have non-zero elements at the main diagonal. Nevertheless, Proposition 5.2 is very important, since the relationship between φ and ψ it discovers will be used in Proposition 5.3 to relate the steady-state PMFs for $SMC(G)$ and the reduced $DTMC(G)$, as well as in Section 8 to prove preservation of the stationary behaviour by a stochastic equivalence.

Example 5.3. *Let E be from Example 3.5. The TPM for $DTMC(\overline{E})$ is*

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

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

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

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

$$\sum_{\tilde{s}\in DR_T(\overline{E})} \psi(\tilde{s}) = \psi(s_1)+\psi(s_2)+\psi(s_4)+\psi(s_5) = \frac{\theta\phi(l+m)+\chi(\phi l+\theta m)}{\theta\phi(1+\chi)(l+m)+\chi(\phi l+\theta m)}.$$

By Proposition 5.2, we have

$$\begin{aligned}
\varphi(s_1) &= 0\cdot\tfrac{\theta\phi(1+\chi)(l+m)+\chi(\phi l+\theta m)}{\theta\phi(l+m)+\chi(\phi l+\theta m)} = 0,\\
\varphi(s_2) &= \tfrac{\theta\phi(l+m)}{\theta\phi(1+\chi)(l+m)+\chi(\phi l+\theta m)}\cdot\tfrac{\theta\phi(1+\chi)(l+m)+\chi(\phi l+\theta m)}{\theta\phi(l+m)+\chi(\phi l+\theta m)} = \tfrac{\theta\phi(l+m)}{\theta\phi(l+m)+\chi(\phi l+\theta m)},\\
\varphi(s_3) &= 0,\\
\varphi(s_4) &= \tfrac{\chi\phi l}{\theta\phi(1+\chi)(l+m)+\chi(\phi l+\theta m)}\cdot\tfrac{\theta\phi(1+\chi)(l+m)+\chi(\phi l+\theta m)}{\theta\phi(l+m)+\chi(\phi l+\theta m)} = \tfrac{\chi\phi l}{\theta\phi(l+m)+\chi(\phi l+\theta m)},\\
\varphi(s_5) &= \tfrac{\chi\theta m}{\theta\phi(1+\chi)(l+m)+\chi(\phi l+\theta m)}\cdot\tfrac{\theta\phi(1+\chi)(l+m)+\chi(\phi l+\theta m)}{\theta\phi(l+m)+\chi(\phi l+\theta m)} = \tfrac{\chi\theta m}{\theta\phi(l+m)+\chi(\phi l+\theta m)}.
\end{aligned}$$

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 5.1 with the use of ψ^* *and SJ.*

Let us now consider the method from [56, 3, 4] that eliminates vanishing states from the EMC (EDTMC, in our terminology) corresponding to the underlying SMC of every GSPN N. The TPM for the resulting *reduced* EDTMC (REDTMC) has smaller size than that for the EDTMC. The method demonstrates that there exists a transformation of the underlying SMC of N into a CTMC, whose states are the tangible markings of N. This CTMC, which is essentially the *reduced* underlying SMC (RSMC) of N, is constructed on the basis of the REDTMC. The CTMC can then be directly solved to get both the transient and the steady-state PMFs over the tangible markings of N.

This method can be easily transferred to dtsiPBC, hence, for every dynamic expression G, we can find a DTMC (since the sojourn time in the tangible states from $DR(G)$ is discrete and geometrically distributed) with the states from $DR_T(G)$, which can be directly solved to find the transient and the steady-state PMFs over the tangible states. We shall demonstrate that such a *reduced* DTMC (RDTMC) of G, denoted by $RDTMC(G)$, can be constructed from $DTMC(G)$, using the method analogous to that designed in [56, 3, 4] in the framework of GSPNs to transform EDTMC into REDTMC. Since the sojourn time in the vanishing states is zero, the state transitions of $RDTMC(G)$ occur in the moments of the global discrete time associated with $SMC(G)$, unlike those of $EDTMC(G)$, which happen only when the current state changes to some *different* one, irrespective of the global time. Therefore, in our case, we can skip the stages of constructing the REDTMC of G, denoted by $REDTMC(G)$, from $EDTMC(G)$, and recovering RSMC of G, denoted by $RSMC(G)$, (which is the sought-for DTMC) from $REDTMC(G)$, since we have $RSMC(G) = RDTMC(G)$.

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

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

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

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

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

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

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

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

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

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

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

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

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

Definition 5.3. Let G be a dynamic expression and $[G]_\approx \in DR_T(G)$. The *reduced discrete time Markov chain (RDTMC)* of G, denoted by $RDTMC(G)$, has the state space $DR_T(G)$, the initial state $[G]_\approx$ and the transitions $s \hookrightarrow_{\mathcal{P}} \tilde{s}$, where $\mathcal{P} = PM^\diamond(s, \tilde{s})$.

RDTMCs of static expressions can be defined as well. For $E \in RegStatExpr$, let $RDTMC(E) = RDTMC(\overline{E})$.

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

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

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

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

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

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

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

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

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

When $RDTMC(G)$ has a single steady state, we have $\psi^\diamond = \lim_{k\to\infty} \psi^\diamond[k]$.

The zero sojourn time in the vanishing states guarantees that the state transitions of $RDTMC(G)$ occur in the moments of the global discrete time associated with $SMC(G)$, i.e. every such state transition occurs after one time unit delay. Hence, the sojourn time in the tangible states is the same for $RDTMC(G)$ and $SMC(G)$. The state transition probabilities of $RDTMC(G)$ are those to move from tangible to tangible states in any number of steps, without traversal of the tangible states. Therefore, $RDTMC(G)$ and $SMC(G)$ have the same transient behaviour over the tangible states, thus, the transient analysis of $SMC(G)$ is possible to accomplish using $RDTMC(G)$.

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

Proposition 5.3. *Let G be a dynamic expression, φ be the steady-state PMF for $SMC(G)$ and $\psi^\diamond$ be the steady-state PMF for $RDTMC(G)$. Then $\forall s \in DR(G)$,*

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

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

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

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

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

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

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

Then the equation system for ψ is decomposed as follows:

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

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

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

We have

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

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

1. There exist *no loops among vanishing states*. We have $\exists l \in \mathbb{N},\ \forall k > l,\ \mathbf{C}^k = \mathbf{0}$ and $\mathbf{G} = \sum_{k=0}^{l} \mathbf{C}^l$.
   Let us right-multiply the first equation of the decomposed equation system for ψ by $\mathbf{G}$:

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

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

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

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

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

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

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

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

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

2. There exist *loops among vanishing states*. We have $\lim_{k\to\infty} \mathbf{C}^k = \mathbf{0}$ and $\mathbf{G} = (\mathbf{I} - \mathbf{C})^{-1}$.
Let us right-multiply the first equation of the decomposed equation system for ψ by $\mathbf{G}$:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

By Proposition 5.2, $\forall s \in DR_T(G),\ \varphi(s) = \frac{\psi(s)}{\sum_{\tilde{s}\in DR_T(G)} \psi(\tilde{s})}$.
Therefore, $\forall s \in DR_T(G)$,

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

Thus, to calculate φ, one can just take all the elements of $\psi^\diamond$ as the steady-state probabilities of the tangible states, instead of abstracting from self-loops to get $\mathbf{P}^*$ and then ψ^*, followed by weighting by *SJ* and normalization. Hence, using *RDTMC*(G) instead of *EDTMC*(G) allows one to avoid such a multistage analysis, but constructing $\mathbf{P}^\diamond$ also requires some efforts, including calculating matrix powers or inverse matrices. Note that *RDTMC*(G) has self-loops, unlike *EDTMC*(G), hence, the behaviour of *RDTMC*(G) may stabilize slower than that of *EDTMC*(G) (if each of them has a single steady state). On the other hand, $\mathbf{P}^\diamond$ is smaller and denser matrix than $\mathbf{P}^*$, since $\mathbf{P}^\diamond$ has additional non-zero elements not only at the main diagonal, but also many of them outside it. Therefore, mostly, we have less time-consuming numerical calculation of $\psi^\diamond$ with respect to ψ^*. At the same time, the complexity of the analytical calculation of $\psi^\diamond$ with respect to ψ^* depends on the model structure, such as the number of vanishing states and loops among them, but usually it is lower, since the matrix size reduction plays an important role in many cases.

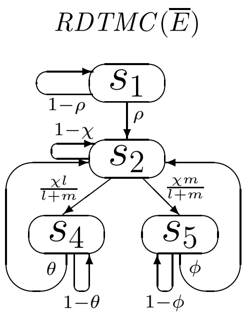

Figure 8: The reduced DTMC of $\overline{E}$ for $E = [(\{a\},\rho) * ((\{b\},\chi); (((\{c\}, l); (\{d\},\theta))[]((\{e\}, m); (\{f\},\phi)))) * \mathsf{Stop}]$.

Example 5.4. *Let E be from Example 3.5. Remember that $DR_T(\overline{E}) = \{s_1, s_2, s_4, s_5\}$ and $DR_V(\overline{E}) = \{s_3\}$. We reorder the states from $DR(\overline{E})$, by moving the vanishing states to the first positions, as follows: s_3, s_1, s_2, s_4, s_5.*

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

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

The result of the decomposing **P** *are the matrices*

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

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

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

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

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

In Figure 8, the reduced DTMC $RDTMC(\overline{E})$ is presented.
Then the steady-state PMF for $RDTMC(\overline{E})$ is

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

Note that $\psi^\diamond = (\psi^\diamond(s_1), \psi^\diamond(s_2), \psi^\diamond(s_4), \psi^\diamond(s_5))$. By Proposition 5.3, we have

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

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 5.1 with the use of ψ^* and SJ.

Note that our reduction of the underlying SMC by eliminating its vanishing states, resulting in the reduced DTMC, resembles the reduction from [51] by removing instantaneous states of stochastically discontinuous Markov reward chains. The latter are "limits" of continuous time Markov chains with state rewards and fast transitions when the rates (speeds) of these transitions tend to infinity, making them immediate. By analogy with that work, we would consider DTMCs extended with instantaneous states instead of SMCs with geometrically distributed or zero sojourn time in the states. However, within dtsiPBC, we have decided to take SMCs as the underlying stochastic process to be able in the perspective to consider not only geometrically distributed and zero residence time in the states, but arbitrary fixed time delays as well.

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 $\overline{E} \neq_{ts} \overline{E'}$, 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 $\{a\}$. Moreover, 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 3.4. Unlike $=_{ts}$, 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'.

Since the semantic equivalence $=_{ts}$ 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 behavior of a system. To define stochastic bisimulation equivalences, we have to 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 of their transition systems, $[G]_\approx$ and $[G']_\approx$, are to be related by a bisimulation having the following transfer property: two states are related if in each of them the same multisets of multiactions can occur, and the resulting states *belong to the same equivalence class*. In addition, the sums of probabilities for all such occurrences should be the same for both states.

Thus, we follow the approaches of [39, 46, 34, 36, 10, 7], but we implement step semantics instead of interleaving one considered in these papers. Recall also that we use the generative probabilistic transition systems, like in [39], in contrast to the reactive model, treated in [46], and we take transition probabilities instead of transition rates from [34, 36, 10, 7]. Thus, step stochastic bisimulation equivalence that we define further is (in the probabilistic sense) comparable only with interleaving probabilistic bisimulation one from [39], and our equivalence is obviously stronger.

In the definition below, we consider $\mathcal{L}(\Upsilon) \in \mathbb{N}_{fin}^{\mathcal{L}}$ for $\Upsilon \in \mathbb{N}_{fin}^{\mathcal{SIL}}$, 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 $\mathcal{H} \subseteq DR(G)$. Then, for any $s \in DR(G)$ and $A \in \mathbb{N}_{fin}^{\mathcal{L}}$, 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

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

We write $s \xrightarrow{A} \mathcal{H}$ if $\exists \mathcal{P},\ s \xrightarrow{A}_{\mathcal{P}} \mathcal{H}$. Further, we write $s \to_{\mathcal{P}} \mathcal{H}$ 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

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

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 (with respect to the stochastic bisimulation). Note that, for $G \neq G'$, transitions starting from the states of $DR(G)$ (or $DR(G')$) always lead to those from the same set, since $DR(G) \cap DR(G') = \emptyset$, and this allows us to "mix" the sets of states in the definition of stochastic bisimulation.

Definition 6.1. 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. $(s_1, s_2) \in \mathcal{R} \Rightarrow \forall \mathcal{H} \in (DR(G) \cup DR(G'))/_{\mathcal{R}},\ \forall A \in \mathbb{N}_{fin}^{\mathcal{L}}$,

$$s_1 \xrightarrow{A}_{\mathcal{P}} \mathcal{H} \Leftrightarrow s_2 \xrightarrow{A}_{\mathcal{P}} \mathcal{H}.$$

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

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 6.1. *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 fact, 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. □

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

Proposition 6.2. *Let G and G' be dynamic expressions and $G \underline{\leftrightarrow}_{ss} G'$. Then $\mathcal{R}_{ss}(G, G')$ is the largest step stochastic bisimulation between G and G'.*

Proof. See Appendix A.1. □

The algorithm for determining bisimulation of transition systems from [64] can be adapted for our framework. This algorithm has time complexity $O(m \log n)$, where n is the number of states and m is the number of transitions.

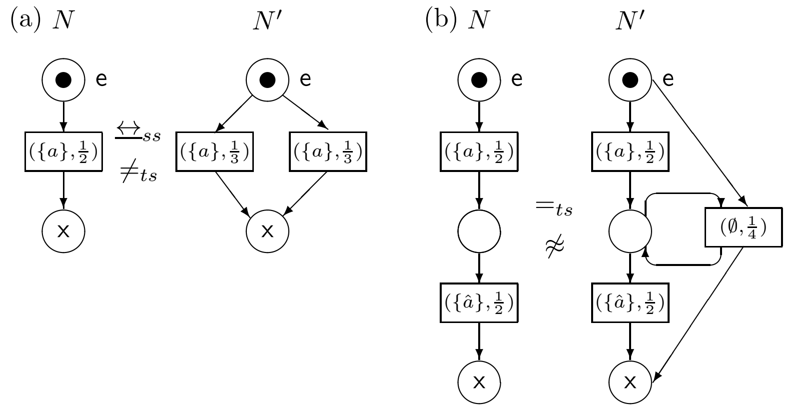

Figure 9: Dtsi-boxes of the dynamic expressions from equivalence examples of Theorem 6.1.

6.2. Interrelations of the stochastic equivalences

Now we compare the discrimination power of the stochastic equivalences.

Theorem 6.1. *For dynamic expressions G and G' the following* strict *implications hold:*

$$G \approx G' \;\Rightarrow\; G =_{ts} G' \;\Rightarrow\; G\underline{\leftrightarrow}_{ss}G'.$$

Proof. Let us check the validity of the implications.

- The implication $=_{ts}\Rightarrow \underline{\leftrightarrow}_{ss}$ is proved as follows. Let $\beta : G =_{ts} G'$. Then it is easy to see that $\mathcal{R} : G\underline{\leftrightarrow}_{ss}G'$, where $\mathcal{R} = \{(s, \beta(s)) \mid s \in DR(G)\}$.
- The implication $\approx\Rightarrow=_{ts}$ 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, i.e. the reverse ones do not work, by the following counterexamples.

(a) Let $E = (\{a\}, \frac{1}{2})$ and $E' = (\{a\}, \frac{1}{3})_1[](\{a\}, \frac{1}{3})_2$. Then $\overline{E}\underline{\leftrightarrow}_{ss}\overline{E'}$, but $\overline{E} \neq_{ts} \overline{E'}$, 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 $\overline{E} =_{ts} \overline{E'}$, but $\overline{E} \not\approx \overline{E'}$, since $\overline{E}$ and $\overline{E'}$ cannot be reached from each other by applying inaction rules. □

Example 6.1. *In Figure 9, the marked dtsi-boxes corresponding to the dynamic expressions from equivalence examples of Theorem 6.1 are presented, i.e. $N = Box_{dtsi}(\overline{E})$ and $N' = Box_{dtsi}(\overline{E'})$ for each picture (a)–(b).*

7. Reduction modulo equivalences

The equivalences which we proposed 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 behaviour 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 $\mathcal{R} : G\underline{\leftrightarrow}_{ss}G$, let $\mathcal{K} \in DR(G)/_{\mathcal{R}}$ and $s_1, s_2 \in \mathcal{K}$. We have $\forall\widetilde{\mathcal{K}} \in DR(G)/_{\mathcal{R}},\ \forall A \in \mathbb{N}^{\mathcal{L}}_{fin}$,

$s_1 \xrightarrow{A}_{\mathcal{P}} \widetilde{\mathcal{K}} \Leftrightarrow s_2 \xrightarrow{A}_{\mathcal{P}} \widetilde{\mathcal{K}}$. 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 $\mathcal{K} \rightarrow \widetilde{\mathcal{K}}$ if $\exists A,\ \mathcal{K} \xrightarrow{A} \widetilde{\mathcal{K}}$. The similar arguments allow us to write $\mathcal{K} \rightarrow_{\mathcal{P}} \widetilde{\mathcal{K}}$, where $\mathcal{P} = PM(\mathcal{K}, \widetilde{\mathcal{K}}) = PM(s_1, \widetilde{\mathcal{K}}) = PM(s_2, \widetilde{\mathcal{K}})$.

By Proposition 6.1, $\mathcal{R} \subseteq (DR_T(G))^2 \uplus (DR_V(G))^2$. Hence, $\forall \mathcal{K} \in DR(G)/_{\mathcal{R}}$, all states from $\mathcal{K}$ are tangible, when $\mathcal{K} \in DR_T(G)/_{\mathcal{R}}$, or all of them are vanishing, when $\mathcal{K} \in DR_V(G)/_{\mathcal{R}}$.

The *average sojourn time in the equivalence class (with respect to $\mathcal{R}$) of states $\mathcal{K}$* is

$$SJ_{\mathcal{R}}(\mathcal{K}) = \begin{cases} \frac{1}{1-PM(\mathcal{K},\mathcal{K})}, & \mathcal{K} \in DR_T(G)/_{\mathcal{R}}; \\ 0, & \mathcal{K} \in DR_V(G)/_{\mathcal{R}}. \end{cases}$$

The *average sojourn time vector for the equivalence classes (with respect to $\mathcal{R}$) of states* of G, denoted by $SJ_{\mathcal{R}}$, has the elements $SJ_{\mathcal{R}}(\mathcal{K}),\ \mathcal{K} \in DR(G)/_{\mathcal{R}}$.

The *sojourn time variance in the equivalence class (with respect to $\mathcal{R}$) of states $\mathcal{K}$* is

$$VAR_{\mathcal{R}}(\mathcal{K}) = \begin{cases} \frac{PM(\mathcal{K},\mathcal{K})}{(1-PM(\mathcal{K},\mathcal{K}))^2}, & \mathcal{K} \in DR_T(G)/_{\mathcal{R}}; \\ 0, & \mathcal{K} \in DR_V(G)/_{\mathcal{R}}. \end{cases}$$

The *sojourn time variance vector for the equivalence classes (with respect to $\mathcal{R}$) of states* of G, denoted by $VAR_{\mathcal{R}}$, has the elements $VAR_{\mathcal{R}}(\mathcal{K}),\ \mathcal{K} \in DR(G)/_{\mathcal{R}}$.

Let $\mathcal{R}_{ss}(G) = \bigcup\{\mathcal{R} \mid \mathcal{R} : G \underline{\leftrightarrow}_{ss} G\}$ be the *union of all step stochastic autobisimulations* on G. By Proposition 6.2, $\mathcal{R}_{ss}(G)$ is the largest step stochastic autobisimulation on G. Based on the equivalence classes with respect to $\mathcal{R}_{ss}(G)$, the quotient (by $\underline{\leftrightarrow}_{ss}$) 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 7.1. 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

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

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

The quotient (by $\underline{\leftrightarrow}_{ss}$) transition systems of static expressions can be defined as well. For $E \in RegStatExpr$, let $TS_{\underline{\leftrightarrow}_{ss}}(E) = TS_{\underline{\leftrightarrow}_{ss}}(\overline{E})$.

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

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

The value $k = 0$ in the summation above corresponds to the case when no self-loops occur. 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 (which is a 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 7.2. 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 $\mathcal{K} \to \widetilde{\mathcal{K}}$ 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 the sojourn time in every $\mathcal{K} \in DR_V(G)/_{\mathcal{R}_{ss}(G)}$ is equal to zero.

The quotient (by $\underline{\leftrightarrow}_{ss}$) underlying SMCs of static expressions can be defined as well. For $E \in RegStatExpr$, let $SMC_{\underline{\leftrightarrow}_{ss}}(E) = SMC_{\underline{\leftrightarrow}_{ss}}(\overline{E})$.

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 $SMC(G)$, respectively.

The quotients of both transition systems and underlying SMCs are the minimal reductions of these objects modulo step stochastic bisimulations. The quotients can be used to simplify analysis of system properties which are 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, excepting that the former method merges states, while the latter one merges places.

Moreover, the algorithms which can be adapted for our framework exist for constructing the quotients of transition systems by bisimulation [64] and those of (discrete or continuous time) Markov chains by ordinary lumping [24]. The algorithms have time complexity $O(m \log n)$ and space complexity $O(m + n)$ (the case of Markov chains), where n is the number of states and m is the number of transitions. As mentioned in [74], the algorithm from [24] can be easily adjusted to produce quotients of labeled probabilistic transition systems by the probabilistic bisimulation equivalence. In [74], the symbolic partition refinement algorithm on state space of CTMCs was proposed. The algorithm can be straightforwardly accommodated to DTMCs, interactive MCs, Markov reward models, Markov decision processes, Kripke structures and labeled probabilistic transition systems. Such a symbolic lumping uses memory efficiently due to compact representation of the state space partition. The symbolic lumping is time efficient, since fast algorithm of the partition representation and refinement is applied.

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

Definition 7.3. 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 $\mathcal{K} \to_{\mathcal{P}} \widetilde{\mathcal{K}}$, where $\mathcal{P} = PM(\mathcal{K},\widetilde{\mathcal{K}})$.

The quotient (by $\underline{\leftrightarrow}_{ss}$) DTMCs of static expressions can be defined as well. For $E \in RegStatExpr$, let $DTMC_{\underline{\leftrightarrow}_{ss}}(E) = DTMC_{\underline{\leftrightarrow}_{ss}}(\overline{E})$.

Eliminating equivalence classes (with respect to $\mathcal{R}_{ss}(G)$) of vanishing states from the quotient (by $\underline{\leftrightarrow}_{ss}$) DTMCs of expressions results in the reductions of such DTMCs.

Definition 7.4. The *reduced quotient (by $\underline{\leftrightarrow}_{ss}$) DTMC* of G, $RDTMC_{\underline{\leftrightarrow}_{ss}}(G)$, is defined like $RDTMC(G)$ in Section 5, but it is constructed from $DTMC_{\underline{\leftrightarrow}_{ss}}(G)$ instead of $DTMC(G)$.

The reduced quotient (by $\underline{\leftrightarrow}_{ss}$) DTMCs of static expressions can be defined as well. For $E \in RegStatExpr$, let $RDTMC_{\underline{\leftrightarrow}_{ss}}(E) = RDTMC_{\underline{\leftrightarrow}_{ss}}(\overline{E})$.

The steady-state PMFs $\psi_{\underline{\leftrightarrow}_{ss}}$ for $DTMC_{\underline{\leftrightarrow}_{ss}}(G)$ and $\psi^\diamond_{\underline{\leftrightarrow}_{ss}}$ for $RDTMC_{\underline{\leftrightarrow}_{ss}}(G)$ are defined like the corresponding notions ψ for $DTMC(G)$ and $\psi^\diamond$ for $RDTMC(G)$, respectively.

Obviously, the relationships between the steady-state PMFs $\psi_{\underline{\leftrightarrow}_{ss}}$ and $\psi^*_{\underline{\leftrightarrow}_{ss}}$, $\varphi_{\underline{\leftrightarrow}_{ss}}$ and $\psi_{\underline{\leftrightarrow}_{ss}}$, as well as $\varphi_{\underline{\leftrightarrow}_{ss}}$ and $\psi^\diamond_{\underline{\leftrightarrow}_{ss}}$, are the same as those determined between their “non-quotient” versions in Theorem 5.1, Proposition 5.2 and Proposition 5.3, respectively.

The comprehensive quotient and reduction example will be presented in Section 9.

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 behavior, i.e. expressions with the iteration operator. Note that the iteration operator does not guarantee infiniteness of behaviour, since there can exist a deadlock 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 "breaked". On the other hand, the iteration body can be left after a finite number of its repeated executions and then the iteration termination is started. To avoid executing any activities after the iteration body, we take Stop as the termination argument of iteration.

Like in the framework of SMCs, in LDTSIPNs the most common systems for performance analysis are *ergodic* (recurrent non-null, aperiodic and irreducible) ones. For ergodic LDTSIPNs, the steady-state marking probabilities exist and can be determined. In [60, 61], the following sufficient (but not necessary) conditions for ergodicity of DTSPNs are stated: *liveness* (for each transition and any reachable marking there exists a sequence of markings from it leading to the marking enabling that transition), *boundedness* (the number of tokens in every place is not greater than some fixed number for any reachable marking) and *nondeterminism* (the transition probabilities are strictly less than 1). Let the dtsi-box of a dynamic expression has no deadlocks in the body of some iteration operator it contains and Stop is the termination argument of this operator. If all the states between the initial and final ones (including both these states) of such an iteration body are tangible, then the three ergodicity conditions are satisfied: the subnet corresponding to this iteration body is live, safe (1-bounded) and nondeterministic (since all markings of the live subnet are tangible and non-terminal, the probabilities of transitions from them are strictly less than 1). Hence, for the dtsi-box, its underlying SMC restricted to the states between the initial and final states of this iteration body is ergodic. The isomorphism between SMCs of expressions and those of the corresponding dtsi-boxes, which is stated by Proposition 5.1, guarantees that the underlying SMC of an expression with infinite behaviour is ergodic, if restricted to the states in which such an iteration body is executed. Since the ergodicity condition above is not necessary, there exist dynamic expressions with vanishing states traversed while executing their iteration bodies, which, nevertheless, have ergodic underlying SMCs, as Example 5.1 demonstrated.

In this section, we consider the expressions such that their underlined SMCs contain one ergodic subset of states to guarantee that a single steady state exists.

8.1. Steady state and equivalences

The following proposition demonstrates that, for two dynamic expressions related by $\underline{\leftrightarrow}_{ss}$, the steady-state probabilities to come in an equivalence class coincide. One can also interpret the result stating that the mean recurrence time for an equivalence class is the same for both expressions.

Proposition 8.1. *Let G, G' be dynamic expressions with $\mathcal{R} : G\underline{\leftrightarrow}_{ss}G'$ and φ be the steady-state PMF for $SMC(G)$, 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 and φ be the steady-state PMF for $SMC(G)$, $\varphi_{\underline{\leftrightarrow}_{ss}}$ be the steady-state PMF for $SMC_{\underline{\leftrightarrow}_{ss}}(G)$. By Proposition 8.1, we have $\forall\mathcal{H} \in DR(G)/_{\mathcal{R}_{ss}(G)},\ \varphi_{\underline{\leftrightarrow}_{ss}}(\mathcal{H}) = \sum_{s\in\mathcal{H}} \varphi(s)$. Thus, for every equivalence class $\mathcal{H} \in DR(G)/_{\mathcal{R}_{ss}(G)}$, the value of $\varphi_{\underline{\leftrightarrow}_{ss}}$ corresponding to $\mathcal{H}$ is the sum of all values of φ corresponding to the states from $\mathcal{H}$. 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 8.1, $\underline{\leftrightarrow}_{ss}$ preserves the quantitative properties of the stationary behaviour (the level of SMCs). Now we intend to demonstrate that the qualitative properties of the stationary behaviour based on the multiaction labels are preserved as well (the level of transition systems).

Definition 8.1. 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 $\exists s \in DR(G),\ s \xrightarrow{\Upsilon_1} s_1 \xrightarrow{\Upsilon_2} \cdots \xrightarrow{\Upsilon_n} s_n,\ \mathcal{L}(\Upsilon_i) = A_i\ (1 \leq i \leq n)$. Then the *probability to execute the derived step trace* Σ *in* s is

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

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

Theorem 8.1. *Let G, G' be dynamic expressions with $\mathcal{R} : G\underline{\leftrightarrow}_{ss}G'$ and 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. □

We now present a result not concerning the steady-state probabilities, but revealing very important properties of residence time in the equivalence classes. The following proposition demonstrates that, for two dynamic expressions related by $\underline{\leftrightarrow}_{ss}$, the sojourn time averages in an equivalence class coincide, as well as the sojourn time variances in it.

Proposition 8.2. *Let G, G' be dynamic expressions with $\mathcal{R} : G\underline{\leftrightarrow}_{ss}G'$. 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. □

Example 8.1. *Let*

$$E = [(\{a\}, \tfrac{1}{2}) * ((\{b\}, \tfrac{1}{2}); ((\{c\}, \tfrac{1}{3})_1[](\{c\}, \tfrac{1}{3})_2)) * \mathsf{Stop}],$$
$$E' = [(\{a\}, \tfrac{1}{2}) * (((\{b\}, \tfrac{1}{3})_1; (\{c\}, \tfrac{1}{2})_1)[]((\{b\}, \tfrac{1}{3})_2; (\{c\}, \tfrac{1}{2})_2)) * \mathsf{Stop}].$$

We have $\overline{E}\underline{\leftrightarrow}_{ss}\overline{E'}$.

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

$$s_1 = [[\overline{(\{a\}, \tfrac{1}{2})} * ((\{b\}, \tfrac{1}{2}); ((\{c\}, \tfrac{1}{3})_1[](\{c\}, \tfrac{1}{3})_2)) * \mathsf{Stop}]]_{\approx},$$
$$s_2 = [[(\{a\}, \tfrac{1}{2}) * (\overline{(\{b\}, \tfrac{1}{2})}; ((\{c\}, \tfrac{1}{3})_1[](\{c\}, \tfrac{1}{3})_2)) * \mathsf{Stop}]]_{\approx},$$
$$s_3 = [[(\{a\}, \tfrac{1}{2}) * ((\{b\}, \tfrac{1}{2}); \overline{((\{c\}, \tfrac{1}{3})_1[](\{c\}, \tfrac{1}{3})_2)}) * \mathsf{Stop}]]_{\approx}.$$

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

$$s_1' = [[\overline{(\{a\}, \tfrac{1}{2})} * (((\{b\}, \tfrac{1}{3})_1; (\{c\}, \tfrac{1}{2})_1)[]((\{b\}, \tfrac{1}{3})_2; (\{c\}, \tfrac{1}{2})_2)) * \mathsf{Stop}]]_{\approx},$$
$$s_2' = [[(\{a\}, \tfrac{1}{2}) * \overline{(((\{b\}, \tfrac{1}{3})_1; (\{c\}, \tfrac{1}{2})_1)[]((\{b\}, \tfrac{1}{3})_2; (\{c\}, \tfrac{1}{2})_2))} * \mathsf{Stop}]]_{\approx},$$
$$s_3' = [[(\{a\}, \tfrac{1}{2}) * (((\{b\}, \tfrac{1}{3})_1; \overline{(\{c\}, \tfrac{1}{2})_1})[]((\{b\}, \tfrac{1}{3})_2; (\{c\}, \tfrac{1}{2})_2)) * \mathsf{Stop}]]_{\approx},$$
$$s_4' = [[(\{a\}, \tfrac{1}{2}) * (((\{b\}, \tfrac{1}{3})_1; (\{c\}, \tfrac{1}{2})_1)[]((\{b\}, \tfrac{1}{3})_2; \overline{(\{c\}, \tfrac{1}{2})_2})) * \mathsf{Stop}]]_{\approx}.$$

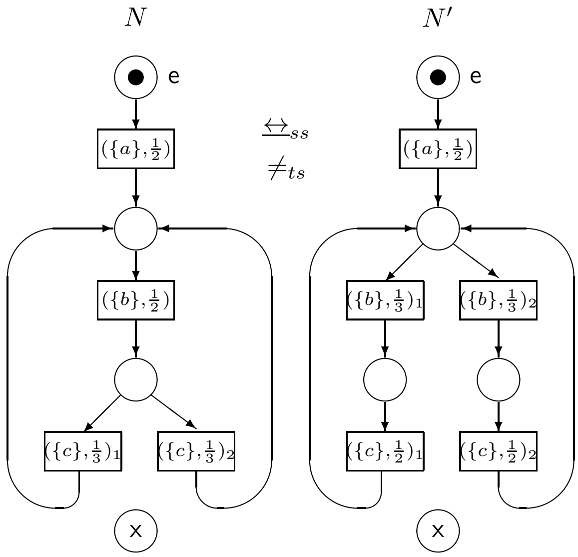

Figure 10: $\underline{\leftrightarrow}_{ss}$ preserves steady-state behaviour and sojourn time properties in the equivalence classes.

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 (with respect to $\mathcal{R}_{ss}(\overline{E}, \overline{E'})$) $\mathcal{H} = \{s_3, s_3', s_4'\}$. 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 come in 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: $SJ_{\mathcal{R}_{ss}(\overline{E},\overline{E'})\cap(DR(\overline{E}))^2}(\mathcal{H} \cap DR(G)) = SJ_{\mathcal{R}_{ss}(\overline{E},\overline{E'})\cap(DR(\overline{E}))^2}(\{s_3\}) = \frac{1}{1-PM(\{s_3\},\{s_3\})} = \frac{1}{1-PM(s_3,s_3)} = \frac{1}{1-\frac{1}{2}} = 2 = \frac{1}{1-\frac{1}{2}} = \frac{1}{1-PM(s_3',s_3')} = \frac{1}{1-PM(s_4',s_4')} = \frac{1}{1-PM(\{s_3',s_4'\},\{s_3',s_4'\})} = SJ_{\mathcal{R}_{ss}(\overline{E},\overline{E'})\cap(DR(\overline{E'}))^2}(\{s_3', s_4'\}) = SJ_{\mathcal{R}_{ss}(\overline{E},\overline{E'})\cap(DR(\overline{E'}))^2}(\mathcal{H} \cap DR(G'))$.

At last, 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 10, the marked dtsi-boxes corresponding to the dynamic expressions above are presented, i.e. $N = Box_{dtsi}(\overline{E})$ and $N' = Box_{dtsi}(\overline{E'})$.

8.2. Preservation of performance and simplification of its analysis

Many performance indices are based on the steady-state probabilities to come in 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 8.1, Theorem 8.1 and Proposition 8.2 guarantee coincidence of the mentioned indices for the expressions related by $\underline{\leftrightarrow}_{ss}$. Thus, $\underline{\leftrightarrow}_{ss}$ (hence, all the stronger equivalences we have considered) preserves performance of stochastic systems modeled by expressions of dtsiPBC.

In addition, it is easier to evaluate performance using an SMC with less states, since in this case the size of the transition probability matrix will be smaller, and we shall solve systems of less equations to calculate 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.

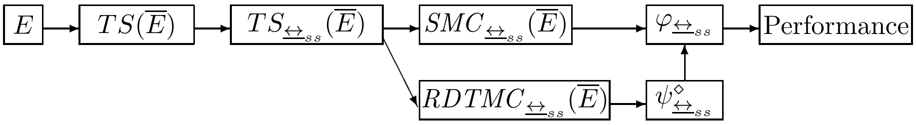

Figure 11: Equivalence-based simplification of performance evaluation.

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 constructed from the quotient transition system.
5. Stationary probabilities and performance indices are calculated using the SMC.

The limitation of the method above is its applicability only to the expressions such that their corresponding SMCs contain one irreducible subset of states, i.e. the existence of exactly one stationary state is required. If an SMC contains several irreducible subsets of states then several steady states may exist which depend on the initial PMF. There is an analytical method to determine the stable states for SMCs of this kind as well [44]. Note that, for every expression, the underlying SMC has by definition only one initial PMF (that at the time moment 0), hence, the stationary state will be only one in this case too. The general steady-state probability is calculated as a sum of the stationary probabilities of the irreducible subsets of states weighted by the probabilities to enter these subsets starting from the initial state and walking through some transient states. It is worth to apply 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 [56, 3, 4] 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 [30] 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 from [38] for generalized SMPs (GSMPs) reduction, allowing one to simplify transient performance analysis (based on the transient probabilities of being in the states of GSMPs).

Alternatively, the results at the end of Section 7 allow us to simplify the steps 4 and 5 of the method above by constructing the reduced quotient DTMC (instead of the quotient underlying SMC) from the quotient transition system, followed by calculating the stationary probabilities of the quotient underlying SMC using this DTMC, and then obtaining the performance indices. We first merge the equivalent states in transition systems and only then eliminate the vanishing states in Markov chains. The reason is that transition systems, being a higher-level formalism than Markov chains, describe both functional (qualitative) and performance (quantitative) aspects of behaviour while Markov chains represent only performance ones. Thus, eliminating vanishing states first would destroy the functional behaviour (which is respected by the equivalence used for quotienting), since the steps with different multiaction parts may lead to or start from different vanishing states.

Figure 11 presents the main stages of the standard and alternative equivalence-based simplification of performance evaluation described above.

9. Shared memory system

In this section with a case study of the shared memory system we show how steady-state distribution can be used for performance evaluation. The example also illustrates the method of performance analysis simplification above.

9.1. The standard system

Consider a model of two processors accessing a common shared memory described in [56, 3, 4] 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). The model works 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

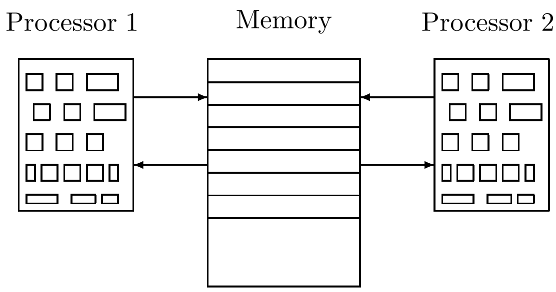

Figure 12: The diagram of the shared memory system.

and another processor should wait until the former one ends its memory operations, and the system returns to the state with both active processors and the available common memory. The diagram of the system is depicted in Figure 12.

Let us explain the meaning of actions from the syntax of dtsiPBC expressions which will specify the system modules. The action a corresponds to the system activation. The actions r_i $(1 \le i \le 2)$ 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 $\mathsf{sy}\ a_1 \cdots \mathsf{sy}\ a_n\ \mathsf{rs}\ a_1 \cdots \mathsf{rs}\ a_n$ to $\mathsf{sr}\ (a_1, \ldots, a_n)$.

The static expression of the first processor is

$$E_1 = [(\{x_1\}, \frac{1}{2}) * ((\{r_1\}, \frac{1}{2}); (\{d_1, y_1\}, 1); (\{m_1, z_1\}, \frac{1}{2})) * \mathsf{Stop}].$$

The static expression of the second processor is

$$E_2 = [(\{x_2\}, \frac{1}{2}) * ((\{r_2\}, \frac{1}{2}); (\{d_2, y_2\}, 1); (\{m_2, z_2\}, \frac{1}{2})) * \mathsf{Stop}].$$

The static expression of the shared memory is

$$E_3 = [(\{a, \widehat{x_1}, \widehat{x_2}\}, \frac{1}{2}) * (((\{\widehat{y_1}\}, 1); (\{\widehat{z_1}\}, \frac{1}{2}))[]((\{\widehat{y_2}\}, 1); (\{\widehat{z_2}\}, \frac{1}{2}))) * \mathsf{Stop}].$$

The static expression of the shared memory system with two processors is

$$E = (E_1 \| E_2 \| E_3)\ \mathsf{sr}\ (x_1, x_2, y_1, y_2, z_1, z_2).$$

Let us illustrate an effect of synchronization. As result of the synchronization of immediate multiactions $(\{d_i, y_i\}, 1)$ and $(\{\widehat{y_i}\}, 1)$ we obtain $(\{d_i\}, 2)$ $(1 \le i \le 2)$. The synchronization of stochastic multiactions $(\{m_i, z_i\}, \frac{1}{2})$ and $(\{\widehat{z_i}\}, \frac{1}{2})$ produces $(\{m_i\}, \frac{1}{4})$ $(1 \le i \le 2)$. The result of synchronization of $(\{a, \widehat{x_1}, \widehat{x_2}\}, \frac{1}{2})$ with $(\{x_1\}, \frac{1}{2})$ is $(\{a, \widehat{x_2}\}, \frac{1}{4})$, and that of synchronization of $(\{a, \widehat{x_1}, \widehat{x_2}\}, \frac{1}{2})$ with $(\{x_2\}, \frac{1}{2})$ is $(\{a, \widehat{x_1}\}, \frac{1}{4})$. After applying synchronization to $(\{a, \widehat{x_2}\}, \frac{1}{4})$ and $(\{x_2\}, \frac{1}{2})$, as well as to $(\{a, \widehat{x_1}\}, \frac{1}{4})$ and $(\{x_1\}, \frac{1}{2})$, we obtain the same activity $(\{a\}, \frac{1}{8})$.

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

$$\begin{aligned} s_1 = [([\overline{(\{x_1\}, \tfrac{1}{2})} * ((\{r_1\}, \tfrac{1}{2}); (\{d_1, y_1\}, 1); (\{m_1, z_1\}, \tfrac{1}{2})) * \mathsf{Stop}] \| \\ \overline{[(\{x_2\}, \tfrac{1}{2})} * ((\{r_2\}, \tfrac{1}{2}); (\{d_2, y_2\}, 1); (\{m_2, z_2\}, \tfrac{1}{2})) * \mathsf{Stop}] \| \\ \overline{[(\{a, \widehat{x_1}, \widehat{x_2}\}, \tfrac{1}{2})} * (((\{\widehat{y_1}\}, 1); (\{\widehat{z_1}\}, \tfrac{1}{2}))[]((\{\widehat{y_2}\}, 1); (\{\widehat{z_2}\}, \tfrac{1}{2}))) * \mathsf{Stop}]) \\ \mathsf{sr}\ (x_1, x_2, y_1, y_2, z_1, z_2)]_{\approx}, \end{aligned}$$

$$\begin{aligned} s_2 = [([(\{x_1\}, \tfrac{1}{2}) * (\overline{(\{r_1\}, \tfrac{1}{2})}; (\{d_1, y_1\}, 1); (\{m_1, z_1\}, \tfrac{1}{2})) * \mathsf{Stop}] \| \\ [(\{x_2\}, \tfrac{1}{2}) * \overline{((\{r_2\}, \tfrac{1}{2})}; (\{d_2, y_2\}, 1); (\{m_2, z_2\}, \tfrac{1}{2})) * \mathsf{Stop}] \| \\ [(\{a, \widehat{x_1}, \widehat{x_2}\}, \tfrac{1}{2}) * \overline{(((\{\widehat{y_1}\}, 1); (\{\widehat{z_1}\}, \tfrac{1}{2}))[]((\{\widehat{y_2}\}, 1); (\{\widehat{z_2}\}, \tfrac{1}{2})))} * \mathsf{Stop}]) \\ \mathsf{sr}\ (x_1, x_2, y_1, y_2, z_1, z_2)]_{\approx}, \end{aligned}$$

$$
\begin{gathered}
s_3 = [([([(\{x_1\},\tfrac{1}{2})\underline{*((\{r_1\},\tfrac{1}{2})};\overline{(\{d_1,y_1\},1)};(\{m_1,z_1\},\tfrac{1}{2}))*\mathsf{Stop}]\|\\
[(\{x_2\},\tfrac{1}{2})*((\{r_2\},\tfrac{1}{2});(\{d_2,y_2\},1);(\{m_2,z_2\},\tfrac{1}{2}))*\mathsf{Stop}]\|\\
[(\{a,\widehat{x_1},\widehat{x_2}\},\tfrac{1}{2})*(\overline{((\{\widehat{y_1}\},1)};(\{\widehat{z_1}\},\tfrac{1}{2}))[]((\{\widehat{y_2}\},1);(\{\widehat{z_2}\},\tfrac{1}{2})))*\mathsf{Stop}])\\
\mathsf{sr}\ (x_1,x_2,y_1,y_2,z_1,z_2)]_{\approx},
\end{gathered}
$$

$$
\begin{gathered}
s_4 = [([([(\{x_1\},\tfrac{1}{2})*\overline{((\{r_1\},\tfrac{1}{2})};(\{d_1,y_1\},1);(\{m_1,z_1\},\tfrac{1}{2}))*\mathsf{Stop}]\|\\
[(\{x_2\},\tfrac{1}{2})*((\{r_2\},\underline{\tfrac{1}{2});(\{d_2,y_2\},1)};(\{m_2,z_2\},\tfrac{1}{2}))*\mathsf{Stop}]\|\\
[(\{a,\widehat{x_1},\widehat{x_2}\},\tfrac{1}{2})*(((\{\widehat{y_1}\},1);(\{\widehat{z_1}\},\tfrac{1}{2}))[]\overline{((\{\widehat{y_2}\},1)};(\{\widehat{z_2}\},\tfrac{1}{2})))*\mathsf{Stop}])\\
\mathsf{sr}\ (x_1,x_2,y_1,y_2,z_1,z_2)]_{\approx},
\end{gathered}
$$

$$
\begin{gathered}
s_5 = [([([(\{x_1\},\tfrac{1}{2})\underline{*((\{r_1\}},\tfrac{1}{2});(\{d_1,y_1\},1);\overline{(\{m_1,z_1\},\tfrac{1}{2})})*\mathsf{Stop}]\|\\
[(\{x_2\},\tfrac{1}{2})*\overline{((\{r_2\},\tfrac{1}{2})};(\underline{\{d_2,y_2\}},1);(\{m_2,z_2\},\tfrac{1}{2}))*\mathsf{Stop}]\|\\
[(\{a,\widehat{x_1},\widehat{x_2}\},\tfrac{1}{2})*(((\{\widehat{y_1}\},1);\overline{(\{\widehat{z_1}\},\tfrac{1}{2})})[]((\{\widehat{y_2}\},1);(\{\widehat{z_2}\},\tfrac{1}{2})))*\mathsf{Stop}])\\
\mathsf{sr}\ (x_1,x_2,y_1,y_2,z_1,z_2)]_{\approx},
\end{gathered}
$$

$$
\begin{gathered}
s_6 = [([([(\{x_1\},\tfrac{1}{2})*((\{r_1\},\underline{\tfrac{1}{2});}\overline{(\{d_1,y_1\},1)};(\{m_1,z_1\},\tfrac{1}{2}))*\mathsf{Stop}]\|\\
[(\{x_2\},\tfrac{1}{2})*\underline{((\{r_2\},\tfrac{1}{2});\overline{(\{d_2,y_2\},1)};(\{m_2,z_2\},\tfrac{1}{2}))*\mathsf{Stop}}]\|\\
[(\{a,\widehat{x_1},\widehat{x_2}\},\tfrac{1}{2})*(((\{\widehat{y_1}\},1);(\{\widehat{z_1}\},\tfrac{1}{2}))[]((\{\widehat{y_2}\},1);(\{\widehat{z_2}\},\tfrac{1}{2})))*\mathsf{Stop}])\\
\mathsf{sr}\ (x_1,x_2,y_1,y_2,z_1,z_2)]_{\approx},
\end{gathered}
$$

$$
\begin{gathered}
s_7 = [([([(\{x_1\},\tfrac{1}{2})*\overline{((\{r_1\},\tfrac{1}{2})};(\{d_1,y_1\},\underline{1);(\{m_1,z_1\}},\tfrac{1}{2}))*\mathsf{Stop}]\|\\
[(\{x_2\},\tfrac{1}{2})*((\{r_2\},\tfrac{1}{2});(\{d_2,y_2\},1);(\{m_2,z_2\},\underline{\tfrac{1}{2}))*}\mathsf{Stop}]\|\\
[(\{a,\widehat{x_1},\widehat{x_2}\},\tfrac{1}{2})*(((\{\widehat{y_1}\},1);(\{\widehat{z_1}\},\tfrac{1}{2}))[]((\{\widehat{y_2}\},1);\overline{(\{\widehat{z_2}\},\tfrac{1}{2})}))*\mathsf{Stop}])\\
\mathsf{sr}\ (x_1,x_2,y_1,y_2,z_1,z_2)]_{\approx},
\end{gathered}
$$

$$
\begin{gathered}
s_8 = [([([(\{x_1\},\tfrac{1}{2})*((\{r_1\},\underline{\tfrac{1}{2});(\{d_1,y_1\},1);}\overline{(\{m_1,z_1\},\tfrac{1}{2}))}*\mathsf{Stop}]\|\\
[(\{x_2\},\tfrac{1}{2})*((\{r_2\},\tfrac{1}{2});\overline{(\underline{\{d_2,y_2\}},1)};(\{m_2,z_2\},\tfrac{1}{2}))*\mathsf{Stop}]\|\\
[(\{a,\widehat{x_1},\widehat{x_2}\},\tfrac{1}{2})*(((\{\widehat{y_1}\},1);\overline{(\{\widehat{z_1}\},\tfrac{1}{2})})[]((\{\widehat{y_2}\},1);(\{\widehat{z_2}\},\tfrac{1}{2})))*\mathsf{Stop}])\\
\mathsf{sr}\ (x_1,x_2,y_1,y_2,z_1,z_2)]_{\approx},
\end{gathered}
$$

$$
\begin{gathered}
s_9 = [([([(\{x_1\},\tfrac{1}{2})*((\{r_1\},\tfrac{1}{2});\overline{(\{d_1,y_1\},\underline{1)};(\{m_1,z_1\}},\tfrac{1}{2}))*\mathsf{Stop}]\|\\
[(\{x_2\},\tfrac{1}{2})*((\{r_2\},\tfrac{1}{2});(\{d_2,y_2\},1);(\{m_2,z_2\},\underline{\tfrac{1}{2}))*}\mathsf{Stop}]\|\\
[(\{a,\widehat{x_1},\widehat{x_2}\},\tfrac{1}{2})*(((\{\widehat{y_1}\},1);(\{\widehat{z_1}\},\tfrac{1}{2}))[]((\{\widehat{y_2}\},1);\overline{(\{\widehat{z_2}\},\tfrac{1}{2})}))*\mathsf{Stop}])\\
\mathsf{sr}\ (x_1,x_2,y_1,y_2,z_1,z_2)]_{\approx}.
\end{gathered}
$$

We have $DR_T(\overline{E}) = \{s_1, s_2, s_5, s_7, s_8, s_9\}$ and $DR_V(\overline{E}) = \{s_3, s_4, s_6\}$.

The states are interpreted as follows: s_1 is the initial state, s_2: the system is activated and the memory is not requested, s_3: the memory is requested by the first processor, s_4: the memory is requested by the second processor, s_5: the memory is allocated to the first processor, s_6: the memory is requested by two processors, s_7: the memory is allocated to the second processor, s_8: the memory is allocated to the first processor and the memory is requested by the second processor, s_9: the memory is allocated to the second processor and the memory is requested by the first processor.

In Figure 13, the transition system $TS(\overline{E})$ is presented. In Figure 14, the underlying SMC $SMC(\overline{E})$ is depicted.

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

$$SJ = \left(8, \frac{4}{3}, 0, 0, \frac{8}{5}, 0, \frac{8}{5}, 4, 4\right).$$

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

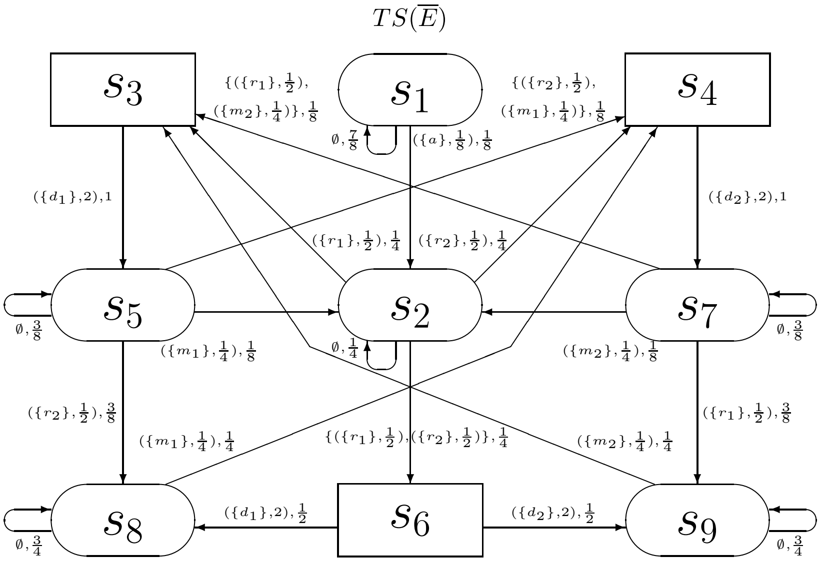

Figure 13: The transition system of the shared memory system.

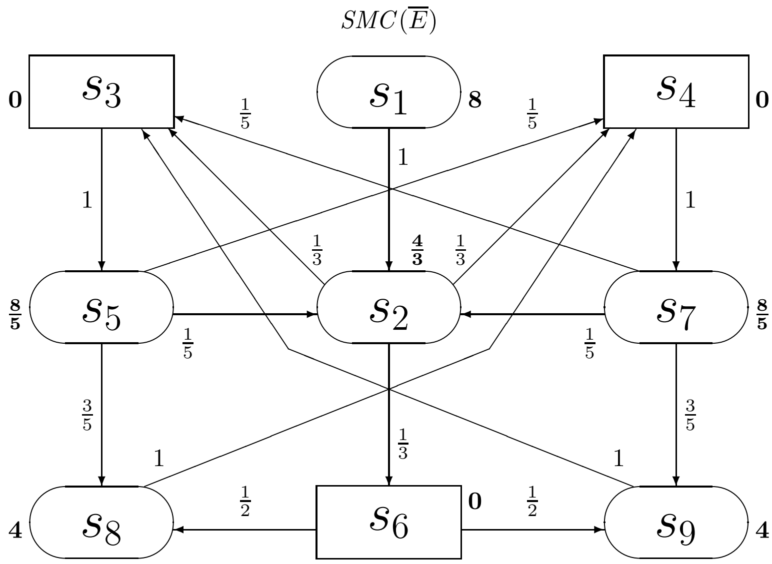

Figure 14: The underlying SMC of the shared memory system.

Table 5: Transient and steady-state probabilities for the EDTMC of the shared memory system.

k	0	5	10	15	20	25	30	35	40	45	50	∞
$\psi_1^*[k]$	1	0	0	0	0	0	0	0	0	0	0	0
$\psi_2^*[k]$	0	0	0.0754	0.0859	0.0677	0.0641	0.0680	0.0691	0.0683	0.0680	0.0681	0.0682
$\psi_3^*[k]$	0	0.2444	0.2316	0.1570	0.1554	0.1726	0.1741	0.1702	0.1696	0.1705	0.1707	0.1705
$\psi_5^*[k]$	0	0.2333	0.0982	0.1516	0.1859	0.1758	0.1672	0.1690	0.1711	0.1708	0.1703	0.1705
$\psi_6^*[k]$	0	0.0444	0.0323	0.0179	0.0202	0.0237	0.0234	0.0226	0.0226	0.0228	0.0228	0.0227
$\psi_8^*[k]$	0	0	0.1163	0.1395	0.1147	0.1077	0.1130	0.1150	0.1139	0.1133	0.1136	0.1136

$$VAR = \left(56, \frac{4}{9}, 0, 0, \frac{24}{25}, 0, \frac{24}{25}, 12, 12\right).$$

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

$$\mathbf{P}^* = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{3} & \frac{1}{3} & 0 & \frac{1}{3} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & \frac{1}{5} & 0 & \frac{1}{5} & 0 & 0 & 0 & \frac{3}{5} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & \frac{1}{5} & \frac{1}{5} & 0 & 0 & 0 & 0 & 0 & \frac{3}{5} \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

In Table 5, the transient and the steady-state probabilities $\psi_i^*[k]$ ($i \in \{1, 2, 3, 5, 6, 8\}$) for the EDTMC of the shared memory system at the time moments $k \in \{0, 5, 10, \ldots, 50\}$ and $k = \infty$ are presented, and in Figure 15, the alteration diagram (evolution in time) for the transient probabilities is depicted. It is sufficient to consider the probabilities for the states $s_1, s_2, s_3, s_5, s_6, s_8$ only, since the corresponding values coincide for s_3, s_4, as well as for s_5, s_7, and for s_8, s_9.

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

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

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

$$\left(0, \frac{1}{11}, 0, 0, \frac{3}{11}, 0, \frac{3}{11}, \frac{5}{11}, \frac{5}{11}\right).$$

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

$$\psi^* SJ^T = \frac{17}{11}.$$

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

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

We can now calculate the main performance indices.

- The average recurrence time in the state s_2, where no processor requests the memory, called the *average system run-through*, is $\frac{1}{\varphi_2} = 17$.

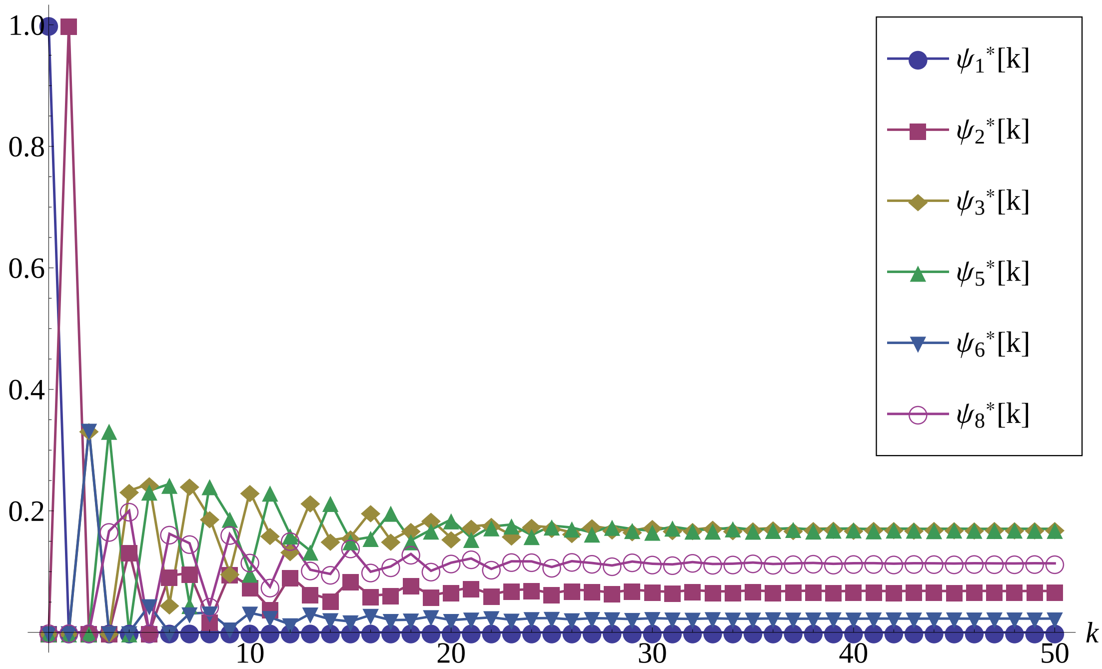

Figure 15: Transient probabilities alteration diagram for the EDTMC of the shared memory system.

- The common memory is available only in the states s_2, s_3, s_4, s_6. The steady-state probability that the memory is available is $\varphi_2 + \varphi_3 + \varphi_4 + \varphi_6 = \frac{1}{17} + 0 + 0 + 0 = \frac{1}{17}$. Then the steady-state probability that the memory is used (i.e. not available), called the *shared memory utilization*, is $1 - \frac{1}{17} = \frac{16}{17}$.
- After activation of the system, we leave the state s_1 for ever, and the common memory is either requested or allocated in every remaining state, with exception of s_2. Thus, the *rate with which the shared memory necessity emerges* coincides with the rate of leaving s_2, calculated as $\frac{\varphi_2}{SJ_2} = \frac{1}{17} \cdot \frac{3}{4} = \frac{3}{68}$.
- The common memory request of the first processor $(\{r_1\}, \frac{1}{2})$ is only possible from the states s_2, s_7. In each of the states, the request probability is the sum of the execution probabilities for all sets of activities containing $(\{r_1\}, \frac{1}{2})$. The *steady-state probability of the shared memory request from the first processor* is $\varphi_2 \sum_{\{\Upsilon \mid (\{r_1\}, \frac{1}{2}) \in \Upsilon\}} PT(\Upsilon, s_2) + \varphi_7 \sum_{\{\Upsilon \mid (\{r_1\}, \frac{1}{2}) \in \Upsilon\}} PT(\Upsilon, s_7) = \frac{1}{17}\left(\frac{1}{4} + \frac{1}{4}\right) + \frac{3}{17}\left(\frac{3}{8} + \frac{1}{8}\right) = \frac{2}{17}$.

In Figure 16, the marked dtsi-boxes corresponding to the dynamic expressions of two processors, shared memory and the shared memory system are presented, i.e. $N_i = Box_{dtsi}(\overline{E_i})$ $(1 \leq i \leq 3)$ and $N = Box_{dtsi}(\overline{E})$.

9.2. The abstract system and its reduction

Let us consider a modification of the shared memory system with abstraction from identifiers of the processors, i.e. such that they are indistinguishable. For example, we can just see that a processor requires memory or the memory is allocated to it but cannot observe which processor is it. We call this system the abstract shared memory one. To implement the abstraction, we replace the actions r_i, d_i, m_i $(1 \leq i \leq 2)$ in the system specification by r, d, m, respectively.

The static expression of the first processor is

$$F_1 = [(\{x_1\}, \frac{1}{2}) * ((\{r\}, \frac{1}{2}); (\{d, y_1\}, 1); (\{m, z_1\}, \frac{1}{2})) * \mathsf{Stop}].$$

The static expression of the second processor is

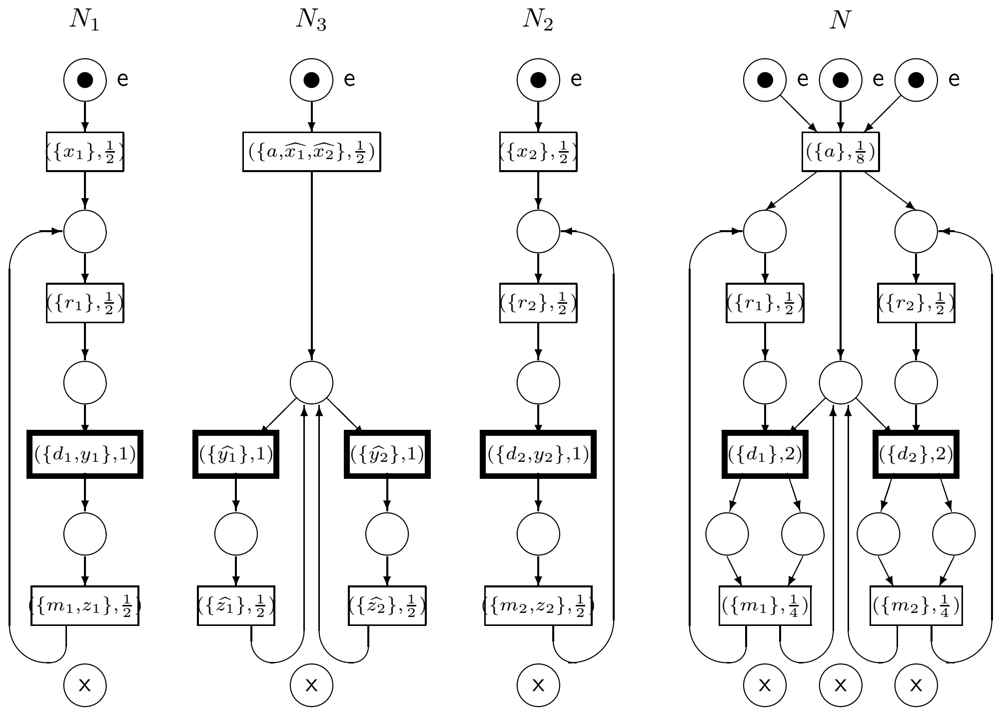

Figure 16: The marked dtsi-boxes of two processors, shared memory and the shared memory system.

$$F_2 = [(\{x_2\}, \frac{1}{2}) * ((\{r\}, \frac{1}{2}); (\{d, y_2\}, 1); (\{m, z_2\}, \frac{1}{2})) * \mathsf{Stop}].$$

The static expression of the shared memory is

$$F_3 = [(\{a, \widehat{x_1}, \widehat{x_2}\}, \frac{1}{2}) * (((\{\widehat{y_1}\}, 1); (\{\widehat{z_1}\}, \frac{1}{2}))[]((\{\widehat{y_2}\}, 1); (\{\widehat{z_2}\}, \frac{1}{2}))) * \mathsf{Stop}].$$

The static expression of the abstract shared memory system with two processors is

$$F = (F_1 \| F_2 \| F_3) \ \mathsf{sr} \ (x_1, x_2, y_1, y_2, z_1, z_2).$$

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

The first, second and third performance indices are the same for the standard and the abstract systems. Let us consider the following performance index which is specific to the abstract system.

- The common memory request of a processor $(\{r\}, \frac{1}{2})$ is only possible from the states s_2, s_5, s_7. In each of the states, the request probability is the sum of the execution probabilities for all sets of activities containing $(\{r\}, \frac{1}{2})$. The *steady-state probability of the shared memory request from a processor* is $\varphi_2 \sum_{\{\Upsilon | (\{r\}, \frac{1}{2}) \in \Upsilon\}} PT(\Upsilon, s_2) + \varphi_5 \sum_{\{\Upsilon | (\{r\}, \frac{1}{2}) \in \Upsilon\}} PT(\Upsilon, s_5) + \varphi_7 \sum_{\{\Upsilon | (\{r\}, \frac{1}{2}) \in \Upsilon\}} PT(\Upsilon, s_7) = \frac{1}{17}\left(\frac{1}{4} + \frac{1}{4} + \frac{1}{4}\right) + \frac{3}{17}\left(\frac{3}{8} + \frac{1}{8}\right) + \frac{3}{17}\left(\frac{3}{8} + \frac{1}{8}\right) = \frac{15}{68}$.

The marked dtsi-boxes corresponding to the dynamic expressions of the standard and the abstract two processors and shared memory are similar, as well as the marked dtsi-boxes corresponding to the dynamic expression of the standard and the abstract shared memory systems.

We have $DR(\overline{F})/_{\mathcal{R}_{ss}(\overline{F})} = \{\mathcal{K}_1, \mathcal{K}_2, \mathcal{K}_3, \mathcal{K}_4, \mathcal{K}_5, \mathcal{K}_6\}$, where $\mathcal{K}_1 = \{s_1\}$ (the initial state), $\mathcal{K}_2 = \{s_2\}$ (the system is activated and the memory is not requested), $\mathcal{K}_3 = \{s_3, s_4\}$ (the memory is requested by one processor), $\mathcal{K}_4 = \{s_5, s_7\}$ (the memory is allocated to a processor), $\mathcal{K}_5 = \{s_6\}$ (the memory is requested by two processors), $\mathcal{K}_6 = \{s_8, s_9\}$ (the memory is allocated to a processor and the memory is requested by another processor).

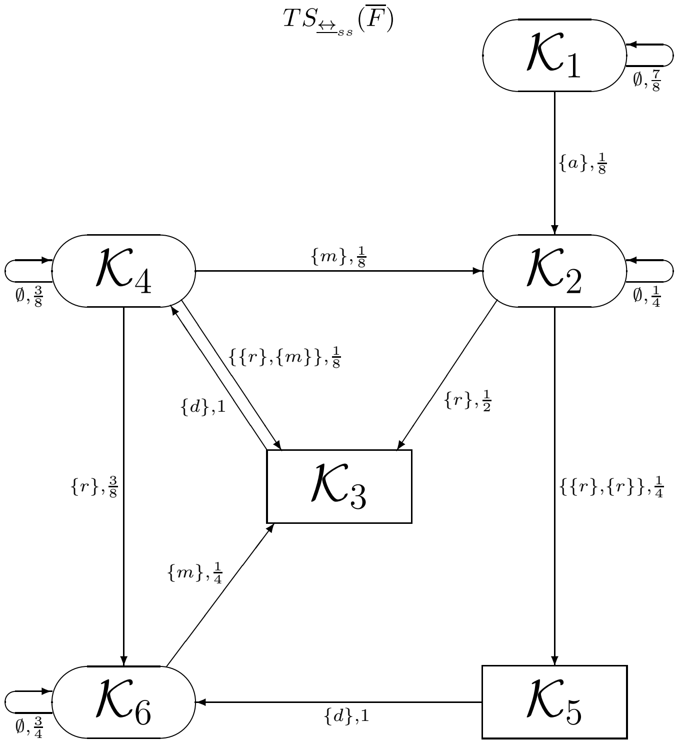

Figure 17: The quotient transition system of the abstract shared memory system.

We also have $DR_T(\overline{F})/_{\mathcal{R}_{ss}(\overline{F})} = \{\mathcal{K}_1, \mathcal{K}_2, \mathcal{K}_4, \mathcal{K}_6\}$ and $DR_V(\overline{F})/_{\mathcal{R}_{ss}(\overline{F})} = \{\mathcal{K}_3, \mathcal{K}_5\}$.

In Figure 17, the quotient transition system $TS_{\underline{\leftrightarrow}_{ss}}(\overline{F})$ is presented. In Figure 18, the quotient underlying SMC $SMC_{\underline{\leftrightarrow}_{ss}}(\overline{F})$ is depicted.

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

$$SJ' = \left(8, \frac{4}{3}, 0, \frac{8}{5}, 0, 4\right).$$

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

$$VAR' = \left(56, \frac{4}{9}, 0, \frac{24}{25}, 0, 12\right).$$

The TPM for $EDTMC_{\underline{\leftrightarrow}_{ss}}(\overline{F})$ is

$$\mathbf{P}'^* = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{2}{3} & 0 & \frac{1}{3} & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & \frac{1}{5} & \frac{1}{5} & 0 & 0 & \frac{3}{5} \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}.$$

In Table 6, the transient and the steady-state probabilities $\psi_i'^*[k]$ $(1 \le i \le 6)$ for the quotient EDTMC of the abstract shared memory system at the time moments $k \in \{0, 5, 10, \ldots, 50\}$ and $k = \infty$ are presented, and in Figure 19, the alteration diagram (evolution in time) for the transient probabilities is depicted.

The steady-state PMF for $EDTMC_{\underline{\leftrightarrow}_{ss}}(\overline{F})$ is

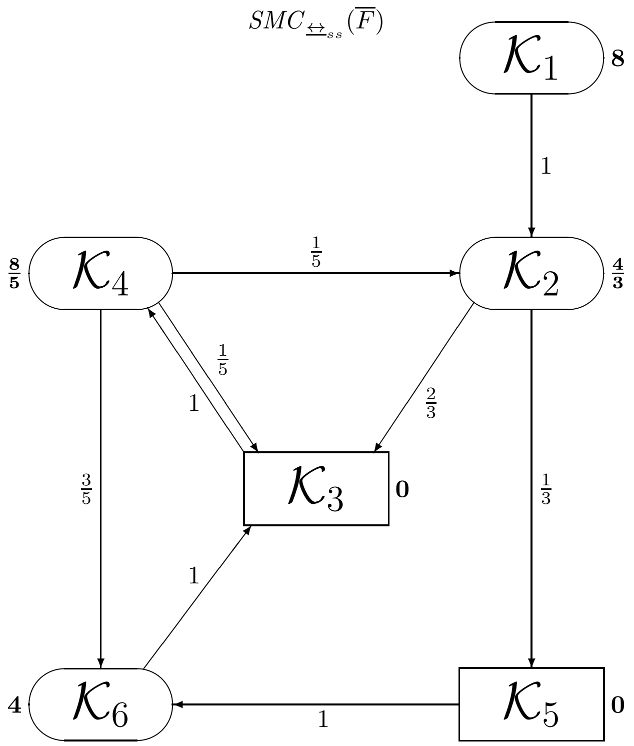

Figure 18: The quotient underlying SMC of the abstract shared memory system.

Table 6: Transient and steady-state probabilities for the quotient EDTMC of the abstract shared memory system.

k	0	5	10	15	20	25	30	35	40	45	50	∞
$\psi_1'^*[k]$	1	0	0	0	0	0	0	0	0	0	0	0
$\psi_2'^*[k]$	0	0	0.0754	0.0859	0.0677	0.0641	0.0680	0.0691	0.0683	0.0680	0.0681	0.0682
$\psi_3'^*[k]$	0	0.4889	0.4633	0.3140	0.3108	0.3452	0.3482	0.3404	0.3392	0.3409	0.3413	0.3409
$\psi_4'^*[k]$	0	0.4667	0.1964	0.3031	0.3719	0.3517	0.3344	0.3380	0.3422	0.3417	0.3407	0.3409
$\psi_5'^*[k]$	0	0.0444	0.0323	0.0179	0.0202	0.0237	0.0234	0.0226	0.0226	0.0228	0.0228	0.0227
$\psi_6'^*[k]$	0	0	0.2325	0.2791	0.2294	0.2154	0.2260	0.2299	0.2277	0.2267	0.2271	0.2273

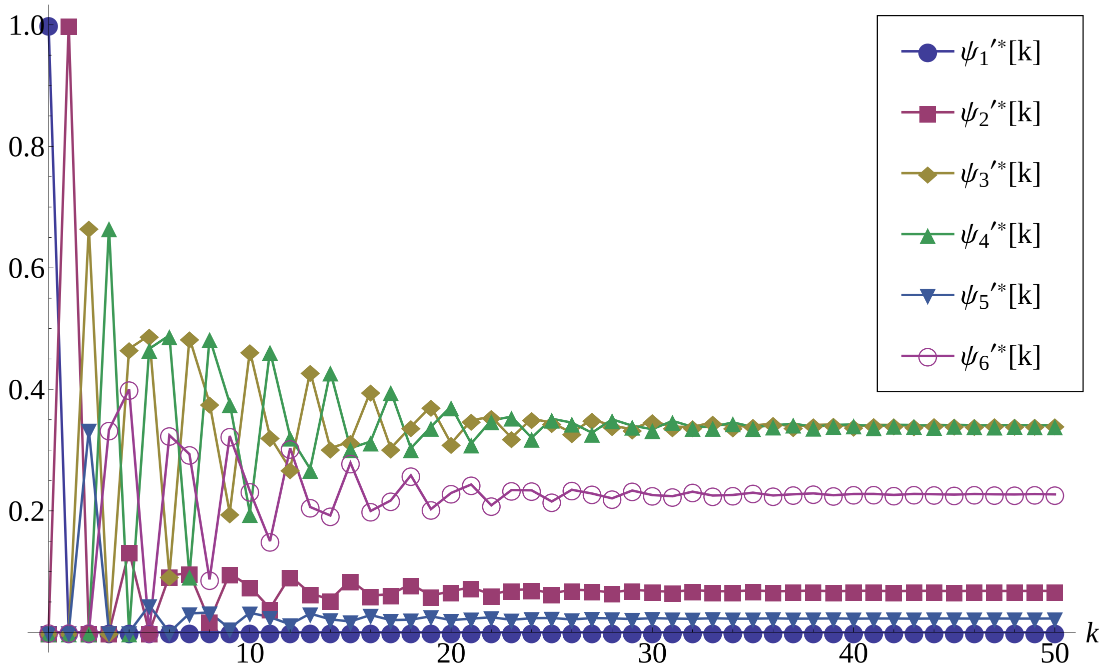

Figure 19: Transient probabilities alteration diagram for the quotient EDTMC of the abstract shared memory system.

$$\psi'^* = \left(0, \frac{3}{44}, \frac{15}{44}, \frac{15}{44}, \frac{1}{44}, \frac{5}{22}\right).$$

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

$$\left(0, \frac{1}{11}, 0, \frac{6}{11}, 0, \frac{10}{11}\right).$$

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

$$\psi'^* SJ'^T = \frac{17}{11}.$$

Thus, the steady-state PMF for $SMC_{\underline{\leftrightarrow}_{ss}}(\overline{F})$ is

$$\varphi' = \left(0, \frac{1}{17}, 0, \frac{6}{17}, 0, \frac{10}{17}\right).$$

Alternatively, from $TS_{\underline{\leftrightarrow}_{ss}}(\overline{F})$, we can construct the reduced quotient DTMC of $\overline{F}$, $RDTMC_{\underline{\leftrightarrow}_{ss}}(\overline{F})$, and then calculate φ' using it.

Remember that $DR_T(\overline{F})/_{\mathcal{R}_{ss}(\overline{F})} = \{\mathcal{K}_1, \mathcal{K}_2, \mathcal{K}_4, \mathcal{K}_6\}$ and $DR_V(\overline{F})/_{\mathcal{R}_{ss}(\overline{F})} = \{\mathcal{K}_3, \mathcal{K}_5\}$. We reorder the elements of $DR(\overline{F})/_{\mathcal{R}_{ss}(\overline{F})}$, by moving the equivalence classes of vanishing states to the first positions: $\mathcal{K}_3, \mathcal{K}_5, \mathcal{K}_1, \mathcal{K}_2, \mathcal{K}_4, \mathcal{K}_6$.

The (reordered) TPM for $DTMC_{\underline{\leftrightarrow}_{ss}}(\overline{F})$ is

$$\mathbf{P}' = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{7}{8} & \frac{1}{8} & 0 & 0 \\ \frac{1}{2} & \frac{1}{4} & 0 & \frac{1}{4} & 0 & 0 \\ \frac{1}{8} & 0 & 0 & \frac{1}{8} & \frac{3}{8} & \frac{3}{8} \\ \frac{1}{4} & 0 & 0 & 0 & 0 & \frac{3}{4} \end{pmatrix}.$$

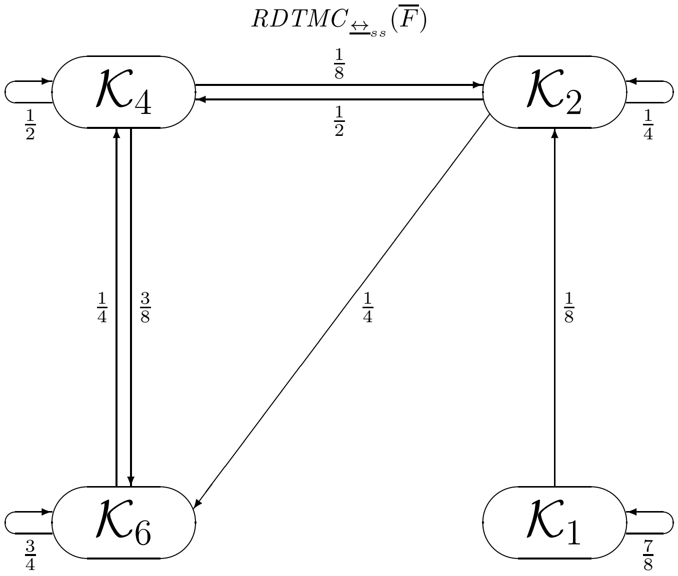

Figure 20: The reduced quotient DTMC of the abstract shared memory system.

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

$$\mathbf{C}' = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix},\ \mathbf{D}' = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},\ \mathbf{E}' = \begin{pmatrix} 0 & 0 \\ \frac{1}{2} & \frac{1}{4} \\ \frac{1}{8} & 0 \\ \frac{1}{4} & 0 \end{pmatrix},\ \mathbf{F}' = \begin{pmatrix} \frac{7}{8} & \frac{1}{8} & 0 & 0 \\ 0 & \frac{1}{4} & 0 & 0 \\ 0 & \frac{1}{8} & \frac{3}{8} & \frac{3}{8} \\ 0 & 0 & 0 & \frac{3}{4} \end{pmatrix}.$$

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

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

Further, the TPM for $RDTMC_{\underline{\leftrightarrow}_{ss}}(\overline{F})$ is

$$\mathbf{P}'^{\diamond} = \mathbf{F}' + \mathbf{E}'\mathbf{G}'\mathbf{D}' = \mathbf{F}' + \mathbf{E}'\mathbf{I}\mathbf{D}' = \mathbf{F}' + \mathbf{E}'\mathbf{D}' = \begin{pmatrix} \frac{7}{8} & \frac{1}{8} & 0 & 0 \\ 0 & \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ 0 & \frac{1}{8} & \frac{1}{2} & \frac{3}{8} \\ 0 & 0 & \frac{1}{4} & \frac{3}{4} \end{pmatrix}.$$

In Figure 20, the reduced quotient DTMC $RDTMC_{\underline{\leftrightarrow}_{ss}}(\overline{F})$ is presented.

In Table 7, the transient and the steady-state probabilities $\psi_i'^{\diamond}[k]$ $(1 \leq i \leq 4)$ for the reduced quotient DTMC of the abstract shared memory system at the time moments $k \in \{0, 5, 10, \ldots, 50\}$ and $k = \infty$ are presented, and in Figure 21, the alteration diagram (evolution in time) for the transient probabilities is depicted.

Then the steady-state PMF for $RDTMC_{\underline{\leftrightarrow}_{ss}}(\overline{F})$ is

$$\psi'^{\diamond} = \left(0, \frac{1}{17}, \frac{6}{17}, \frac{10}{17}\right).$$

Note that $\psi'^{\diamond} = (\psi'^{\diamond}(\mathcal{K}_1), \psi'^{\diamond}(\mathcal{K}_2), \psi'^{\diamond}(\mathcal{K}_4), \psi'^{\diamond}(\mathcal{K}_6))$. By the "quotient" analogue of Proposition 5.3, we have

$$\varphi'(\mathcal{K}_1) = 0, \quad \varphi'(\mathcal{K}_2) = \tfrac{1}{17}, \quad \varphi'(\mathcal{K}_3) = 0, \quad \varphi'(\mathcal{K}_4) = \tfrac{6}{17}, \quad \varphi'(\mathcal{K}_5) = 0, \quad \varphi'(\mathcal{K}_6) = \tfrac{10}{17}.$$

Thus, the steady-state PMF for $SMC_{\underline{\leftrightarrow}_{ss}}(\overline{F})$ is

Table 7: Transient and steady-state probabilities for the reduced quotient DTMC of the abstract shared memory system.

k	0	5	10	15	20	25	30	35	40	45	50	∞
$\psi_1'^{\diamond}[k]$	1	0.5129	0.2631	0.1349	0.0692	0.0355	0.0182	0.0093	0.0048	0.0025	0.0013	0
$\psi_2'^{\diamond}[k]$	0	0.1244	0.0931	0.0764	0.0679	0.0635	0.0612	0.0600	0.0594	0.0591	0.0590	0.0588
$\psi_3'^{\diamond}[k]$	0	0.1726	0.2614	0.3060	0.3289	0.3406	0.3466	0.3497	0.3513	0.3521	0.3525	0.3529
$\psi_4'^{\diamond}[k]$	0	0.1901	0.3824	0.4826	0.5341	0.5605	0.5740	0.5810	0.5845	0.5863	0.5872	0.5882

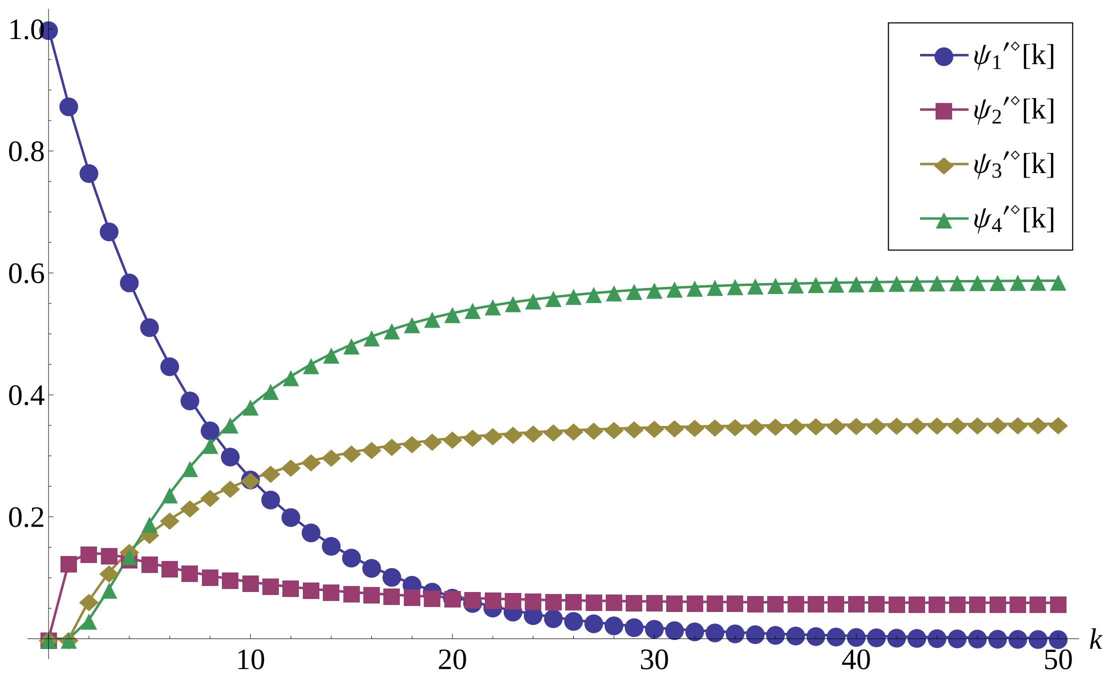

Figure 21: Transient probabilities alteration diagram for the reduced quotient DTMC of the abstract shared memory system.

$$\varphi' = \left(0, \frac{1}{17}, 0, \frac{6}{17}, 0, \frac{10}{17}\right).$$

This coincides with the result obtained with the use of ψ'^* and SJ'.
We can now calculate the main performance indices.

- The average recurrence time in the state $\mathcal{K}_2$, where no processor requests the memory, called the *average system run-through*, is $\frac{1}{\varphi'_2} = \frac{17}{1} = 17$.
- The common memory is available only in the states $\mathcal{K}_2, \mathcal{K}_3, \mathcal{K}_5$. The steady-state probability that the memory is available is $\varphi'_2 + \varphi'_3 + \varphi'_5 = \frac{1}{17} + 0 + 0 = \frac{1}{17}$. Then the steady-state probability that the memory is used (i.e. not available), called the *shared memory utilization*, is $1 - \frac{1}{17} = \frac{16}{17}$.
- After activation of the system, we leave the state $\mathcal{K}_1$ for ever, and the common memory is either requested or allocated in every remaining state, with exception of $\mathcal{K}_2$. Thus, the *rate with which the shared memory necessity emerges* coincides with the rate of leaving $\mathcal{K}_2$, calculated as $\frac{\varphi'_2}{SJ'_2} = \frac{1}{17} \cdot \frac{3}{4} = \frac{3}{68}$.
- The common memory request of a processor $\{r\}$ is only possible from the states $\mathcal{K}_2, \mathcal{K}_4$. 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 $\varphi'_2 \sum_{\{A,\mathcal{K}|\{r\}\in A,\ \mathcal{K}_2 \xrightarrow{A} \mathcal{K}\}} PM_A(\mathcal{K}_2, \mathcal{K}) + \varphi'_4 \sum_{\{A,\mathcal{K}|\{r\}\in A,\ \mathcal{K}_4 \xrightarrow{A} \mathcal{K}\}} PM_A(\mathcal{K}_4, \mathcal{K}) = \frac{1}{17}\left(\frac{1}{2} + \frac{1}{4}\right) + \frac{6}{17}\left(\frac{3}{8} + \frac{1}{8}\right) = \frac{15}{68}$.

One can see that the performance indices are the same for the complete and the quotient abstract shared memory systems. The coincidence of the first, second and third performance indices obviously illustrates the results of Proposition 8.1 and Proposition 8.2. The coincidence of the fourth performance index is due to Theorem 8.1: one should just apply its result to the derived step traces $\{\{r\}\}$, $\{\{r\}, \{r\}\}$, $\{\{r\}, \{m\}\}$ of the expression $\overline{F}$ and itself, and then sum the left and right parts of the three resulting equalities.

9.3. The generalized system

Now we obtain the performance indices taking general values for all multiaction probabilities and weights. Let us suppose that all the mentioned multiactions have the same generalized probability $\rho \in (0;1)$, and generalized weight $l \in \mathbb{N}_{\geq 1}$. The resulting specification K of the generalized shared memory system is defined as follows.

The static expression of the first processor is

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

The static expression of the second processor is

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

The static expression of the shared memory is

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

The static expression of the generalized shared memory system with two processors is

$$K = (K_1 \| K_2 \| K_3) \text{ sr } (x_1, x_2, y_1, y_2, z_1, z_2).$$

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 states are interpreted as follows: $\tilde{s}_1$ 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.

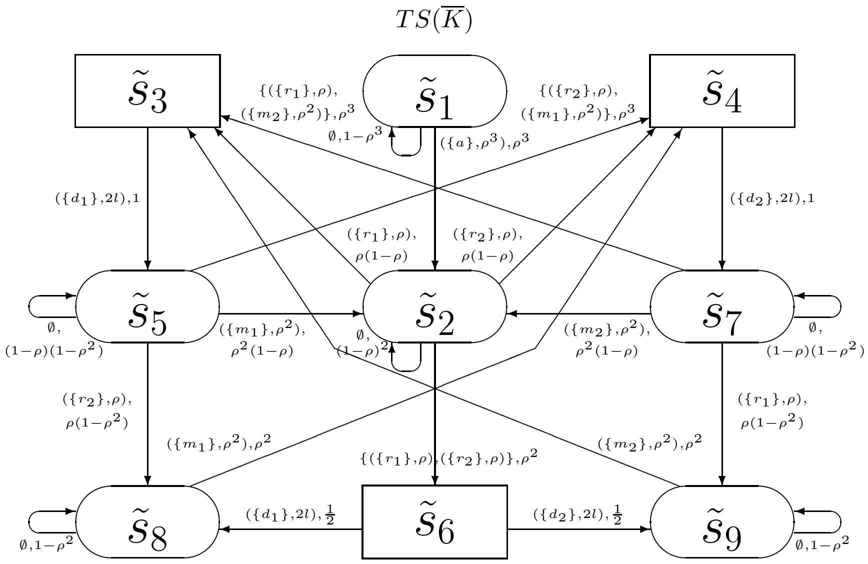

Figure 22: The transition system of the generalized shared memory system.

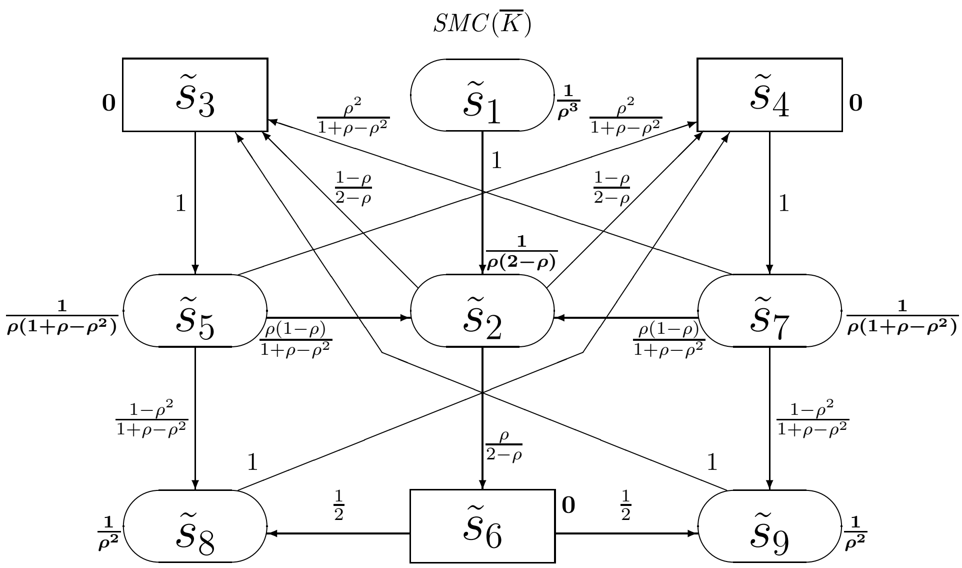

Figure 23: The underlying SMC of the generalized shared memory system.

In Figure 22, the transition system $TS(\overline{K})$ is presented. In Figure 23, the underlying SMC $SMC(\overline{K})$ is depicted. 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

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

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

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

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

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

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

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

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 $\tilde{\varphi}_2 + \tilde{\varphi}_3 + \tilde{\varphi}_4 + \tilde{\varphi}_6 = \frac{\rho^2(1-\rho)}{2+\rho-\rho^2-\rho^3} + 0 + 0 + 0 = \frac{\rho^2(1-\rho)}{2+\rho-\rho^2-\rho^3}$. Then the steady-state probability that the memory is used (i.e. not available), called the *shared memory utilization*, is $1 - \frac{\rho^2(1-\rho)}{2+\rho-\rho^2-\rho^3} = \frac{2+\rho-2\rho^2}{2+\rho-\rho^2-\rho^3}$.
- After activation of the system, we leave the state $\tilde{s}_1$ for ever, and the common memory is either requested or allocated in every remaining state, with exception of $\tilde{s}_2$. Thus, the *rate with which the shared memory necessity emerges* coincides with the rate of leaving $\tilde{s}_2$, calculated as $\frac{\tilde{\varphi}_2}{\widetilde{SJ}_2} = \frac{\rho^2(1-\rho)}{2+\rho-\rho^2-\rho^3} \cdot \frac{\rho(2-\rho)}{1} = \frac{\rho^3(1-\rho)(2-\rho)}{2+\rho-\rho^2-\rho^3}$.

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

9.4. The abstract generalized system and its reduction

Let us consider a modification of the generalized shared memory system with abstraction from identifiers of the processors. We call this system the abstract generalized shared memory one.

The static expression of the first processor is

$$L_1=[(\{x_1\},\rho)*((\{r\},\rho);(\{d,y_1\},l);(\{m,z_1\},\rho))*\mathsf{Stop}].$$

The static expression of the second processor is

$$L_2=[(\{x_2\},\rho)*((\{r\},\rho);(\{d,y_2\},l);(\{m,z_2\},\rho))*\mathsf{Stop}].$$

The static expression of the shared memory is

$$L_3=[(\{a,\widehat{x_1},\widehat{x_2}\},\rho)*(((\{\widehat{y_1}\},l);(\{\widehat{z_1}\},\rho))[]((\{\widehat{y_2}\},l);(\{\widehat{z_2}\},\rho)))*\mathsf{Stop}].$$

The static expression of the abstract generalized shared memory system with two processors is

$$L=(L_1\|L_2\|L_3)\ \mathsf{sr}\ (x_1,x_2,y_1,y_2,z_1,z_2).$$

$DR(\overline{L})$ 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 abstract modification. Let us consider the following performance index which is again 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 $(\{r\},\rho)$. 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).

We also have $DR_T(\overline{L})/_{\mathcal{R}_{ss}(\overline{L})}=\{\widetilde{\mathcal{K}}_1,\widetilde{\mathcal{K}}_2,\widetilde{\mathcal{K}}_4,\widetilde{\mathcal{K}}_6\}$ and $DR_V(\overline{L})/_{\mathcal{R}_{ss}(\overline{L})}=\{\widetilde{\mathcal{K}}_3,\widetilde{\mathcal{K}}_5\}$.

In Figure 24, the quotient transition system $TS_{\underline{\leftrightarrow}_{ss}}(\overline{L})$ is presented. In Figure 25, the quotient underlying SMC $SMC_{\underline{\leftrightarrow}_{ss}}(\overline{L})$ is depicted.

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

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

The TPM for $EDTMC_{\underline{\leftrightarrow}_{ss}}(\overline{L})$ is

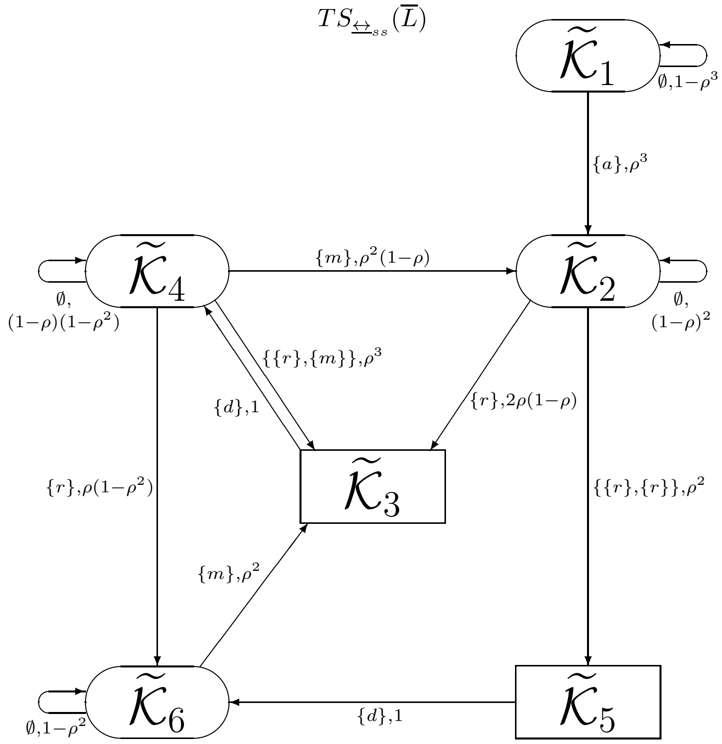

Figure 24: The quotient transition system of the abstract generalized shared memory system.

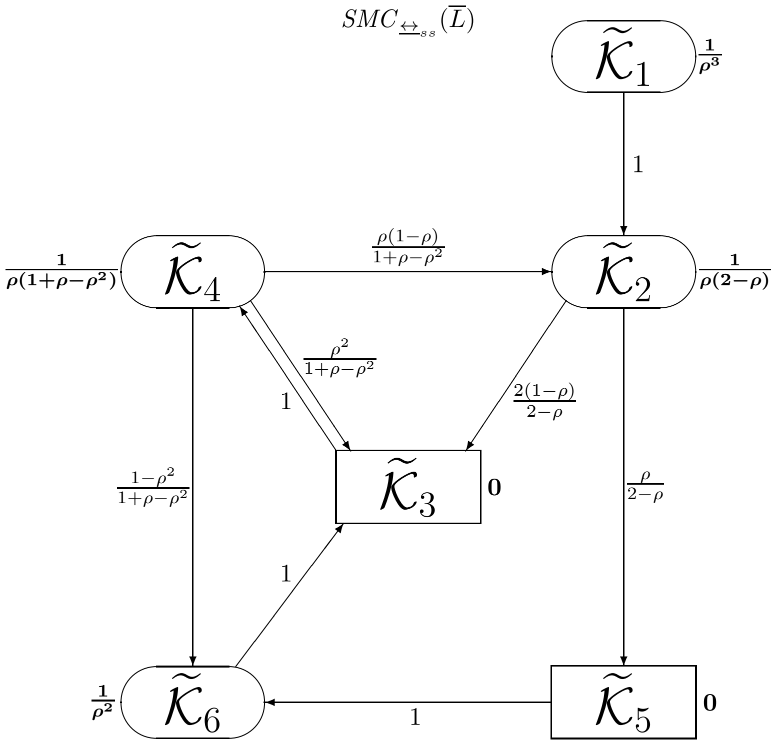

Figure 25: The quotient underlying SMC of the abstract generalized shared memory system.

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

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

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

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

$$\tilde{\psi}'^{*}\widetilde{SJ}'^{T} = \frac{2+\rho-\rho^2-\rho^3}{\rho^2(6+3\rho-9\rho^2+2\rho^3)}.$$

Thus, the steady-state PMF for $SMC_{\underline{\leftrightarrow}_{ss}}(\overline{L})$ is

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

Alternatively, from $TS_{\underline{\leftrightarrow}_{ss}}(\overline{L})$, we can construct the reduced quotient DTMC of $\overline{L}$, $RDTMC_{\underline{\leftrightarrow}_{ss}}(\overline{L})$, and then calculate $\tilde{\varphi}'$ using it.

Remember that $DR_T(\overline{L})/_{\mathcal{R}_{ss}(\overline{L})} = \{\widetilde{\mathcal{K}}_1, \widetilde{\mathcal{K}}_2, \widetilde{\mathcal{K}}_4, \widetilde{\mathcal{K}}_6\}$ and $DR_V(\overline{L})/_{\mathcal{R}_{ss}(\overline{L})} = \{\widetilde{\mathcal{K}}_3, \widetilde{\mathcal{K}}_5\}$. We reorder the elements of $DR(\overline{L})/_{\mathcal{R}_{ss}(\overline{L})}$, by moving the equivalence classes of vanishing states to the first positions: $\widetilde{\mathcal{K}}_3, \widetilde{\mathcal{K}}_5, \widetilde{\mathcal{K}}_1, \widetilde{\mathcal{K}}_2, \widetilde{\mathcal{K}}_4, \widetilde{\mathcal{K}}_6$.

The (reordered) TPM for $DTMC_{\underline{\leftrightarrow}_{ss}}(\overline{L})$ is

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

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

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

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

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

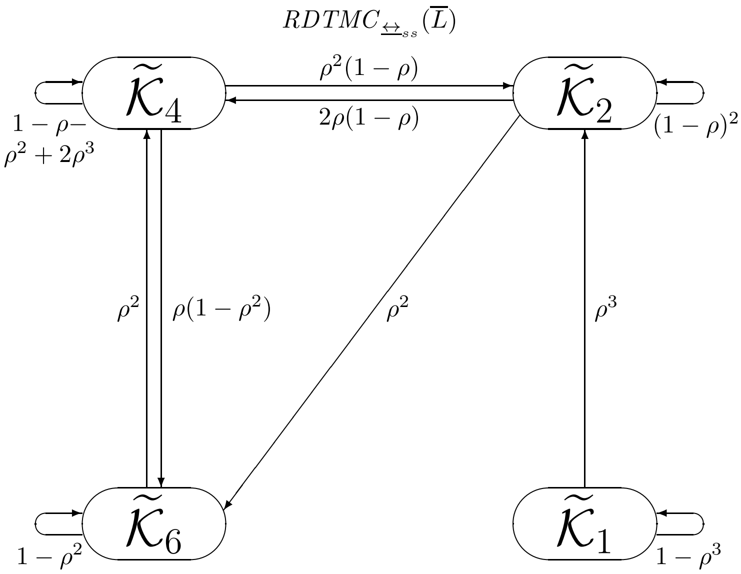

Figure 26: The reduced quotient DTMC of the abstract generalized shared memory system.

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

Further, the TPM for $RDTMC_{\underline{\leftrightarrow}_{ss}}(\overline{L})$ is

$$\widetilde{\mathbf{P}}'^{\diamond} = \widetilde{\mathbf{F}}' + \widetilde{\mathbf{E}}'\widetilde{\mathbf{G}}'\widetilde{\mathbf{D}}' = \widetilde{\mathbf{F}}' + \widetilde{\mathbf{E}}'\mathbf{I}\widetilde{\mathbf{D}}' = \widetilde{\mathbf{F}}' + \widetilde{\mathbf{E}}'\widetilde{\mathbf{D}}' = \begin{pmatrix} 1-\rho^3 & \rho^3 & 0 & 0 \\ 0 & (1-\rho)^2 & 2\rho(1-\rho) & \rho^2 \\ 0 & \rho^2(1-\rho) & 1-\rho-\rho^2+2\rho^3 & \rho(1-\rho^2) \\ 0 & 0 & \rho^2 & 1-\rho^2 \end{pmatrix}.$$

In Figure 26, the reduced quotient DTMC $RDTMC_{\underline{\leftrightarrow}_{ss}}(\overline{L})$ is presented.
Then the steady-state PMF for $RDTMC_{\underline{\leftrightarrow}_{ss}}(\overline{L})$ is

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

Note that $\tilde{\psi}'^{\diamond} = (\tilde{\psi}'^{\diamond}(\widetilde{\mathcal{K}}_1), \tilde{\psi}'^{\diamond}(\widetilde{\mathcal{K}}_2), \tilde{\psi}'^{\diamond}(\widetilde{\mathcal{K}}_4), \tilde{\psi}'^{\diamond}(\widetilde{\mathcal{K}}_6))$. By the "quotient" analogue of Proposition 5.3, we have

$$\tilde{\varphi}'(\widetilde{\mathcal{K}}_1) = 0, \quad \tilde{\varphi}'(\widetilde{\mathcal{K}}_2) = \frac{\rho^2(1-\rho)}{2+\rho-\rho^2-\rho^3}, \quad \tilde{\varphi}'(\widetilde{\mathcal{K}}_3) = 0, \quad \tilde{\varphi}'(\widetilde{\mathcal{K}}_4) = \frac{\rho(2-\rho)}{2+\rho-\rho^2-\rho^3}, \quad \tilde{\varphi}'(\widetilde{\mathcal{K}}_5) = 0, \quad \tilde{\varphi}'(\widetilde{\mathcal{K}}_6) = \frac{2-\rho-\rho^2}{2+\rho-\rho^2-\rho^3}.$$

Thus, the steady-state PMF for $SMC_{\underline{\leftrightarrow}_{ss}}(\overline{L})$ is

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

This coincides with the result obtained with the use of $\tilde{\psi}'^{*}$ and $\widetilde{SJ}'$.
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 $\tilde{\varphi}'_2 + \tilde{\varphi}'_3 + \tilde{\varphi}'_5 = \frac{\rho^2(1-\rho)}{2+\rho-\rho^2-\rho^3} + 0 + 0 = \frac{\rho^2(1-\rho)}{2+\rho-\rho^2-\rho^3}$. Then 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 $\widetilde{\mathcal{K}}_2$. Thus, the *rate with which the shared memory necessity emerges* coincides with the rate of leaving $\widetilde{\mathcal{K}}_2$, calculated as $\frac{\tilde{\varphi}'_2}{\widetilde{SJ}'_2} = \frac{\rho^2(1-\rho)}{2+\rho-\rho^2-\rho^3} \cdot \frac{\rho(2-\rho)}{1} = \frac{\rho^3(1-\rho)(2-\rho)}{2+\rho-\rho^2-\rho^3}$.

- The common memory request of 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 $\tilde{\varphi}'_2 \sum_{\{A,\widetilde{\mathcal{K}}|\{r\}\in A,\ \widetilde{\mathcal{K}}_2 \xrightarrow{A} \widetilde{\mathcal{K}}\}} PM_A(\widetilde{\mathcal{K}}_2, \widetilde{\mathcal{K}}) + \tilde{\varphi}'_4 \sum_{\{A,\widetilde{\mathcal{K}}|\{r\}\in A,\ \widetilde{\mathcal{K}}_4 \xrightarrow{A} \widetilde{\mathcal{K}}\}} PM_A(\widetilde{\mathcal{K}}_4, \widetilde{\mathcal{K}}) = \frac{\rho^2(1-\rho)}{2+\rho-\rho^2-\rho^3}(2\rho(1-\rho)+\rho^2) + \frac{\rho(2-\rho)}{2+\rho-\rho^2-\rho^3}(\rho(1-\rho^2)+\rho^3) = \frac{\rho^2(2-\rho)(1+\rho-\rho^2)}{2+\rho-\rho^2-\rho^3}$.

One can see that 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 Proposition 8.1 and Proposition 8.2. The coincidence of the fourth performance index is due to Theorem 8.1: one should just apply its 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.

Let us consider what is 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 multiaction of the system. The closer is ρ to 0, the less is the probability to execute some activities at every discrete time step, 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 step, 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 27, the graphs for $\tilde{\varphi}'_2$, $\tilde{\varphi}'_4$, $\tilde{\varphi}'_6$ as functions of ρ are depicted. Notice that, however, 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*.

Further, $\tilde{\varphi}'_2$, $\tilde{\varphi}'_6$ tend to 0 and $\tilde{\varphi}'_4$ tends to 1 when ρ approaches 1. Thus, 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 $\rho \approx 0.7433$. In this case, the probability that the system is activated and the memory is not requested is maximal, i.e. the *maximal shared memory availability* is about 8%.

In Figure 28, the graph for the average system run-through, calculated as $\frac{1}{\tilde{\varphi}'_2}$, as a function of ρ is depicted. One can see that 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.

The first graph in Figure 29 represents the shared memory utilization, calculated as $1 - \tilde{\varphi}'_2 - \tilde{\varphi}'_3 - \tilde{\varphi}'_5$, as a function of ρ. One can see that 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 graph in Figure 29 represents the rate with which the shared memory necessity emerges, calculated as $\frac{\tilde{\varphi}'_2}{\widetilde{SJ}'_2}$, as a function of ρ. One can see that 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 $\rho \approx 0.7743$. Thus, the *maximal rate with which the shared memory necessity emerges* is about $\frac{1}{13}$. To decrease the mentioned rate, one should take the parameter ρ closer to 0 or 1.

The third graph in Figure 29 represents the steady-state probability of the shared memory request from a processor, calculated as $\tilde{\varphi}'_2\widetilde{\Sigma}'_2 + \tilde{\varphi}'_4\widetilde{\Sigma}'_4$, where $\widetilde{\Sigma}'_i = \sum_{\{A,\widetilde{\mathcal{K}}|\{r\}\in A,\ \widetilde{\mathcal{K}}_i \xrightarrow{A} \widetilde{\mathcal{K}}\}} PM_A(\widetilde{\mathcal{K}}_i, \widetilde{\mathcal{K}})$, $i \in \{2,4\}$, as a function of ρ. 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 mentioned probability, one should take the parameter ρ closer to 1.

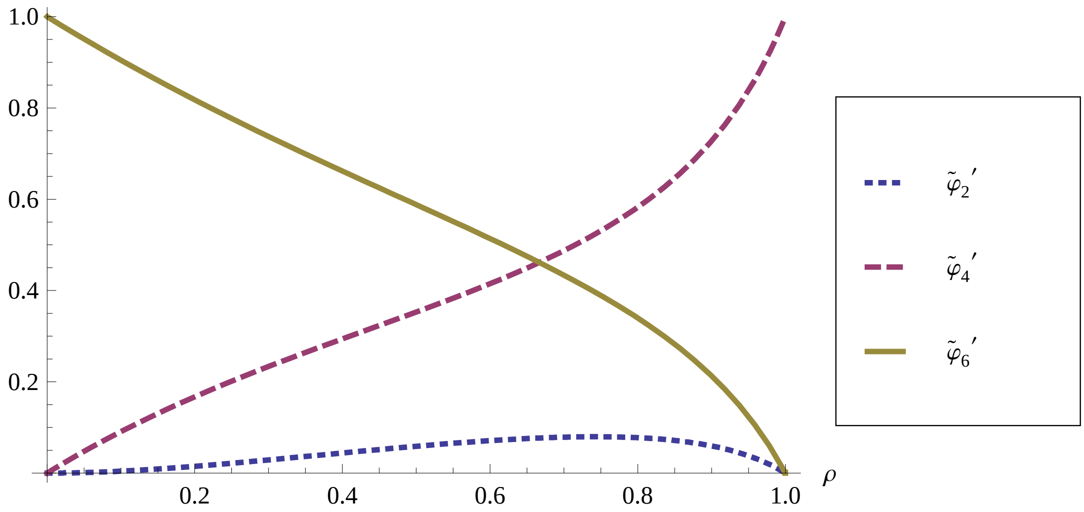

Figure 27: Steady-state probabilities $\tilde{\varphi}_2'$, $\tilde{\varphi}_4'$, $\tilde{\varphi}_6'$ as functions of the parameter ρ.

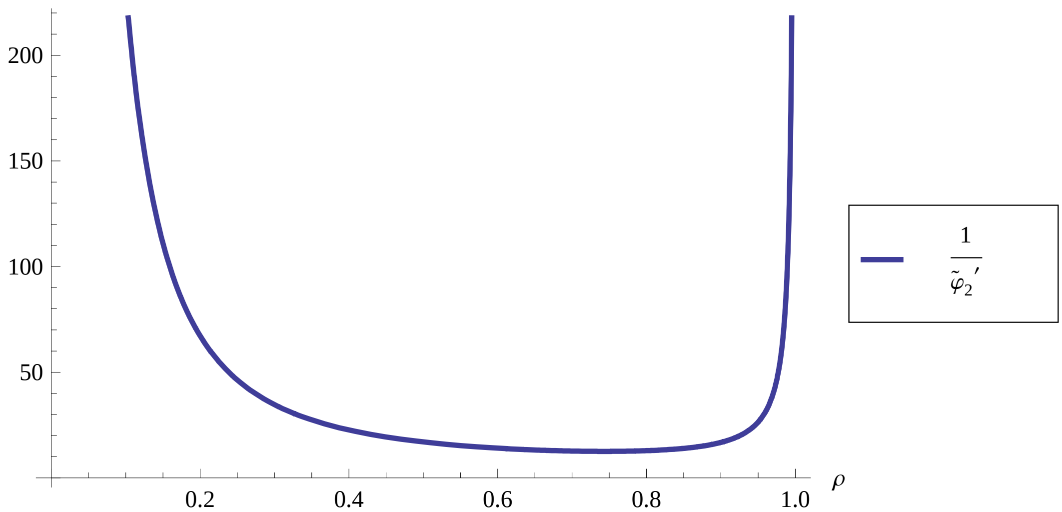

Figure 28: Average system run-through $\frac{1}{\tilde{\varphi}_2'}$ as a function of the parameter ρ.

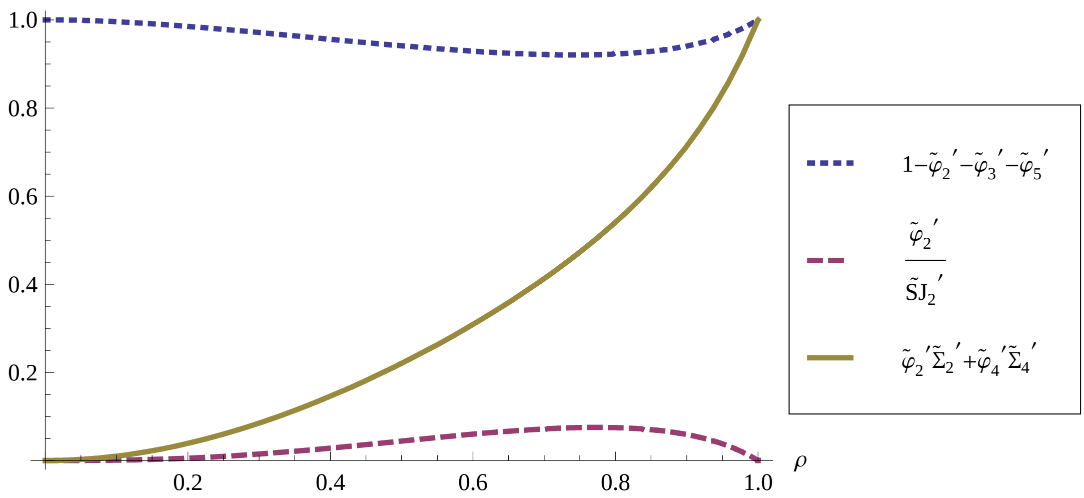

Figure 29: Some performance indices as functions of the parameter ρ.

10. Related work

In this section, we consider in detail differences and similarities between dtsiPBC and other well-known or similar SPAs for the purpose of subsequent determining the specific advantages of dtsiPBC.

10.1. Continuous time and interleaving semantics

Let us compare dtsiPBC with classical SPAs: Markovian Timed Processes for Performance Evaluation (MTIPP) [34], Performance Evaluation Process Algebra (PEPA) [36] and Extended Markovian Process Algebra (EMPA) [10].

In MTIPP, every activity is a pair consisting of the action name (including the symbol τ for the *internal*, invisible action) and the parameter of exponential distribution of the action delay (the *rate*). The operations are *prefix*, *choice*, *parallel* composition including *synchronization* on the specified action set and *recursion*. It is possible to specify processes by recursive equations as well. The interleaving semantics is defined on the basis of Markovian (i.e. extended with the specification of rates) labeled transition systems. Note that we have the interleaving behaviour here because the exponential PDF is a continuous one, and a simultaneous firing of any two activities has zero probability according to the properties of continuous distributions. The continuous time Markov chains (CTMCs) can be derived from the mentioned transition systems to analyze the performance.

In PEPA, activities are the pairs consisting of action types (including the *unknown*, unimportant type τ) and activity rates. The rate is either the parameter of exponential distribution of the activity duration or it is *unspecified*, denoted by $\top$. An activity with unspecified rate is *passive* by its action type. The set of operations includes *prefix*, *choice*, *cooperation*, *hiding* and constants whose meaning is given by the defining equations including the *recursive* ones. The cooperation is accomplished on the set of action types (the cooperation set) on which the components must *synchronize* or cooperate. If the cooperation set is empty, the cooperation operator turns into the *parallel* combinator. The 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 EMPA, each action is 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 generalized SPNs, GSPNs) and *passive* ones (similar to passive actions of PEPA). Timed actions specify activities that are relevant for performance analysis. Immediate actions model logical events and the activities that are irrelevant from the performance viewpoint or much faster than others. Passive actions model activities waiting for the synchronization with timed or immediate ones, and express nondeterministic choice. The set of operators consist of *prefix*, functional *abstraction*, functional *relabeling*, *alternative* composition and *parallel* composition ones. Parallel composition includes *synchronization* on the set of action types like in TCSP [37]. The syntax also includes *recursive* definitions given by means of constants. The semantics is interleaving and based on the labeled transition systems enriched with the information on 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 dtsiPBC, every activity is 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 (the activity is called an immediate multiaction in this case). Immediate multiactions in dtsiPBC are similar to immediate actions in EMPA, but all the immediate multiactions have the same priority 1 (with the purpose to execute them always before stochastic multiactions, all having the same priority 0), whereas the immediate actions in EMPA can have different priority levels. There are no immediate actions in MTIPP and PEPA. Immediate actions are available only in iPEPA [33], where they are analogous to immediate multiactions in dtsiPBC, and in a variant of TIPP [29] discussed while constructing the calculus PM-TIPP in [67], but there immediate activities are used just to specify probabilistic branching and they cannot be synchronized.

dtsiPBC has the sequence operation in contrast to the prefix one in the classical SPAs. One can combine arbitrary expressions with the sequence operator, i.e. it is more flexible than the prefix one, where the first argument should be a single activity. The choice operation in dtsiPBC is analogous to that in MTIPP and PEPA, as well as to the alternative composition in EMPA, in the sense that the choice is probabilistic, but a discrete probability function is used in dtsiPBC, unlike continuous ones in the classical calculi. Concurrency and synchronization in dtsiPBC are different operations (this feature is inherited from PBC), unlike the situation in the classical SPAs where parallel composition (combinator) has a synchronization capability. Relabeling in dtsiPBC is analogous to that in EMPA, but it is additionally extended to conjugated actions. The restriction operation in dtsiPBC differs from hiding in PEPA and functional abstraction in EMPA, where the hidden actions are labeled with a symbol of "silent" action τ. In dtsiPBC, restriction by an action means that, for a given expression, any process behaviour containing the action or its conjugate is not allowed. The synchronization on an elementary action in dtsiPBC collects all the pairs consisting of this elementary action and its conjugate which are contained in the multiactions from the synchronized activities. The operation produces new activities such that the first element of every resulting activity is the union of the multiactions from which all the mentioned pairs of conjugated actions are removed. The second element is either the product of the probabilities of the synchronized stochastic multiactions or the sum of the weights of the synchronized immediate multiactions. This differs from the way synchronization is applied in the classical SPAs where it is accomplished over identical action names, and every resulting activity consists of the same action name and the rate calculated via some expression (including sums, minimums and products) on the rates of the initial activities, such as the apparent rate in PEPA. dtsiPBC has no recursion operation or recursive definitions, but it includes the iteration operation to specify infinite looping behaviour with the explicitly defined start and termination.

dtsiPBC has a discrete time semantics, and time delays in the tangible states are geometrically distributed, unlike the classical SPAs with continuous time semantics and exponentially distributed activity delays. As a consequence, the semantics of dtsiPBC is the step one in contrast to the interleaving semantics of the classical SPAs. The performance is investigated via the underlying semi-Markov chains (SMCs) and (reduced) DTMCs extracted from the labeled probabilistic transition systems associated with expressions of dtsiPBC. In the classical SPAs, continuous time Markov chains (CTMCs) are usually used for performance evaluation. In [26], 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. In [9, 6], 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 discrete time Markov chains (DTMCs) (but when there are only immediate transitions). dtsiPBC has a denotational semantics in terms of LDTSIPNs from which the underlying SMCs and embedded DTMCs (EDTMCs) can be derived.

10.2. Continuous time and non-interleaving semantics

Only a few non-interleaving SPAs were proposed among non-Markovian ones [42]. The semantics of all Markovian calculi is interleaving and their action delays have exponential distribution, which is the only continuous probability distribution with memoryless (Markovian) property.

In [17], Generalized Stochastic Process Algebra (GSPA) was introduced. It has a true-concurrent denotational semantics in terms of generalized stochastic event structures (GSESs) with non-Markovian stochastic delays of events. In that paper, no operational semantics or performance evaluation methods for GSPA were presented. Later, in [41], generalized semi-Markov processes (GSMPs) were extracted from GSESs to analyze performance.

In [65], generalized Stochastic π-calculus (Sπ) with general continuous distributions of activity delays was defined. It has a proved operational semantics with transitions labeled by encodings of their deduction trees. No well-established underlying performance model for this version of Sπ was described.

In [15, 14], Generalized Semi-Markovian Process Algebra (GSMPA) was developed with ST-operational semantics and non-Markovian action delays. The performance analysis in GSMPA is accomplished via GSMPs.

Again, the first fundamental difference between dtsiPBC and the calculi GSPA, Sπ and GSMPA is that dtsiPBC is based on PBC, whereas GSPA is an extension of Process Algebra (PA) from [17], Sπ extends π-calculus [59] and GSMPA is an enrichment of EMPA. Therefore, both GSPA and GSMPA have *prefixing*, *choice* (*alternative* composition), *parallel* composition, *renaming* (*relabeling*) and *hiding* (*abstraction*) operations, but only GSMPA permits *constants*. Unlike dtsiPBC, GSPA has neither iteration or recursion, GSMPA allows only *recursive* definitions, whereas Sπ additionally has operations to specify *mobility*. Note also that GSPA, Sπ and GSMPA do not specify instant events or activities while dtsiPBC has immediate multiactions.

The second significant difference is that 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. As a consequence, 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 are often annotated with an additional information to keep concurrency data. The transition labels in the operational semantics of Sπ encode the action causality information and allow one to derive the enabling relations and the firing distributions of concurrent transitions from the transition sequences. At the same time, abstracting from stochastic delays leads to the classical early interleaving semantics of π-calculus. 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 are the interleaving ones while the termination transitions are the step ones. As a result, the decorated interleaving / step transition systems are obtained. dtsiPBC has an SPNs-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

In [1], a class of compositional DTSPNs with generally distributed discrete time transition delays was proposed, called dts-nets. The denotational semantics of a stochastic extension (we call it stochastic ACP or sACP) of a subset of Algebra of Communicating Processes (ACP) [5] can be constructed via dts-nets. There are two types of transitions in dts-nets: immediate (timeless) ones, with zero delays, and time ones, whose delays are random variables having general discrete distributions. The top-down synthesis of dts-nets consists in the substitution of their transitions by blocks (dts-subnets) corresponding to the sequence, choice, parallelism and iteration operators. It was explained how to calculate the throughput time of dts-nets using the service time (defined as 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 approach was implemented.

In [52, 53], an SPA called Theory of Communicating Processes with discrete stochastic time (TCP^{dst}) was introduced. Its actions have (deterministic) discrete real time delays (including zero time delays) or 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

Table 8: 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)

graphs which are essentially the reward transition systems with probabilistic states having finite number of outgoing probabilistic 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.

The first difference between dtsiPBC and the algebras sACP and TCP^{dst} is that dtsiPBC is based on PBC, but sACP and TCP^{dst} are the extensions of ACP. sACP has taken from ACP only *sequence*, *choice*, *parallelism* and *iteration* operations, whereas dtsiPBC has additionally relabeling, restriction and synchronization ones, inherited from PBC. In TCP^{dst}, besides standard action *prefixing*, *alternative*, *parallel* composition, *encapsulation* (similar to *restriction*) and *recursive* variables, there are also *timed delay prefixing*, *dependent delays scope* and the *maximal time progress* operators, which are new both for ACP and dtsiPBC.

The second difference is that zero or geometrically distributed delays are associated with process states in dtsiPBC, unlike zero or generally distributed discrete time delays of actions in sACP and deterministic or generally distributed stochastic delays of actions in TCP^{dst}. Neither formal syntax nor operational semantics for sACP are defined and it is not explained how to derive Markov chains from the algebraic expressions or the corresponding dts-nets to analyze performance. It is 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. In addition, no denotational semantics was defined for TCP^{dst}.

Table 8 summarizes the SPAs comparison above and that from Section 1 (the calculi sPBC, gsPBC and dtsPBC), 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 in parentheses 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 Sπ 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.

An existence of the analytical solution also permits to interpret quantitative values (rates, probabilities 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) [45]. Note that DTMCs whose transition probabilities are parameters were introduced in [23]. CTMCs with the transition rates treated as parameters were investigated in [31]. 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 [45], 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^*, $PM^\diamond$ define probability distributions, hence, they always return probabilities belonging to $(0; 1]$ for any parameters from $(0; 1)$.

11.2. Application area

From the application viewpoint, one considers what kind of systems are more appropriate to be modeled and analyzed within SPAs. MTIPP and PEPA are well-suited for the interleaving continuous time systems such that the activity rates or the average sojourn time in the states are known in advance and exponential distribution approximates well the activity delay distributions, whereas EMPA 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 the continuous time systems with a capability to keep the activity causality information, and with the known activity delay distributions, which cannot be approximated accurately by exponential distribution, while Sπ 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 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 instant probabilistic choice or urgency, hence, dtsiPBC can be taken as a non-interleaving discrete time counterpart of EMPA.

11.3. Concurrency interpretation

One can see that 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. 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, which has attracted special attention last years. Nevertheless, not so many formal stochastic models of parallel systems were defined whose underlying stochastic processes are based on DTMCs. As mentioned in [25], such models are more difficult to analyze, since a lot of 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. As observed in [38], 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 stochastic processes have the additional transitions corresponding to the simultaneous execution of concurrent (i.e. non-synchronized) activities. The transitions of that kind allow 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 should be equal to 1, are distributed among more possibilities. Therefore, the systems with parallel stochastic processes usually have smaller average run-through. In case 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.

11.4. Advantages of dtsiPBC

Thus, the main advantages of dtsiPBC are the flexible multiaction 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 the concurrent step operational semantics based on labeled probabilistic transition systems and the 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 with respect to 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 the shared memory system has been presented as 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, such as business processes, neural and transportation networks, computer and communication systems, whose discrete states change with a global time tick, as well as for those, in which the distributed architecture or the concurrency level should be preserved while modeling and analysis (remember that, in step semantics, we have additional transitions due to concurrent executions).

Future work will consist in constructing a congruence for dtsiPBC, i.e. the equivalence that withstands application of all operations of the algebra. The first possible candidate is a stronger version of $\underline{\leftrightarrow}_{ss}$ defined via transition systems equipped with two extra transitions skip and redo like those from [48]. We also plan to extend the calculus with deterministically timed multiactions having a fixed time delay (including the zero one which is the case of immediate multiactions) to enhance expressiveness of the calculus and to extend application area of the associated analysis techniques. Moreover, recursion could be added to dtsiPBC to increase further specification power of the algebra.

Acknowledgements. The first author thanks Eike Best for the qualified consideration, encouraging discussions and many valuable advices related to the subject of the paper.

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 (2000) 237–255. http://tmitwww.tm.tue.nl/staff/hreijers/H.A. Reijers Bestanden/Statistica.pdf.

[2] C. Autant, P. Schnoebelen, Place bisimulations in Petri nets, in: Proc. 13^{th} ICATPN 1992, volume 616 of *Lect. Notes Comp. Sci.*, Springer, 1992, pp. 45–61.
[3] G. Balbo, Introduction to stochastic Petri nets, in: Proc. 1^{st} EEF/Euro Summer School of Trends in Comp. Sci. 2000, volume 2090 of *Lect. Notes Comp. Sci.*, Springer, 2001, pp. 84–155.
[4] G. Balbo, Introduction to generalized stochastic Petri nets, in: Proc. 7^{th} SFM 2007, volume 4486 of *Lect. Notes Comp. Sci.*, Springer, 2007, pp. 83–131.
[5] J.A. Bergstra, J.W. Klop, Algebra of communicating processes with abstraction, Theor. Comput. Sci. 37 (1985) 77–121.
[6] M. Bernardo, Theory and application of extended Markovian process algebra, University of Bologna, Italy, 1999. Ph. D. thesis, 276 pages, http://www.sti.uniurb.it/bernardo/documents/phdthesis.pdf.
[7] M. Bernardo, A survey of Markovian behavioral equivalences, in: Proc. 7^{th} SFM 2007, volume 4486 of *Lect. Notes Comp. Sci.*, Springer, 2007, pp. 180–219. http://www.sti.uniurb.it/bernardo/documents/sfm07pe.pdf.
[8] M. Bernardo, M. Bravetti, Reward based congruences: can we aggregate more?, in: Proc. PAPM-PROBMIV 2001, volume 2165 of *Lect. Notes Comp. Sci.*, Springer, 2001, pp. 136–151. http://www.cs.unibo.it/~bravetti/papers/papm01b.ps.
[9] 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 (1998) 83–154. http://www.sti.uniurb.it/bernardo/documents/ic144.pdf.
[10] M. Bernardo, R. Gorrieri, A tutorial on EMPA: a theory of concurrent processes with nondeterminism, priorities, probabilities and time, Theor. Comput. Sci. 202 (1998) 1–54. http://www.sti.uniurb.it/bernardo/documents/tcs202.pdf.
[11] E. Best, R. Devillers, J.G. Hall, The box calculus: a new causal algebra with multi-label communication, in: Advances in Petri Nets 1992, volume 609 of *Lect. Notes Comp. Sci.*, Springer, 1992, pp. 21–69.
[12] E. Best, R. Devillers, M. Koutny, Petri net algebra, EATCS Monographs on Theor. Comput. Sci., Springer, 2001. 378 pages.
[13] E. Best, M. Koutny, A refined view of the box algebra, in: Proc. 16^{th} ICATPN 1995, volume 935 of *Lect. Notes Comp. Sci.*, Springer, 1995, pp. 1–20. http://parsys.informatik.uni-oldenburg.de/~best/publications/pn95.ps.gz.
[14] M. Bravetti, Specification and analysis of stochastic real-time systems, University of Bologna, Italy, 2002. Ph. D. thesis, 432 pages, http://www.cs.unibo.it/~bravetti/papers/phdthesis.ps.gz.
[15] M. Bravetti, M. Bernardo, R. Gorrieri, Towards performance evaluation with general distributions in process algebras, in: Proc. 9^{th} CONCUR 1998, volume 1466 of *Lect. Notes Comp. Sci.*, Springer, 1998, pp. 405–422. http://www.cs.unibo.it/~bravetti/papers/concur98.ps.
[16] E. Brinksma, H. Hermanns, Process algebra and Markov chains, in: Proc. 1^{st} EEF/Euro Summer School of Trends in Comp. Sci. 2000, volume 2090 of *Lect. Notes Comp. Sci.*, Springer, 2001, pp. 183–231.
[17] E. Brinksma, J.P. Katoen, R. Langerak, D. Latella, A stochastic causality-based process algebra, Comp. J. 38 (1995) 552–565. http://eprints.eemcs.utwente.nl/6387/01/552.pdf.
[18] P. Buchholz, Markovian process algebra: composition and equivalence, in: Proc. 2^{nd} Int. Workshop on Process Algebras and Performance Modelling (PAPM) 1994, number 27 in Arbeitsberichte des IMMD, University of Erlangen, Germany, 1994, pp. 11–30.
[19] P. Buchholz, A notion of equivalence for stochastic Petri nets, in: Proc. 16^{th} ICATPN 1995, volume 935 of *Lect. Notes Comp. Sci.*, Springer, 1995, pp. 161–180.
[20] P. Buchholz, Iterative decomposition and aggregation of labeled GSPNs, in: Proc. 19^{th} ICATPN 1998, volume 1420 of *Lect. Notes Comp. Sci.*, Springer, 1998, pp. 226–245.
[21] 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. Novosibirsk, Russia, http://itar.iis.nsk.su/files/itar/pages/spnpancc.pdf.
[22] I. Christoff, Testing equivalence and fully abstract models of probabilistic processes, in: Proc. 1^{st} CONCUR 1990, volume 458 of *Lect. Notes Comp. Sci.*, Springer, 1990, pp. 126–140.
[23] C. Daws, Symbolic and parametric model checking of discrete-time Markov chains, in: Proc. 1^{st} ICTAC 2004, volume 3407 of *Lect. Notes Comp. Sci.*, Springer, 2005, pp. 280–294.
[24] S. Derisavi, H. Hermanns, W.H. Sanders, Optimal state-space lumping of Markov chains, Information Processing Letters 87 (2003) 309–315.
[25] J.M. Fourneau, Collaboration of discrete-time Markov chains: Tensor and product form, Performance Evaluation 67 (2010) 779–796.
[26] S. Gilmore, J. Hillston, L. Kloul, M. Ribaudo, PEPA nets: a structured performance modelling formalism, Performance Evaluation 54 (2003) 79–104. http://www.dcs.ed.ac.uk/pepa/pepanetsJournal.pdf.
[27] R.J. van Glabbeek, The linear time – branching time spectrum II: the semantics of sequential systems with silent moves. Extended abstract, in: Proc. 4^{th} CONCUR 1993, volume 715 of *Lect. Notes Comp. Sci.*, Springer, 1993, pp. 66–81.
[28] R.J. van Glabbeek, S.A. Smolka, B. Steffen, Reactive, generative, and stratified models of probabilistic processes, Information and Computation 121 (1995) 59–80. http://boole.stanford.edu/pub/prob.ps.gz.
[29] N. Götz, U. Herzog, M. Rettelbach, Multiprocessor and distributed system design: the integration of functional specification and performance analysis using stochastic process algebras, in: Proc. 16^{th} Performance 1993, volume 729 of *Lect. Notes Comp. Sci.*, Springer, 1993, pp. 121–146.
[30] 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.
[31] T. Han, J.P. Katoen, A. Mereacre, Approximate parameter synthesis for probabilistic time-bounded reachability, in: Proc. 29^{th} IEEE Real-Time Systems Symposium (RTSS) 2008, IEEE Computer Society Press, New York, USA, 2008, pp. 173–182.
[32] B.R. Haverkort, Markovian models for performance and dependability evaluation, in: Proc. 1^{st} EEF/Euro Summer School of Trends in Comp. Sci. 2000, volume 2090 of *Lect. Notes Comp. Sci.*, Springer, 2001, pp. 38–83. http://www-i2.informatik.rwth-aachen.de/Teaching/Seminar/VOSS2005/have01.pdf.
[33] 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 (2013) 97–118. http://pubs.doc.ic.ac.uk/fluid-unified-stochastic-probes/fluid-unified-stochastic-probes.pdf.
[34] H. Hermanns, M. Rettelbach, Syntax, semantics, equivalences and axioms for MTIPP, in: Proc. 2^{nd} Int. Workshop on Process Algebras and Performance Modelling (PAPM) 1994, number 27 in Arbeitsberichte des IMMD, University of Erlangen, Germany, 1994, pp. 71–88.

[35] J. Hillston, The nature of synchronisation, in: Proc. 2^{nd} Int. Workshop on Process Algebras and Performance Modelling (PAPM) 1994, number 27 in Arbeitsberichte des IMMD, University of Erlangen, Germany, 1994, pp. 51–70. http://www.dcs.ed.ac.uk/pepa/synchronisation.pdf.

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

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

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

[39] C.C. Jou, S.A. Smolka, Equivalences, congruences and complete axiomatizations for probabilistic processes, in: Proc. 1^{st} CONCUR 1990, volume 458 of *Lect. Notes Comp. Sci.*, Springer, 1990, pp. 367–383.

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

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

[42] J.P. Katoen, P.R. D'Argenio, General distributions in process algebra, in: Proc. 1^{st} EEF/Euro Summer School of Trends in Comp. Sci. 2000, volume 2090 of *Lect. Notes Comp. Sci.*, Springer, 2001, pp. 375–429.

[43] M. Koutny, A compositional model of time Petri nets, in: Proc. 21^{st} ICATPN 2000, volume 1825 of *Lect. Notes Comp. Sci.*, Springer, 2000, pp. 303–322.

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

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

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

[47] H. Macià, V. Valero, D. Cazorla, F. Cuartero, Introducing the iteration in sPBC, in: Proc. 24^{th} FORTE 2004, volume 3235 of *Lect. Notes Comp. Sci.*, Springer, 2004, pp. 292–308. http://www.info-ab.uclm.es/retics/publications/2004/forte04.pdf.

[48] H. Macià, V. Valero, F. Cuartero, D. de-Frutos, A congruence relation for sPBC, Formal Methods in System Design 32 (2008) 85–128. Springer, The Netherlands.

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

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

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

[52] J. Markovski, E.P. de Vink, Extending timed process algebra with discrete stochastic time, in: Proc. 12^{th} AMAST 2008, volume 5140 of *Lect. Notes Comp. Sci.*, Springer, 2008, pp. 268–283.

[53] J. Markovski, E.P. de Vink, Performance evaluation of distributed systems based on a discrete real- and stochastic-time process algebra, Fundamenta Informaticae 95 (2009) 157–186. IOS Press, Amsterdam, The Netherlands.

[54] O. Marroquín, D. de-Frutos, Extending the Petri box calculus with time, in: Proc. 22^{nd} ICATPN 2001, volume 2075 of *Lect. Notes Comp. Sci.*, Springer, 2001, pp. 303–322.

[55] M.A. Marsan, Stochastic Petri nets: an elementary introduction, in: Advances in Petri Nets 1989, volume 424 of *Lect. Notes Comp. Sci.*, Springer, 1990, pp. 1–29.

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

[57] P.M. Merlin, D.J. Farber, Recoverability of communication protocols: implications of a theoretical study, IEEE Transactions on Communications 24 (1976) 1036–1043.

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

[59] R.A.J. Milner, J. Parrow, D. Walker, A calculus of mobile processes (i and ii), Information and Computation 100 (1992) 1–77.

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

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

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

[63] A. Niaouris, An algebra of Petri nets with arc-based time restrictions, in: Proc. 1^{st} ICTAC 2004, volume 3407 of *Lect. Notes Comp. Sci.*, Springer, 2005, pp. 447–462.

[64] R. Paige, R.E. Tarjan, Three partition refinement algorithms, SIAM J. Comput. 16 (1987) 973–989.

[65] C. Priami, Stochastic π-calculus with general distributions, in: Proc. 4^{th} Int. Workshop on Process Algebra and Performance Modelling (PAPM) 1996, CLUT Press, Torino, Italy, 1996, pp. 41–57.

[66] C. Ramchandani, Perfomance evaluation of asynchronous concurrent systems by timed Petri nets, Massachusetts Institute of Technology, Cambridge, USA, 1973. Ph. D. thesis.

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

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

[69] 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. NCC Publisher, Novosibirsk, Russia, http://itar.iis.nsk.su/files/itar/pages/dtsitncc.pdf.

[70] I.V. Tarasyuk, Stochastic Petri box calculus with discrete time, Fundamenta Informaticae 76 (2007) 189–218. IOS Press, Amsterdam, The

Netherlands.
[71] I.V. Tarasyuk, Investigating equivalence relations in dtsPBC, Berichte aus dem Department für Informatik 5/08, Carl von Ossietzky Universität Oldenburg, Germany, 2008. 57 pages, http://itar.iis.nsk.su/files/itar/pages/dtspbcit_cov.pdf.
[72] I.V. Tarasyuk, H. Macià, V. Valero, Discrete time stochastic Petri box calculus with immediate multiactions, Technical Report DIAB-10-03-1, Department of Computer Systems, High School of Computer Science Engineering, University of Castilla-La Mancha, Albacete, Spain, 2010. 25 pages, http://itar.iis.nsk.su/files/itar/pages/dtsipbc.pdf.
[73] I.V. Tarasyuk, H. Macià, V. Valero, Discrete time stochastic Petri box calculus with immediate multiactions dtsiPBC, in: Proc. 6^{th} Int. Workshop on Practical Applications of Stochastic Modelling (PASM) 2012, London, UK, volume 296 of *Electronic Notes in Theor. Comput. Sci.*, Elsevier, 2013, pp. 229–252. http://itar.iis.nsk.su/files/itar/pages/dtsipbcentcs.pdf.
[74] R. Wimmer, S. Derisavi, H. Hermanns, Symbolic partition refinement with automatic balancing of time and space, Performance Evaluation 67 (2010) 816–836.
[75] R. Zijal, G. Ciardo, G. Hommel, Discrete deterministic and stochastic Petri nets, in: Proc. 9^{th} ITG/GI Professional Meeting "Messung, Modellierung und Bewertung von Rechen- und Kommunikationssystemen" (MMB) 1997, Freiberg, Germany, VDE-Verlag, Berlin, Germany, 1997, pp. 103–117. http://www.cs.ucr.edu/~ciardo/pubs/1997MMB-DDSPN.pdf.
[76] A. Zimmermann, J. Freiheit, G. Hommel, Discrete time stochastic Petri nets for modeling and evaluation of real-time systems, in: Proc. 9^{th} Int. Workshop on Parallel and Distributed Real Time Systems (WPDRTS) 2001, San Francisco, USA, pp. 282–286. http://pdv.cs.tu-berlin.de/~azi/texte/WPDRTS01.pdf.

Appendix A. Proofs

Appendix A.1. Proof of Proposition 6.2

Like it has been done for strong equivalence in Proposition 8.2.1 from [36], we shall prove the following fact about step stochastic bisimulation. Let us have $\forall j \in \mathcal{J},\ \mathcal{R}_j : G\underline{\leftrightarrow}_{ss}G'$ 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 $\mathcal{R} : G\underline{\leftrightarrow}_{ss}G'$.

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 $\mathcal{R} : G\underline{\leftrightarrow}_{ss}G'$ by induction on n.

It is clear that $\forall j \in \mathcal{J},\ \mathcal{R}_j : G\underline{\leftrightarrow}_{ss}G'$ 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 $\forall \mathcal{H} \in (DR(G) \cup DR(G'))/_{\mathcal{R}},\ \forall A \in \mathbb{N}_{fin}^{\mathcal{L}},\ PM_A(s_1, \mathcal{H}) = PM_A(s_2, \mathcal{H})$.

- $n = 1$

  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 $\mathcal{R}_j : G\underline{\leftrightarrow}_{ss}G'$, we get $\forall \mathcal{H} \in (DR(G) \cup DR(G'))/_{\mathcal{R}},\ \forall A \in \mathbb{N}_{fin}^{\mathcal{L}}$,

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

- $n \to n + 1$

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

  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, $\forall \mathcal{H} \in (DR(G) \cup DR(G'))/_{\mathcal{R}},\ \forall A \in \mathbb{N}_{fin}^{\mathcal{L}}$,

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

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

Appendix A.2. Proof of Proposition 8.1

By Proposition 6.1, $(DR(G)\cup DR(G'))/_{\mathcal{R}} = ((DR_T(G)\cup DR_T(G'))/_{\mathcal{R}}) \uplus ((DR_V(G)\cup DR_V(G'))/_{\mathcal{R}})$. Hence, $\forall\mathcal{H}\in (DR(G)\cup DR(G'))/_{\mathcal{R}}$, all states from $\mathcal{H}$ are tangible, when $\mathcal{H}\in (DR_T(G)\cup DR_T(G'))/_{\mathcal{R}}$, or all of them are vanishing, when $\mathcal{H}\in (DR_V(G)\cup DR_V(G'))/_{\mathcal{R}}$.

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

By Proposition 5.2, $\forall s\in DR_T(G),\ \varphi(s)=\frac{\psi(s)}{\sum_{\tilde{s}\in DR_T(G)}\psi(\tilde{s})}$ and $\forall s'\in DR_T(G'),\ \varphi'(s')=\frac{\psi'(s')}{\sum_{\tilde{s}'\in DR_T(G')}\psi'(\tilde{s}')}$, where ψ and ψ' are the steady-state PMFs for $DTMC(G)$ and $DTMC(G')$, respectively. Thus, $\forall\mathcal{H},\widetilde{\mathcal{H}}\in (DR_T(G)\cup DR_T(G'))/_{\mathcal{R}}$,
$\sum_{s\in\mathcal{H}\cap DR(G)}\varphi(s)=\sum_{s\in\mathcal{H}\cap DR_T(G)}\varphi(s)=\sum_{s\in\mathcal{H}\cap DR_T(G)}\left(\frac{\psi(s)}{\sum_{\tilde{s}\in DR_T(G)}\psi(\tilde{s})}\right)=\frac{\sum_{s\in\mathcal{H}\cap DR_T(G)}\psi(s)}{\sum_{\tilde{s}\in DR_T(G)}\psi(\tilde{s})}=\frac{\sum_{s\in\mathcal{H}\cap DR_T(G)}\psi(s)}{\sum_{\widetilde{\mathcal{H}}}\sum_{\tilde{s}\in\widetilde{\mathcal{H}}\cap DR_T(G)}\psi(\tilde{s})}$ and
$\sum_{s'\in\mathcal{H}\cap DR(G')}\varphi'(s')=\sum_{s'\in\mathcal{H}\cap DR_T(G')}\varphi'(s')=\sum_{s'\in\mathcal{H}\cap DR_T(G')}\left(\frac{\psi'(s')}{\sum_{\tilde{s}'\in DR_T(G')}\psi'(\tilde{s}')}\right)=\frac{\sum_{s'\in\mathcal{H}\cap DR_T(G')}\psi'(s')}{\sum_{\tilde{s}'\in DR_T(G')}\psi'(\tilde{s}')}=\frac{\sum_{s'\in\mathcal{H}\cap DR_T(G')}\psi'(s')}{\sum_{\widetilde{\mathcal{H}}}\sum_{\tilde{s}'\in\widetilde{\mathcal{H}}\cap DR_T(G')}\psi'(\tilde{s}')}$.

It remains to prove that $\forall\mathcal{H}\in (DR_T(G)\cup DR_T(G'))/_{\mathcal{R}},\ \sum_{s\in\mathcal{H}\cap DR_T(G)}\psi(s)=\sum_{s'\in\mathcal{H}\cap DR_T(G')}\psi'(s')$. Since $(DR(G)\cup DR(G'))/_{\mathcal{R}} = ((DR_T(G)\cup DR_T(G'))/_{\mathcal{R}}) \uplus ((DR_V(G)\cup DR_V(G'))/_{\mathcal{R}})$, 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 $\forall\widetilde{\mathcal{H}}\in (DR(G)\cup DR(G'))/_{\mathcal{R}},\ \forall A\in I\!N_{fin}^{\mathcal{L}},\ s_1\xrightarrow{A}_{\mathcal{P}}\widetilde{\mathcal{H}}\ \Leftrightarrow\ s_2\xrightarrow{A}_{\mathcal{P}}\widetilde{\mathcal{H}}$. Therefore, $PM(s_1,\widetilde{\mathcal{H}})=\sum_{\{\Upsilon\mid\exists\tilde{s}_1\in\widetilde{\mathcal{H}},\ s_1\xrightarrow{\Upsilon}\tilde{s}_1\}}PT(\Upsilon,s_1)=\sum_{A\in I\!N_{fin}^{\mathcal{L}}}\sum_{\{\Upsilon\mid\exists\tilde{s}_1\in\widetilde{\mathcal{H}},\ s_1\xrightarrow{\Upsilon}\tilde{s}_1,\ \mathcal{L}(\Upsilon)=A\}}PT(\Upsilon,s_1)=\sum_{A\in I\!N_{fin}^{\mathcal{L}}}PM_A(s_1,\widetilde{\mathcal{H}})=\sum_{A\in I\!N_{fin}^{\mathcal{L}}}PM_A(s_2,\widetilde{\mathcal{H}})=\sum_{A\in I\!N_{fin}^{\mathcal{L}}}\sum_{\{\Upsilon\mid\exists\tilde{s}_2\in\widetilde{\mathcal{H}},\ s_2\xrightarrow{\Upsilon}\tilde{s}_2,\ \mathcal{L}(\Upsilon)=A\}}PT(\Upsilon,s_2)=\sum_{\{\Upsilon\mid\exists\tilde{s}_2\in\widetilde{\mathcal{H}},\ s_2\xrightarrow{\Upsilon}\tilde{s}_2\}}PT(\Upsilon,s_2)=PM(s_2,\widetilde{\mathcal{H}})$. Since we have the previous equality for all $s_1,s_2\in\mathcal{H}$, we can denote $PM(\mathcal{H},\widetilde{\mathcal{H}})=PM(s_1,\widetilde{\mathcal{H}})=PM(s_2,\widetilde{\mathcal{H}})$. Note that 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 $DR(G')$.

  By induction hypothesis, $\sum_{s\in\mathcal{H}\cap DR(G)}\psi[k](s)=\sum_{s'\in\mathcal{H}\cap DR(G')}\psi'[k](s')$. Further,
  $\sum_{\tilde{s}\in\widetilde{\mathcal{H}}\cap DR(G)}\psi[k+1](\tilde{s})=\sum_{\tilde{s}\in\widetilde{\mathcal{H}}\cap DR(G)}\sum_{s\in DR(G)}\psi[k](s)PM(s,\tilde{s})=\sum_{s\in DR(G)}\sum_{\tilde{s}\in\widetilde{\mathcal{H}}\cap DR(G)}\psi[k](s)PM(s,\tilde{s})=$
  $\sum_{s\in DR(G)}\psi[k](s)\sum_{\tilde{s}\in\widetilde{\mathcal{H}}\cap DR(G)}PM(s,\tilde{s})=\sum_{\mathcal{H}}\sum_{s\in\mathcal{H}\cap DR(G)}\psi[k](s)\sum_{\tilde{s}\in\widetilde{\mathcal{H}}\cap DR(G)}PM(s,\tilde{s})=$
  $\sum_{\mathcal{H}}\sum_{s\in\mathcal{H}\cap DR(G)}\psi[k](s)\sum_{\tilde{s}\in\widetilde{\mathcal{H}}\cap DR(G)}\sum_{\{\Upsilon\mid s\xrightarrow{\Upsilon}\tilde{s}\}}PT(\Upsilon,s)=\sum_{\mathcal{H}}\sum_{s\in\mathcal{H}\cap DR(G)}\psi[k](s)\sum_{\{\Upsilon\mid\exists\tilde{s}\in\widetilde{\mathcal{H}}\cap DR(G),\ s\xrightarrow{\Upsilon}\tilde{s}\}}PT(\Upsilon,s)=$
  $\sum_{\mathcal{H}}\sum_{s\in\mathcal{H}\cap DR(G)}\psi[k](s)PM(s,\widetilde{\mathcal{H}})=\sum_{\mathcal{H}}\sum_{s\in\mathcal{H}\cap DR(G)}\psi[k](s)PM(\mathcal{H},\widetilde{\mathcal{H}})=$
  $\sum_{\mathcal{H}}PM(\mathcal{H},\widetilde{\mathcal{H}})\sum_{s\in\mathcal{H}\cap DR(G)}\psi[k](s)=\sum_{\mathcal{H}}PM(\mathcal{H},\widetilde{\mathcal{H}})\sum_{s'\in\mathcal{H}\cap DR(G')}\psi'[k](s')=$
  $\sum_{\mathcal{H}}\sum_{s'\in\mathcal{H}\cap DR(G')}\psi'[k](s')PM(\mathcal{H},\widetilde{\mathcal{H}})=\sum_{\mathcal{H}}\sum_{s'\in\mathcal{H}'\cap DR(G')}\psi'[k](s')PM(s',\widetilde{\mathcal{H}})=$
  $\sum_{\mathcal{H}}\sum_{s'\in\mathcal{H}\cap DR(G')}\psi'[k](s')\sum_{\{\Upsilon\mid\exists\tilde{s}'\in\widetilde{\mathcal{H}}\cap DR(G'),\ s'\xrightarrow{\Upsilon}\tilde{s}'\}}PT(\Upsilon,s')=$
  $\sum_{\mathcal{H}}\sum_{s'\in\mathcal{H}\cap DR(G')}\psi'[k](s')\sum_{\tilde{s}'\in\widetilde{\mathcal{H}}\cap DR(G')}\sum_{\{\Upsilon\mid\exists\tilde{s}',\ s'\xrightarrow{\Upsilon}\tilde{s}'\}}PT(\Upsilon,s')=$
  $\sum_{\mathcal{H}}\sum_{s'\in\mathcal{H}\cap DR(G')}\psi'[k](s')\sum_{\tilde{s}'\in\widetilde{\mathcal{H}}\cap DR(G')}PM(s',\tilde{s}')=\sum_{s'\in DR(G')}\psi'[k](s')\sum_{\tilde{s}'\in\widetilde{\mathcal{H}}\cap DR(G')}PM(s',\tilde{s}')=$
  $\sum_{s'\in DR(G')}\sum_{\tilde{s}'\in\widetilde{\mathcal{H}}\cap DR(G')}\psi'[k](s')PM(s',\tilde{s}')=\sum_{\tilde{s}'\in\widetilde{\mathcal{H}}\cap DR(G')}\sum_{s'\in DR(G')}\psi'[k](s')PM(s',\tilde{s}')=$
  $\sum_{\tilde{s}'\in\widetilde{\mathcal{H}}\cap DR(G')}\psi'[k+1](\tilde{s}')$. $\square$

Appendix A.3. Proof of Theorem 8.1

Let $\mathcal{H}\in (DR(G)\cup DR(G'))/_{\mathcal{R}}$ and $s,\bar{s}\in\mathcal{H}$. We have $\forall\widetilde{\mathcal{H}}\in (DR(G)\cup DR(G'))/_{\mathcal{R}},\ \forall A\in I\!N_{fin}^{\mathcal{L}},\ s\xrightarrow{A}_{\mathcal{P}}\widetilde{\mathcal{H}}\ \Leftrightarrow\ \bar{s}\xrightarrow{A}_{\mathcal{P}}\widetilde{\mathcal{H}}$. The previous equality is valid for all $s,\bar{s}\in\mathcal{H}$, hence, we can rewrite it as $\mathcal{H}\xrightarrow{A}_{\mathcal{P}}\widetilde{\mathcal{H}}$ and denote

$PM_A(\mathcal{H},\widetilde{\mathcal{H}}) = PM_A(s,\widetilde{\mathcal{H}}) = PM_A(\bar{s},\widetilde{\mathcal{H}})$. Note that 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 is true for $DR(G')$.

Let $\Sigma = A_1 \cdots A_n$ be a derived step trace of G and G'. Then $\exists \mathcal{H}_0,\dots,\mathcal{H}_n \in (DR(G) \cup DR(G'))/_{\mathcal{R}},\ \mathcal{H}_0 \xrightarrow{A_1}_{\mathcal{P}_1} \mathcal{H}_1 \xrightarrow{A_2}_{\mathcal{P}_2} \cdots \xrightarrow{A_n}_{\mathcal{P}_n} \mathcal{H}_n$. Now we intend to 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 $\mathcal{H}_1,\dots,\mathcal{H}_n$ is equal to the product of $\mathcal{P}_1,\dots,\mathcal{P}_n$:

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

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

- $n = 1$

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

- $n \to n+1$

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

Let $s_0, \bar{s}_0 \in \mathcal{H}_0$. We have $PT(A_1 \cdots A_n, s_0) = \sum_{\{\Upsilon_1,\dots,\Upsilon_n \mid s_0 \xrightarrow{\Upsilon_1} \cdots \xrightarrow{\Upsilon_n} s_n,\ \mathcal{L}(\Upsilon_i)=A_i,\ (1 \le i \le n)\}} \prod_{i=1}^{n} PT(\Upsilon_i, s_{i-1}) =$
$\sum_{\mathcal{H}_1,\dots,\mathcal{H}_n} \sum_{\{\Upsilon_1,\dots,\Upsilon_n \mid s_0 \xrightarrow{\Upsilon_1} \cdots \xrightarrow{\Upsilon_n} s_n,\ \mathcal{L}(\Upsilon_i)=A_i,\ s_i \in \mathcal{H}_i\ (1 \le i \le n)\}} \prod_{i=1}^{n} PT(\Upsilon_i, s_{i-1}) = \sum_{\mathcal{H}_1,\dots,\mathcal{H}_n} \prod_{i=1}^{n} PM_{A_i}(\mathcal{H}_{i-1},\mathcal{H}_i) =$
$\sum_{\mathcal{H}_1,\dots,\mathcal{H}_n} \sum_{\{\bar{\Upsilon}_1,\dots,\bar{\Upsilon}_n \mid \bar{s}_0 \xrightarrow{\bar{\Upsilon}_1} \cdots \xrightarrow{\bar{\Upsilon}_n} \bar{s}_n,\ \mathcal{L}(\bar{\Upsilon}_i)=A_i,\ \bar{s}_i \in \mathcal{H}_i\ (1 \le i \le n)\}} \prod_{i=1}^{n} PT(\bar{\Upsilon}_i, \bar{s}_{i-1}) =$
$\sum_{\{\bar{\Upsilon}_1,\dots,\bar{\Upsilon}_n \mid \bar{s}_0 \xrightarrow{\bar{\Upsilon}_1} \cdots \xrightarrow{\bar{\Upsilon}_n} \bar{s}_n,\ \mathcal{L}(\bar{\Upsilon}_i)=A_i,\ (1 \le i \le n)\}} \prod_{i=1}^{n} PT(\bar{\Upsilon}_i, \bar{s}_{i-1}) = PT(A_1 \cdots A_n, \bar{s}_0)$.
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 8.1, $\sum_{s \in \mathcal{H} \cap DR(G)} \varphi(s) = \sum_{s' \in \mathcal{H} \cap DR(G')} \varphi'(s')$. Now we can complete the proof:
$\sum_{s \in \mathcal{H} \cap DR(G)} \varphi(s) PT(\Sigma, s) = \sum_{s \in \mathcal{H} \cap DR(G)} \varphi(s) PT(\Sigma, \mathcal{H}) = PT(\Sigma, \mathcal{H}) \sum_{s \in \mathcal{H} \cap DR(G)} \varphi(s) =$
$PT(\Sigma, \mathcal{H}) \sum_{s' \in \mathcal{H} \cap DR(G')} \varphi'(s') = \sum_{s' \in \mathcal{H} \cap DR(G')} \varphi'(s') PT(\Sigma, \mathcal{H}) = \sum_{s' \in \mathcal{H} \cap DR(G')} \varphi'(s') PT(\Sigma, s')$. □

Appendix A.4. Proof of Proposition 8.2

Let us present two facts, which will be used in the proof.

1. By Proposition 6.1, $(DR(G) \cup DR(G'))/_{\mathcal{R}} = ((DR_T(G) \cup DR_T(G'))/_{\mathcal{R}}) \uplus ((DR_V(G) \cup DR_V(G'))/_{\mathcal{R}})$. Hence, $\forall \mathcal{H} \in (DR(G) \cup DR(G'))/_{\mathcal{R}}$, all states from $\mathcal{H}$ are tangible, when $\mathcal{H} \in (DR_T(G) \cup DR_T(G'))/_{\mathcal{R}}$, or all of them are vanishing, when $\mathcal{H} \in (DR_V(G) \cup DR_V(G'))/_{\mathcal{R}}$.
2. Let $\mathcal{H} \in (DR(G) \cup DR(G'))/_{\mathcal{R}}$ and $s_1, s_2 \in \mathcal{H}$. We have $\forall \widetilde{\mathcal{H}} \in (DR(G) \cup DR(G'))/_{\mathcal{R}},\ \forall A \in \mathbb{N}_{fin}^{\mathcal{L}},\ s_1 \xrightarrow{A}_{\mathcal{P}} \widetilde{\mathcal{H}} \Leftrightarrow s_2 \xrightarrow{A}_{\mathcal{P}} \widetilde{\mathcal{H}}$. Therefore, $PM(s_1,\widetilde{\mathcal{H}}) = \sum_{\{\Upsilon \mid \exists \tilde{s}_1 \in \widetilde{\mathcal{H}},\ s_1 \xrightarrow{\Upsilon} \tilde{s}_1\}} PT(\Upsilon, s_1) = \sum_{A \in \mathbb{N}_{fin}^{\mathcal{L}}} \sum_{\{\Upsilon \mid \exists \tilde{s}_1 \in \widetilde{\mathcal{H}},\ s_1 \xrightarrow{\Upsilon} \tilde{s}_1,\ \mathcal{L}(\Upsilon)=A\}} PT(\Upsilon, s_1) =$
   $\sum_{A \in \mathbb{N}_{fin}^{\mathcal{L}}} PM_A(s_1,\widetilde{\mathcal{H}}) = \sum_{A \in \mathbb{N}_{fin}^{\mathcal{L}}} PM_A(s_2,\widetilde{\mathcal{H}}) = \sum_{A \in \mathbb{N}_{fin}^{\mathcal{L}}} \sum_{\{\Upsilon \mid \exists \tilde{s}_2 \in \widetilde{\mathcal{H}},\ s_2 \xrightarrow{\Upsilon} \tilde{s}_2,\ \mathcal{L}(\Upsilon)=A\}} PT(\Upsilon, s_2) =$
   $\sum_{\{\Upsilon \mid \exists \tilde{s}_2 \in \widetilde{\mathcal{H}},\ s_2 \xrightarrow{\Upsilon} \tilde{s}_2\}} PT(\Upsilon, s_2) = PM(s_2,\widetilde{\mathcal{H}})$. Since 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 $\mathcal{H} \in (DR_V(G) \cup DR_V(G'))/_{\mathcal{R}}$.

  Then 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 $\mathcal{H} \in (DR_T(G) \cup DR_T(G'))/_{\mathcal{R}}$.

  Then 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 to that for the averages. □