University of Castilla-La Mancha

A publication of the

Department of Computer Science

Stochastic equivalence for modular performance evaluation in dtsiPBC

by

Igor V. Tarasyuk, Hermenegilda Macià, Valentín Valero

Technical Report **#DIAB-11-06-2** June 2011

DEPARTAMENTO DE INFORMÁTICA
ESCUELA SUPERIR DE INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE CASTILLA-LA MANCHA
Campus Universitario s/n
Albacete - 02071 - Spain
Phone +34.967.599200, Fax +34.967.599224

Stochastic equivalence for modular performance evaluation in dtsiPBC

Igor V. Tarasyuk*, Hermenegilda Macià and Valentín Valero†

Abstract

We propose the extension of discrete time stochastic Petri box calculus (dtsPBC) presented by I.V. Tarasyuk with immediate multiactions. dtsPBC is a discrete time analog 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 both semantics is demonstrated. In order to evaluate performance, the corresponding semi-Markov chains are analyzed. We define step stochastic bisimulation equivalence of expressions and explain how it can be used to reduce their transition systems and underlying semi-Markov chains as well as to compare the stationary behaviour. We prove that the introduced equivalence guarantees a coincidence of performance indices and can be used for performance analysis simplification. 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 Petri net, stochastic process algebra, Petri box calculus, discrete time, immediate multiaction, transition system, operational semantics, immediate transition, dtsi-box, denotational semantics, Markov chain, performance evaluation, stochastic equivalence, reduction, shared memory system.

1 Introduction

Algebraic process calculi like CSP [17], ACP [8] and CCS [28] 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 [18], PEPA [16] and EMPA [7]. 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).

Petri box calculus (PBC) [4, 5] is a flexible and expressive process algebra developed as a tool for specification of Petri nets (PNs) structure and their interrelations. Its goal was also to propose a compositional semantics for high level constructs of concurrent programming languages in terms of elementary PNs. Formulas of PBC are combined not from single (visible or invisible) actions and variables only, 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) [28]. 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 [4, 5]. Last years, several extensions of PBC with a deterministic or a stochastic model of time were presented.

*The author was supported by a grant within the "Invited researchers" programme of University of Castilla-La Mancha (UCLM), Spain, with the project "Stochastic equivalences for modular performance analysis in dtsiPBC", by Deutsche Forschungsgemeinschaft (DFG), grant 436 RUS 113/1002/01, and by Russian Foundation for Basic Research (RFBR), grant 09-01-91334.

†The authors were supported by Spanish government (co-financed by FEDER founds) with the project "Modeling and analyzing composite Web services using formal methods", with reference TIN2009-14312-C02-02.

A deterministic time model is considered in time Petri box calculus (tPBC) [20], in timed Petri box calculus (TPBC) [26] and in arc time Petri box calculus (atPBC) [34]. In tPBC each action has a time interval associated (the earliest and the latest firing time), and an interleaving operational semantics is defined. The denotational semantics is then defined in terms of a subclass of labeled time PNs (LtPNs), based on tPNs [27], 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 [36], 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 then defined on a subclass of arc time PNs (atPNs), where time restrictions are associated with the arcs, called arc time Petri boxes (at-boxes).

A stochastic extension of PBC, called stochastic Petri box calculus (sPBC), was proposed in [33, 22]. In sPBC, multiactions have stochastic durations 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 labeled transition systems. Its denotational semantics was defined in terms of a subclass of labeled continuous time stochastic PNs (LCTSPNs), based on CTSPNs [23, 2], and called stochastic Petri boxes (s-boxes). In [30], the iteration operator was added to sPBC. In [31], a number of new equivalence relations were proposed for regular terms of sPBC to choose later a suitable candidate for a congruence. sPBC with iteration was enriched further with immediate multiactions in [32]. A denotational semantics of such an sPBC extension was defined via a subclass of labeled generalized SPNs (LGSPNs), based on GSPNs [23, 2, 3], and called generalized stochastic Petri boxes (gs-boxes).

In [39], 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 [29], 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 [38, 40], we constructed an enrichment of dtsPBC with the iteration operator used to specify infinite 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 [41], we presented the extension dtsiPBC of the latter calculus with immediate multiactions.

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, equivalences for SPAs 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.

Interleaving probabilistic weak trace equivalence was introduced in [12]. Interleaving probabilistic strong bisimulation equivalence was proposed in [21] on labeled probabilistic transition systems, in [18] on labeled CTMCs and in [16] on probabilistic process algebras. Interleaving probabilistic equivalences were defined for probabilistic processes in [19, 15]. Interleaving probabilistic weak bisimulation equivalence was considered in [9] on Markovian process algebras, in [10] on labeled CTSPNs and in [11] on GSPNs. In [6], a comparison of a variety 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.

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 if terms of a subclass of labeled discrete time stochastic and immediate PNs (LDTSPNs with immediate transitions, LDTSIPNs), based on an extension of DTSPNs somewhat similar to discrete time deterministic and stochastic PNs (DTDSPNs) [42], and called dtsi-boxes. A consistency of both semantics is demonstrated. The corresponding stochastic process, which is a semi-Markov chain (SMC), is constructed and investigated, with the purpose of performance evaluation, which is the same for both semantics. Further, we propose step stochastic bisimulation equivalence allowing one to identify stochastic 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 calculus. 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. This property implies a 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. At last, we present a case study of the system with two processors and a common shared memory explaining how to model concurrent systems within the calculus and analyze their performance as well as in which way to reduce the systems while preserving their performance indices and making simpler the performance evaluation. First results on this subject can be found in [41].

Let us compare dtsiPBC with the classical SPAs MTIPP, PEPA and EMPA. The first main difference between them and dtsiPBC 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 are discrete probabilities. The third main difference are immediate multiactions. Let us explain this in more detail. 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 under condition that no other multiaction can occur at the current discrete time moment (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. 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 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 consist 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 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 issues can be investigated based on the discrete time Markov chain (DTMC) extracted from the labeled probabilistic transition system associated with each expression of dtsiPBC. In the classical SPAs, continuous time Markov chains (CTMCs) are used for performance evaluation. dtsiPBC has a denotational semantics in terms of LDTSPNs from which the corresponding DTMCs

can be derived as well. Thus, the multiaction labels and the set of flexible and powerful operations, as well as a step operational and a Petri net denotational semantics allowing for concurrent execution of activities (or transitions) are the main advantages of dtsiPBC.

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. Finally, Section 10 summarizes the results obtained and outlines research perspectives in this area.

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 X by $\mathbb{N}_f^X$. The *cardinality* of a multiset 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 is a proper set such that $M \subseteq X$. 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 $a \neq \hat{a}$ and $\hat{\hat{a}} = a$. Let $\mathcal{A} = Act \cup \widehat{Act}$ be the set of *all actions*, and $\mathcal{L} = \mathbb{N}_f^{\mathcal{A}}$ be the set of *all multiactions*. Note that $\emptyset \in \mathcal{L}$, this corresponds to an internal activity, 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 *conditional probability* of the multiaction α. The multiaction probabilities are used to calculate the probabilities of state changes (steps) at discrete time moments. The probabilities of stochastic multiactions are required not to be equal to 1, since this value is left for immediate multiactions which will be defined later. 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 $l \in \mathbb{N} \setminus \{0\}$ is the non-zero *weight* of the multiaction α. 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 concurrent step, i.e., the steps consisting only of immediate multiactions or those including only stochastic multiactions are allowed. Let $\mathcal{IL}$ be the set of *all immediate multiactions*.

Let us note that the same multiaction $\alpha \in \mathcal{L}$ may have different probabilities and weights in the same specification. It is easy to differentiate between probabilities and weights, hence, between stochastic and immediate multiactions, since the probabilities of stochastic multiactions belong to the interval $(0; 1)$, and the weights of immediate multiactions are non-zero (positive) natural numbers from $\mathbb{N} \setminus \{0\} = \{1, 2, \ldots\}$. 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 $(\alpha, \kappa) \in \mathcal{SIL}$ is defined as $\mathcal{A}(\alpha, \kappa) = \mathcal{A}(\alpha)$. The *alphabet* of $\Upsilon \in \mathbb{N}_f^{\mathcal{SIL}}$ is defined as $\mathcal{A}(\Upsilon) = \cup_{(\alpha,\kappa) \in \Upsilon} \mathcal{A}(\alpha)$. For $(\alpha, \kappa) \in \mathcal{SIL}$, 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 $\Upsilon \in \mathbb{N}_f^{\mathcal{SIL}}$ is defined as $\mathcal{L}(\Upsilon) = \sum_{(\alpha,\kappa) \in \Upsilon} \alpha$.

Activities are combined into formulas 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 the standard interpretation like in other process algebras, but parallelism does not include synchronization unlike the corresponding operation in CCS [28].

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 $f(\alpha) = \sum_{x \in \alpha} f(x)$. Let $\Upsilon \in \mathbb{N}_f^{\mathcal{SIL}}$. Relabeling is extended to the multisets of activities as follows: for $\Upsilon \in \mathbb{N}_f^{\mathcal{SIL}}$, $f(\Upsilon) = \sum_{(\alpha,\kappa) \in \Upsilon} (f(\alpha), \kappa)$.

Restriction over an action a 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 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

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

We may synchronize multiactions of the same type only: either both stochastic 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).

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, here and further we prefer the PBC approach and add them to resolve ambiguities only.

To avoid inconsistency of the iteration operator, we should not allow any concurrency in 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. Later on, in Example 4.2, we shall demonstrate that such inconsistency can result to nets which are not safe, see also [5] 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, see [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 in the corresponding LDTSIPN, hence, no transitions are fired and 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 instantaneously.

First, in Table 1, we define inaction rules for the 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 the 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

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

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 3.1 *Let $\approx = (\Rightarrow \cup \Leftarrow)^*$ be the 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 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 then 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 conditional 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 multiactions describe dynamic expression transformations due to the execution of non-empty multisets of stochastic multiactions. The rules represent the possible state changes of the specified processes when some non-empty multisets of stochastic multiactions are executed. As we shall see, the application of an action rule with stochastic multiactions to a dynamic expression leads to a discrete time step in the corresponding LDTSIPN at which some stochastic transitions are fired and the current marking is changed, unless there is a self-loop produced by the iterative execution of a non-empty multiset (which should

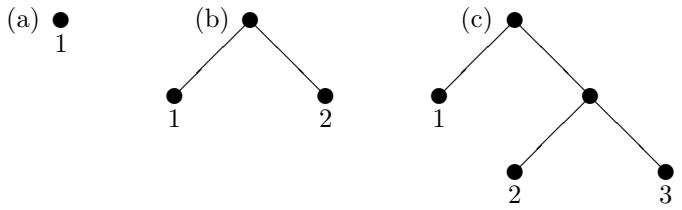

Figure 1: The binary trees encoded with the numberings 1, (1)(2) and (1)((2)(3))

be additionally the one-element one, i.e., the single stochastic multiaction, since we do not allow concurrency in the highest level of the second argument of iteration).

Action rules with immediate multiactions describe dynamic expression transformations due to the execution of non-empty multisets of immediate multiactions. The rules represent the possible state changes of the specified processes when some non-empty multisets of immediate multiactions are executed. As we shall see, the application of an action rule with immediate multiactions to a dynamic expression leads in the corresponding LDTSIPN to the instantaneous firing of some immediate transitions and changing of the the current marking, unless there is a self-loop produced by the iterative execution of a non-empty multiset (which should be additionally the one-element one, i.e., the single immediate multiaction, since we do not allow concurrency in 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 the 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 are fired 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. The PBC rule $G \stackrel{\emptyset}{\rightarrow} G$ from [5] in our setting would correspond to the rule $G \Rightarrow G$ describing the stay in the current state when no time elapses, but we do not need it to transform dynamic expressions into operative ones, hence, we do not introduce it in dtsPBC.

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 instantaneously 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 instantaneously 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 3.2 *Let $\iota \in I\!N$. The* numbering *of expressions is defined as*

$$\iota ::= \ \iota \mid (\iota)(\iota).$$

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

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

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

After the enumeration, the multisets of activities from the expressions will be then the proper sets. In the following, we suppose that the identical activities are enumerated when it is 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 $\exists E \in RegStatExpr,\ G \in [\underline{E}]_{\approx}$.

Let $G \in OpRegDynExpr$. We now define the *set of all sets of non-conflicting activities which can be 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 $\Upsilon \in Can(G \circ E),\ \Upsilon \in Can(E \circ G)$ $(\circ \in \{;, []\}),\ \Upsilon \in Can(G \| H),\ \Upsilon \in Can(H \| G)$, $f(\Upsilon) \in Can(G[f]),\ \Upsilon \in Can(G\ \mathsf{rs}\ a)$ (when $a, \hat{a} \notin \mathcal{A}(\Upsilon)$), $\Upsilon \in Can(G\ \mathsf{sy}\ a),\ \Upsilon \in Can([G * E * F])$, $\Upsilon \in Can([E * G * F]),\ \Upsilon \in Can([E * F * G])$.
4. If $\Upsilon \in Can(G)$ and $\Xi \in Can(H)$ then $\Upsilon + \Xi \in Can(G \| H)$.
5. If $\Upsilon \in Can(G\ \mathsf{sy}\ a)$ and $(\alpha, \kappa), (\beta, \lambda) \in \Upsilon$ are different activities such that $a \in \alpha,\ \hat{a} \in \beta$ then
   (a) $(\Upsilon + \{(\alpha \oplus_a \beta, \kappa \cdot \lambda)\}) \setminus \{(\alpha, \kappa), (\beta, \lambda)\} \in Can(G\ \mathsf{sy}\ a)$, if $\kappa, \lambda \in (0; 1)$;
   (b) $(\Upsilon + \{(\alpha \oplus_a \beta, \kappa + \lambda)\}) \setminus \{(\alpha, \kappa), (\beta, \lambda)\} \in Can(G\ \mathsf{sy}\ a)$, if $\kappa, \lambda \in I\!N \setminus \{0\}$.
   When we synchronize the same set of activities in different orders, we obtain several activities with the same multiaction and probability or weight parts, but with different numberings having the same content. Then we only consider a single one of the resulting activities 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 $Can(G\ \mathsf{sy}\ a)$.

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

The expression $G \in OpRegDynExpr$ is *tangible*, denoted by $tang(G)$, if $Can(G)$ contains only sets of stochastic multiactions (possibly including the empty set), i.e., $\forall \Upsilon \in Can(G),\ \Upsilon \in I\!N_f^{\mathcal{SL}}$. Otherwise, G is *vanishing*, denoted by $vanish(G)$, meaning that there are immediate multiactions in the sets from $Can(G)$, hence, according to the note above, there are non-empty sets of immediate multiactions in $Can(G)$ as well, i.e., $\exists \Upsilon \in Can(G),\ \Upsilon \in I\!N_f^{\mathcal{IL}} \setminus \{\emptyset\}$. Obviously, immediate multiactions are only executable from vanishing operative dynamic expressions. Stochastic multiactions are only executable from tangible ones, since no stochastic multiactions can be executed from a vanishing operative dynamic expression G, even if $Can(G)$ contains sets of stochastic multiactions. The reason is that immediate multiactions have a priority over stochastic ones, and should be executed first.

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, $\Gamma, \Delta \in I\!N_f^{\mathcal{SL}} \setminus \{\emptyset\},\ \Gamma' \in I\!N_f^{\mathcal{SL}},\ I, J \in I\!N_f^{\mathcal{IL}} \setminus \{\emptyset\},\ I' \in I\!N_f^{\mathcal{IL}}$ and $\Upsilon \in I\!N_f^{\mathcal{SIL}} \setminus \{\emptyset\}$. The names of the action rules with immediate multiactions have suffix '**i**'.

Rule **Sy2** establishes that the synchronization of the two stochastic multiactions is made by taking the product of their probabilities, since we are considering that both must occur for the synchronization to happen, so this corresponds to the probability of the events intersection. In rule **Sy2i**, we sum the weights of two synchronized immediate multiactions, since the weights can be interpreted as the rewards, thus, we collect the rewards. Moreover, we express that the synchronized execution of immediate multiactions has more importance than that of every single one. Since execution of immediate multiactions takes no time, we prefer to execute in a

Table 3: Action and empty loop rules

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

step as much synchronized immediate multiactions as possible to get more significant progress in computation, this aspect will be used late while performance evaluation.

Observe also that we do not have self-synchronization, i.e., the 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, see [5].

3.3 Transition systems

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

Definition 3.3 *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 the sets of activities executable in* s is defined as $Exec(s) = \{\Upsilon \mid \exists H \in s,\ \exists \widetilde{H},\ H \xrightarrow{\Upsilon} \widetilde{H}\}$. The state s is *tangible*, if $Exec(s) \subseteq I\!N_f^{\mathcal{SL}}$. For tangible states we may have $Exec(s) = \emptyset$. Otherwise, the state s is *vanishing*, and in this case $Exec(s) \subseteq I\!N_f^{\mathcal{IL}} \setminus \{\emptyset\}$. The set of *all tangible states from* $DR(G)$ is denoted by $DR_T(G)$, and the set of *all vanishing states from* $DR(G)$ is denoted by $DR_V(G)$. Obviously, $DR(G) = DR_T(G) \uplus DR_V(G)$ ($\uplus$ denotes disjoint union).

Let $\Upsilon \in Exec(s) \setminus \{\emptyset\}$. The *probability of the set of stochastic multiactions* 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

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

Thus, if $s \in DR_T(G)$ and $Exec(s) \neq \emptyset$, then $PF(\Upsilon, s)$ could be interpreted as a *joint* probability of independent events. Each such an event is interpreted as readiness or not readiness for execution of a particular stochastic multiaction from Υ. The multiplication in the definition is used because it reflects the probability of the event intersection. 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 $PF(\Upsilon, s)$ could 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. In addition, this means that concurrent execution of the immediate multiactions has more importance than that of every single one. Since the execution of immediate multiactions takes no time, we prefer to execute in a step as much parallel immediate multiactions as possible to get more significant progress in computation. 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 [3]. 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 the GSPNs possessing interleaving semantics, we are to 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 $\Upsilon \in Exec(s)$. The *probability to execute the set of activities* Υ *in* s is

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

Thus, $PT(\Upsilon, s)$ is the probability of the set of stochastic multiactions or the weight of the set of immediate multiactions Υ which is ready for execution in s *normalized* by the probabilities or the weights of *all* the sets executable in s. The denominator of the fraction above is a sum since it reflects the probability of the events union.

If s is tangible, then $PT(\emptyset, s) \in (0; 1]$, hence, there is a non-zero probability to stay in the state s in the next time moment, and the residence time in s is at least 1 discrete time unit.

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

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

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

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

Definition 3.4 *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 $L_G \subseteq 2^{\mathcal{SIL}} \times (0; 1]$;*
- *the set of* transitions *is $\mathcal{T}_G = \{(s, (\Upsilon, PT(\Upsilon, s)), \tilde{s}) \mid s \in DR(G),\ \exists H \in s,\ \exists \widetilde{H} \in \tilde{s},\ H \stackrel{\Upsilon}{\to} \widetilde{H}\}$;*
- *the* initial state *is $s_G = [G]_{\approx}$.*

The 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, according to the classification from [15]. 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 that occur at moments of discrete time 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 instantaneously after one discrete time unit delay. Each step consisting of immediate multiactions occurs instantaneously without any delay. The step can change the current state to another one. The states are the

structural equivalence classes of dynamic expressions obtained by application of action rules starting from the expressions belonging to $[G]_{\approx}$. A transition $(s,(\Upsilon,\mathcal{P}),\tilde{s}) \in \mathcal{T}_G$ will be written as $s \xrightarrow{\Upsilon}_{\mathcal{P}} \tilde{s}$. 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 the expressions from the equivalence class. We have to keep track of such executions, called *empty loops*, because they have non-zero probabilities. This follows from the definition of $PF(\emptyset, s)$ and the fact that multiaction probabilities cannot be equal to 1 as they belong to the interval $(0;1)$. For vanishing states Υ cannot be the empty set, since we must execute some immediate multiactions from them in the current time moment.

The step probabilities belong to the interval $(0;1]$, being 1 in the case when we cannot leave a tangible state s and 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 3.5 *Let G, G' be dynamic expressions and $TS(G) = (S_G, L_G, \mathcal{T}_G, s_G), TS(G') = (S_{G'}, L_{G'}, \mathcal{T}_{G'}, s_{G'})$ be their transition systems. A mapping $\beta : S_G \to S_{G'}$ is an* isomorphism *between $TS(G)$ and $TS(G')$, denoted by $\beta : TS(G) \simeq TS(G')$, if*

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

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

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

Definition 3.6 *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 3.2 *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 conditional 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 [31]. The latter is a continuous time stochastic analogue of the* stop *process proposed in [5].* Stop *is a discrete time stochastic analogue of the* stop*. Then, let*

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

$DR(\overline{E})$ 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).

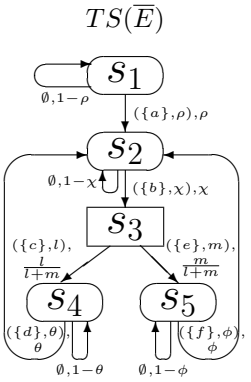

Figure 2: The transition system of $\overline{E}$ for $E = [(\{a\},\rho)*((\{b\},\chi);((((\{c\},l);(\{d\},\theta))[](\{e\},m);(\{f\},\phi))))*\mathsf{Stop}]$

4.1 Labeled DTSIPNs

Let us introduce a class of labeled discrete time stochastic and immediate Petri nets. 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* $T_N = Ts_N \cup Ti_N$ *are finite sets of* places *and* stochastic and immediate transitions*, respectively, such that* $P_N \cup T_N \neq \emptyset$, $P_N \cap T_N = \emptyset$ *and* $Ts_N \cap Ti_N = \emptyset$*;*
- $W_N : (P_N \times T_N) \cup (T_N \times P_N) \to \mathbb{N}$ *is a function providing the* weights of arcs *between places and transitions;*
- $\Omega_N : T_N \to (0;1) \cup (\mathbb{N} \setminus \{0\})$ *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;*
- $M_N \in \mathbb{N}_f^{P_N}$ *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 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 $t \in T_N$, $U \in \mathbb{N}_f^{T_N}$. 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 $U^\bullet = \sum_{t \in U} t^\bullet$.

Let N be an LDTSIPN and $M, \widetilde{M} \in \mathbb{N}_f^{P_N}$.

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 either immediate or stochastic one, and in the latter case 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 $Ena(M) \subseteq Ts_N$ or $Ena(M) = \emptyset$. 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 of the set of stochastic transitions* or the *weight of the set of immediate transitions U which is ready for firing in M* is

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

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

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

Thus, if $tang(M)$ and $Ena(M) \neq \emptyset$, then $PF(U,M)$ could be interpreted as a *joint* probability of independent events. Each such an event is interpreted as readiness or not readiness for firing of a particular transition from U. The multiplication in the definition is used because it reflects the probability of the events intersection. When no transitions are enabled in M, i.e., $Ena(M) = \emptyset$, we take $PF(\emptyset,M) = 1$, since we stay in M in this case. Note that if $tang(M)$ then we have $PF(\emptyset,M) \in (0;1]$, hence, we can stay in M at the next time moment with a certain positive probability.

If $vanish(M)$ then $PF(U,M)$ could be interpreted as the *overall* weight of the immediate transitions from U, i.e., the sum of all their weights.

Let $U \subseteq Ena(M),\ U \neq \emptyset$ and ${}^\bullet U \subseteq M$. The concurrent firing of the transitions from U changes the marking M to $\widetilde{M} = M - {}^\bullet U + U^\bullet$, denoted by $M \xrightarrow{U}_{\mathcal{P}} \widetilde{M}$, where $\mathcal{P} = PT(U,M)$ is the *probability that the set of transitions U fires in M* defined as

$$PT(U,M) = \frac{PF(U,M)}{\sum\limits_{\{V \mid {}^\bullet V \subseteq M\}} PF(V,M)}.$$

In the case $U = \emptyset$ and $tang(M)$ we have $M = \widetilde{M}$ and

$$PT(\emptyset,M) = \frac{PF(\emptyset,M)}{\sum\limits_{\{V \mid {}^\bullet V \subseteq M\}} PF(V,M)}.$$

Thus, $PT(U,M)$ is the probability of the set of stochastic transitions or the weight of the set of immediate transitions U which is ready for firing in M *normalized* by the probabilities or weights of *all* the sets enabled in M. The denominator of the fraction above is a sum since it reflects the probability of the events union.

If $tang(M)$ then $PT(\emptyset,M) \in (0;1]$, hence, there is a non-zero probability to stay in the marking in the next time moment, and the residence time in M is at least 1 discrete time unit.

Note that for all markings of an LDTSIPN N, the sum of outgoing probabilities is equal to 1. More formally, $\forall M \in I\!N_f^{P_N},\ PT(\emptyset,M) + \sum_{\{U \mid {}^\bullet U \subseteq M\}} PT(U,M) = 1$. 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 $M \to \widetilde{M}$ if $\exists U,\ M \xrightarrow{U} \widetilde{M}$.

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

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

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

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 $M \to \widetilde{M}$ 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 $M \stackrel{U}{\to}_{\mathcal{P}} \widetilde{M}$.*

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 $P_N \cap T_N = \emptyset$;*
- *$W_N : (P_N \times T_N) \cup (T_N \times P_N) \to I\!N$ is a function providing the* weights of arcs *between places and transitions and vice versa;*
- *Λ_N is the* place and transition labeling *function such that*
  - *$\Lambda_N|_{P_N} : P_N \to \{\mathsf{e},\mathsf{i},\mathsf{x}\}$ (it specifies* entry, internal *and* exit *places, respectively);*
  - *$\Lambda_N|_{T_N} : T_N \to \{\varrho \mid \varrho \subseteq 2^{\mathcal{SIL}} \times \mathcal{SIL}\}$ (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 the constant relabeling that will be defined later. In the case of 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 $M_N \in I\!N_f^{P_N}$ is the *initial 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 $L_N(t) = \mathcal{L}(\Lambda_N(t))$. The behaviour of marked dtsi-boxes follows from the firing rule of LDTSIPNs. A plain dtsi-box N is *n-bounded* ($n \in I\!N$) 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.

To define the semantic function that associates a plain dtsi-box with every static expression of dtsiPBC, we introduce the *enumeration* function $Enu : T_N \to Num$, which associates the numberings with transitions of a plain dtsi-box N 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.

The structure of the plain dtsi-box corresponding to a static expression is constructed like in PBC, see [5], 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. 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 denotes *operator box* and u denotes *transition name* from PBC setting.

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

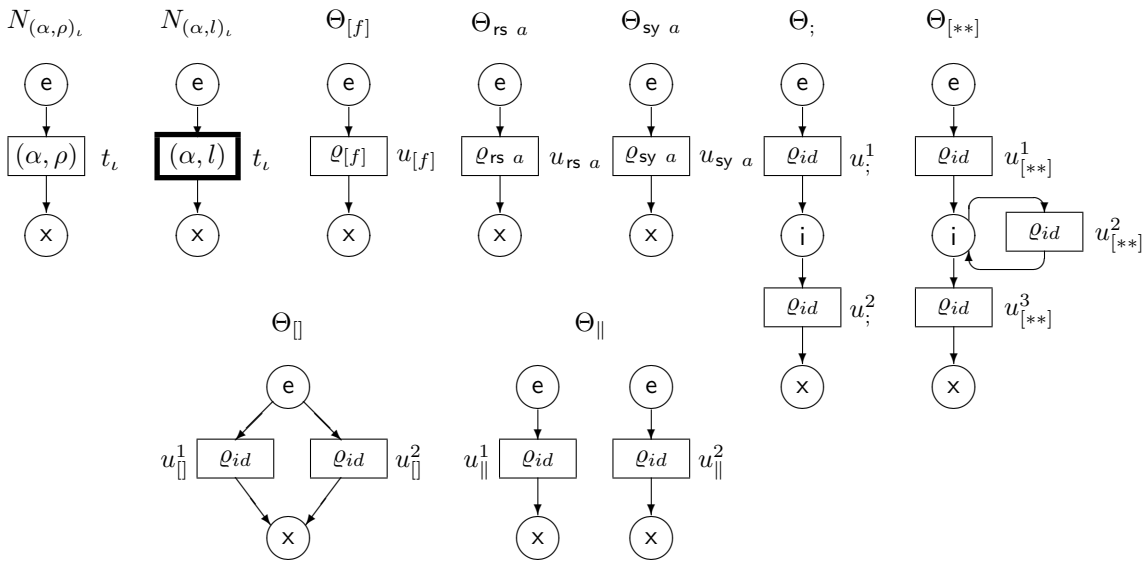

Figure 3: The plain and operator dtsi-boxes

- $\varrho_{id} = \{(\{(\alpha,\kappa)\},(\alpha,\kappa)) \mid (\alpha,\kappa) \in \mathcal{SIL}\}$ 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 in ϱ_{id} such that if $(\Upsilon,(\alpha,\kappa)),(\Xi,(\beta,\lambda)) \in \varrho_{\mathsf{sy}\ a}$ and $a \in \alpha,\ \hat{a} \in \beta$ 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 $\kappa,\lambda \in I\!N \setminus \{0\}$.

The plain and operator dtsi-boxes are presented in Figure 3. Note that the label i of internal places is usually omitted.

In the case of the synchronization, a decision that we must take is the selection of the operator box that we shall use for the iteration, since we have two proposals in plain PBC for that purpose [5]. 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 the operator box. 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 case cannot occur, so that the net obtained will be always safe.

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 be the enumeration function for T_E. We shall use the analogous notation for static expressions F and K.

- $Box_{dtsi}(E \circ F) = \Theta_\circ(Box_{dtsi}(E), Box_{dtsi}(F)),\ \circ \in \{;,[],\|\}$. 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 one from the resulting transitions 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. $Box_{dtsi}(E \circ a) = \Theta_{\circ a}(Box_{dtsi}(E)),\ \circ \in \{\text{rs}, \text{sy}\}$;
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 $Box_{dtsi}(\underline{E}) = \underline{Box_{dtsi}(E)}$.

Observe 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 (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 relates 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.

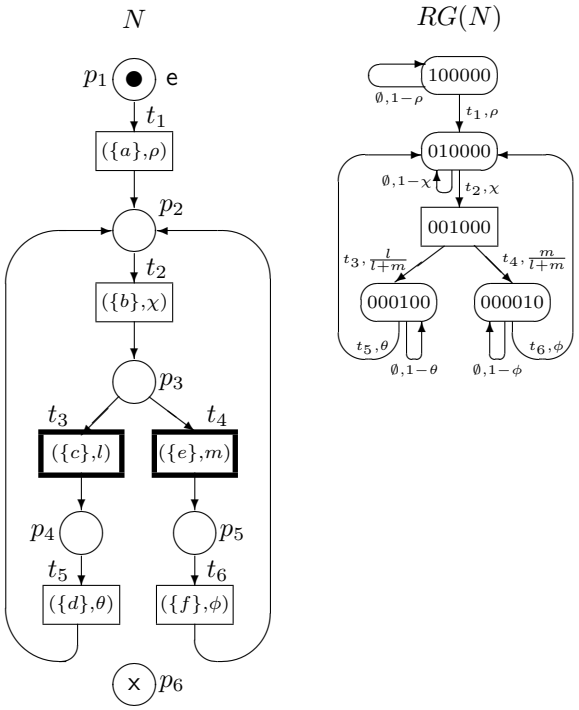

Figure 4: The marked dtsi-box $N = Box_{dtsi}(\overline{E})$ for $E = [(\{a\}, \rho) * ((\{b\}, \chi); ((((\{c\}, l); (\{d\}, \theta))[]((\{e\}, m); (\{f\}, \phi)))) * \mathsf{Stop}]$ and its reachability graph

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 3.2. 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. Symmetrically, in the marking $(0, 1, 1, 2, 0, 0)$ there are 2 tokens in the place p_4. 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, see [5]. The origin of the problem is that N has as a self-loop with two subnets which can function independently. This explains why do we consider regular expressions only.*

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.

For a dynamic expression G, a discrete random variable is associated with every tangible state from $DR(G)$. The variable captures a residence time in the state. One can interpret staying in a state in the next discrete

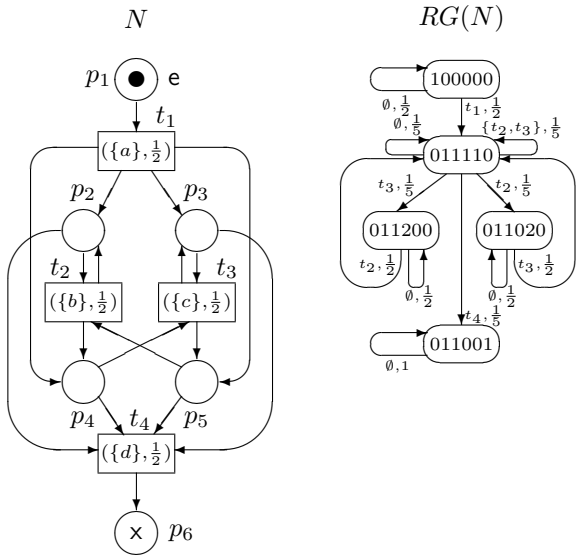

Figure 5: The marked dtsi-box $N = Box_{dtsi}(\overline{E})$ for $E = [((\{a\}, \frac{1}{2}) * ((\{b\}, \frac{1}{2}) \| (\{c\}, \frac{1}{2})) * (\{d\}, \frac{1}{2})]$ and its reachability graph

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, since the probability to stay in a tangible state s for $k-1$ time moments and leave it at moment $k \geq 1$ is $PM(s,s)^{k-1}(1 - PM(s,s))$ (the residence time is k in this case). The mean value formula for geometrical distribution allows us to calculate the average sojourn time in a tangible state s as $\frac{1}{1-PM(s,s)}$. Obviously, the average sojourn time in a vanishing state is zero. Thus, 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, is that with the elements $SJ(s),\ s \in DR(G)$.

Analogously, the *sojourn time variance in the state* s is

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

The *sojourn time variance vector* of G, denoted by VAR, is that with 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) [37], 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 similar to that applied in the context of generalized stochastic PNs (GSPNs) in [23, 2, 3], and also in the framework of discrete time deterministic and stochastic PNs (DTDSPNs) in [42]. $EDTMC(G)$ only describes the state changes of $SMC(G)$ while ignoring any time characteristics.

Thus, to construct the EDTMC, we should abstract from all time aspects of behaviour, i.e., from the sojourn time in the states. The sojourn time in every state of the EDTMC is deterministic and it is equal to one discrete time unit. Let G be a dynamic expression. A 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 G be a dynamic expression and $s, \tilde{s} \in DR(G)$.

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

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

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

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

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

is the *self-loops abstraction factor*. The *self-loops abstraction vector* of G, denoted by SL, is that with 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.

$PM^*(s,\tilde{s})$ 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 [14] 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 $DR(G)$ and the transitions $s \twoheadrightarrow_{\mathcal{P}} \tilde{s}$, if $s \to \tilde{s}$ and $s \neq \tilde{s}$, where $\mathcal{P} = PM^*(s,\tilde{s})$.*

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

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

The transient (k-step, $k \in I\!N$) probability mass function (PMF) $\psi^*[k] = (\psi^*[k](s_1), \ldots, \psi^*[k](s_n))$ for $EDTMC(G)$ is the solution of the equation system

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

where $\psi^*[0] = (\psi^*[0](s_1), \ldots, \psi^*[0](s_n))$ is the initial PMF defined as

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

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

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

$$\left\{ \begin{array}{l} \psi^*(\mathbf{P}^* - \mathbf{E}) = \mathbf{0} \\ \psi^* \mathbf{1}^T = 1 \end{array} \right. ,$$

where $\mathbf{E}$ is the unitary matrix of dimension n and $\mathbf{0}$ is the vector with n values 0, $\mathbf{1}$ is that with n values 1.

When $EDTMC(G)$ has the 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 $\psi^*(s_i)$ $(1 \leq i \leq n)$ by the average sojourn time $SJ(s_i)$ in the state s_i, after which we normalize the resulting values. Remember that for a vanishing state $s \in DR_V(G)$ we have $SJ(s) = 0$.

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

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

Example 5.1 *Let E be from Example 3.2. In Figure 6, the underlying SMC $SMC(\overline{E})$ is presented. 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

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

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

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

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

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

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

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

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

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

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

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

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

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

Let us also 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 $DR(G)$ and the transitions $s \to_{\mathcal{P}} \tilde{s}$, 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 $\mathcal{P}_{ij}$ $(1 \leq i, j \leq n = |DR(G)|)$ of (one-step) transition probability matrix (TPM) $\mathbf{P}$ for $DTMC(G)$ are defined as

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

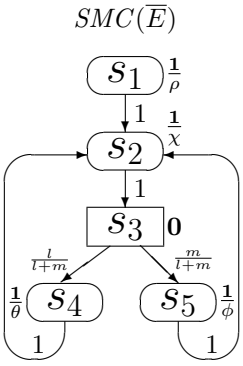

Figure 6: The underlying SMC of $\overline{E}$ for $E = [(\{a\}, \rho) * ((\{b\}, \chi); ((((\{c\}, l); (\{d\}, \theta))[]((\{e\}, m); (\{f\}, \phi)))) * \mathsf{Stop}]$

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

Let us determine a relationship between steady-state PMFs for DTMCs and EDTMCs. The following theorem proposes the equation that relates the mentioned steady-state PMFs.

First, we introduce some helpful notation. For a vector $v = (v_1, \ldots, v_n)$, let $Diag(v)$ be a diagonal matrix of dimension 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: $\forall s \in DR(G)$,*

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

Proof. Let PSL be a vector with the elements

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

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

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

After replacement of $\mathbf{P}^*$ by the expression with $\mathbf{P}$ we obtain

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

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

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

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

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

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

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

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 the solution of the equation system

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

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

Proposition 5.1 *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 $\forall s \in DR(G)$,*

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

Proof. By Theorem 5.1, since $\forall s \in DR_T(G),\ SL(s) = SJ(s)$. □

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 $SMC(G)$, see, in particular, [24].

- The *average recurrence (return) time in the state s* (i.e., the number of discrete time units or steps required for this) is $\frac{1}{\varphi(s)}$.
- The *fraction of residence time in the state s* is $\varphi(s)$.
- The *fraction of residence time in the set of states $S \subseteq DR(G)$* or the *probability of the event determined by a condition that is true for all states from S* is $\sum_{s\in S} \varphi(s)$.
- The *relative fraction of residence time in the set of states S with respect to that in $\widetilde{S}$* is $\frac{\sum_{s\in S} \varphi(s)}{\sum_{\tilde{s}\in \widetilde{S}} \varphi(\tilde{s})}$.
- The *rate of leaving the state s* is $\frac{\varphi(s)}{SJ(s)}$.
- 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 $M, \widetilde{M} \in I\!N_f^{P_N}$. 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 relates their initial states.

Proposition 5.2 *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 3.2. In Figure 7, the underlying SMC $SMC(N)$ is presented. It is easy to see that $SMC(\overline{E})$ and $SMC(N)$ are isomorphic. Thus, the steady-state PMF for $SMC(N)$ is the same as for $SMC(\overline{E})$.*

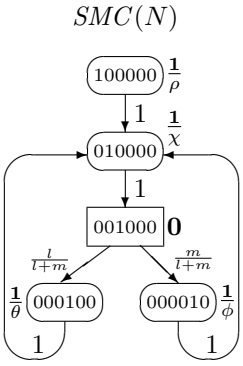

Figure 7: The underlying SMC of $N = Box_{dtsi}(\overline{E})$ for $E = [(\{a\}, \rho) * ((\{b\}, \chi); ((((\{c\}, l); (\{d\}, \theta))[]((\{e\}, m); (\{f\}, \phi)))) * \mathsf{Stop}]$

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. 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 which 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 [19, 21, 6].

In the definition below, we consider $\mathcal{L}(\Upsilon) \in I\!N_f^{\mathcal{L}}$ for $\Upsilon \in I\!N_f^{\mathcal{SIL}}$, i.e., (possibly empty) multisets of multiactions. The multiactions can be empty, then $\mathcal{L}(\Upsilon)$ contains the elements $\emptyset$, and 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 $A \in I\!N_f^{\mathcal{L}}$ 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 $s \rightarrow_{\mathcal{P}} \mathcal{H}$ if $\exists A,\ s \xrightarrow{A} \mathcal{H}$, where $\mathcal{P} = PM(s, \mathcal{H})$ is the *overall probability to move from s into the set of states $\mathcal{H}$ via any steps* defined as

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

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. $(s_1, s_2) \in \mathcal{R} \ \Rightarrow\ \forall \mathcal{H} \in (DR(G) \cup DR(G'))/_{\mathcal{R}},\ \forall A \in I\!N_f^{\mathcal{L}}$,

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

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 relates 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. □

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 the reverse implications 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 8, 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).*

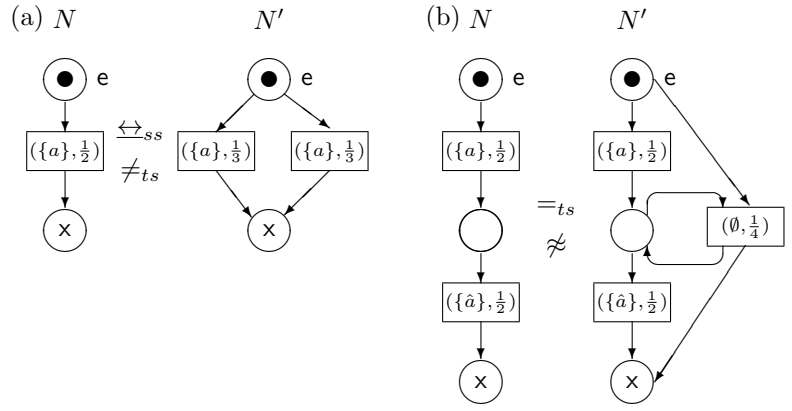

Figure 8: Dtsi-boxes of the dynamic expressions from equivalence examples of Theorem 6.1

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 to 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 behavioural and performance analysis of systems.

An *autobisimulation* is a bisimulation between an expression and itself. For a dynamic expression G and a step stochastic autobisimulation on it $\mathcal{R} : G\underline{\leftrightarrow}_{ss}G$ let $\mathcal{K} \in DR(G)/_{\mathcal{R}}$ and $s_1, s_2 \in \mathcal{K}$. We have $\forall\widetilde{\mathcal{K}} \in DR(G)/_{\mathcal{R}},\ \forall A \in \mathbb{N}_f^{\mathcal{L}},\ s_1 \stackrel{A}{\rightarrow}_{\mathcal{P}} \widetilde{\mathcal{K}} \Leftrightarrow s_2 \stackrel{A}{\rightarrow}_{\mathcal{P}} \widetilde{\mathcal{K}}$. The previous equality is valid for all $s_1, s_2 \in \mathcal{K}$, hence, we can rewrite it as $\mathcal{K} \stackrel{A}{\rightarrow}_{\mathcal{P}} \widetilde{\mathcal{K}}$, 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 $\mathcal{K} \stackrel{A}{\rightarrow} \widetilde{\mathcal{K}}$ if $\exists\mathcal{P},\ \mathcal{K} \stackrel{A}{\rightarrow}_{\mathcal{P}} \widetilde{\mathcal{K}}$ and $\mathcal{K} \rightarrow \widetilde{\mathcal{K}}$ if $\exists A,\ \mathcal{K} \stackrel{A}{\rightarrow} \widetilde{\mathcal{K}}$. 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 $SJ_{\mathcal{R}}$, is that with 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

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

The *sojourn time variance vector for the equivalence classes (with respect to $\mathcal{R}$) of states* of G, denoted by $VAR_{\mathcal{R}}$, is that with 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*

- $S_{\underline{\leftrightarrow}_{ss}} = DR(G)/_{\mathcal{R}_{ss}(G)}$;

- $L_{\underline{\leftrightarrow}_{ss}} \subseteq I\!N_f^{\mathcal{L}} \times (0;1]$;
- $\mathcal{T}_{\underline{\leftrightarrow}_{ss}} = \{(\mathcal{K}, (A, PM_A(\mathcal{K}, \widetilde{\mathcal{K}})), \widetilde{\mathcal{K}}) \mid \mathcal{K} \in DR(G)/_{\mathcal{R}_{ss}(G)},\ \mathcal{K} \stackrel{A}{\to} \widetilde{\mathcal{K}}\}$;
- $s_{\underline{\leftrightarrow}_{ss}} = \{[G]_{\approx}\}$.

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

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

Let $\mathcal{K} \to \widetilde{\mathcal{K}}$ 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

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

The value $k = 0$ in the summation above corresponds to the case when no self-loops occur. Note that $\forall \mathcal{K} \in DR_T(G)/_{\mathcal{R}_{ss}(G)},\ PM^*(\mathcal{K}, \widetilde{\mathcal{K}}) = SJ(\mathcal{K}) PM(\mathcal{K}, \widetilde{\mathcal{K}})$, since we always have the empty loop (which is a self-loop) $\mathcal{K} \stackrel{\emptyset}{\to} \mathcal{K}$ 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 $DR(G)/_{\mathcal{R}_{ss}(G)}$ 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* quotient (by $\underline{\leftrightarrow}_{ss}$) average sojourn time vector *of G, defined as $SJ_{\underline{\leftrightarrow}_{ss}} = SJ_{\mathcal{R}_{ss}(G)}$.*

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

The quotients of both transition systems and underlying SMCs are the minimal reductions of the mentioned 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 [1] based on place bisimulation equivalence for PNs. Moreover, the algorithms which can be adapted for our framework exist for constructing the quotients of transition systems by bisimulation [35] and those of (discrete or continuous time) Markov chains by ordinary lumping [13]. The algorithms have time complexity $O(m \lg n)$ 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. The comprehensive 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.

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 [29], the following sufficient (but not necessary) conditions for ergodicity of DTSPNs are stated: *liveness* (for each transition and any reachable marking there exist a sequence of markings from it leading to the marking enabling that transition), *boundedness* (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). For the dtsi-box of a dynamic expression with no deadlocks in at least one of the bodies of the iteration operators it contains these three conditions are satisfied: the subnet corresponding to the deadlock-free iteration body is live, safe (1-bounded) and nondeterministic (since all markings of the live subnet are non-terminal, the probabilities of transitions from them are strictly less than 1). Hence, its SMC restricted to the states between the initial and final states of the deadlock-free iteration body is ergodic. The isomorphism between SMCs of expressions and those of the corresponding dtsi-boxes which is stated by

Proposition 5.2 guarantees that the underlying SMC of an expression with infinite behaviour is ergodic if restricted to the states in which a deadlock-free iteration body is executed.

In this section, we consider the expressions such that their underlined SMCs contain one ergodic subset of states to guarantee that the 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 $\mathcal{R} : G \underline{\leftrightarrow}_{ss} G'$. 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. □

In the proof of Proposition 8.1 a limit construction was used to go from transient to stationary case. Thus, the result of the proposition is valid if we replace steady-state probabilities with transient ones.

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

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

The following theorem demonstrates that for two dynamic expressions related by $\underline{\leftrightarrow}_{ss}$ the steady-state probabilities to 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 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. □

In the proof of Theorem 8.1 a limit construction was used to go from transient to stationary case. Thus, the result of the theorem is valid if we replace steady-state probabilities with transient ones.

Example 8.1 *The expression* $\mathsf{Stop} = (\{c\}, \frac{1}{2})$ rs c *specifies the non-terminating process that performs only empty loops with probability* 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

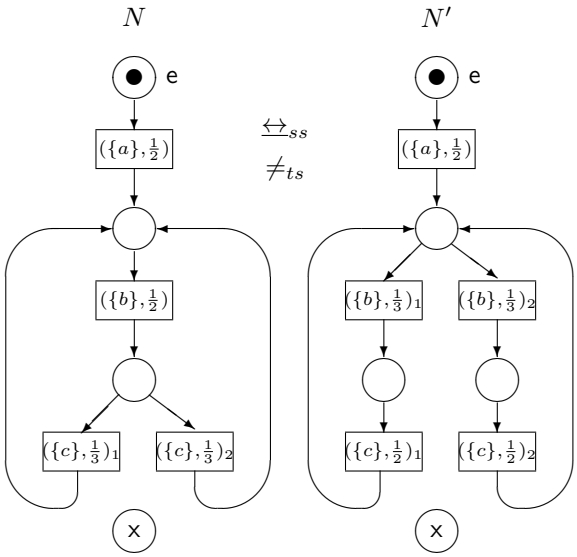

Figure 9: $\underline{\leftrightarrow}_{ss}$ implies a coincidence of the steady-state probabilities to come in an equivalence class and start a trace from it

$$s'_1 = [[\overline{(\{a\},\tfrac{1}{2})} * ((((\{b\},\tfrac{1}{3})_1;(\{c\},\tfrac{1}{2})_1)[](((\{b\},\tfrac{1}{3})_2;(\{c\},\tfrac{1}{2})_2)) * \mathsf{Stop}]]_{\approx},$$
$$s'_2 = [[(\{a\},\tfrac{1}{2}) * \overline{((((\{b\},\tfrac{1}{3})_1;(\{c\},\tfrac{1}{2})_1)[](((\{b\},\tfrac{1}{3})_2;(\{c\},\tfrac{1}{2})_2))} * \mathsf{Stop}]]_{\approx},$$
$$s'_3 = [[(\{a\},\tfrac{1}{2}) * ((((\{b\},\tfrac{1}{3})_1;\overline{(\{c\},\tfrac{1}{2})_1})[](((\{b\},\tfrac{1}{3})_2;(\{c\},\tfrac{1}{2})_2)) * \mathsf{Stop}]]_{\approx},$$
$$s'_4 = [[(\{a\},\tfrac{1}{2}) * ((((\{b\},\tfrac{1}{3})_1;(\{c\},\tfrac{1}{2})_1)[](((\{b\},\tfrac{1}{3})_2;\overline{(\{c\},\tfrac{1}{2})_2})) * \mathsf{Stop}]]_{\approx}.$$

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 step trace Σ from it coincide as well: $\varphi(s_3)(PT(\{(\{c\},\frac{1}{3})_1\},s_3) + PT(\{(\{c\},\frac{1}{3})_2\},s_3)) = \frac{1}{2}\left(\frac{1}{4}+\frac{1}{4}\right) = \frac{1}{4} = \frac{1}{4}\cdot\frac{1}{2}+\frac{1}{4}\cdot\frac{1}{2} = \varphi'(s'_3)PT(\{(\{c\},\frac{1}{2})_1\},s'_3) + \varphi'(s'_4)PT(\{(\{c\},\frac{1}{2})_2\},s'_4)$.

In Figure 9, 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 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 and Theorem 8.1 guarantee a 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 the 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 dimension 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.

2. The transition system of the expression is constructed.

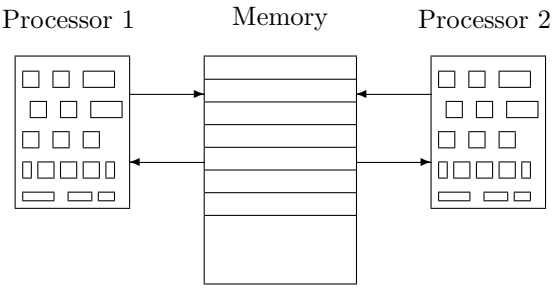

Figure 10: The diagram of the shared memory system

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.

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 a SMC contains several irreducible subsets of states then several steady states can exist which depend on the initial PMF. There is an analytical method to determine the stable states for SMCs of this kind as well. 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. In addition, it is worth to apply the method only to the systems with similar subprocesses.

9 Shared memory system

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

9.1 The standard system

Consider a model of two processors accessing a common shared memory described in [25, 2, 3] 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. The model performs 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 an acquisition of the memory 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 10.

Let us explain the meaning of actions from syntax of the dtsiPBC expressions which will specify the system modules. The action a corresponds to the system activation. The actions r_i $(1 \leq i \leq 2)$ represent the common memory request of processor i. The instantaneous actions d_i correspond to the decision on the memory allocation in favour of the processor i. The actions m_i represent the common memory access of processor i. The other actions are used for communication purposes only via synchronization, and we abstract from them later using restriction.

The 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

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

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

$$s_1 = \underline{[(\overline{[(\{x_1\}, \frac{1}{2})} * ((\{r_1\}, \frac{1}{2}); (\{d_1, y_1\}, 1); (\{m_1, z_1\}, \frac{1}{2})) * \mathsf{Stop}]\|}$$
$$\underline{[(\{x_2\}, \frac{1}{2})} * ((\{r_2\}, \frac{1}{2}); (\{d_2, y_2\}, 1); (\{m_2, z_2\}, \frac{1}{2})) * \mathsf{Stop}]\|$$
$$\overline{[(\{a, \widehat{x_1}, \widehat{x_2}\}, \frac{1}{2}) * ((((\{\widehat{y_1}\}, 1); (\{\widehat{z_1}\}, \frac{1}{2}))[]((\{\widehat{y_2}\}, 1); (\{\widehat{z_2}\}, \frac{1}{2}))) * \mathsf{Stop}])}$$
$$\mathsf{sy}\ x_1\ \mathsf{sy}\ x_2\ \mathsf{sy}\ y_1\ \mathsf{sy}\ y_2\ \mathsf{sy}\ z_1\ \mathsf{sy}\ z_2\ \mathsf{rs}\ x_1\ \mathsf{rs}\ x_2\ \mathsf{rs}\ y_1\ \mathsf{rs}\ y_2\ \mathsf{rs}\ z_1\ \mathsf{rs}\ z_2]_{\approx},$$

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

$$s_3 = [([(\{x_1\}, \frac{1}{2}) * ((\{r_1\}, \frac{1}{2}); \overline{(\{d_1, y_1\}, 1)}; (\{m_1, z_1\}, \frac{1}{2})) * \mathsf{Stop}]\|$$
$$[(\{x_2\}, \frac{1}{2}) * ((\{r_2\}, \frac{1}{2}); (\{d_2, y_2\}, 1); (\{m_2, z_2\}, \frac{1}{2})) * \mathsf{Stop}]\|$$
$$[(\{a, \widehat{x_1}, \widehat{x_2}\}, \frac{1}{2}) * ((((\{\widehat{y_1}\}, 1); (\{\widehat{z_1}\}, \frac{1}{2}))[]((\{\widehat{y_2}\}, 1); (\{\widehat{z_2}\}, \frac{1}{2}))) * \mathsf{Stop}])$$
$$\mathsf{sy}\ x_1\ \mathsf{sy}\ x_2\ \mathsf{sy}\ y_1\ \mathsf{sy}\ y_2\ \mathsf{sy}\ z_1\ \mathsf{sy}\ z_2\ \mathsf{rs}\ x_1\ \mathsf{rs}\ x_2\ \mathsf{rs}\ y_1\ \mathsf{rs}\ y_2\ \mathsf{rs}\ z_1\ \mathsf{rs}\ z_2]_{\approx},$$

$$s_4 = [([(\{x_1\}, \frac{1}{2}) * (\overline{(\{r_1\}, \frac{1}{2})}; (\{d_1, y_1\}, 1); (\{m_1, z_1\}, \frac{1}{2})) * \mathsf{Stop}]\|$$
$$[(\{x_2\}, \frac{1}{2}) * ((\{r_2\}, \frac{1}{2}); \overline{(\{d_2, y_2\}, 1)}; (\{m_2, z_2\}, \frac{1}{2})) * \mathsf{Stop}]\|$$
$$[(\{a, \widehat{x_1}, \widehat{x_2}\}, \frac{1}{2}) * ((((\{\widehat{y_1}\}, 1); (\{\widehat{z_1}\}, \frac{1}{2}))[](\overline{(\{\widehat{y_2}\}, 1)}; (\{\widehat{z_2}\}, \frac{1}{2}))) * \mathsf{Stop}])$$
$$\mathsf{sy}\ x_1\ \mathsf{sy}\ x_2\ \mathsf{sy}\ y_1\ \mathsf{sy}\ y_2\ \mathsf{sy}\ z_1\ \mathsf{sy}\ z_2\ \mathsf{rs}\ x_1\ \mathsf{rs}\ x_2\ \mathsf{rs}\ y_1\ \mathsf{rs}\ y_2\ \mathsf{rs}\ z_1\ \mathsf{rs}\ z_2]_{\approx},$$

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

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

$$s_7 = [([(\{x_1\}, \frac{1}{2}) * (\overline{(\{r_1\}, \frac{1}{2})}; (\{d_1, y_1\}, 1); (\{m_1, z_1\}, \frac{1}{2})) * \mathsf{Stop}]\|$$
$$[(\{x_2\}, \frac{1}{2}) * ((\{r_2\}, \frac{1}{2}); (\{d_2, y_2\}, 1); \overline{(\{m_2, z_2\}, \frac{1}{2})) * \mathsf{Stop}]}\|$$
$$[(\{a, \widehat{x_1}, \widehat{x_2}\}, \frac{1}{2}) * ((((\{\widehat{y_1}\}, 1); (\{\widehat{z_1}\}, \frac{1}{2}))[]((\{\widehat{y_2}\}, 1); \overline{(\{\widehat{z_2}\}, \frac{1}{2})})) * \mathsf{Stop}])$$
$$\mathsf{sy}\ x_1\ \mathsf{sy}\ x_2\ \mathsf{sy}\ y_1\ \mathsf{sy}\ y_2\ \mathsf{sy}\ z_1\ \mathsf{sy}\ z_2\ \mathsf{rs}\ x_1\ \mathsf{rs}\ x_2\ \mathsf{rs}\ y_1\ \mathsf{rs}\ y_2\ \mathsf{rs}\ z_1\ \mathsf{rs}\ z_2]_{\approx},$$

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

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

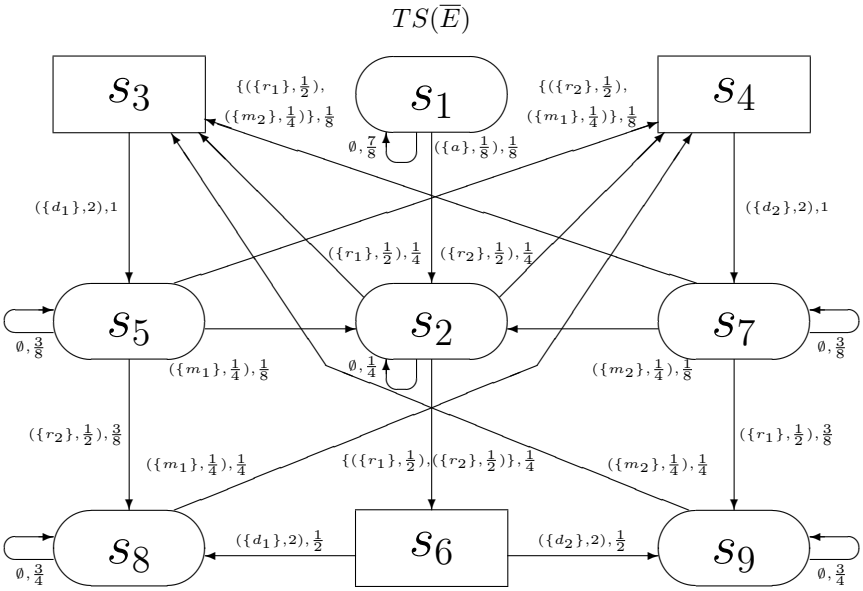

Figure 11: The transition system of the shared memory system

We have $DR_T(\overline{E}) = \{s_1, s_2, s_5, s_5, s_8, s_9\}$ 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 11, the transition system $TS(\overline{E})$ is presented. In Figure 12, 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

$$VAR = \left(64, \frac{16}{9}, 0, 0, \frac{64}{25}, 0, \frac{64}{25}, 16, 16\right).$$

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

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

In Table 4, the transient and the steady-state probabilities $\psi_i^*[k]$ ($i \in \{1, 2, 3, 5, 6, 8\}$) for the EDTMC of the shared memory system at the time moments k ($0 \leq k \leq 10$) and $k = \infty$ are presented, and in Figure 13, 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 as well as for s_8, s_9.

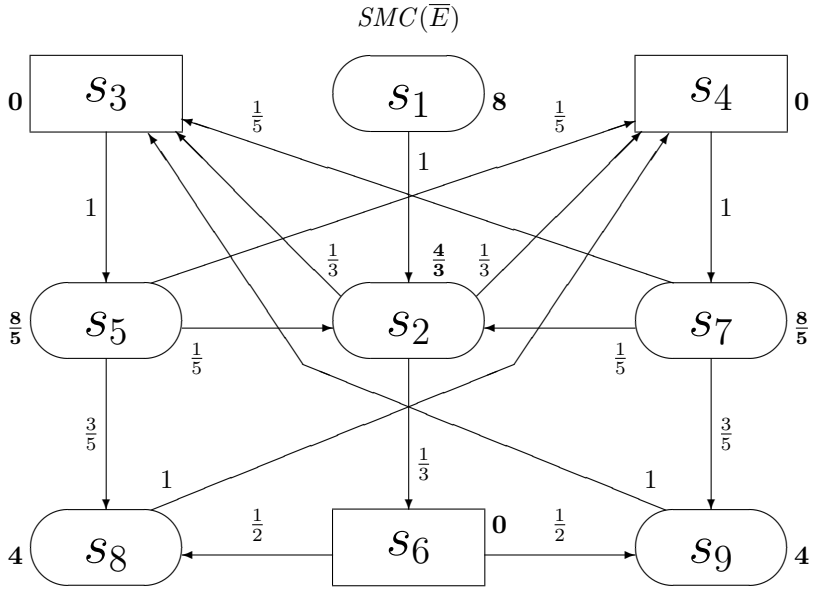

Figure 12: The underlying SMC of the shared memory system

Table 4: Transient and steady-state probabilities for the EDTMC of the shared memory system

k	0	1	2	3	4	5	6	7	8	9	10	∞
$\psi_1^*[k]$	1	0	0	0	0	0	0	0	0	0	0	0
$\psi_2^*[k]$	0	1	0	0	0.1333	0	0.0933	0.0978	0.0187	0.0969	0.0754	0.0682
$\psi_3^*[k]$	0	0	0.3333	0	0.2333	0.2444	0.0467	0.2422	0.1886	0.0982	0.2316	0.1705
$\psi_5^*[k]$	0	0	0	0.3333	0	0.2333	0.2444	0.0467	0.2422	0.1886	0.0982	0.1705
$\psi_6^*[k]$	0	0	0.3333	0	0	0.0444	0	0.0311	0.0326	0.0062	0.0323	0.0227
$\psi_8^*[k]$	0	0	0	0.1667	0.2000	0	0.1622	0.1467	0.0436	0.1616	0.1163	0.1136

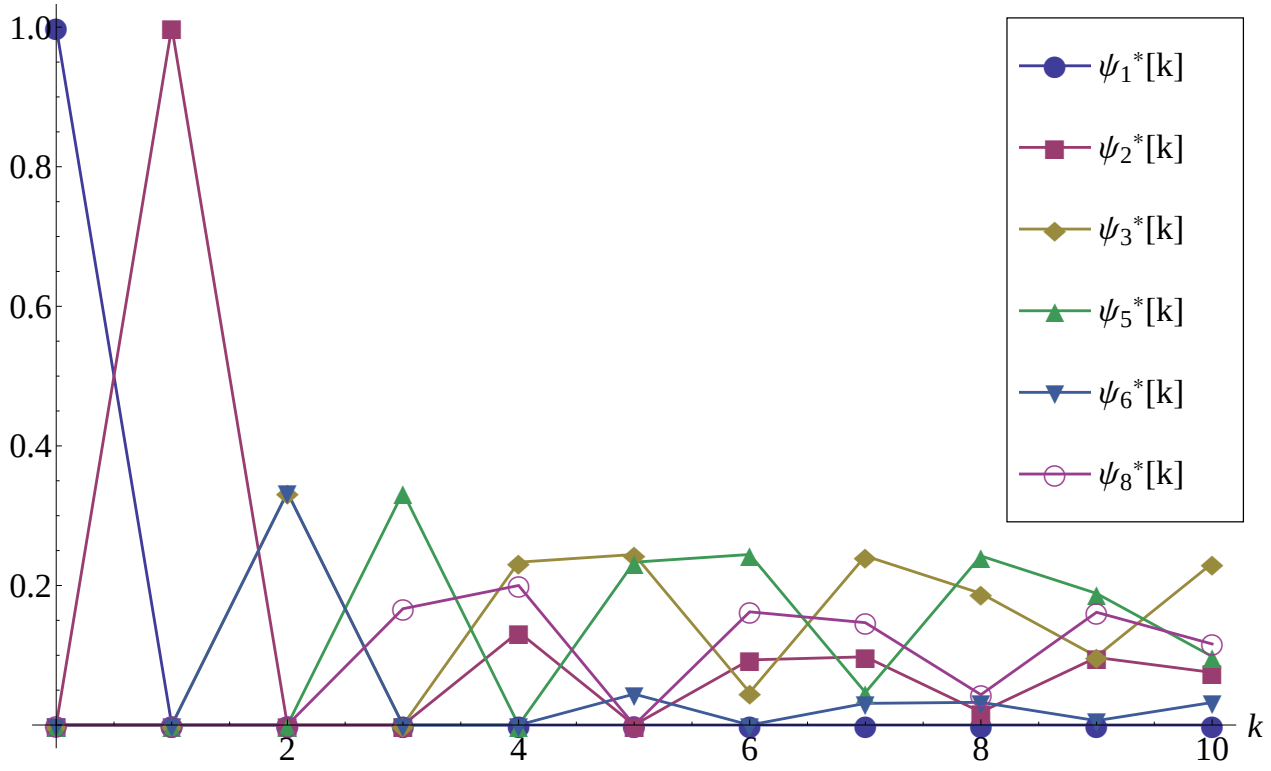

Figure 13: Transient probabilities alteration diagram for the EDTMC of the shared memory system

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 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 $\frac{1}{\varphi_2} = 17$.
- The common memory is available only in the states s_2, s_3, s_4, s_6. Then the steady-state probability that the memory is available is $\varphi_2 + \varphi_3 + \varphi_4 + \varphi_6 = \frac{1}{17} + 0 + 0 + 0 = \frac{1}{17}$. 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 $\frac{\varphi_2}{SJ_2} = \frac{1}{17} \cdot \frac{3}{4} = \frac{3}{68}$.

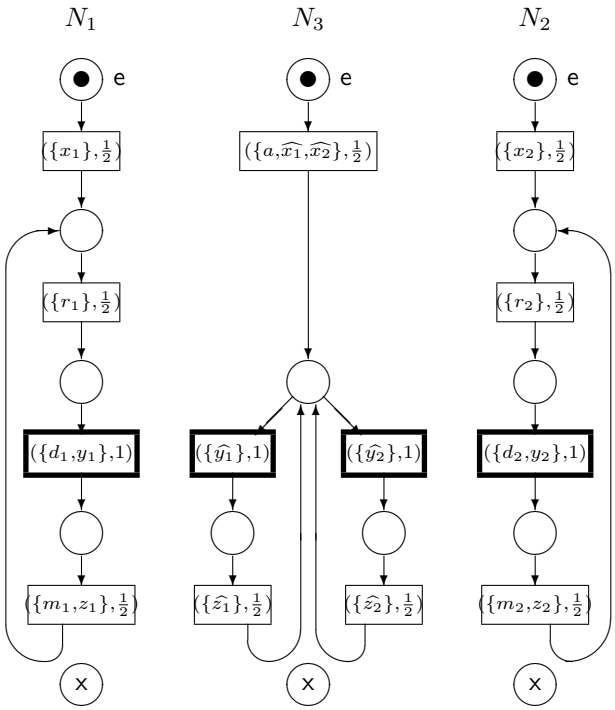

Figure 14: The marked dtsi-boxes of two processors and shared memory

- 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 $\varphi_2 \sum_{\{\Upsilon|(\{r_1\},\frac{1}{2})\in\Upsilon\}} PT(\Upsilon,s_2) + \varphi_7 \sum_{\{\Upsilon|(\{r_1\},\frac{1}{2})\in\Upsilon\}} PT(\Upsilon,s_7) = \frac{1}{17}\left(\frac{1}{4}+\frac{1}{4}\right) + \frac{3}{17}\left(\frac{3}{8}+\frac{1}{8}\right) = \frac{2}{17}$.

In Figure 14, the marked dtsi-boxes corresponding to the dynamic expressions of two processors and shared memory are presented, i.e., $N_i = Box_{dtsi}(\overline{E_i})$ $(1 \le i \le 3)$. In Figure 15, the marked dtsi-box corresponding to the dynamic expression of the shared memory system is depicted, i.e., $N = Box_{dtsi}(\overline{E})$.

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 the processors 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.

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

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

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

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

$DR(\overline{F})$ resembles $DR(\overline{E})$, and $TS(\overline{F})$ is similar to $TS(\overline{E})$. We have $SMC(\overline{F}) = SMC(\overline{E})$. 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.

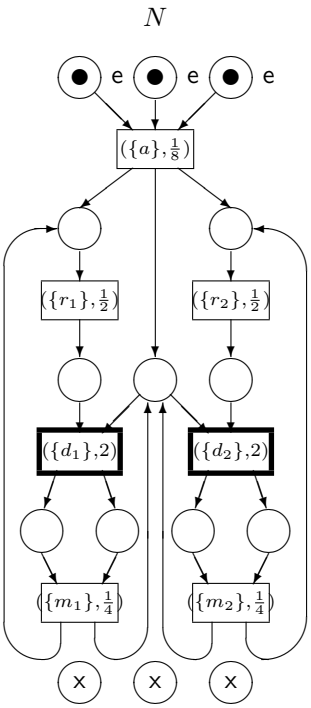

Figure 15: The marked dtsi-box of the shared memory system

The first and second performance indices are the same for the standard and the abstract systems. Let us consider the following performance index which is a 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).

In Figure 16, the quotient transition system $TS_{\underline{\leftrightarrow}_{ss}}(\overline{F})$ is presented. In Figure 17, 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

$$VAR' = \left(64, \frac{16}{9}, 0, \frac{64}{25}, 0, 16\right).$$

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

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

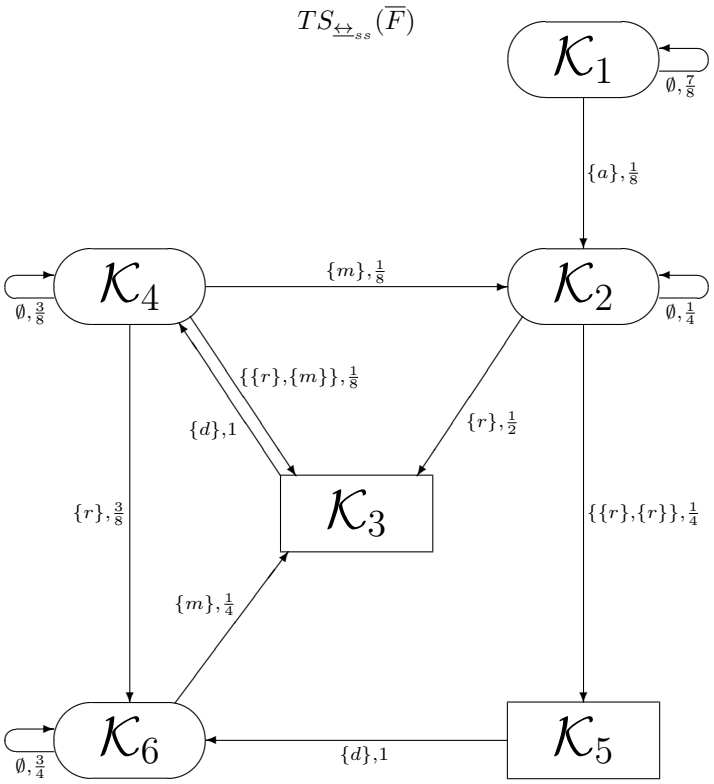

Figure 16: The quotient transition system of the abstract shared memory system

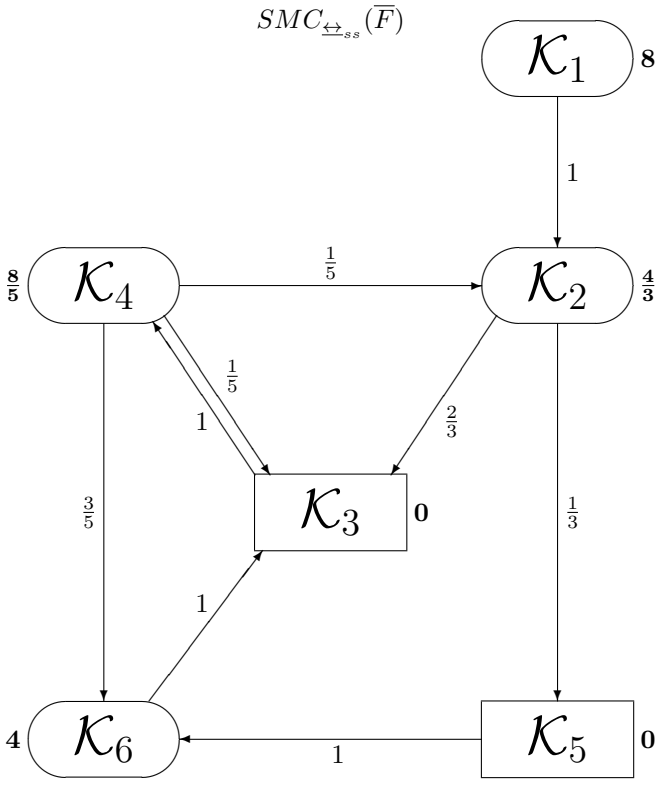

Figure 17: The quotient underlying SMC of the abstract shared memory system

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

k	0	1	2	3	4	5	6	7	8	9	10	∞
${\psi'_1}^*[k]$	1	0	0	0	0	0	0	0	0	0	0	0
${\psi'_2}^*[k]$	0	1	0	0	0.1333	0	0.0933	0.0978	0.0187	0.0969	0.0754	0.0682
${\psi'_3}^*[k]$	0	0	0.6667	0	0.4667	0.4889	0.0933	0.4844	0.3772	0.1964	0.4633	0.3409
${\psi'_4}^*[k]$	0	0	0	0.6667	0	0.4667	0.4889	0.0933	0.4844	0.3772	0.1964	0.3409
${\psi'_5}^*[k]$	0	0	0.3333	0	0	0.0444	0	0.0311	0.0326	0.0062	0.0323	0.0227
${\psi'_6}^*[k]$	0	0	0	0.3333	0.4000	0	0.3244	0.2933	0.0871	0.3233	0.2325	0.2273

In Table 5, the transient and the steady-state probabilities ${\psi'_i}^*[k]$ $(1 \leq i \leq 6)$ for the quotient EDTMC of the abstract shared memory system at the time moments k $(0 \leq k \leq 10)$ and $k = \infty$ are presented, and in Figure 18, the alteration diagram (evolution in time) for the transient probabilities is depicted.

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

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

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 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

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

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 $\mathcal{K}_2, \mathcal{K}_3, \mathcal{K}_5$. Then the steady-state probability that the memory is available is $\varphi'_2 + \varphi'_3 + \varphi'_5 = \frac{1}{17} + 0 + 0 = \frac{1}{17}$. 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 $\{r\}$. Thus, the *steady-state probability of the shared memory request from a processor* is $\varphi'_2 \sum_{\{A,\widetilde{\mathcal{K}} | \{r\} \in A,\ \mathcal{K}_2 \xrightarrow{A} \widetilde{\mathcal{K}}\}} PM_A(\mathcal{K}_2, \widetilde{\mathcal{K}}) + \varphi'_4 \sum_{\{A,\widetilde{\mathcal{K}} | \{r\} \in A,\ \mathcal{K}_4 \xrightarrow{A} \widetilde{\mathcal{K}}\}} PM_A(\mathcal{K}, \widetilde{\mathcal{K}}) = \frac{1}{17}\left(\frac{1}{2} + \frac{1}{4}\right) + \frac{6}{17}\left(\frac{3}{8} + \frac{1}{8}\right) = \frac{15}{68}$.

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 and second performance indices obviously illustrates the result of Proposition 8.1. The coincidence of the third performance index is due to Theorem 8.1: one should just apply its result to the 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.

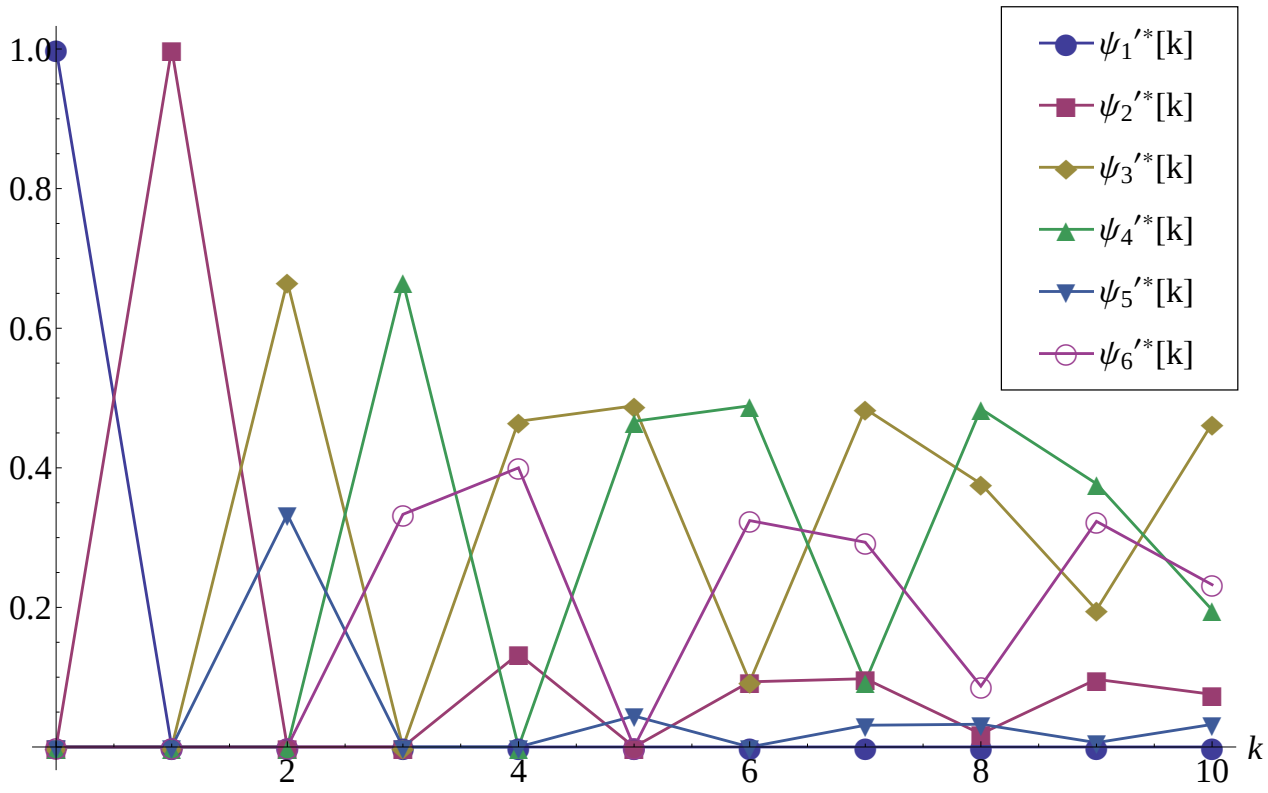

Figure 18: Transient probabilities alteration diagram for the quotient EDTMC of the abstract shared memory system

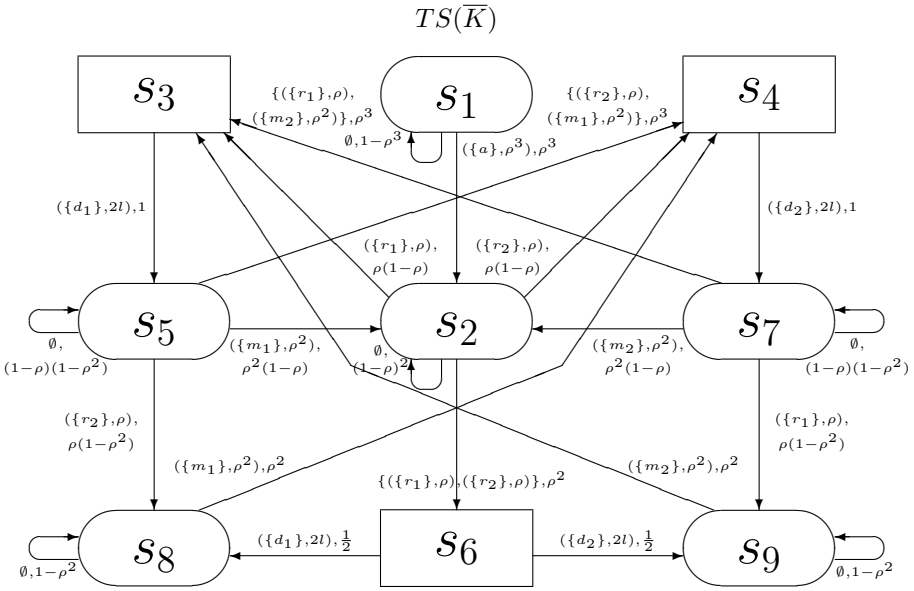

Figure 19: The transition system of the generalized shared memory system

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 ρ, and generalized weight l. 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

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

In Figure 19, the transition system $TS(\overline{K})$ is presented. In Figure 20, 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

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

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

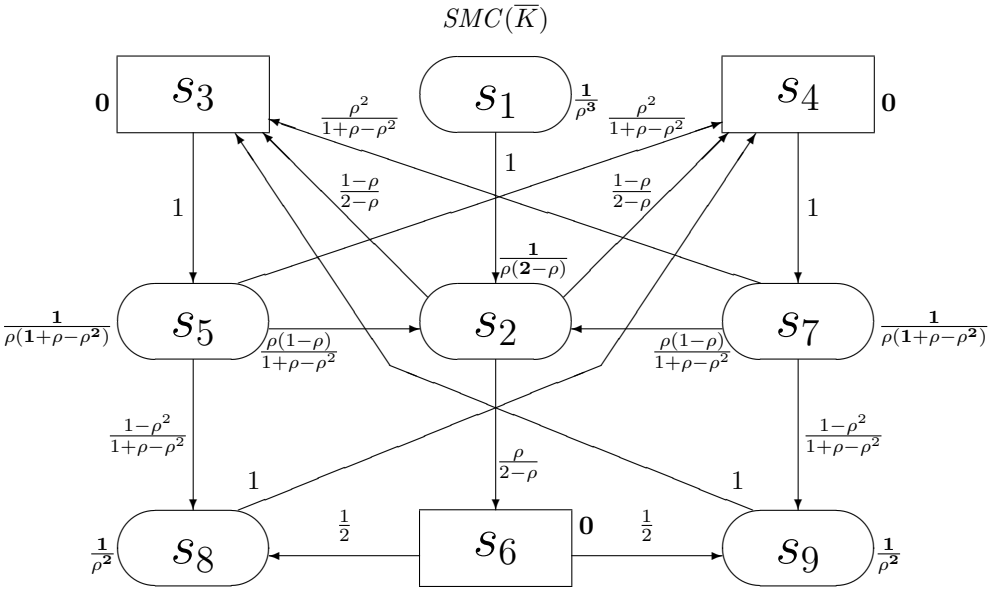

Figure 20: The underlying SMC of the generalized shared memory system

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

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 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 $\tilde{s}_2, \tilde{s}_3, \tilde{s}_4, \tilde{s}_6$. Then 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 $(\{r_1\}, \rho)$. Thus, the *steady-state probability of the shared memory request from the first processor* is $\tilde{\varphi}_2 \sum_{\{\Upsilon \mid (\{r_1\}, \frac{1}{2}) \in \Upsilon\}} PT(\Upsilon, \tilde{s}_2) + \tilde{\varphi}_7 \sum_{\{\Upsilon \mid (\{r_1\}, \frac{1}{2}) \in \Upsilon\}} PT(\Upsilon, \tilde{s}_7) = \frac{\rho^2(1-\rho)}{2+\rho-\rho^2-\rho^3}(\rho(1-\rho) + \rho(1-\rho)) + \frac{\rho(2-\rho)}{2(2+\rho-\rho^2-\rho^3)}(\rho(1-\rho^2) + \rho^3) = \frac{\rho^2(2+3\rho-8\rho^2+4\rho^3)}{2(2+\rho-\rho^2-\rho^3)}$.

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

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

$DR(\overline{L})$ resembles $DR(\overline{K})$, and $TS(\overline{L})$ is similar to $TS(\overline{K})$. We have $SMC(\overline{L}) = SMC(\overline{K})$. 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 and second 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 $(\{r\}, \rho)$. Thus, the *steady-state probability of the shared memory request from a processor* is $\tilde{\varphi}_2 \sum_{\{\Upsilon \mid (\{r\}, \rho) \in \Upsilon\}} PT(\Upsilon, \tilde{s}_2) + \tilde{\varphi}_5 \sum_{\{\Upsilon \mid (\{r\}, \rho) \in \Upsilon\}} PT(\Upsilon, \tilde{s}_5) + \tilde{\varphi}_7 \sum_{\{\Upsilon \mid (\{r\}, \rho) \in \Upsilon\}} PT(\Upsilon, \tilde{s}_7) = \frac{\rho^2(1-\rho)}{2+\rho-\rho^2-\rho^3}(\rho(1-\rho) + \rho(1-\rho) + \rho^2) + \frac{\rho(2-\rho)}{2(2+\rho-\rho^2-\rho^3)}(\rho(1-\rho^2) + \rho^3) + \frac{\rho(2-\rho)}{2(2+\rho-\rho^2-\rho^3)}(\rho(1-\rho^2) + \rho^3) = \frac{\rho^2(2-\rho)(1+\rho-\rho^2)}{2+\rho-\rho^2-\rho^3}$.

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).

In Figure 21, the quotient transition system $TS_{\underline{\leftrightarrow}_{ss}}(\overline{L})$ is presented. In Figure 22, 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

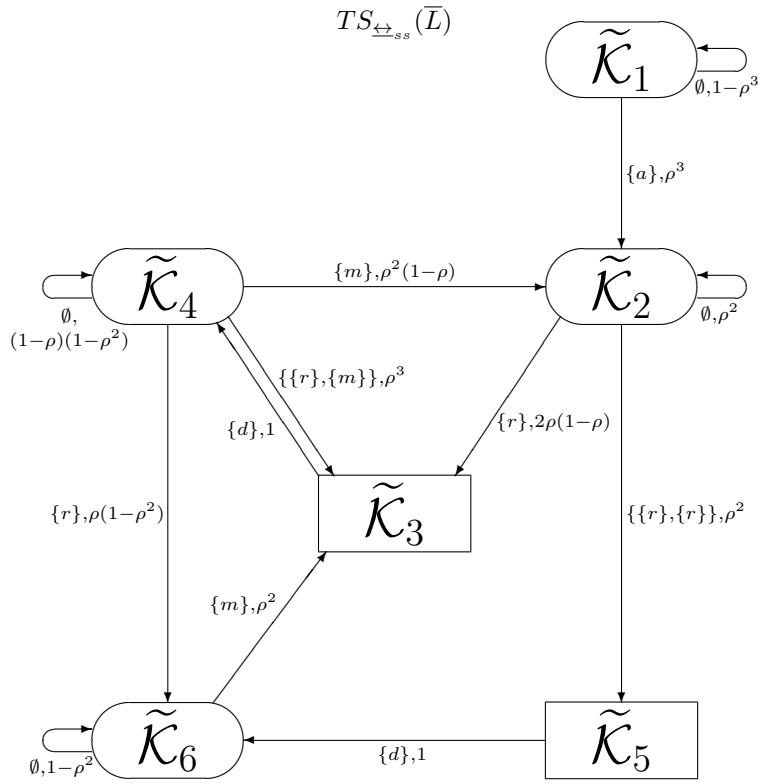

Figure 21: The quotient transition system of the abstract generalized shared memory system

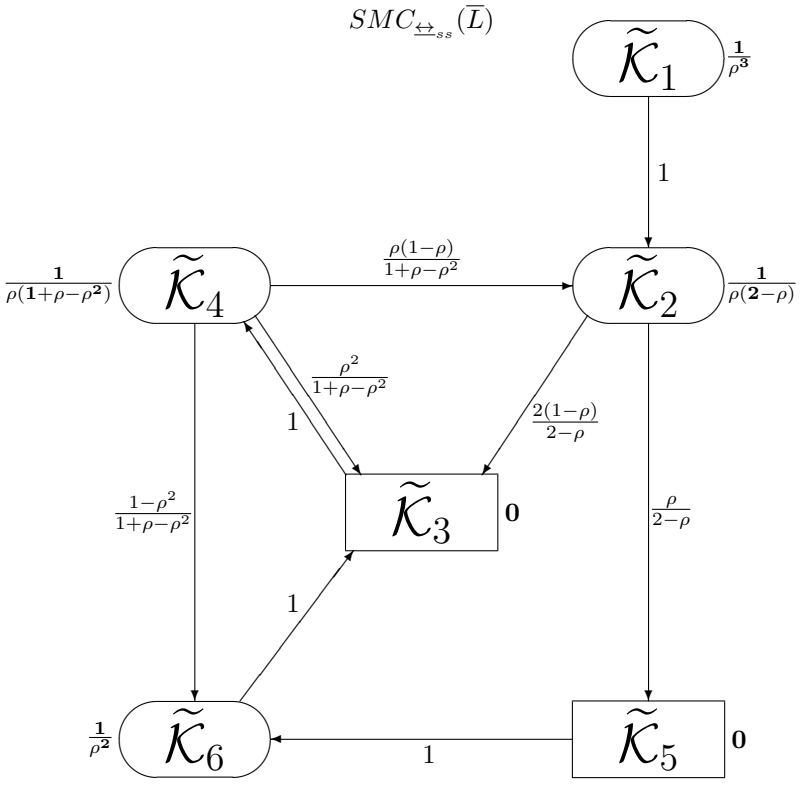

Figure 22: The quotient underlying SMC of the abstract generalized shared memory system

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

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

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

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 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

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

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 $\widetilde{\mathcal{K}}_2, \widetilde{\mathcal{K}}_3, \widetilde{\mathcal{K}}_5$. Then the steady-state probability that the memory is available is $\tilde{\varphi}'_2+\tilde{\varphi}'_3+\tilde{\varphi}'_5 = \frac{\rho^2(1-\rho)}{2+\rho-\rho^2-\rho^3}+0+0 = \frac{\rho^2(1-\rho)}{2+\rho-\rho^2-\rho^3}$. Thee 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 $\{r\}$. Thus, the *steady-state probability of the shared memory request from a processor* is $\tilde{\varphi}'_2 \sum_{\{A,\widetilde{\mathcal{K}} | \{r\} \in A,\ \mathcal{K}_2 \xrightarrow{A} \widetilde{\mathcal{K}}\}} PM_A(\widetilde{\mathcal{K}}_2, \widetilde{\mathcal{K}}) + \tilde{\varphi}'_4 \sum_{\{A,\widetilde{\mathcal{K}} | \{r\} \in A,\ \widetilde{\mathcal{K}}_4 \xrightarrow{A} \widetilde{\mathcal{K}}\}} PM_A(\widetilde{\mathcal{K}}, \widetilde{\mathcal{K}}) = \frac{\rho^2(1-\rho)}{2+\rho-\rho^2-\rho^3}(2\rho(1-\rho)+\rho^2) + \frac{\rho(2-\rho)}{2+\rho-\rho^2-\rho^3}(\rho(1-\rho^2)+\rho^3) = \frac{\rho^2(2-\rho)(1+\rho-\rho^2)}{2+\rho-\rho^2-\rho^3}$.

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 and second performance indices obviously illustrates the result of Proposition 8.1. The coincidence of the third performance index is due to Theorem 8.1: one should just apply its result to the 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 which effect have 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

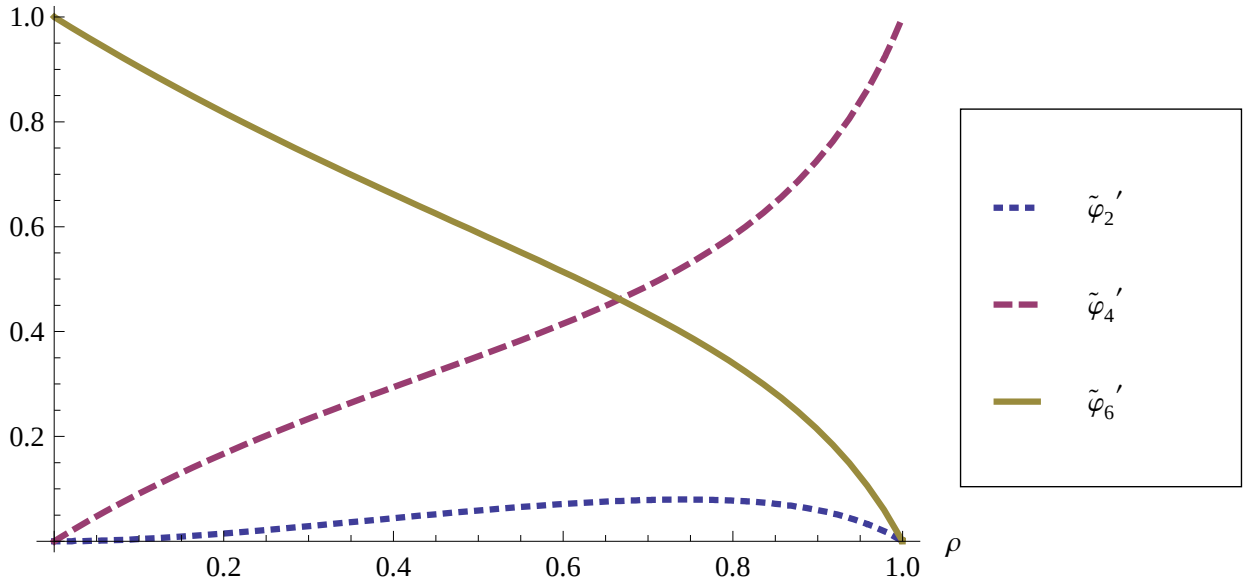

Figure 23: Alteration diagrams for the steady-state probabilities $\tilde{\varphi}'_2,\ \tilde{\varphi}'_4,\ \tilde{\varphi}'_6$ depending on the parameter ρ

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 23, the alteration diagrams are depicted for $\tilde{\varphi}'_2,\ \tilde{\varphi}'_4,\ \tilde{\varphi}'_6$ considered as the functions depending on ρ. 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 $\rho = 0.7433$. 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 24, the alteration diagram are depicted for the shared memory utilization calculated as $1 - \tilde{\varphi}'_2 - \tilde{\varphi}'_3 - \tilde{\varphi}'_5$ and considered as the function depending on 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 $\rho = 0.7433$. 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 influence of value ρ to the remaining performance indices presented before is investigated according to the same pattern as above.

10 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 thus preserves performance measures. A case study of the shared memory system has been presented as an example of modeling, performance evaluation and performance

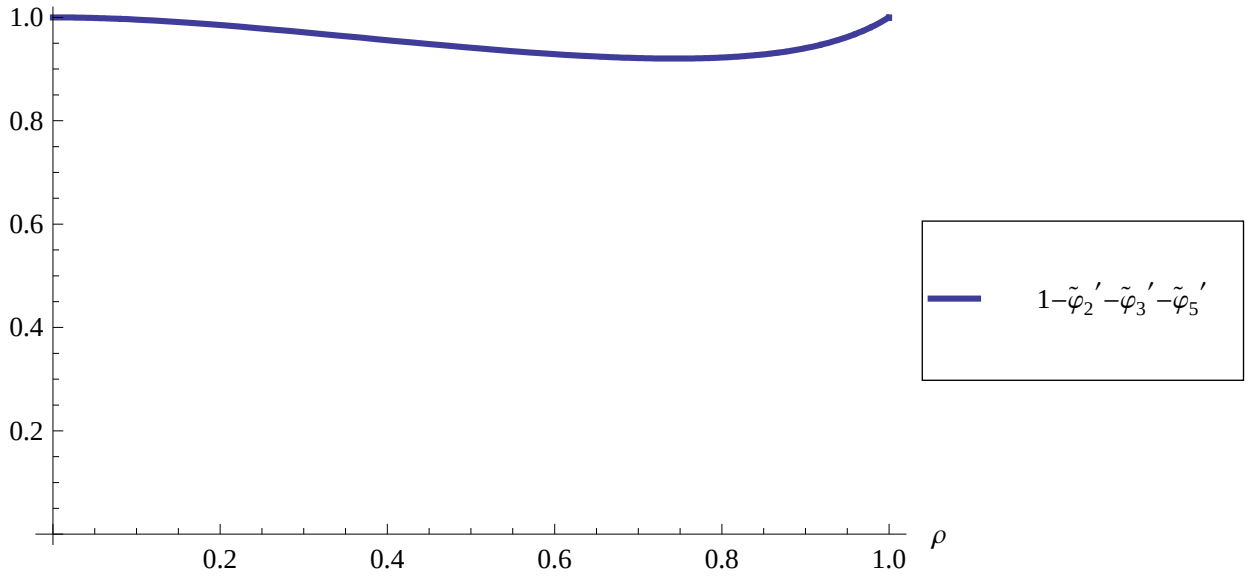

Figure 24: Alteration diagram for the shared memory utilization $1 - \tilde{\varphi}'_2 - \tilde{\varphi}'_3 - \tilde{\varphi}'_5$ depending on the parameter ρ

preserving reduction in the framework of the calculus.

Future work will consist in constructing a congruence relation 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 [31]. 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 operation could be added to dtsiPBC to increase further specification power of the algebra.

References

[1] Autant C., Schnoebelen Ph. *Place bisimulations in Petri nets. Lecture Notes in Computer Science* **616**, p. 45–61, June 1992.

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

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

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

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

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

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

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

[9] Buchholz P. *Markovian process algebra: composition and equivalence.* In: Herzog U. and Rettelbach M., editors, *Proceedings of* 2^{nd} *Workshop on Process Algebras and Performance Modelling, Arbeitsberichte des IMMD* **27**, p. 11–30, University of Erlangen, Germany, 1994.

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

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

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

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

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

[15] van Glabbeek R.J., Smolka S.A., Steffen B. *Reactive, generative, and stratified models of probabilistic processes. Information and Computation* **121(1)**, p. 59–80, 1995, `http://boole.stanford.edu/pub/prob.ps.gz`.

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

[17] Hoare C.A.R. *Communicating sequential processes.* Prentice-Hall, London, 1985.

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

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

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

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

[22] Macià H. *sPBC: Una extensión Markoviana del Petri box calculus. Ph.D. thesis*, 249 p., Departamento de Informática, Universidad de Castilla-La Mancha, Albacete, Spain, December 2003 (in Spanish), `http://www.info-ab.uclm.es/retics/publications/2003/sPBCthesis03.pdf`.

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

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

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

[26] Marroquín O., de-Frutos D. *Extending the Petri box calculus with time. Lecture Notes in Computer Science* **2075**, p. 303–322, 2001.

[27] Merlin P., Farber D.J. *Recoverability of communication protocols: implications of a theoretical study. IEEE Transactions on Communications* **24(9)**, p. 1036–1043, 1976.

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

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

[30] Macià H., Valero V., Cazorla D., Cuartero F. *Introducing the iteration in sPBC. Proceedings of the 24^{th} International Conference on Formal Techniques for Networked and Distributed Systems - 04 (FORTE'04)*, Madrid, Spain, *Lecture Notes in Computer Science* **3235**, p. 292–308, October 2004, `http://www.info-ab.uclm.es/retics/publications/2004/forte04.pdf`.

[31] Macià H., Valero V., Cuartero F., de-Frutos D. *A congruence relation for sPBC. Formal Methods in System Design* **32(2)**, p. 85–128, Springer, The Netherlands, April 2008.

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

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

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

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

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

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

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

[39] Tarasyuk I.V. *Stochastic Petri box calculus with discrete time. Fundamenta Informaticae* **76(1–2)**, p. 189–218, IOS Press, Amsterdam, The Netherlands, February 2007.

[40] Tarasyuk I.V. *Investigating equivalence relations in dtsPBC. Berichte aus dem Department für Informatik* **5/08**, 57 p., Carl von Ossietzky Universität Oldenburg, Germany, October 2008, `http://db.iis.nsk.su/persons/itar/dtspbcit_cov.pdf`.

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

[42] Zimmermann A., Freiheit J., Hommel G. *Discrete time stochastic Petri nets for modeling and evaluation of real-time systems. Proceedings of Workshop on Parallel and Distributed Real Time Systems*, San Francisco, USA, 6 p., 2001, `http://pdv.cs.tu-berlin.de/~azi/texte/WPDRTS01.pdf`.

A Proofs

A.1 Proof of Proposition 6.2

Like it has been done for strong equivalence in Proposition 8.2.1 from [16], we shall prove the following fact about step stochastic bisimulation. Let for some index set $\mathcal{J}$ we have $\forall j \in \mathcal{J},\ \mathcal{R}_j : G \underline{\leftrightarrow}_{ss} G'$. Then the transitive closure of the union of all the 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 over 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 $SJ(s_1) = SJ(s_2)$ and $\forall \mathcal{H} \in (DR(G) \cup DR(G'))/_{\mathcal{R}},\ \forall A \in I\!N_f^{\mathcal{L}},\ PM_A(s_1, \mathcal{H}) = PM_A(s_2, \mathcal{H})$.

- $n = 1$

  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 $SJ(s_1) = SJ(s_2)$ and $\forall \mathcal{H} \in (DR(G) \cup DR(G'))/_{\mathcal{R}},\ \forall A \in I\!N_f^{\mathcal{L}}$,

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

- $n \to n + 1$

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

  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 $SJ(s_1) = SJ(s_3)$ and $PM_A(s_1, \mathcal{H}) = PM_A(s_3, \mathcal{H})$. By the induction hypothesis, we get $SJ(s_3) = SJ(s_2)$ and $PM_A(s_3, \mathcal{H}) = PM_A(s_2, \mathcal{H})$. Thus, $SJ(s_1) = SJ(s_3) = SJ(s_2)$ and $\forall \mathcal{H} \in (DR(G) \cup DR(G'))/_{\mathcal{R}},\ \forall A \in I\!N_f^{\mathcal{L}}$,

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

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 above that $\mathcal{R}_{ss}(G, G') : G \underline{\leftrightarrow}_{ss} G'$. □

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 5.1, $\forall s \in DR_T(G),\ \varphi(s) = \psi(s)$ and $\forall s' \in DR_V(G'),\ \varphi'(s') = \psi'(s')$, where ψ and ψ' are the steady-state PMFs for $DTMC(G)$ and $DTMC(G')$, respectively. Thus, $\forall \mathcal{H} \in (DR_T(G) \cup DR_T(G'))/_{\mathcal{R}}$, $\sum_{s \in \mathcal{H} \cap DR(G)} \varphi(s) = \sum_{s \in \mathcal{H} \cap DR_T(G)} \varphi(s) = \sum_{s \in \mathcal{H} \cap DR_T(G)} \psi(s)$ and $\sum_{s' \in \mathcal{H} \cap DR(G')} \varphi'(s') = \sum_{s' \in \mathcal{H} \cap DR_T(G')} \varphi'(s') = \sum_{s' \in \mathcal{H} \cap DR_T(G')} \psi'(s')$.

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 one: $\forall \mathcal{H} \in (DR(G) \cup DR(G'))/_{\mathcal{R}},\ \sum_{s \in \mathcal{H} \cap DR(G)} \psi(s) = \sum_{s' \in \mathcal{H} \cap DR(G')} \psi'(s')$. 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.

- $k = 0$

  Note that the only nonzero values of the initial PMFs of $DTMC(G)$ and $DTMC(G')$ are $\psi[0]([G]_{\approx})$ and $\psi[0]([G']_{\approx})$. The only equivalence class containing $[G]_{\approx}$ or $[G']_{\approx}$ is $\mathcal{H}_0 = \{[G]_{\approx}, [G']_{\approx}\}$. Thus, $\sum_{s \in \mathcal{H}_0 \cap DR(G)} \psi[0](s) = \psi[0]([G]_{\approx}) = 1 = \psi'[0]([G']_{\approx}) = \sum_{s' \in \mathcal{H}_0 \cap DR(G')} \psi'[0](s')$.

  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 $\forall \widetilde{\mathcal{H}} \in (DR(G) \cup DR(G'))/_{\mathcal{R}},\ \forall A \in I\!N_f^{\mathcal{L}}$, $s_1 \xrightarrow{A}_{\mathcal{P}} \widetilde{\mathcal{H}} \ \Leftrightarrow\ s_2 \xrightarrow{A}_{\mathcal{P}} \widetilde{\mathcal{H}}$. Therefore, $PM(s_1, \widetilde{\mathcal{H}}) = \sum_{\{\Upsilon \mid \exists \tilde{s}_1 \in \widetilde{\mathcal{H}},\ s_1 \xrightarrow{\Upsilon} \tilde{s}_1\}} PT(\Upsilon, s_1) = \sum_{A \in I\!N_f^{\mathcal{L}}} \sum_{\{\Upsilon \mid \exists \tilde{s}_1 \in \widetilde{\mathcal{H}},\ s_1 \xrightarrow{\Upsilon} \tilde{s}_1,\ \mathcal{L}(\Upsilon) = A\}} PT(\Upsilon, s_1) = \sum_{A \in I\!N_f^{\mathcal{L}}} PM_A(s_1, \widetilde{\mathcal{H}}) = \sum_{A \in I\!N_f^{\mathcal{L}}} PM_A(s_2, \widetilde{\mathcal{H}}) = \sum_{A \in I\!N_f^{\mathcal{L}}} \sum_{\{\Upsilon \mid \exists \tilde{s}_2 \in \widetilde{\mathcal{H}},\ s_2 \xrightarrow{\Upsilon} \tilde{s}_2,\ \mathcal{L}(\Upsilon) = A\}} PT(\Upsilon, s_2) = \sum_{\{\Upsilon \mid \exists \tilde{s}_2 \in \widetilde{\mathcal{H}},\ s_2 \xrightarrow{\Upsilon} \tilde{s}_2\}} PT(\Upsilon, s_2) = PM(s_2, \widetilde{\mathcal{H}})$. 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,
$\sum_{\tilde{s}\in\widetilde{\mathcal{H}}\cap DR(G)}\psi[k+1](\tilde{s})=\sum_{\tilde{s}\in\widetilde{\mathcal{H}}\cap DR(G)}\sum_{s\in DR(G)}\psi[k](s)PM(s,\tilde{s})=$
$\sum_{s\in DR(G)}\sum_{\tilde{s}\in\widetilde{\mathcal{H}}\cap DR(G)}\psi[k](s)PM(s,\tilde{s})=\sum_{s\in DR(G)}\psi[k](s)\sum_{\tilde{s}\in\widetilde{\mathcal{H}}\cap DR(G)}PM(s,\tilde{s})=$
$\sum_{\mathcal{H}}\sum_{s\in\mathcal{H}\cap DR(G)}\psi[k](s)\sum_{\tilde{s}\in\widetilde{\mathcal{H}}\cap DR(G)}PM(s,\tilde{s})=$
$\sum_{\mathcal{H}}\sum_{s\in\mathcal{H}\cap DR(G)}\psi[k](s)\sum_{\tilde{s}\in\widetilde{\mathcal{H}}\cap DR(G)}\sum_{\{\Upsilon|s\xrightarrow{\Upsilon}\tilde{s}\}}PT(\Upsilon,s)=$
$\sum_{\mathcal{H}}\sum_{s\in\mathcal{H}\cap DR(G)}\psi[k](s)\sum_{\{\Upsilon|\exists\tilde{s}\in\widetilde{\mathcal{H}}\cap DR(G),\ s\xrightarrow{\Upsilon}\tilde{s}\}}PT(\Upsilon,s)=$
$\sum_{\mathcal{H}}\sum_{s\in\mathcal{H}\cap DR(G)}\psi[k](s)PM(s,\widetilde{\mathcal{H}})=\sum_{\mathcal{H}}\sum_{s\in\mathcal{H}\cap DR(G)}\psi[k](s)PM(\mathcal{H},\widetilde{\mathcal{H}})=$
$\sum_{\mathcal{H}}PM(\mathcal{H},\widetilde{\mathcal{H}})\sum_{s\in\mathcal{H}\cap DR(G)}\psi[k](s)=\sum_{\mathcal{H}}PM(\mathcal{H},\widetilde{\mathcal{H}})\sum_{s'\in\mathcal{H}\cap DR(G')}\psi'[k](s')=$
$\sum_{\mathcal{H}}\sum_{s'\in\mathcal{H}\cap DR(G')}\psi'[k](s')PM(\mathcal{H},\widetilde{\mathcal{H}})=\sum_{\mathcal{H}}\sum_{s'\in\mathcal{H}'\cap DR(G')}\psi'[k](s')PM(s',\widetilde{\mathcal{H}})=$
$\sum_{\mathcal{H}}\sum_{s'\in\mathcal{H}\cap DR(G')}\psi'[k](s')\sum_{\{\Upsilon|\exists\tilde{s}'\in\widetilde{\mathcal{H}}\cap DR(G'),\ s'\xrightarrow{\Upsilon}\tilde{s}'\}}PT(\Upsilon,s')=$
$\sum_{\mathcal{H}}\sum_{s'\in\mathcal{H}\cap DR(G')}\psi'[k](s')\sum_{\tilde{s}'\in\widetilde{\mathcal{H}}\cap DR(G')}\sum_{\{\Upsilon|\exists\tilde{s}',\ s'\xrightarrow{\Upsilon}\tilde{s}'\}}PT(\Upsilon,s')=$
$\sum_{\mathcal{H}}\sum_{s'\in\mathcal{H}\cap DR(G')}\psi'[k](s')\sum_{\tilde{s}'\in\widetilde{\mathcal{H}}\cap DR(G')}PM(s',\tilde{s}')=$
$\sum_{s'\in DR(G')}\psi'[k](s')\sum_{\tilde{s}'\in\widetilde{\mathcal{H}}\cap DR(G')}PM(s',\tilde{s}')=\sum_{s'\in DR(G')}\sum_{\tilde{s}'\in\widetilde{\mathcal{H}}\cap DR(G')}\psi'[k](s')PM(s',\tilde{s}')=$
$\sum_{\tilde{s}'\in\widetilde{\mathcal{H}}\cap DR(G')}\sum_{s'\in DR(G')}\psi'[k](s')PM(s',\tilde{s}')=\sum_{\tilde{s}'\in\widetilde{\mathcal{H}}\cap DR(G')}\psi'[k+1](\tilde{s}')$. □

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 $\forall\widetilde{\mathcal{H}}\in(DR(G)\cup DR(G'))/_{\mathcal{R}},\ \forall A\in\mathbb{N}_f^{\mathcal{L}},\ s\xrightarrow{A}_{\mathcal{P}}\widetilde{\mathcal{H}}\ \Leftrightarrow\ \bar{s}\xrightarrow{A}_{\mathcal{P}}\widetilde{\mathcal{H}}$. 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 step trace of G and G'. We have $\exists\mathcal{H}_0,\ldots,\exists\mathcal{H}_n\in(DR(G)\cup DR(G'))/_{\mathcal{R}},\ \mathcal{H}_0\xrightarrow{A_1}_{\mathcal{P}_1}\mathcal{H}_1\xrightarrow{A_2}_{\mathcal{P}_2}\cdots\xrightarrow{A_n}_{\mathcal{P}_n}\mathcal{H}_n$. 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 $\mathcal{H}_1,\ldots,\mathcal{H}_n$ is equal to the product of $\mathcal{P}_1,\ldots,\mathcal{P}_n$:

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

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

- $n=1$
  $\sum_{\{\Upsilon_1|s_0\xrightarrow{\Upsilon_1}s_1,\ \mathcal{L}(\Upsilon_1)=A_1,\ s_1\in\mathcal{H}_1\}}PT(\Upsilon_1,s_0)=PM_{A_1}(s_0,\mathcal{H}_1)=PM_{A_1}(\mathcal{H}_0,\mathcal{H}_1)$.
- $n\to n+1$
  $\sum_{\{\Upsilon_1,\ldots,\Upsilon_n,\Upsilon_{n+1}|s_0\xrightarrow{\Upsilon_1}\cdots\xrightarrow{\Upsilon_n}s_n\xrightarrow{\Upsilon_{n+1}}s_{n+1},\ \mathcal{L}(\Upsilon_i)=A_i,\ s_i\in\mathcal{H}_i\ (1\le i\le n+1)\}}\prod_{i=1}^{n+1}PT(\Upsilon_i,s_{i-1})=$
  $\sum_{\{\Upsilon_{n+1}|s_n\xrightarrow{\Upsilon_{n+1}}s_{n+1},\ \mathcal{L}(\Upsilon_{n+1})=A_{n+1},\ s_n\in\mathcal{H}_n,\ s_{n+1}\in\mathcal{H}_{n+1}\}}\sum_{\{\Upsilon_1,\ldots,\Upsilon_n|s_0\xrightarrow{\Upsilon_1}\cdots\xrightarrow{\Upsilon_n}s_n,\ \mathcal{L}(\Upsilon_i)=A_i,\ s_i\in\mathcal{H}_i\ (1\le i\le n)\}}$
  $\prod_{i=1}^{n}PT(\Upsilon_i,s_{i-1})PT(\Upsilon_{n+1},s_n)=$
  $\sum_{\{\Upsilon_1,\ldots,\Upsilon_n|s_0\xrightarrow{\Upsilon_1}\cdots\xrightarrow{\Upsilon_n}s_n,\ \mathcal{L}(\Upsilon_i)=A_i,\ s_i\in\mathcal{H}_i\ (1\le i\le n)\}}$
  $\left[\prod_{i=1}^{n}PT(\Upsilon_i,s_{i-1})\sum_{\{\Upsilon_{n+1}|s_n\xrightarrow{\Upsilon_{n+1}}s_{n+1},\ \mathcal{L}(\Upsilon_{n+1})=A_{n+1},\ s_n\in\mathcal{H}_n,\ s_{n+1}\in\mathcal{H}_{n+1}\}}PT(\Upsilon_{n+1},s_n)\right]=$
  $\sum_{\{\Upsilon_1,\ldots,\Upsilon_n|s_0\xrightarrow{\Upsilon_1}\cdots\xrightarrow{\Upsilon_n}s_n,\ \mathcal{L}(\Upsilon_i)=A_i,\ s_i\in\mathcal{H}_i\ (1\le i\le n)\}}\prod_{i=1}^{n}PT(\Upsilon_i,s_{i-1})PM_{A_{n+1}}(s_n,\mathcal{H}_{n+1})=$
  $\sum_{\{\Upsilon_1,\ldots,\Upsilon_n|s_0\xrightarrow{\Upsilon_1}\cdots\xrightarrow{\Upsilon_n}s_n,\ \mathcal{L}(\Upsilon_i)=A_i,\ s_i\in\mathcal{H}_i\ (1\le i\le n)\}}\prod_{i=1}^{n}PT(\Upsilon_i,s_{i-1})PM_{A_{n+1}}(\mathcal{H}_n,\mathcal{H}_{n+1})=$
  $PM_{A_{n+1}}(\mathcal{H}_n,\mathcal{H}_{n+1})\sum_{\{\Upsilon_1,\ldots,\Upsilon_n|s_0\xrightarrow{\Upsilon_1}\cdots\xrightarrow{\Upsilon_n}s_n,\ \mathcal{L}(\Upsilon_i)=A_i,\ s_i\in\mathcal{H}_i\ (1\le i\le n)\}}\prod_{i=1}^{n}PT(\Upsilon_i,s_{i-1})=$
  $PM_{A_{n+1}}(\mathcal{H}_n,\mathcal{H}_{n+1})\prod_{i=1}^{n}PM_{A_i}(\mathcal{H}_{i-1},\mathcal{H}_i)=\prod_{i=1}^{n+1}PM_{A_i}(\mathcal{H}_{i-1},\mathcal{H}_i)$.

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

Since we have the previous equality for all $s_0, \bar{s}_0 \in \mathcal{H}_0$, we can denote $PT(A_1 \cdots A_n, \mathcal{H}_0) = PT(A_1 \cdots A_n, s_0) = PT(A_1 \cdots A_n, \bar{s}_0)$.

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