Discrete time stochastic Petri box calculus with immediate multiactions

Igor V. Tarasyuk[a,1,3], Hermenegilda Macià[b,2,4] and Valentín Valero[b,2,5]

[a] *A.P. Ershov Institute of Informatics Systems SB RAS, Novosibirsk, Russian Federation*

[b] *High School of Computer Science Engineering, UCLM, Albacete, Spain*

Abstract

We propose discrete time stochastic Petri box calculus extended with immediate multiactions, called dtsiPBC. The step operational semantics is constructed via labeled probabilistic transition systems. The denotational semantics is defined via labeled discrete time stochastic Petri nets with immediate transitions (LDTSIPNs). A consistency of both semantics is demonstrated. In order to evaluate performance, the corresponding semi-Markov chains are analyzed. In a case study, performance of the shared memory system is evaluated.

Keywords: Stochastic process algebra, Petri box calculus, discrete time, immediate multiaction, probabilistic transition system, LDTSIPN, performance evaluation, shared memory system.

1 Introduction

Algebraic process calculi are a recognized formal model for specification of computing systems and analysis of their behaviour. Petri Box Calculus (PBC) [2] is a flexible and expressive process algebra developed as a tool for specification of Petri nets 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 Petri nets. PBC has a step operational semantics in terms of labeled transition systems. Its denotational

[1] Supported by Deutsche Forschungsgemeinschaft (DFG), 436 RUS 113/1002/01.

[2] Supported by Spanish government (co-financed by FEDER founds), the project "Modeling and analyzing composite Web services using formal methods", TIN2009-14312-C02-02.

[3] Email:itar@iis.nsk.su

[4] Email:Hermenegilda.Macia@uclm.es

[5] Email:Valentin.Valero@uclm.es

This paper is electronically published in
Electronic Notes in Theoretical Computer Science
URL: www.elsevier.nl/locate/entcs

semantics was proposed in terms of a subclass of Petri nets (PNs) equipped with interface and considered up to isomorphism called Petri boxes.

There are timed extensions of PBC considering a deterministic time model: time Petri box calculus (tPBC) [6], in which an interleaving semantics is considered and *actions* have a time interval associated; timed Petri box calculus (TPBC) [10], where a step semantics is considered and multiactions have time durations associated, and arc time Petri box calculus (atPBC) [14], in which a step semantics is also considered and time intervals for the multiaction delays. There are also stochastic extensions of PBC: stochastic Petri box calculus (sPBC) [7,8], with a continuous time model and multiaction delays that follow a negative exponential distribution. A discrete time stochastic extension of finite PBC was presented in [15], dtsPBC, providing a step operational semantics and a denotational semantics based on *dts-boxes*, a subclass of labeled discrete time stochastic PNs (LDTSPNs).

In this paper, dtsPBC is extended with the iteration operator and immediate multiactions. This new language, *discrete time stochastic and immediate Petri box calculus* (dtsiPBC), is a discrete time analog of sPBC. Immediate multiactions increase the specification capability: they can model instant probabilistic choices and activities with negligible durations. They are also used to specify urgent activities and the ones, which not relevant for performance evaluation. In many cases, they result in a more clear system representation. We define a step operational semantics by using labeled probabilistic transition systems, and a corresponding denotational semantics in terms of a subclass of LDTSPNs with immediate transitions (LDTSIPNs), called *dtsi-boxes*. Consistency of both semantics is then demonstrated. The corresponding stochastic process, semi-Markov chain (SMC), is constructed and investigated, with the purpose of performance evaluation. At last, a case study of a system with two processors and a common shared memory explains how to model and analyze performance of concurrent systems with dtsiPBC.

There are many well-known works related to stochastic extensions of process algebras (SPAs). Due to the lack of space we can only mention a few of them, as MTIPP [4], PEPA [5] and EMPA [1]. The first difference between dtsiPBC and these classical SPAs comes from PBC, since dtsiPBC is based on that calculus: all operations and a notion of multiaction are inherited from PBC. The second difference is discrete conditional probabilities of activities in dtsiPBC due to its discrete time semantics, whereas the action rates are used in the standard SPAs with continuous time semantics. Discrete time operational semantics of dtsiPBC allows for concurrent execution of activities in steps. In continuous time semantics, concurrency is simulated by interleaving, since simultaneous occurrence of any two events has zero probability according to the properties of continuous probability distributions. The third difference are immediate multiactions in dtsiPBC which have the same priority while

immediate actions in EMPA can have different priorities. There exist no immediate actions in MTIPP and PEPA. There is a recent work by Markovski and de Vink [9], where a SPA with discrete time is defined, providing for it an interleaving semantics, but in this work immediate actions are not considered.

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. In Section 6, an illustrative example of the shared memory system is presented and investigated as a case study. Section 7 summarizes the results obtained and outlines research perspectives.

2 Syntax

We denote the *set of all finite multisets* over a set X by $\mathbb{N}_f^X$ and the *set of all subsets* of X 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)* s.t. $a \neq \hat{a}$ 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 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. These 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. On the other hand, notice that zero probabilities are not allowed for 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. These are clearly identifiable from stochastic multiactions, because of the natural number instead of a real number in the interval $(0; 1)$. Stochastic and immediate multiactions cannot be executed together in some concurrent step, i.e. the steps can only consist either of stochastic or immediate multiactions, the latter having a priority over stochastic ones. Thus, in a state where both kinds of multiactions can occur, immediate multiactions always occur before stochastic ones. Let $\mathcal{IL}$ be the set of *all immediate multiactions*.

Notice that the same multiaction $\alpha \in \mathcal{L}$ may have different probabilities and weights in the same specification. 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 $\Omega(\alpha, \kappa) = \kappa$.

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, iteration 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 the standard process calculi. Relabeling functions $f : \mathcal{A} \to \mathcal{A}$ are bijections preserving conjugates, i.e. $\forall x \in \mathcal{A},\ f(\hat{x}) = \widehat{f(x)}$. Relabeling is extended to multiactions: for $\alpha \in \mathcal{L}$, we define $f(\alpha) = \sum_{x \in \alpha} f(x)$. Relabeling is extended to multisets of activities: for $\Upsilon \in \mathbb{N}_f^{\mathcal{SIL}}$, we define $f(\Upsilon) = \sum_{(\alpha,\kappa)\in\Upsilon}(f(\alpha),\kappa)$. Restriction over an action $a \in Act$ means that any process behaviour containing a or $\hat{a}$ is not allowed.

Synchronization of multiactions is defined for multiactions belonging to the same class (stochastic or immediate). Taking into account this requirement, let $\alpha, \beta \in \mathcal{L}$, and $a \in Act$ s.t. $a \in \alpha$ and $\hat{a} \in \beta$ or $\hat{a} \in \alpha$ and $a \in \beta$. Synchronization of α and β by a is defined as $\alpha \oplus_a \beta = \gamma$, where

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

As in PBC, static expressions specify the structure of processes and correspond to unmarked LDTSIPNs.

Definition 2.1 Let $(\alpha, \kappa) \in \mathcal{SIL}$, $a \in Act$. A *static expression* of dtsiPBC is

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

StatExpr will denote the set of *all static expressions* of dtsiPBC.

A restriction must be introduced to avoid inconsistency of the iteration operator. We do not allow any concurrency at the highest level of the second argument of iteration. This is not a severe restriction, since we can prefix parallel expressions by an activity with the empty multiaction. The mentioned inconsistency can result in non-safe nets [3].

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

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

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

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

Dynamic expressions specify process states, and correspond to LDTSIPNs (marked by definition). Dynamic expressions are obtained from static ones which are annotated with upper or lower bars and specify active components of the system at the current instant. $\overline{E}$ denotes the *initial*, $\underline{E}$ 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 the bars from it.

Definition 2.3 Let $E \in StatExpr$, $a \in Act$. A *dynamic expression* of dtsiPBC is

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

DynExpr will denote the set of *all dynamic expressions* of dtsiPBC. If the underlying static expression of a dynamic one is not regular, the corresponding LDTSIPN can be non-safe (but it is 2-bounded in the worst case [3]). A dynamic expression is *regular* if its underlying static expression is regular. *RegDynExpr* denotes the set of *all regular dynamic expressions* of dtsiPBC.

3 Operational semantics

Inaction Rules. These describe structural transformations for dynamic expressions, but not changing 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. These transformations do not take any time, and their application to a dynamic expression will not modify the corresponding marking in the associated LDTSIPN. No transitions are therefore fired in relation with these transformations. In Table 1, we define the inaction rules for the regular dynamic expressions in the form of overlined and underlined static ones, where $E, F, K \in \mathit{RegStatExpr}$ and $a \in \mathit{Act}$. Inaction rules for arbitrary regular dynamic expressions are defined in Table 2, where $E, F \in \mathit{RegStatExpr}$, $a \in \mathit{Act}$ and $G, H, \widetilde{G}, \widetilde{H} \in \mathit{RegDynExpr}$.

Table 1
Inaction rules for overlined and underlined regular static expressions

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

Table 2
Inaction rules for arbitrary regular dynamic expressions

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

A regular dynamic expression G is *operative* if no inaction rule can be applied to it. *OpRegDynExpr* denotes 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. We shall 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.

Action and empty loop rules. With action rules the execution of activities is captured. The prioritization of immediate multiactions w.r.t. stochastic ones is also captured by these action rules. We also have the *empty loop rule*, which is used to capture a delay of one time unit at any state when no immediate multiactions are executable. In this case, the empty multiset of activities is considered to be executed. Action rules with stochastic multiactions define dynamic expression transformations due to the execution of non-empty multisets of stochastic multiactions, and are time consuming, they take one time unit, whereas action rules with immediate multiactions define instantaneous dynamic expression transformations due to the execution of non-empty multisets of immediate multiactions. Action rules with either stochastic or immediate multiactions respectively correspond to stochastic or immediate transition firings in the corresponding LDTSIPN. The firing of a set of stochastic transitions is time consuming, one time unit elapses with their firing, whereas immediate transitions take no time in their firing.

With the empty loop rule $G \xrightarrow{\emptyset} G$ (rule **El** in Table 3) we capture the possibility to stay at a *tangible state* (only stochastic movements are possible) without firing any activities. This is defined as an empty movement that takes one time unit. This rule reflects a non-zero probability to stay at the current state at the next time moment, which is an essential feature of discrete time stochastic processes. This is a new rule that has no prototype among inaction rules of PBC, since it represents a time delay. The PBC rule $G \xrightarrow{\emptyset} G$ from [3] in our setting would correspond to a rule $G \Rightarrow G$, but notice that our model is strongly based on the transformation of dynamic expressions into operative ones by the bars movements, hence, we do not introduce it in dtsiPBC.

Thus, an application of every action rule with stochastic multiactions or the empty loop rule requires one 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 instant.

Expressions of dtsiPBC can contain identical activities. Thus, 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 obtained from synchronization will be annotated with concatenation of numberings of the activities they come from, hence, the numbering we use has a tree structure to reflect the effect of multiple synchronizations. But notice that the new activities resulting from synchronizations in different orders should be considered up to permutation of their numbering. In this way, we can recognize different instances of the same activity.

Due to the lack of space we omit a formalization of the numbering mechanism, which is straightforward. From now onwards, we will assume that the identical activities are enumerated when needed to avoid ambiguity. This enumeration is considered to be implicit.

Let $\mathcal{E} \subseteq X^2$ be an equivalence relation on a set X. The *equivalence class* (w.r.t. $\mathcal{E}$) of $x \in X$ is $[x]_{\mathcal{E}} = \{y \in X \mid (x,y) \in \mathcal{E}\}$. The equivalence $\mathcal{E}$ partitions X into the *set of equivalence classes* $X/_{\mathcal{E}} = \{[x]_{\mathcal{E}} \mid x \in X\}$.

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

Let $G \in OpRegDynExpr$. We now define the *set of all sets of 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 $a \in Act$.

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

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

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

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

(v) If $\Upsilon \in Can(G\ \mathsf{sy}\ a)$ and $(\alpha,\kappa),(\beta,\lambda) \in \Upsilon$ are different, $a \in \alpha, \hat{a} \in \beta$ then
  (a) $(\Upsilon + \{(\alpha \oplus_a \beta, \kappa \cdot \lambda)\}) \setminus \{(\alpha,\kappa),(\beta,\lambda)\} \in Can(G\ \mathsf{sy}\ a)$, 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 $\kappa, \lambda \in \mathbb{N} \setminus \{0\}$.

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.

By definition of $Can(G)$, $\Upsilon \in Can(G)$ implies $\forall \Xi \subseteq \Upsilon,\ \Xi \neq \emptyset,\ \Xi \in Can(G)$. 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. $\forall \Upsilon \in Can(G),\ \Upsilon \in \mathbb{N}_f^{\mathcal{SL}}$. 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 $Can(G)$, i.e. $\exists \Upsilon \in Can(G),\ \Upsilon \in \mathbb{N}_f^{\mathcal{IL}} \setminus \{\emptyset\}$. Clearly, 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 must be executed first.

In Table 3, we define the action and empty loop rules, where $(\alpha,\rho),(\beta,\chi)\in\mathcal{SL}$, $(\alpha,l),(\beta,m)\in\mathcal{IL}$, $(\alpha,\kappa)\in\mathcal{SIL}$. Further, $E,F\in RegStatExpr, G,H\in OpRegDynExpr$, $\widetilde{G},\widetilde{H}\in RegDynExpr$, $a\in Act$. Moreover, $\Gamma,\Delta\in\mathbb{N}_f^{\mathcal{SL}}\setminus\{\emptyset\}$, $\Gamma'\in\mathbb{N}_f^{\mathcal{SL}}$, $I,J\in\mathbb{N}_f^{\mathcal{IL}}\setminus\{\emptyset\}$, $I'\in\mathbb{N}_f^{\mathcal{IL}}$ and $\Upsilon\in\mathbb{N}_f^{\mathcal{SIL}}\setminus\{\emptyset\}$. The names of the action rules with immediate multiactions have suffix '**i**'.

Table 3
Action and empty loop rules

El $\dfrac{tang(G)}{G\stackrel{\emptyset}{\to}G}$ **B** $\overline{(\alpha,\kappa)}\stackrel{\{(\alpha,\kappa)\}}{\longrightarrow}\underline{(\alpha,\kappa)}$ **S** $\dfrac{G\stackrel{\Upsilon}{\to}\widetilde{G}}{G;E\stackrel{\Upsilon}{\to}\widetilde{G};E\ \ E;G\stackrel{\Upsilon}{\to}E;\widetilde{G}}$ **L** $\dfrac{G\stackrel{\Upsilon}{\to}\widetilde{G}}{G[f]\stackrel{f(\Upsilon)}{\longrightarrow}\widetilde{G}[f]}$

Rs $\dfrac{G\stackrel{\Upsilon}{\to}\widetilde{G},\ a,\hat{a}\notin\mathcal{A}(\Upsilon)}{G\ \mathsf{rs}\ a\stackrel{\Upsilon}{\to}\widetilde{G}\ \mathsf{rs}\ a}$ **C** $\dfrac{G\stackrel{\Gamma}{\to}\widetilde{G},\ \neg init(G)\vee(init(G)\wedge tang(\overline{E}))}{G[]E\stackrel{\Gamma}{\to}\widetilde{G}[]E\ \ E[]G\stackrel{\Gamma}{\to}E[]\widetilde{G}}$

Ci $\dfrac{G\stackrel{I}{\to}\widetilde{G}}{G[]E\stackrel{I}{\to}\widetilde{G}[]E\ \ E[]G\stackrel{I}{\to}E[]\widetilde{G}}$ **P1** $\dfrac{G\stackrel{\Gamma}{\to}\widetilde{G},\ tang(H)}{G\|H\stackrel{\Gamma}{\to}\widetilde{G}\|H\ \ H\|G\stackrel{\Gamma}{\to}H\|\widetilde{G}}$

P1i $\dfrac{G\stackrel{I}{\to}\widetilde{G}}{G\|H\stackrel{I}{\to}\widetilde{G}\|H\ \ H\|G\stackrel{I}{\to}H\|\widetilde{G}}$ **P2** $\dfrac{G\stackrel{\Gamma}{\to}\widetilde{G},\ H\stackrel{\Delta}{\to}\widetilde{H},\ tang(G)\wedge tang(H)}{G\|H\stackrel{\Gamma+\Delta}{\longrightarrow}\widetilde{G}\|\widetilde{H}}$ **P2i** $\dfrac{G\stackrel{I}{\to}\widetilde{G},\ H\stackrel{J}{\to}\widetilde{H}}{G\|H\stackrel{I+J}{\longrightarrow}\widetilde{G}\|\widetilde{H}}$

I1 $\dfrac{G\stackrel{\Upsilon}{\to}\widetilde{G}}{[G*E*F]\stackrel{\Upsilon}{\to}[\widetilde{G}*E*F]}$ **I2** $\dfrac{G\stackrel{\Gamma}{\to}\widetilde{G},\ \neg init(G)\vee(init(G)\wedge tang(\overline{F}))}{[E*G*F]\stackrel{\Gamma}{\to}[E*\widetilde{G}*F]}$

I2i $\dfrac{G\stackrel{I}{\to}\widetilde{G}}{[E*G*F]\stackrel{I}{\to}[E*\widetilde{G}*F]}$ **I3** $\dfrac{G\stackrel{\Gamma}{\to}\widetilde{G},\ \neg init(G)\vee(init(G)\wedge tang(\overline{F}))}{[E*F*G]\stackrel{\Gamma}{\to}[E*F*\widetilde{G}]}$ **I3i** $\dfrac{G\stackrel{I}{\to}\widetilde{G}}{[E*F*G]\stackrel{I}{\to}[E*F*\widetilde{G}]}$

Sy1 $\dfrac{G\stackrel{\Upsilon}{\to}\widetilde{G}}{G\ \mathsf{sy}\ a\stackrel{\Upsilon}{\to}\widetilde{G}\ \mathsf{sy}\ a}$ **Sy2** $\dfrac{G\ \mathsf{sy}\ a\xrightarrow{\Gamma'+\{(\alpha,\rho)\}+\{(\beta,\chi)\}}\widetilde{G}\ \mathsf{sy}\ a,\ a\in\alpha,\ \hat{a}\in\beta,\ tang(G\ \mathsf{sy}\ a)}{G\ \mathsf{sy}\ a\xrightarrow{\Gamma'+\{(\alpha\oplus_a\beta,\rho\cdot\chi)\}}\widetilde{G}\ \mathsf{sy}\ a}$

Sy2i $\dfrac{G\ \mathsf{sy}\ a\xrightarrow{I'+\{(\alpha,l)\}+\{(\beta,m)\}}\widetilde{G}\ \mathsf{sy}\ a,\ a\in\alpha,\ \hat{a}\in\beta}{G\ \mathsf{sy}\ a\xrightarrow{I'+\{(\alpha\oplus_a\beta,l+m)\}}\widetilde{G}\ \mathsf{sy}\ a}$

Rule **Sy2** establishes that the synchronization of two stochastic multiactions is made by taking the product of their probabilities, since we are considering that both must occur for the synchronization to happen, so this corresponds to the probability of the event 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 step as many synchronized immediate multiactions as possible to get more progress in behaviour, this aspect will be used later, while performance evaluation based on the embedded discrete time Markov chains. 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 many technical difficulties, see [3].

Transition systems. Now we construct labeled probabilistic transition systems of dynamic expressions to define later the operational semantics.

Definition 3.2 The *derivation set* $DR(G)$ of a dynamic expression G is the minimal set s.t. $[G]_{\approx} \in DR(G)$ or, if $[H]_{\approx} \in DR(G)$ and $\exists \Gamma,\ H \xrightarrow{\Gamma} \widetilde{H}$, 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 $Exec(s) = \{\Upsilon \mid \exists H \in s,\ \exists \widetilde{H},\ H \xrightarrow{\Upsilon} \widetilde{H}\}$. Note that if $\Upsilon \in Exec(s)$, then $\exists H \in s,\ \Upsilon \in Can(H)$. The state s is *tangible*, if $Exec(s) \subseteq \mathbb{N}_f^{\mathcal{SL}}$. For tangible states we may have $Exec(s) = \emptyset$. Otherwise, the state s is *vanishing*, and in this case $Exec(s) \subseteq \mathbb{N}_f^{\mathcal{IL}} \setminus \{\emptyset\}$. 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 $DR_V(G)$. Clearly, $DR(G) = DR_T(G) \uplus DR_V(G)$, where $\uplus$ 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 which is ready for execution in* s is $PF(\Upsilon, s) = \begin{cases} \prod_{(\alpha,\rho)\in\Upsilon} \rho \cdot \prod_{\{\{(\beta,\chi)\}\in Exec(s) \mid (\beta,\chi)\notin\Upsilon\}} (1-\chi), & s \in DR_T(G); \\ \sum_{(\alpha,l)\in\Upsilon} l, & s \in DR_V(G). \end{cases}$

For $\Upsilon = \emptyset$ and $s \in DR_T(G)$, let $PF(\emptyset, s) = \begin{cases} \prod_{\{(\beta,\chi)\}\in Exec(s)} (1-\chi), & Exec(s) \neq \emptyset; \\ 1, & Exec(s) = \emptyset. \end{cases}$

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 independent 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 execution of immediate multiactions takes no time, we prefer to execute in a step as many parallel immediate multiactions as possible to get more progress in behaviour of the embedded discrete time Markov chains of expressions while performance evaluation. This reasoning is the same as that used to define the probability of synchronized immediate multiactions in the rule **Sy2i**. The definition of $PF(\Gamma, s)$ (and those of other probability functions we shall present) is based on the (implicit) enumeration of activities.

Let $\Upsilon \in Exec(s)$. The *probability to execute the set of activities* Υ *in* s is $PT(\Upsilon, s) = \frac{PF(\Upsilon,s)}{\sum_{\Xi\in Exec(s)} PF(\Xi,s)}$. 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 is a sum, since it reflects the probability of the mutually exclusive event union.

If s is tangible, then $PT(\emptyset, s) \in (0; 1]$, hence, there is a non-zero probability to stay at the state s in the next time moment, and the residence time in s is at least 1 time unit. Observe that $\forall s \in DR(G)$, $\sum_{\Upsilon \in Exec(s)} PT(\Upsilon, s) = 1$, by definition of $PT(\Upsilon, s)$; hence, it defines a probability distribution.

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

Definition 3.3 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 $\mathcal{T}_G = \{(s, (\Upsilon, PT(\Upsilon, s)), \tilde{s}) \mid s \in DR(G),\ \exists H \in s,\ \exists \widetilde{H} \in \tilde{s},\ H \xrightarrow{\Upsilon} \widetilde{H}\}$;
- the *initial state* is $s_G = [G]_{\approx}$.

The definition of $TS(G)$ is correct: 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 [16], since 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 the activities included) 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 discrete time moments with some (one-step) probability and consist of sets of activities. Every step consisting of stochastic multiactions or the empty step (i.e. that consisting of the empty set of activities) occurs 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. 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 $(s, (\Upsilon, \mathcal{P}), \tilde{s}) \in \mathcal{T}_G$ is written as $s \xrightarrow{\Upsilon}_{\mathcal{P}} \tilde{s}$, interpreted as the probability to change s to $\tilde{s}$ by executing Υ is $\mathcal{P}$.

For tangible states, Υ can be the empty set, and its execution does not change the current state (i.e. the equivalence class), since we 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 at the current instant.

The step probabilities belong to the interval $(0;1]$, being 1 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 there is just one transition from a vanishing state.

We write $s \xrightarrow{\Upsilon} \tilde{s}$ if $\exists \mathcal{P},\ s \xrightarrow{\Upsilon}_{\mathcal{P}} \tilde{s}$ and $s \to \tilde{s}$ if $\exists \Upsilon,\ s \xrightarrow{\Upsilon} \tilde{s}$. For $E \in \mathit{RegStatExpr}$, let $TS(E) = TS(\overline{E})$.

Example 3.4 The expression $\mathsf{Stop} = (\{g\}, \frac{1}{2})\ \mathsf{rs}\ g$ specifies a non-terminating process that is only able to perform empty loops with probability 1.
Let $E = [(\{a\}, \rho) * ((\{b\}, \chi); ((((\{c\}, l); (\{d\}, \theta))[]((\{e\}, m); (\{f\}, \phi)))) * \mathsf{Stop}]$.

In Figure 1 we can see its corresponding $TS(\overline{E})$. $DR(\overline{E})$ consists of the five equivalence classes, where $DR_T(\overline{E}) = \{s_1, s_2, s_4, s_5\}$ and $DR_V(\overline{E}) = \{s_3\}$.

4 Denotational semantics

Labeled DTSIPNs. Let us introduce a class of labeled discrete time stochastic and immediate Petri nets. Let us present a formal definition of LDTSIPNs.

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

- P_N and $T_N = Ts_N \uplus Ti_N$ are sets of *places* and *stochastic and immediate transitions*, s.t. $P_N \cup T_N \neq \emptyset$ and $P_N \cap T_N = \emptyset$. Let $M \in \mathbb{N}_f^{P_N}$ be markings.
- $W_N : (P_N \times T_N) \cup (T_N \times P_N) \to \mathbb{N}$ 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 *probability/weight* function associating stochastic transitions with probabilities and immediate ones with weights;
- $L_N : T_N \to \mathcal{L}$ is the *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, square boxes of normal thickness depict stochastic transitions, and those with thick borders represent immediate transitions. 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 $U^\bullet = \sum_{t \in U} t^\bullet$. Immediate transitions have a priority over stochastic ones, thus they fire first if they can. A transition $t \in T_N$ is *enabled* at marking M if ${}^\bullet t \subseteq M$, and one of the following holds: $t \in Ti_N$ or $\forall u \in T_N,\ {}^\bullet u \subseteq M \ \Rightarrow\ u \in Ts_N$. A transition is therefore enabled

at a marking M if there are enough tokens on its precondition places in the usual sense, but if it is stochastic there cannot be any immediate transition enabled. Let $Ena(M)$ be the set of *all transitions enabled at* M. By definition, it follows that $Ena(M) \subseteq Ti_N$ or $Ena(M) \subseteq Ts_N$. A set of transitions $U \subseteq Ena(M)$ is *enabled* at 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).

A 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 $U \subseteq Ena(M)$, $U \neq \emptyset$, ${}^\bullet U \subseteq M$. The *probability of the set of stochastic transitions* or the *weight of the set of immediate transitions* U *which is ready for firing at* M is $PF(U,M) = \begin{cases} \prod_{t\in U} \Omega_N(t) \cdot \prod_{u\in Ena(M)\setminus U}(1-\Omega_N(u)), & tang(M); \\ \sum_{t\in U} \Omega_N(t), & vanish(M). \end{cases}$

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

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 independent event intersection. When no transitions are enabled at 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 $U \subseteq Ena(M)$, $U \neq \emptyset$, ${}^\bullet U \subseteq M$. 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 to fire the set of transitions* U *in* M defined as $PT(U,M) = \frac{PF(U,M)}{\sum_{\{V|{}^\bullet V\subseteq M\}} PF(V,M)}$.

For $U = \emptyset$, $tang(M)$, we have $M = \widetilde{M}$ and $PT(\emptyset, M) = \frac{PF(\emptyset,M)}{\sum_{\{V|{}^\bullet V\subseteq M\}} PF(V,M)}$.

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 at M *normalized* by the probabilities or weights of *all* the sets enabled at M. The denominator of the fraction above is a sum, since it reflects the probability of the mutually exclusive event union.

If $tang(M)$ then $PT(\emptyset,M) \in (0;1]$, hence, there is a non-zero probability to stay at M in the next moment, and the residence time in M is at least 1 time unit. The sum of all outgoing probabilities is 1, i.e. $\forall M \in \mathbb{N}_f^{P_N},\ PT(\emptyset,M) + \sum_{\{U \mid {}^\bullet U \subseteq M\}} PT(U,M) = 1$, hence, it defines a probability distribution.

We write $M \stackrel{U}{\to} \widetilde{M}$ if $\exists \mathcal{P},\ M \stackrel{U}{\to}_{\mathcal{P}} \widetilde{M}$ and $M \to \widetilde{M}$ if $\exists U,\ M \stackrel{U}{\to} \widetilde{M}$. The *probability to move from M to $\widetilde{M}$ by firing any set of transitions* is $PM(M,\widetilde{M}) = \sum_{\{U \mid M \stackrel{U}{\to} \widetilde{M}\}} PT(U,M)$. Since $PM(M,\widetilde{M})$ is the probability for *any* (possibly empty) transition set to change M to $\widetilde{M}$, we use summation.

Definition 4.2 Let N be an LDTSIPN.

- The *reachability set* $RS(N)$ of N is the minimal set of markings s.t. $M_N \in RS(N)$ or, if $M \in RS(N)$ and $M \to \widetilde{M}$, then $\widetilde{M} \in RS(N)$.
- The *reachability graph* $RG(N)$ of N is a directed labeled graph with the nodes $RS(N)$ and the arcs labeled by $(U,\mathcal{P})$ between $M,\ \widetilde{M}$ iff $M \stackrel{U}{\to}_{\mathcal{P}} \widetilde{M}$.

$RS_T(N)$ denotes the set of *all tangible markings* and $RS_V(N)$ denotes that of *all vanishing markings* from $RS(N)$. Thus, $RS(N) = RS_T(N) \uplus RS_V(N)$.

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

Moreover, $\forall t \in T_N,\ {}^\bullet t \neq \emptyset \neq t^\bullet$. Next, for the set of *entry* places of N, defined as ${}^\circ N = \{p \in P_N \mid \Lambda_N(p) = \mathsf{e}\}$, and for the set of *exit* places of N, defined as $N^\circ = \{p \in P_N \mid \Lambda_N(p) = \mathsf{x}\}$, it 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 $M_N \in \mathbb{N}_f^{P_N}$ is its marking.

We shall use the following notation: $\overline{N} = (N, {}^\circ N)$ and $\underline{N} = (N, N^\circ)$. Note that a marked plain dtsi-box $(P_N, T_N, W_N, \Lambda_N, M_N)$ could be interpreted as the LDTSIPN $(P_N, T_N, W_N, \Omega_N, L_N, M_N)$, where functions Ω_N and L_N are defined as follows: $\forall t \in T_N,\ \Omega_N(t) = \Omega(\Lambda_N(t))$ and $L_N(t) = \mathcal{L}(\Lambda_N(t))$. The behaviour of marked dtsi-boxes follows from the firing rule of LDTSIPNs. A plain dtsi-box N is *n-bounded* ($n \in \mathbb{N}$) if $\overline{N}$ is so, i.e. $\forall M \in RS(\overline{N}),\ \forall p \in P_N,\ M(p) \leq n$, and it is *safe* if it is 1-bounded. A plain dtsi-box N is *clean* if $\forall M \in RS(\overline{N})$, ${}^\circ N \subseteq M \ \Rightarrow\ M = {}^\circ N$ and $N^\circ \subseteq M \ \Rightarrow\ M = N^\circ$, i.e. if there are tokens in all its entry (exit) places, then no other places have tokens.

The structure of the plain dtsi-box corresponding to a static expression is constructed as in PBC [3], 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. As we are taking the same structure for the resulting Petri net as in PBC, the obtained plain dtsi-boxes are safe and clean.

The denotational semantics is obtained considering the same standard constructions used for PBC. The relabeling relations $\varrho \subseteq 2^{\mathcal{SIL}} \times \mathcal{SIL}$ are:

- $\varrho_{id} = \{(\{(\alpha,\kappa)\}, (\alpha,\kappa)) \mid (\alpha,\kappa) \in \mathcal{SIL}\}$ is the *identity relabeling*;
- $\varrho_{(\alpha,\kappa)} = \{(\emptyset, (\alpha,\kappa))\}$ is the *constant relabeling* identified with $(\alpha,\kappa) \in \mathcal{SIL}$;
- $\varrho_{[f]} = \{(\{(\alpha,\kappa)\}, (f(\alpha),\kappa)) \mid (\alpha,\kappa) \in \mathcal{SIL}\}$;
- $\varrho_{\mathsf{rs}\ a} = \{(\{(\alpha,\kappa)\}, (\alpha,\kappa)) \mid (\alpha,\kappa) \in \mathcal{SIL},\ a, \hat{a} \notin \alpha\}$;
- $\varrho_{\mathsf{sy}\ a}$ is the least relabeling relation containing ϱ_{id} s.t. if $(\Upsilon, (\alpha,\kappa))$, $(\Xi, (\beta,\lambda)) \in \varrho_{\mathsf{sy}\ a},\ a \in \alpha,\ \hat{a} \in \beta$ then
  - $(\Upsilon + \Xi, (\alpha \oplus_a \beta, \kappa \cdot \lambda)) \in \varrho_{\mathsf{sy}\ a}$, if $\kappa, \lambda \in (0;1)$;
  - $(\Upsilon + \Xi, (\alpha \oplus_a \beta, \kappa + \lambda)) \in \varrho_{\mathsf{sy}\ a}$, if $\kappa, \lambda \in \mathbb{N} \setminus \{0\}$.

We omit the graphical representation of these operator dtsi-boxes, since it is similar to that of PBC [3]. An enumeration function can also be defined in accordance with the activity numbering. All transitions maintain their numbering when they are preserved as result of an operation (assuming they are different from each other), and those obtained from synchronization are assigned the concatenation of the parenthesized numberings of the synchronized transitions. The main novelty here is the computation of the probability/weight for the synchronization, so let us see how we compute it. Let $Box_{dtsi}(E) = (P_E, T_E, W_E, \Lambda_E)$, then $Box_{dtsi}(E\ \mathsf{sy}\ a) = \Theta_{\mathsf{sy}\ a}(Box_{dtsi}(E))$. Now, $\forall v, w \in T_E$, s.t. $\Lambda_E(v) = (\alpha,\kappa),\ \Lambda_E(w) = (\beta,\lambda)$ 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.

By definition of $\varrho_{\mathsf{sy}\ a}$, 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. We only consider a single one from the resulting transitions in the plain dtsi-box to avoid introducing redundant ones. Let us 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:

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

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

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

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

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

For $E\in RegStatExpr$, let $Box_{dtsi}(\overline{E})=\overline{Box_{dtsi}(E)}$, $Box_{dtsi}(\underline{E})=\underline{Box_{dtsi}(E)}$. Let $\simeq$ denote isomorphism between transition systems and reachability graphs relating their initial states. The names of transitions of the dtsi-box of a static expression could be identified with the enumerated activities of the latter.

Theorem 4.5 *For any static expression E, $TS(\overline{E})\simeq RG(Box_{dtsi}(\overline{E}))$.*

Proof. For the qualitative behaviour, we have the same isomorphism as in PBC. The quantitative behaviour is the same, since 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, and we use analogous probability or weight functions to construct the corresponding transition systems and reachability graphs. □

Example 4.6 Let E be from Example 3.4. In Figure 1, the marked dtsi-box $N=Box_{dtsi}(\overline{E})$ and its reachability graph $RG(N)$ are depicted. It is easy to see that $TS(\overline{E})$ and $RG(N)$ are isomorphic.

5 Performance evaluation

Let us see how Markov chains corresponding to the dynamic expressions 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 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 the 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 the 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*

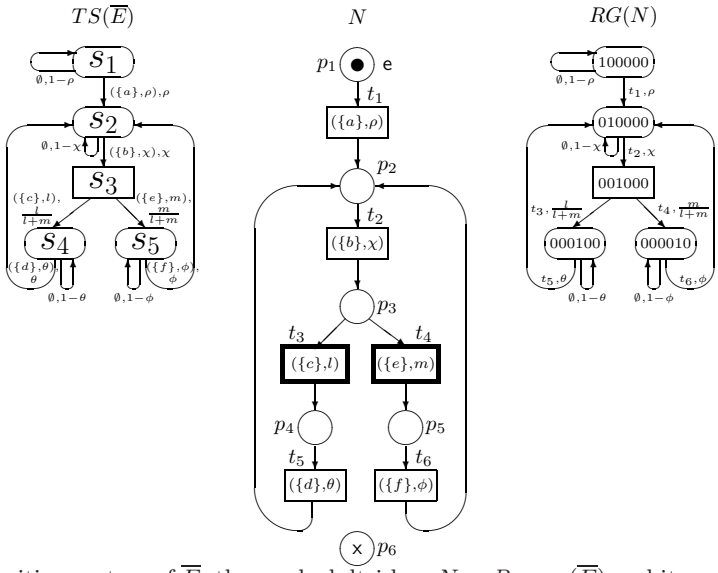

Fig. 1. The transition system of $\overline{E}$, the marked dtsi-box $N = Box_{dtsi}(\overline{E})$ and its reachability graph for $E = [(\{a\},\rho) * ((\{b\},\chi);((((\{c\},l);(\{d\},\theta))[](\{e\},m);(\{f\},\phi)))) * \mathsf{Stop}]$

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* SJ of G has the elements $SJ(s),\ s \in DR(G)$.

To evaluate performance of the system specified by a dynamic expression G, we should investigate the stochastic process associated with it. The process is the underlying semi-Markov chain (SMC), $SMC(G)$, which can be analyzed by extracting from it the embedded (absorbing) discrete time Markov chain (EDTMC) corresponding to G, $EDTMC(G)$. The construction of the latter is similar to that applied in the context of generalized stochastic PNs (GSPNs) in [11]. $EDTMC(G)$ only describes the state changes of $SMC(G)$ while ignoring its time characteristics. Thus, to construct the EDTMC, we should abstract from all time aspects of behaviour of the SMC, i.e. from the sojourn time in its states. 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

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

Notice that $PM^*(s,\tilde{s})$ defines a probability distribution, since $\forall s \in DR(G)$, s.t. s is not a terminal state, we have $\sum_{\{\tilde{s} | s \to \tilde{s},\ s \neq \tilde{s}\}} PM^*(s,\tilde{s}) = \frac{1}{1-PM(s,s)} \sum_{\{\tilde{s} | s \to \tilde{s},\ s \neq \tilde{s}\}} PM(s,\tilde{s}) = \frac{1}{1-PM(s,s)}(1 - PM(s,s)) = 1$.

Definition 5.1 Let G be a dynamic expression. The *embedded (absorbing) discrete time Markov chain (EDTMC)* of G, $EDTMC(G)$, 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 $\mathcal{P} = PM^*(s,\tilde{s})$.

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

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

The transient (k-step, $k \in \mathbb{N}$) probability mass function (PMF) $\psi^*[k] = (\psi^*[k](s_1), \ldots, \psi^*[k](s_n))$ for $EDTMC(G)$ is a 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 $\psi^*[0](s_i) = \begin{cases} 1,\ s_i = [G]_{\approx}; \\ 0,\ \text{otherwise.} \end{cases}$ Note that $\psi^*[k+1] = \psi^*[k]\mathbf{P}^*$ $(k \in \mathbb{N})$.

The steady-state PMF $\psi^* = (\psi^*(s_1), \ldots, \psi^*(s_n))$ for $EDTMC(G)$ is a solution of the equation system $\begin{cases} \psi^*(\mathbf{P}^* - \mathbf{E}) = \mathbf{0} \\ \psi^* \mathbf{1}^T = 1 \end{cases}$, where $\mathbf{E}$ is the identity matrix of size n and $\mathbf{0}$ is a row 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

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

Standard *performance indices (measures)* can be calculated based on φ [13].

6 Shared memory system

We now demonstrate how steady-state probability mass function (PMF) is used for performance evaluation, with the shared memory system case study. This illustrative example demonstrates our modeling and analysis technique. More complex systems can be easily constructed with the flexible and powerful operations of dtsiPBC, taking advantage of the algebraic compositionality.

Consider a model of two processors accessing a common shared memory in the continuous time setting on GSPNs [12]. We shall analyze this shared memory system in the discrete time stochastic setting of dtsiPBC, where concurrent execution of activities is possible. The model works as follows. After activation of the system (turning the computer on), two processors are active, and the common memory is available. Each processor can request an access to the memory after which the instantaneous decision is made. When the decision is made in favour of one processor, it starts acquisition of memory, the other processor must therefore wait until the first one terminates its memory operations, the system then returning to a state in which memory is available and both processors are active. The diagram of the system is in Figure 2.

Let us explain the meaning of actions from the dtsiPBC expressions specifying the system modules. Action a corresponds to system activation. Actions

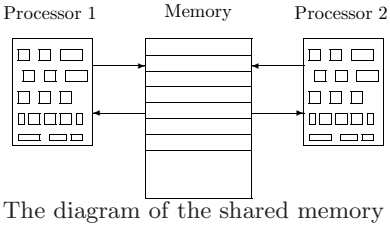

Fig. 2. The diagram of the shared memory system

r_i $(1 \leq i \leq 2)$ represent the common memory request of processor i. Instantaneous actions d_i correspond to the decision on the memory allocation in favour of processor i. Actions m_i represent the common memory access of processor i. The other actions are used for communication purposes only via synchronization, so we will abstract from them by using the 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
$E = (E_1 \| E_2 \| E_3)\ \mathsf{sy}\ x_1\ \mathsf{sy}\ x_2\ \mathsf{sy}\ y_1\ \mathsf{sy}\ y_2\ \mathsf{sy}\ z_1\ \mathsf{sy}\ z_2\ \mathsf{rs}\ x_1\ \mathsf{rs}\ x_2\ \mathsf{rs}\ y_1\ \mathsf{rs}\ y_2\ \mathsf{rs}\ z_1\ \mathsf{rs}\ z_2$.

$DR(\overline{E})$ consists of 9 equivalence classes $s_1, \ldots, s_9$, 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. We have $DR_T(\overline{E}) = \{s_1, s_2, s_5, s_7, s_8, s_9\}$ and $DR_V(\overline{E}) = \{s_3, s_4, s_6\}$.

In Figure 3, the transition system $TS(\overline{E})$ is presented. In Figure 4, the underlying SMC $SMC(\overline{E})$ is depicted. 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 $SJ = \big(8, \frac{4}{3}, 0, 0, \frac{8}{5}, 0, \frac{8}{5}, 4, 4\big)$.

The TPM for $EDTMC(\overline{E})$ is $\mathbf{P}^* = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{3} & \frac{1}{3} & 0 & \frac{1}{3} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & \frac{1}{5} & 0 & \frac{1}{5} & 0 & 0 & 0 & \frac{3}{5} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & \frac{1}{5} & \frac{1}{5} & 0 & 0 & 0 & 0 & 0 & \frac{3}{5} \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$.

The steady-state PMF for $EDTMC(\overline{E})$ is $\psi^* = \big(0, \frac{3}{44}, \frac{15}{88}, \frac{15}{88}, \frac{15}{88}, \frac{1}{44}, \frac{15}{88}, \frac{5}{44}, \frac{5}{44}\big)$.
The steady-state PMF ψ^* weighted by SJ is $\big(0, \frac{1}{11}, 0, 0, \frac{3}{11}, 0, \frac{3}{11}, \frac{5}{11}, \frac{5}{11}\big)$.
It remains to normalize the steady-state weighted PMF dividing it by the sum

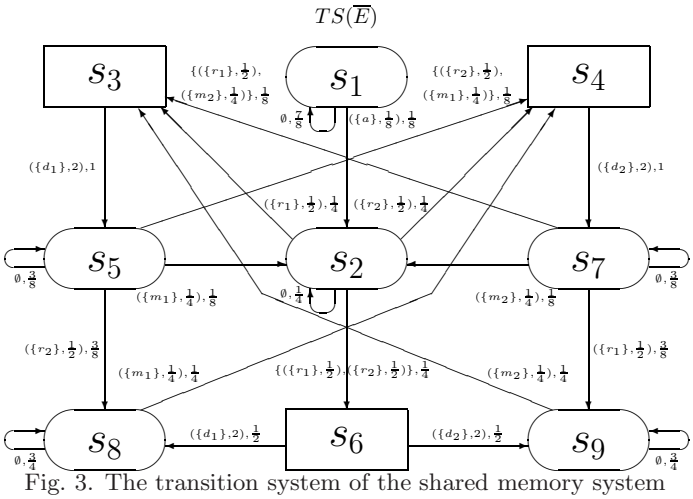

Fig. 3. The transition system of the shared memory system

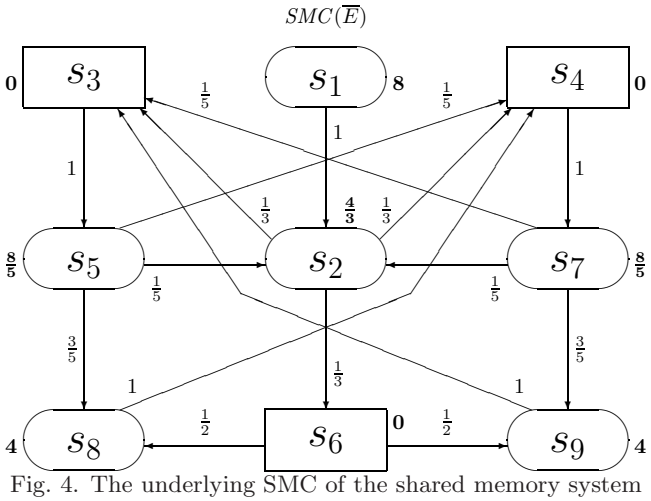

Fig. 4. The underlying SMC of the shared memory system

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 calculate now some 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. The steady-state probability for the memory to be available is $\varphi_2 + \varphi_3 + \varphi_4 + \varphi_6 = \frac{1}{17} + 0 + 0 + 0 = \frac{1}{17}$. The steady-state probability for the memory to be used (i.e. not to be 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 of emerging the shared memory necessity* coincides with the rate of leaving s_2, calculated as $\frac{\varphi_2}{SJ_2} = \frac{1}{17} \cdot \frac{3}{4} = \frac{3}{68}$.
- The common memory request of the first processor $(\{r_1\}, \frac{1}{2})$ is only possible from the states s_2, s_7. At both states, the request probability is the sum of

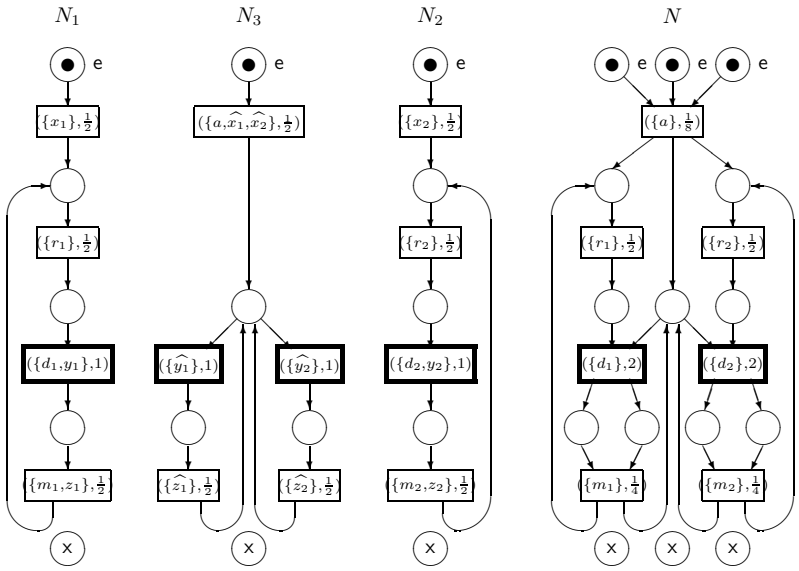

Fig. 5. The marked dtsi-boxes of two processors, shared memory and the shared memory system

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

In Figure 5, the marked dtsi-boxes corresponding to the dynamic expressions of two processors, shared memory and the shared memory system are presented, i.e. $N_i = Box_{dtsi}(\overline{E_i})$ $(1 \leq i \leq 3)$ and $N = Box_{dtsi}(\overline{E})$.

7 Conclusions

We have proposed a discrete time stochastic extension dtsiPBC of a finite part of PBC enriched with iteration and immediate multiactions. The calculus is equipped with a step operational semantics based on labeled probabilistic transition systems and a denotational semantics in terms of a subclass of LDTSIPNs. A method of performance evaluation in the framework of the calculus has been presented applied to the shared memory system case study.

The advantage of our framework is twofold. First, one can specify in it concurrent composition and synchronization of (multi)actions, whereas this is not possible in classical Markov chains. Second, algebraic formulas represent processes in a more compact way than PNs and allow one to apply syntactic transformations and comparisons. Process algebras are compositional by definition and their operations naturally correspond to operators of programming languages. Hence, it is much easier to construct a complex model in the algebraic setting than in PNs. The complexity of PNs generated for practical models in the literature demonstrates that it is not straightforward to construct such PNs directly from the system specifications. Strong points of dtsiPBC are the multiaction labels, immediate multiactions, powerful operations, a step operational and a Petri net denotational semantics allowing for concurrent execution of activities (transitions), as well as analytical performance evaluation. dtsiPBC is well suited for the discrete time applications,

such as computer or communication systems, whose discrete states change with a global time tick, and for those, in which the distributed architecture or the concurrency level should be preserved while modeling and analysis (in step semantics, we have additional transitions due to concurrent executions).

Our future work will consist in constructing a congruence for dtsiPBC, i.e. the equivalence withstanding application of all operations of the algebra. The first candidate is a stronger version of step stochastic bisimulation equivalence defined via transition systems equipped with two extra transitions skip and redo, like in [3]. 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 the application area of the associated analysis techniques. Further, recursion may be added to dtsiPBC to increase its specification power.

References

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

[2] Best, E., R. Devillers and J. G. Hall, *The box calculus: a new causal algebra with multi-label communication*, Lect. Notes Comp. Sci. **609** (1992), pp. 21–69.

[3] Best, E., R. Devillers and M. Koutny, "Petri net algebra," EATCS Monographs on Theor. Comput. Sci., Springer Verlag, 2001.

[4] Hermanns, H. and M. Rettelbach, *Syntax, semantics, equivalences and axioms for MTIPP*, in: *Proc. of* 2^{nd} *Workshop on PAPM*, number 27 in Arbeitsberichte des IMMD (1994), pp. 71–88.

[5] Hillston, J., "A compositional approach to performance modelling," Cambridge University Press, Great Britain, 1996.

[6] Koutny, M., *A compositional model of time Petri nets*, Lect. Notes Comp. Sci. **1825** (2000), pp. 303–322.

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

[8] Macià, H., V. Valero and D. de-Frutos, *sPBC: a Markovian extension of finite Petri box calculus*, in: *Proc. of* 9^{th} *IEEE International Workshop on PNPM* (2001), pp. 207–216.

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

[10] Marroquín, O. and D. de-Frutos, *Extending the Petri box calculus with time*, Lect. Notes Comp. Sci. **2075** (2001), pp. 303–322.

[11] Marsan, M. A., *Stochastic Petri nets: an elementary introduction*, Lect. Notes Comp. Sci. **424** (1990), pp. 1–29.

[12] Marsan, M. A., G. Balbo, G. Conte, S. Donatelli and G. Franceschinis, "Modelling with generalized stochastic Petri nets," John Wiley and Sons, 1995.

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

[14] Niaouris, A., *An algebra of Petri nets with arc-based time restrictions*, Lect. Notes Comp. Sci. **3407** (2005), pp. 447–462.

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

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