Performance analysis of the dining philosophers system in dtsPBC *

Igor V. Tarasyuk[1]

A.P. Ershov Institute of Informatics Systems SB RAS, Novosibirsk, Russia
itar@iis.nsk.su

Abstract. Algebra dtsPBC is a discrete time stochastic extension of finite Petri box calculus (PBC) enriched with iteration. In this paper, within dtsPBC, a method of modeling and performance evaluation based on stationary behaviour analysis for concurrent computing systems is outlined applied to the dining philosophers system.
Keywords: stochastic process algebra, Petri box calculus, discrete time, iteration, stationary behaviour, performance evaluation, dining philosophers system.

1 Introduction

Algebraic process calculi are a recognized formal model for specification of computing systems and analysis of their behaviour. In such process algebras (PAs), systems and processes are specified by formulas, and verification of their properties is accomplished at a syntactic level via equivalences, axioms and inference rules. In the last decades, stochastic extensions of PAs were proposed. Stochastic process algebras (SPAs) do not just specify actions which can happen as usual process algebras (qualitative features), but they associate some quantitative parameters with actions (quantitative characteristics). The well-known SPAs are MTIPP [5], PEPA [4] and EMPA [3].

Petri box calculus (PBC) [1] 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 semantics was proposed in terms of a subclass of Petri nets (PNs) equipped with interface and considered up to isomorphism called Petri boxes.

A stochastic extension of PBC called stochastic Petri box calculus (sPBC) was proposed in [8]. Only a finite part of PBC was used for the stochastic enrichment, i.e., sPBC has neither refinement nor recursion nor iteration operations. The calculus has an interleaving operational semantics in terms of labeled transition systems. Its denotational semantics was defined in terms of a subclass of labeled continuous time stochastic PNs (LCTSPNs) called s-boxes. In [6], the

* This work was supported in part by Deutsche Forschungsgemeinschaft (DFG), 436 RUS 113/1002/01, and Russian Foundation for Basic Research (RFBR), 09-01-91334

iteration was added to sPBC and the producer / consumer system with a buffer of capacity n was specified within the calculus. In [7], the special multiactions with zero time delay were added to sPBC and a manufacturing system with 3 machines and an assembler, as well as the AUY-protocol with a sender, a receiver and 2 channels, were modeled. The mentioned example systems had just interleaving semantics and no performance indices were calculated for them.

In [10], 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 with dts-boxes, a subclass of labeled discrete time stochastic PNs (LDTSPNs). The probabilistic equivalences were proposed and their interrelations were studied.

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 naturally, like in PBC, whereas in sPBC parallelism is simulated by interleaving obliging one to collect the information on causal independence of activities before constructing the semantics. Thus, parallelism is preserved in the semantics of all example systems considered as the case studies within dtsPBC.

In this paper, we use dtsPBC with iteration as a basic model. First, we present syntax of the calculus. Second, we describe its operational semantics in terms of labeled transition systems and denotational semantics based on a subclass of LDTSPNs. Further, the stationary behaviour of infinite stochastic processes within dtsPBC is described. Finally, for a system with five dining philosophers the performance indices are calculated based on the steady-state probabilities.

The paper is organized as follows. The syntax of dtsPBC extended with iteration operator is presented in Section 2. Section 3 describes its operational semantics and Section 4 presents its denotational semantics. The method of performance evaluation of dtsPBC processes is outlined in Section 5. In Section 6, the performance of the dining philosophers system is analyzed. The concluding Section 7 summarizes the results obtained and outlines research perspectives.

2 Syntax

In this section, we propose the syntax of discrete time stochastic PBC (dtsPBC).

We denote the *set of all finite multisets* over X by $\mathbb{N}_f^X$. 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 the execution of a multiaction that contains no visible action names. The *alphabet* of $\alpha \in \mathcal{L}$ is defined as $\mathcal{A}(\alpha) = \{x \in \mathcal{A} \mid \alpha(x) > 0\}$.

An *activity (stochastic multiaction)* is a pair (α, ρ), where $\alpha \in \mathcal{L}$ and $\rho \in (0; 1)$ is the probability of the multiaction α. The multiaction probabilities are used to calculate probabilities of state changes (steps) at discrete time moments. Let

$\mathcal{SL}$ be the set of *all activities*. Let us note that the same multiaction $\alpha \in \mathcal{L}$ may have different probabilities in the same specification. The *alphabet* of $(\alpha, \rho) \in \mathcal{SL}$ is defined as $\mathcal{A}(\alpha, \rho) = \mathcal{A}(\alpha)$. For $(\alpha, \rho) \in \mathcal{SL}$, we define its *multiaction part* as $\mathcal{L}(\alpha, \rho) = \alpha$ and its *probability part* as $\Omega(\alpha, \rho) = \rho$.

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 the standard process algebras.

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 a usual way: for $\alpha \in \mathcal{L}$ we define $f(\alpha) = \sum_{x \in \alpha} f(x)$. Remember that sums are considered with the multiplicity when applied to multisets.

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

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

Static expressions specify the structure of processes. The expressions correspond to unmarked LDTSPNs (which are marked by definition).

Definition 1. *Let* $(\alpha, \rho) \in \mathcal{SL}$, $a \in Act$. *A* static expression *of dtsPBC is*

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

StatExpr denotes the set of *all static expressions* of dtsPBC.

To avoid inconsistency of the iteration operator, we do not allow any concurrency in 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 to non-safe nets [2].

Definition 2. *Let* $(\alpha, \rho) \in \mathcal{SL}$, $a \in Act$. *A* regular static expression *of dtsPBC is*

$$\begin{aligned} E ::= \ & (\alpha, \rho) \mid E; E \mid E[]E \mid E \| E \mid E[f] \mid E\ \mathsf{rs}\ a \mid E\ \mathsf{sy}\ a \mid [E * D * E], \\ \textit{where } D ::= \ & (\alpha, \rho) \mid D; E \mid D[]D \mid D[f] \mid D\ \mathsf{rs}\ a \mid D\ \mathsf{sy}\ a \mid [D * D * E]. \end{aligned}$$

RegStatExpr denotes the set of *all regular static expressions* of dtsPBC.

Dynamic expressions specify the states of processes. The expressions correspond to LDTSPNs (which are marked by default). Dynamic expressions are combined from static ones which are annotated with upper or lower bars and

specify active components of the system at the current instant of time. $\overline{E}$ denotes the *initial*, and $\underline{E}$ 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 upper and lower bars from it.

Definition 3. *Let $E \in StatExpr$, $a \in Act$. A* dynamic expression *of dtsPBC is*

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

DynExpr denotes the set of *all dynamic expressions* of dtsPBC.

A dynamic expression is *regular* if its underlying static expression is regular. *RegDynExpr* denotes the set of *all regular dynamic expressions* of dtsPBC.

3 Operational semantics

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

3.1 Inaction rules

Inaction rules for dynamic expressions describe their structural transformations which do not change the states of the specified processes. 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 LDTSPN, 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.

In Table 1, we define inaction rules for the regular dynamic expressions in the form of overlined and underlined regular 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

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

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

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

A regular dynamic expression G is *operative* if no inaction rule can be applied to it. *OpRegDynExpr* denotes the set of *all operative regular dynamic expressions* of dtsPBC. Note that any regular 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 4. *Let $\approx\ = (\Rightarrow \cup \Leftarrow)^*$ be the structural equivalence of dynamic expressions in dtsPBC. 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

Action rules describe dynamic expression transformations due to the execution of non-empty multisets of activities. The rules represent the possible state changes of the specified processes when some non-empty multisets of activities are executed. As we shall see, the application of an action rule to a dynamic expression leads to a discrete time step in the corresponding LDTSPN at which some 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 be additionally the one-element one, i.e., the single activity, since we do not allow concurrency in the highest level of the second argument of iteration).

The empty loop rule $G \xrightarrow{\emptyset} G$ 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 LDTSPN 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 $G \xrightarrow{\emptyset} G$ from [2] in our setting would correspond to the rule $G \Rightarrow G$ describing the stay in the current state when no time elapses, but it is not needed to transform dynamic expressions into operative ones.

Thus, an application of every action rule or the empty loop rule requires one discrete time unit delay, i.e., the execution of a (possibly empty) multiset of activities resulting to the dynamic expression transformation described by the rule is accomplished instantaneously after one unit of time elapses.

Let $\Gamma \in I\!N_f^{\mathcal{SL}}$. Relabeling is extended to multisets of activities as follows: $f(\Gamma) = \sum_{(\alpha,\rho)\in\Gamma}(f(\alpha),\rho)$. The *alphabet* of Γ is defined as $\mathcal{A}(\Gamma) = \cup_{(\alpha,\rho)\in\Gamma}\mathcal{A}(\alpha)$.

In Table 3, we define the action and empty loop rules. In this table, (α, ρ), $(\beta, \chi) \in \mathcal{SL}$, $E, F \in RegStatExpr$, $G, H \in OpRegDynExpr$, $\widetilde{G}, \widetilde{H} \in RegDynExpr$ and $a \in Act$. Moreover, $\Gamma, \Delta \in \mathbb{N}_f^{\mathcal{SL}} \setminus \{\emptyset\}$ and $\Gamma' \in \mathbb{N}_f^{\mathcal{SL}}$.

Table 3. Action and empty loop rules

El $G \xrightarrow{\emptyset} G$	**B** $\overline{(\alpha,\rho)} \xrightarrow{\{(\alpha,\rho)\}} \underline{(\alpha,\rho)}$	**SC1** $\frac{G \xrightarrow{\Gamma} \widetilde{G},\ \circ \in \{;,[]\}}{G \circ E \xrightarrow{\Gamma} \widetilde{G} \circ E}$	**SC2** $\frac{G \xrightarrow{\Gamma} \widetilde{G},\ \circ \in \{;,[]\}}{E \circ G \xrightarrow{\Gamma} E \circ \widetilde{G}}$	
P1 $\frac{G \xrightarrow{\Gamma} \widetilde{G}}{G \| H \xrightarrow{\Gamma} \widetilde{G} \| H}$	**P2** $\frac{H \xrightarrow{\Gamma} \widetilde{H}}{G \| H \xrightarrow{\Gamma} G \| \widetilde{H}}$	**P3** $\frac{G \xrightarrow{\Gamma} \widetilde{G},\ H \xrightarrow{\Delta} \widetilde{H}}{G \| H \xrightarrow{\Gamma + \Delta} \widetilde{G} \| \widetilde{H}}$	**L** $\frac{G \xrightarrow{\Gamma} \widetilde{G}}{G[f] \xrightarrow{f(\Gamma)} \widetilde{G}[f]}$	
Rs $\frac{G \xrightarrow{\Gamma} \widetilde{G},\ a, \hat{a} \notin \mathcal{A}(\Gamma)}{G \text{ rs } a \xrightarrow{\Gamma} \widetilde{G} \text{ rs } a}$	**I1** $\frac{G \xrightarrow{\Gamma} \widetilde{G}}{[G*E*F] \xrightarrow{\Gamma} [\widetilde{G}*E*F]}$	**I2** $\frac{G \xrightarrow{\Gamma} \widetilde{G}}{[E*G*F] \xrightarrow{\Gamma} [E*\widetilde{G}*F]}$	**I3** $\frac{G \xrightarrow{\Gamma} \widetilde{G}}{[E*F*G] \xrightarrow{\Gamma} [E*F*\widetilde{G}]}$	
Sy1 $\frac{G \xrightarrow{\Gamma} \widetilde{G}}{G \text{ sy } a \xrightarrow{\Gamma} \widetilde{G} \text{ sy } a}$	**Sy2** $\frac{G \text{ sy } a \xrightarrow{\Gamma' + \{(\alpha,\rho)\} + \{(\beta,\chi)\}} \widetilde{G} \text{ sy } a,\ a \in \alpha,\ \hat{a} \in \beta}{G \text{ sy } a \xrightarrow{\Gamma' + \{(\alpha \oplus_a \beta, \rho \cdot \chi)\}} \widetilde{G} \text{ sy } a}$			

In the rule **Sy2**, we multiply the probabilities of synchronized multiactions, since this corresponds to the probability of the events intersection. We do not allow a self-synchronization, i.e., a synchronization of an activity with itself. The purpose is to avoid unexpected behaviour and many technical difficulties, see [2].

3.3 Transition systems

Now we define labeled probabilistic transition systems associated with dynamic expressions and used to define their operational semantics.

Note that expressions of dtsPBC can contain identical activities. To avoid technical difficulties, we must enumerate coinciding activities, for instance, from left to right in the syntax of expressions. The new activities resulting 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 5. *Let* $n \in \mathbb{N}$. *The* numbering *of expressions is* $\iota ::= n \mid (\iota)(\iota)$.

Num denotes the set of *all numberings* of expressions.

The new activities resulting from applications of the second rule for synchronization **Sy2** in different orders should be considered up to permutation of their numbering. In this way, we shall recognize different instances of the same activity. If we compare the contents of different numberings, i.e., the sets of natural numbers in them, we shall be able to identify the mentioned instances. The *content* of a numbering $\iota \in Num$ is $Cont(\iota) = \begin{cases} \{\iota\}, & \iota \in \mathbb{N}; \\ Cont(\iota_1) \cup Cont(\iota_2), & \iota = (\iota_1)(\iota_2). \end{cases}$

After we apply the enumeration, the multisets of activities from the expressions will be 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.

Definition 6. *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. The* derivation set *of a dynamic expression G, denoted by $DR(G)$, is the minimal set such that $[G]_{\approx} \in DR(G)$ 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 multisets of activities executable in s* is defined as $Exec(s) = \{\Gamma \mid \exists H \in s\ \exists \widetilde{H}\ H \xrightarrow{\Gamma} \widetilde{H}\}$.

Let $\Gamma \in Exec(s) \backslash \{\emptyset\}$. The *probability that the multiset of activities Γ is ready for execution in s* is $PF(\Gamma, s) = \prod_{(\alpha,\rho)\in\Gamma} \rho \cdot \prod_{\{\{(\beta,\chi)\}\in Exec(s) \mid (\beta,\chi)\notin\Gamma\}} (1-\chi)$.

For $\Gamma = \emptyset$, we define $PF(\emptyset, s) = \begin{cases} \prod_{\{(\beta,\chi)\}\in Exec(s)} (1-\chi), & Exec(s) \neq \{\emptyset\}; \\ 1, & \text{otherwise.} \end{cases}$

The definition of $PF(\Gamma, s)$ (and those of other probability functions we shall present) is based on the enumeration of activities which is considered implicit. Let $\Gamma \in Exec(s)$. The *probability to execute the multiset of activities Γ in s* is $PT(\Gamma, s) = \frac{PF(\Gamma,s)}{\sum_{\Delta\in Exec(s)} PF(\Delta,s)}$.

The *probability to move from s to $\tilde{s}$ by executing any multiset of activities* is $PM(s, \tilde{s}) = \sum_{\{\Gamma \mid \exists H\in s\ \exists \widetilde{H}\in\tilde{s}\ H\xrightarrow{\Gamma}\widetilde{H}\}} PT(\Gamma, s)$.

Definition 7. *Let G be a dynamic expression. The* (labeled probabilistic) transition system *of G is a quadruple $TS(G) = (S_G, L_G, \mathcal{T}_G, s_G)$, where*

- *the set of* states *is $S_G = DR(G)$;*
- *the set of* labels *is $L_G \subseteq \mathbb{N}_f^{\mathcal{SL}} \times (0; 1]$;*
- *the set of* transitions *is $\mathcal{T}_G = \{(s, (\Gamma, PT(\Gamma, s)), \tilde{s}) \mid s \in DR(G),\ \exists H \in s\ \exists \widetilde{H} \in \tilde{s}\ H \xrightarrow{\Gamma} \widetilde{H}\}$;*
- *the* initial state *is $s_G = [G]_{\approx}$.*

The transition system $TS(G)$ associated with a dynamic expression G describes all steps that happen at discrete time moments with some (one-step) probability and consist of multisets of activities. Every step happens instantaneously after one discrete time unit 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 $(s, (\Gamma, \mathcal{P}), \tilde{s}) \in \mathcal{T}_G$ will be written as $s \xrightarrow{\Gamma}_{\mathcal{P}} \tilde{s}$. The interpretation is: the probability to change the state s to $\tilde{s}$ in the result of executing Γ is $\mathcal{P}$.

Note that Γ can be the empty multiset, 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 state s to itself in the result of executing the empty multiset. This corresponds to 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 nonzero 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 $(0; 1)$. The step probabilities belong to the interval

$(0;1]$. The step probability is 1 in the case when we cannot leave a state s, hence, there exists only one transition from it, namely, the empty loop transition $s \stackrel{\emptyset}{\rightarrow}_1 s$.

We write $s \stackrel{\Gamma}{\rightarrow} \tilde{s}$ if $\exists \mathcal{P}\ s \stackrel{\Gamma}{\rightarrow}_{\mathcal{P}} \tilde{s}$ and $s \rightarrow \tilde{s}$ if $\exists \Gamma\ s \stackrel{\Gamma}{\rightarrow} \tilde{s}$.

Isomorphism is a coincidence of systems up to renaming of their components. $\simeq$ denotes the isomorphism between transition systems relating their initial states.

Definition 8. *Let G be a dynamic expression. The* underlying discrete time Markov chain (DTMC) *of G, denoted by $DTMC(G)$, has the state space $DR(G)$ and the transitions $s \rightarrow_{\mathcal{P}} \tilde{s}$, if $s \rightarrow \tilde{s}$ and $\mathcal{P} = PM(s, \tilde{s})$.*

For a dynamic expression G, a discrete random variable is associated with every state of $DTMC(G)$. 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 the state $s \in DR(G)$ 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 the state s* as $SJ(s) = \frac{1}{1-PM(s,s)}$. The *average sojourn time vector* of G, denoted by SJ, is that with the elements $SJ(s),\ s \in DR(G)$. The *sojourn time variance in the state s* is $VAR(s) = \frac{1}{(1-PM(s,s))^2}$. The *sojourn time variance vector* of G, denoted by VAR, is that with the elements $VAR(s),\ s \in DR(G)$.

4 Denotational semantics

In this section, we define the denotational semantics in terms of a subclass of LDTSPNs called discrete time stochastic Petri boxes (dts-boxes).

Definition 9. *A* discrete time stochastic Petri box (dts-box) *is a tuple $N = (P_N, T_N, W_N, \Lambda_N)$, where*

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

Let $t \in T_N$, $U \in \mathbb{N}_f^{T_N}$. 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 $U^\bullet = \sum_{t \in U} t^\bullet$. We require that $\forall t \in T_N\ {}^\bullet t \neq \emptyset \neq t^\bullet$. 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 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 dts-box is *plain* if $\forall t \in T_N \ \Lambda_N(t) \in \mathcal{SL}$, i.e., $\Lambda_N(t)$ is the constant relabeling that will be defined later. In case of constant relabeling, the shorthand notation (by an activity) for $\Lambda_N(t)$ will be used. A *marked plain dts-box* is a pair (N, M_N), where N is a plain dts-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 dts-box $(P_N, T_N, W_N, \Lambda_N, M_N)$ could be interpreted as the LDTSPN $(P_N, T_N, W_N, \Omega_N, L_N, M_N)$, 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 dts-boxes follows from the firing rule of LDTSPNs.

To define a semantic function that associates a plain dts-box with every static expression of dtsPBC, we shall propose the *enumeration* function $Enu : T_N \to Num$ which numberings with transitions of the plain dts-box N according to those of activities. In the case of synchronization, the function associates concatenation of the parenthesized numberings of the synchronized transitions with a resulting new transition.

The structure of the plain dts-box corresponding to a static expression is constructed like in PBC, see [2], i.e., we use a simultaneous refinement and relabeling meta-operator (net refinement) in addition to the *operator dts-boxes* corresponding to the algebraic operations of dtsPBC and featuring transformational transition relabelings. In the definition of the denotational semantics, we shall apply standard constructions used for PBC. Let Θ denote an *operator box* and u denote a *transition name* from PBC setting.

The relabeling relations $\varrho \subseteq I\!N_f^{\mathcal{SL}} \times \mathcal{SL}$ are defined as follows:

- $\varrho_{id} = \{(\{(\alpha, \rho)\}, (\alpha, \rho)) \mid (\alpha, \rho) \in \mathcal{SL}\}$ is the *identity relabeling* keeping the interface as it is;
- $\varrho_{(\alpha,\rho)} = \{(\emptyset, (\alpha, \rho))\}$ is the *constant relabeling* identified with $(\alpha, \rho) \in \mathcal{SL}$;
- $\varrho_{[f]} = \{(\{(\alpha, \rho)\}, (f(\alpha), \rho)) \mid (\alpha, \rho) \in \mathcal{SL}\}$;
- $\varrho_{\mathsf{rs}\ a} = \{(\{(\alpha, \rho)\}, (\alpha, \rho)) \mid (\alpha, \rho) \in \mathcal{SL},\ a, \hat{a} \notin \alpha\}$;
- $\varrho_{\mathsf{sy}\ a}$ is the least relabeling relation containing in ϱ_{id} such that if $(\Gamma, (\alpha, \rho))$, $(\Delta, (\beta, \chi)) \in \varrho_{\mathsf{sy}\ a}$ and $a \in \alpha,\ \hat{a} \in \beta$, then $(\Gamma + \Delta, (\alpha \oplus_a \beta, \rho \cdot \chi)) \in \varrho_{\mathsf{sy}\ a}$.

The plain and operator dts-boxes are presented in Figure 1. Note that the symbol i is usually omitted.

Now we define the enumeration function Enu for every operator of dtsPBC. Let $Box_{dts}(E) = (P_E, T_E, W_E, \Lambda_E)$ be the plain dts-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.

- $Box_{dts}(E \circ F) = \Theta_\circ(Box_{dts}(E), Box_{dts}(F)),\ \circ \in \{;, [], \|\}$. Since we do not introduce any new transitions, we preserve the initial numbering:
  $Enu(t) = \begin{cases} Enu_E(t), t \in T_E; \\ Enu_F(t), t \in T_F. \end{cases}$
- $Box_{dts}(E[f]) = \Theta_{[f]}(Box_{dts}(E))$. Since we only replace the labels of some multiactions by a bijection, we preserve the initial numbering:
  $Enu(t) = Enu_E(t),\ t \in T_E$.

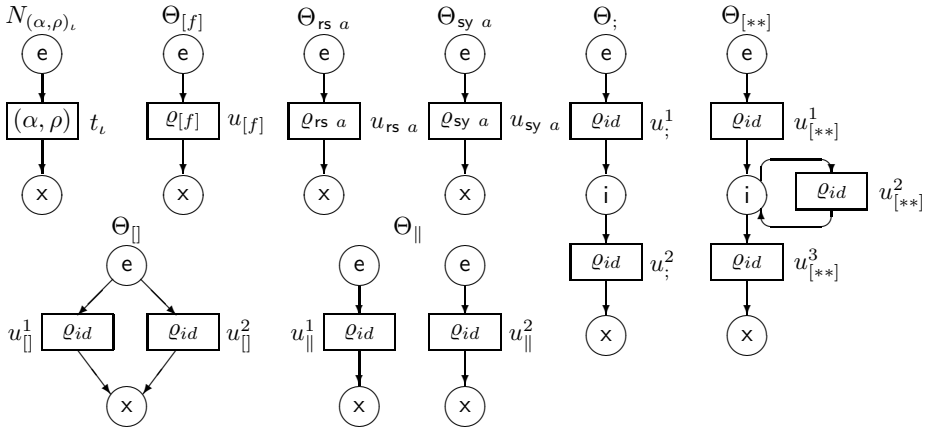

Fig. 1. The plain and operator dts-boxes

- $Box_{dts}(E\ \mathsf{rs}\ a) = \Theta_{\mathsf{rs}\ a}(Box_{dts}(E))$. Since we remove all transitions labeled with multiactions containing a or $\hat{a}$, this does not change the numbering of the remaining transitions: $Enu(t) = Enu_E(t),\ t \in T_E,\ a, \hat{a} \notin \mathcal{L}(\Lambda_E(t))$.
- $Box_{dts}(E\ \mathsf{sy}\ a) = \Theta_{\mathsf{sy}\ a}(Box_{dts}(E))$. Note that $\forall v, w \in T_E$ such that $\Lambda_E(v) = (\alpha, \rho),\ \Lambda_E(w) = (\beta, \chi)$ and $a \in \alpha,\ \hat{a} \in \beta$, the new transition t resulting from synchronization of v and w has the label $\Lambda(t) = (\alpha \oplus_a \beta, \rho \cdot \chi)$ and the numbering $Enu(t) = (Enu_E(v))(Enu_E(w))$. Thus, the enumeration function is $Enu(t) = \begin{cases} Enu_E(t), & t \in T_E; \\ (Enu_E(v))(Enu_E(w)), & t \text{ results from} \\ & \text{synchronization of } v \text{ and } w. \end{cases}$
  When we synchronize the same set of transitions in different orders, we obtain several resulting transitions with the same label and probability, but with different numberings having the same content. In this case, we shall consider only a single one from the resulting transitions in the plain dts-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 $\Lambda(t) = (\alpha \oplus_a \beta, \rho \cdot \chi) = \Lambda(u)$, 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 $Box_{dts}(E\ \mathsf{sy}\ a)$.
- $Box_{dts}([E * F * K]) = \Theta_{[**]}(Box_{dts}(E), Box_{dts}(F), Box_{dts}(K))$. Since we do not introduce any 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 10. *Let $(\alpha, \rho) \in \mathcal{SL},\ a \in Act$ and $E, F, K \in RegStatExpr$. The* denotational semantics *of dtsPBC is a mapping Box_{dts} from $RegStatExpr$ into the area of plain dts-boxes defined as follows:*

1. $Box_{dts}((\alpha,\rho)_\iota) = N_{(\alpha,\rho)_\iota}$;
2. $Box_{dts}(E \circ F) = \Theta_\circ(Box_{dts}(E), Box_{dts}(F)),\ \circ \in \{;, [], \|\}$;
3. $Box_{dts}(E[f]) = \Theta_{[f]}(Box_{dts}(E))$;
4. $Box_{dts}(E \circ a) = \Theta_{\circ a}(Box_{dts}(E)),\ \circ \in \{\mathsf{rs}, \mathsf{sy}\}$;
5. $Box_{dts}([E * F * K]) = \Theta_{[**]}(Box_{dts}(E), Box_{dts}(F), Box_{dts}(K))$.

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

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 dts-boxes, which can be joined according to the term structure (definition of Box_{dts}), the resulting plain box being marked in the places that were marked in the argument nets.

Let $\simeq$ denote the isomorphism between transition systems or between DTMCs and reachability graphs that relates the initial states. Note that the names of transitions of the dts-box corresponding to a static expression could be identified with the enumerated activities of the latter. For a dts-box N, we denote its *reachability graph* by $RG(N)$ and its *underlying DTMC* by $DTMC(N)$.

Theorem 1. *For any static expression* E, $TS(\overline{E}) \simeq RG(Box_{dts}(\overline{E}))$.

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 probability parts coinciding with the probabilities of the transitions belonging to the corresponding dts-box and, both in stochastic processes specified by expressions and dts-boxes, conflicts are resolved via the same probability functions. □

Proposition 1. *For any static expression* E,

$$DTMC(\overline{E}) \simeq DTMC(Box_{dts}(\overline{E})).$$

Proof. By Theorem 1 and definitions of underlying DTMC for dynamic expressions and LDTSPNs, since transition probabilities of the associated DTMCs are the sums of those belonging to transition systems or reachability graphs. □

5 Performance evaluation

Stationary distribution is usually used for performance evaluation. Performance indices are then calculated based on the steady-state probabilities. Let us describe the stationary behaviour of infinite stochastic processes specified by dynamic expressions whose underlined DTMCs contain one ergodic subset of states.

Let G be a dynamic expression. The elements $\mathcal{P}_{ij}$ $(1 \le i, j \le n = |DR(G)|)$ of the (one-step) transition probability matrix (TPM) $\mathbf{P}$ for $DTMC(G)$ are defined as $\mathcal{P}_{ij} = \begin{cases} PM(s_i, s_j), & s_i \to s_j; \\ 0, & \text{otherwise.} \end{cases}$

The transient (k-step, $k \in \mathbb{N}$) probability mass function (PMF) $\psi[k] = (\psi_1[k], \ldots, \psi_n[k])$ for $DTMC(G)$ is the solution of the equation system $\psi[k] = \psi[0]\mathbf{P}^k$, s.t. $\psi[0] = (\psi_1[0], \ldots, \psi_n[0])$ is the initial PMF $\psi_i[0] = \begin{cases} 1, s_i = [G]_{\approx}; \\ 0, \text{otherwise.} \end{cases}$

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

The steady-state PMF $\psi = (\psi_1, \ldots, \psi_n)$ for $DTMC(G)$ is the 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 unitary matrix of dimension n and $\mathbf{0}$ is the vector with n values 0, $\mathbf{1}$ is that with n values 1.

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

Let G be a dynamic expression and $s, \tilde{s} \in DR(G)$, $S, \widetilde{S} \subseteq DR(G)$. The following *performance indices* are based on the steady-state PMF for $DTMC(G)$.

- The *average recurrence (return) time in the state* s is $\frac{1}{\psi(s)}$.
- The *fraction of residence time in the state* s is $\psi(s)$.
- 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 $\sum_{s\in S} \psi(s)$.
- The *relative fraction of residence time in* S *w.r.t. that in* $\widetilde{S}$ is $\frac{\sum_{s\in S} \psi(s)}{\sum_{\tilde{s}\in \widetilde{S}} \psi(\tilde{s})}$.
- The *steady-state probability to perform a step with an activity* (α, ρ) is $\sum_{s\in DR(G)} \psi(s) \sum_{\{\Gamma|(\alpha,\rho)\in\Gamma\}} PT(\Gamma, s)$.
- The *probability of the event determined by a reward function* r *on the states* is $\sum_{s\in DR(G)} \psi(s) r(s)$.

6 Dining philosophers system

Consider a model of five dining philosophers, for which the Petri net interpretation was proposed in [9]. We investigate this dining philosophers system in the discrete time stochastic setting of dtsPBC allowing one to model parallelism naturally. The philosophers occupy a round table, and there is one fork between every neighboring persons, hence, there are five forks on the table. A philosopher needs two forks to eat, namely, his left and right ones. Hence, all five philosophers cannot eat together, since otherwise there will not be enough forks available, but only one of two of them who are not neighbors. The model performs as follows. After the activation of the system (the philosophers come in the dining room), five forks are placed on the table. If the left and right forks are available for a philosopher, he takes them simultaneously and begins eating. At the end of eating, the philosopher places both his forks simultaneously back on the table. The strategy to pick up and release two forks simultaneously prevents the situation when a philosopher takes one fork but is not able to pick up the second one since their neighbor has already done so. In particular, we avoid a deadlock when all the philosophers take their left (right) forks and wait until their right (left) forks will be available. The diagram of the system is depicted in Figure 2.

The meaning of actions from the expressions specifying the system modules is as follows. The action a corresponds to the system activation. The actions b_i and

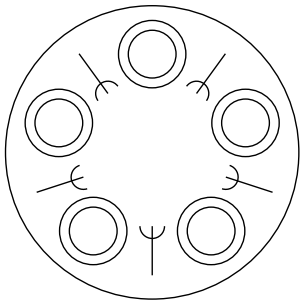

Fig. 2. The diagram of the dining philosophers system

e_i correspond to the beginning and the end, respectively, of eating of philosopher i ($1 \leq i \leq 5$). The other actions are used for communication purposes only via synchronization, and we abstract from them later using restriction. Note that the expression of each philosopher includes two alternative subexpressions such that the second one specifies a resource (fork) sharing with the right neighbor.

The static expression of the philosopher i ($1 \leq i \leq 4$) is $E_i = [(\{x_i\}, \frac{1}{2}) * ((((\{b_i, \widehat{y_i}\}, \frac{1}{2}); (\{e_i, \widehat{z_i}\}, \frac{1}{2}))[]((\{y_{i+1}\}, \frac{1}{2}); (\{z_{i+1}\}, \frac{1}{2}))) * \mathsf{Stop}]$.

The static expression of the philosopher 5 is $E_5 = [(\{a, \widehat{x_1}, \widehat{x_2}, \widehat{x_2}, \widehat{x_4}\}, \frac{1}{2}) * ((((\{b_5, \widehat{y_5}\}, \frac{1}{2}); (\{e_5, \widehat{z_5}\}, \frac{1}{2}))[]((\{y_1\}, \frac{1}{2}); (\{z_1\}, \frac{1}{2}))) * \mathsf{Stop}]$.

The static expression of the dining philosophers system is $E = (E_1 \| E_2 \| E_3 \| E_4 \| E_5)$ sy x_1 sy x_2 sy x_3 sy x_4 sy y_1 sy y_2 sy y_3 sy y_4 sy y_5 sy z_1 sy z_2 sy z_3 sy z_4 sy z_5 rs x_1 rs x_2 rs x_3 rs x_4 rs y_1 rs y_2 rs y_3 rs y_4 rs y_5 rs z_1 rs z_2 rs z_3 rs z_4 rs z_5.

Let us illustrate synchronization. The result of synchronization of the activities $(\{b_i, y_i\}, \frac{1}{2})$ and $(\{\widehat{y_i}\}, \frac{1}{2})$ is the new activity $(\{b_i\}, \frac{1}{4})$ ($1 \leq i \leq 5$). The synchronization of $(\{e_i, z_i\}, \frac{1}{2})$ and $(\{\widehat{z_i}\}, \frac{1}{2})$ produces $(\{e_i\}, \frac{1}{4})$ ($1 \leq i \leq 5$). The synchronization of $(\{a, \widehat{x_1}, \widehat{x_2}, \widehat{x_3}, \widehat{x_4}\}, \frac{1}{2})$ and $(\{x_1\}, \frac{1}{2})$ gives $(\{a, \widehat{x_2}, \widehat{x_3}, \widehat{x_4}\}, \frac{1}{4})$. The result of synchronization of $(\{a, \widehat{x_2}, \widehat{x_3}, \widehat{x_4}\}, \frac{1}{4})$ and $(\{x_2\}, \frac{1}{2})$ is $(\{a, \widehat{x_3}, \widehat{x_4}\}, \frac{1}{8})$. The result of synchronization of $(\{a, \widehat{x_3}, \widehat{x_4}\}, \frac{1}{8})$ and $(\{x_3\}, \frac{1}{2})$ is $(\{a, \widehat{x_4}\}, \frac{1}{16})$. The result of synchronization of $(\{a, \widehat{x_4}\}, \frac{1}{16})$ and $(\{x_4\}, \frac{1}{2})$ is $(\{a\}, \frac{1}{32})$.

$DR(\overline{E})$ consists of 12 equivalence classes: s_1 is the initial state, s_2: the system is activated and no philosophers dine, s_3: philosopher 1 dines, s_4: philosophers 1 and 4 dine, s_5: philosophers 1 and 3 dine, s_6: philosopher 4 dines, s_7: philosopher 3 dines, s_8: philosophers 2 and 4 dine, s_9: philosophers 3 and 5 dine, s_{10}: philosopher 2 dines, s_{11}: philosopher 5 dines, s_{12}: philosophers 2 and 5 dine.

In Figure 3, the transition system $TS(\overline{E})$ is presented.

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

$$SJ = \left(32, \frac{29}{20}, \frac{20}{11}, \frac{16}{7}, \frac{16}{7}, \frac{20}{11}, \frac{20}{11}, \frac{16}{7}, \frac{16}{7}, \frac{20}{11}, \frac{20}{11}, \frac{16}{7}\right).$$

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

$$VAR = \left(1024, \frac{841}{400}, \frac{400}{121}, \frac{256}{49}, \frac{256}{49}, \frac{400}{121}, \frac{400}{121}, \frac{256}{49}, \frac{256}{49}, \frac{400}{121}, \frac{400}{121}, \frac{256}{49}\right).$$

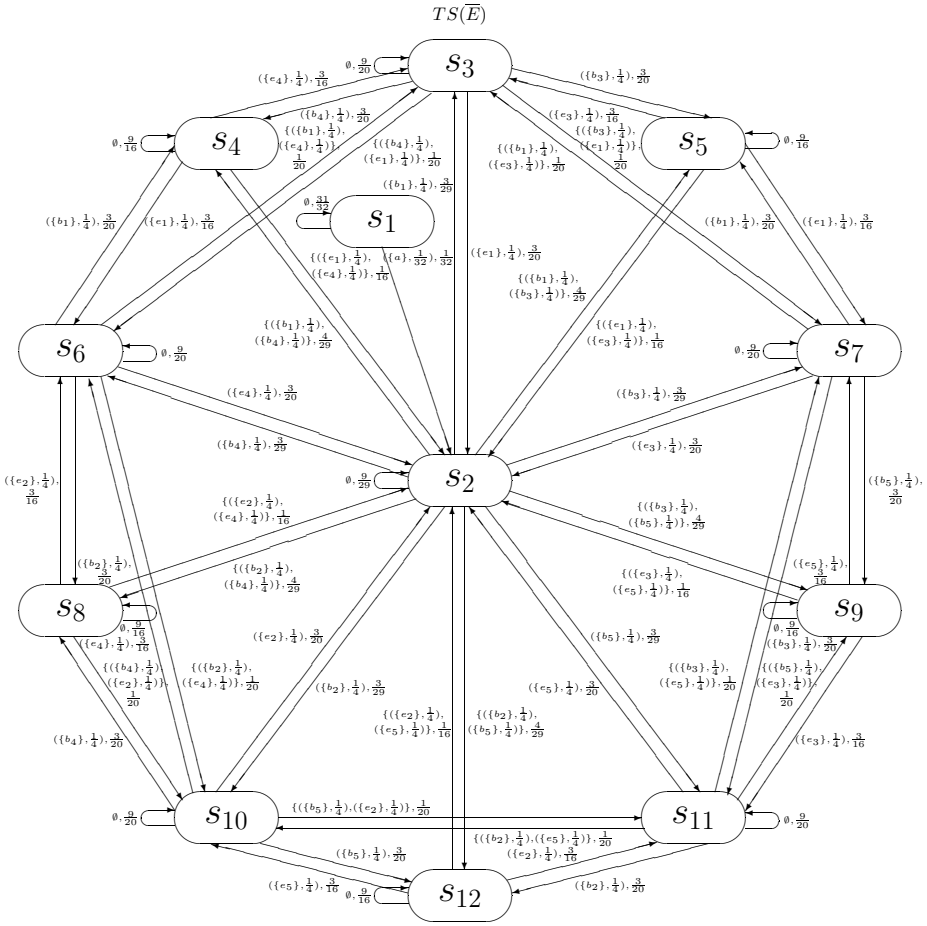

Fig. 3. The transition system of the dining philosophers system

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

$$\mathbf{P} = \begin{bmatrix}
\frac{31}{32} & \frac{1}{32} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{9}{29} & \frac{3}{29} & \frac{1}{29} & \frac{1}{29} & \frac{3}{29} & \frac{3}{29} & \frac{1}{29} & \frac{1}{29} & \frac{3}{29} & \frac{3}{29} & \frac{1}{29} \\
0 & \frac{3}{20} & \frac{9}{20} & \frac{3}{20} & \frac{3}{20} & \frac{1}{20} & \frac{1}{20} & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{1}{16} & \frac{3}{16} & \frac{9}{16} & 0 & \frac{3}{16} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{1}{16} & \frac{3}{16} & 0 & \frac{9}{16} & 0 & \frac{3}{16} & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{3}{20} & \frac{1}{20} & \frac{3}{20} & 0 & \frac{9}{20} & 0 & \frac{3}{20} & 0 & \frac{1}{20} & 0 & 0 \\
0 & \frac{3}{20} & \frac{1}{20} & 0 & \frac{3}{20} & 0 & \frac{9}{20} & 0 & \frac{3}{20} & 0 & \frac{1}{20} & 0 \\
0 & \frac{1}{16} & 0 & 0 & 0 & \frac{3}{16} & 0 & \frac{9}{16} & 0 & \frac{3}{16} & 0 & 0 \\
0 & \frac{1}{16} & 0 & 0 & 0 & 0 & \frac{3}{16} & 0 & \frac{9}{16} & 0 & \frac{3}{16} & 0 \\
0 & \frac{3}{20} & 0 & 0 & 0 & \frac{1}{20} & 0 & \frac{3}{20} & 0 & \frac{9}{20} & \frac{1}{20} & \frac{3}{20} \\
0 & \frac{3}{20} & 0 & 0 & 0 & 0 & \frac{1}{20} & 0 & \frac{3}{20} & \frac{1}{20} & \frac{9}{20} & \frac{3}{20} \\
0 & \frac{1}{16} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{3}{16} & \frac{3}{16} & \frac{9}{16}
\end{bmatrix}.$$

In Table 4, the transient and the steady-state probabilities $\psi_i[k]$ $(1 \leq i \leq 4)$ of the dining philosophers system at the time moments $k \in \{0, 20, 40, \ldots, 200\}$ and $k = \infty$ are presented, and in Figure 4, the alteration diagram (evolution in time) for the transient probabilities is depicted. It is sufficient to consider the probabilities for the states $s_1, \ldots, s_4$ only, since the corresponding values coincide for $s_3, s_6, s_7, s_{10}, s_{11}$ as well as for $s_4, s_5, s_8, s_9, s_{12}$.

Table 4. Transient and steady-state probabilities of the dining philosophers system

k	0	20	40	60	80	100	120	140	160	180	200	∞
$\psi_1[k]$	1	0.5299	0.2808	0.1488	0.0789	0.0418	0.0222	0.0117	0.0062	0.0033	0.0017	0
$\psi_2[k]$	0	0.0842	0.1098	0.1234	0.1306	0.1345	0.1365	0.1375	0.1381	0.1384	0.1386	0.1388
$\psi_3[k]$	0	0.0437	0.0681	0.0811	0.0880	0.0916	0.0935	0.0945	0.0951	0.0954	0.0955	0.0957
$\psi_4[k]$	0	0.0335	0.0537	0.0645	0.0701	0.0732	0.0748	0.0756	0.0760	0.0763	0.0764	0.0766

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

$$\psi = \left(0, \frac{29}{209}, \frac{20}{209}, \frac{16}{209}, \frac{16}{209}, \frac{20}{209}, \frac{20}{209}, \frac{16}{209}, \frac{16}{209}, \frac{20}{209}, \frac{20}{209}, \frac{16}{209}\right).$$

We can now calculate the main performance indices.

- The average recurrence time in the state s_2, where all the forks are available, called the *average system run-through*, is $\frac{1}{\psi_2} = \frac{209}{29} = 7\frac{6}{29}$.
- Nobody eats in the state s_2. Then, the *fraction of time when no philosophers dine* is $\psi_2 = \frac{29}{209}$.
  Only one philosopher eats in the states $s_3, s_6, s_7, s_{10}, s_{11}$. Then, the *fraction of time when only one philosopher dines* is $\psi_3 + \psi_6 + \psi_7 + \psi_{10} + \psi_{11} = \frac{20}{209} + \frac{20}{209} + \frac{20}{209} + \frac{20}{209} + \frac{20}{209} = \frac{100}{209}$.

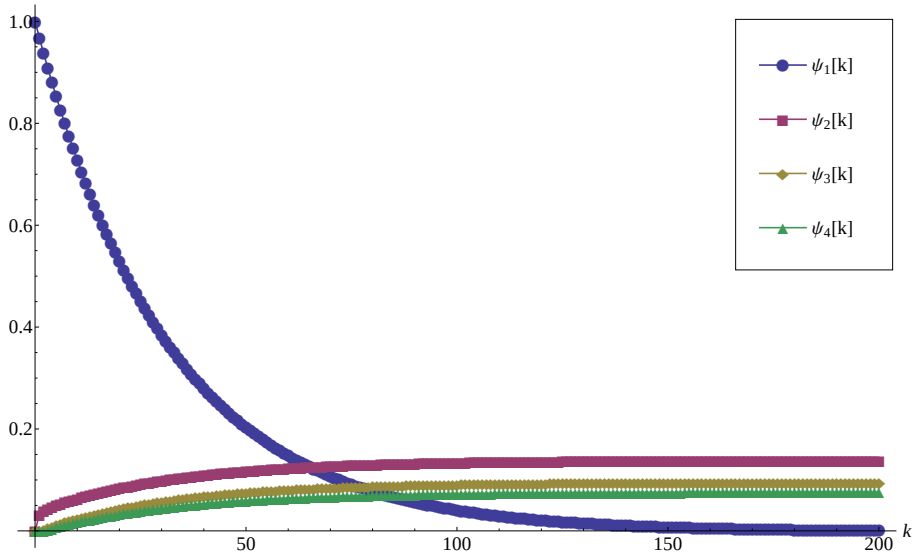

Fig. 4. Transient probabilities alteration diagram of the dining philosophers system

Two philosophers eat together in the states $s_4, s_5, s_8, s_9, s_{12}$. Then, the *fraction of time when two philosophers dine* is $\psi_4 + \psi_5 + \psi_8 + \psi_9 + \psi_{12} = \frac{16}{209} + \frac{16}{209} + \frac{16}{209} + \frac{16}{209} + \frac{16}{209} = \frac{80}{209}$.
The *relative fraction of time when two philosophers dine w.r.t. when only one philosopher dines* is $\frac{80}{209} \cdot \frac{209}{100} = \frac{4}{5}$.

– The beginning of eating of first philosopher $(\{b_1\}, \frac{1}{4})$ is only possible from the states s_2, s_6, s_7. In each of the states the beginning of eating probability is the sum of the execution probabilities for all multisets of activities containing $(\{b_1\}, \frac{1}{4})$. Thus, the *steady-state probability of the beginning of eating of first philosopher* is $\psi_2 \sum_{\{\Gamma|(\{b_1\},\frac{1}{4})\in\Gamma\}} PT(\Gamma, s_2) + \psi_6 \sum_{\{\Gamma|(\{b_1\},\frac{1}{4})\in\Gamma\}} PT(\Gamma, s_6) + \psi_7 \sum_{\{\Gamma|(\{b_1\},\frac{1}{4})\in\Gamma\}} PT(\Gamma, s_7) = \frac{29}{209}\left(\frac{3}{29} + \frac{1}{29} + \frac{1}{29}\right) + \frac{20}{209}\left(\frac{3}{20} + \frac{1}{20}\right) + \frac{20}{209}\left(\frac{3}{20} + \frac{1}{20}\right) = \frac{13}{209}$.

In Figure 5, the marked dts-box corresponding to the dynamic expression of the dining philosophers system is depicted, i.e., $N = Box_{dts}(\overline{E})$.

7 Conclusion

In this paper, within dtsPBC with iteration, a method of performance evaluation of concurrent stochastic systems was proposed based on steady-state probabilities analysis and applied to the dining philosophers system.

We plan to define and investigate stochastic equivalences of dtsPBC which allow one to identify stochastic processes with similar behaviour that are differentiated by too strict notion of the semantic equivalence. Moreover, we would like to extend dtsPBC with recursion to enhance specification power of the calculus.

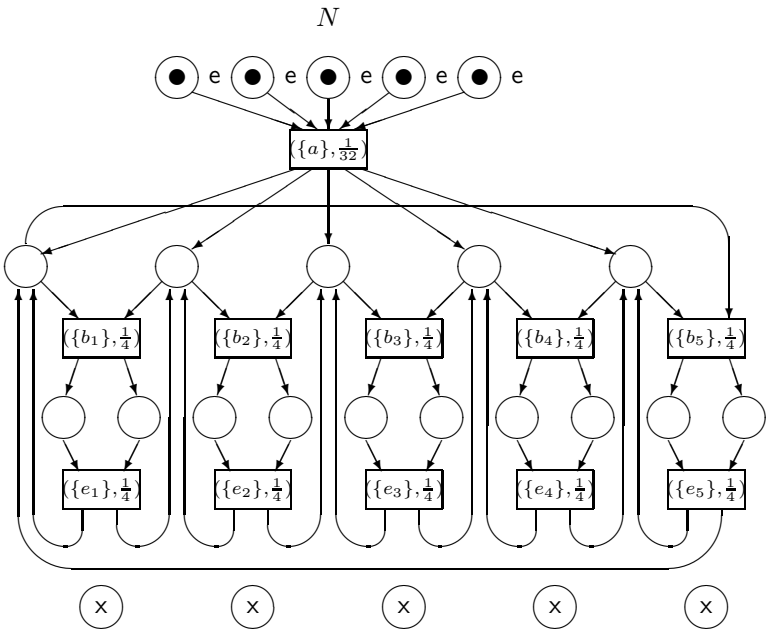

Fig. 5. The marked dts-box of the dining philosophers system

References

1. Best E., Devillers R., Hall J.G. *The box calculus: a new causal algebra with multi-label communication. Lect. Notes Comp. Sci.* **609**, p. 21–69, 1992.
2. Best E., Devillers R., Koutny M. *Petri net algebra. EATCS Monographs on Theoretical Computer Science*, 378 p., Springer Verlag, 2001.
3. Bernardo M., Gorrieri R. *A tutorial on EMPA: a theory of concurrent processes with nondeterminism, priorities, probabilities and time. Theor. Comput. Sci.* **202**, p. 1–54, 1998.
4. Hillston J. *A compositional approach to performance modelling.* 158 p., Cambridge University Press, Great Britain, 1996.
5. Hermanns H., Rettelbach M. *Syntax, semantics, equivalences and axioms for MTIPP. Proc.* 2^{nd} *Workshop on Process Algebras and Performance Modelling, Arbeitsberichte des IMMD* **27**, p. 71–88, University of Erlangen, Germany, 1994.
6. Macià H.S., Valero V.R., Cazorla D.L., Cuartero F.G. *Introducing the iteration in sPBC. Lect. Notes Comp. Sci.* **3235**, p. 292–308, 2004.
7. Macià H.S., Valero V.R., Cuartero F.G., Ruiz M.C.D. *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.
8. Macià H.S., Valero V.R., de Frutos D.E. *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.
9. Peterson J.L. *Petri net theory and modeling of systems.* Prentice-Hall, 1981.
10. Tarasyuk I.V. *Stochastic Petri box calculus with discrete time. Fundamenta Informaticae* **76(1–2)**, p. 189–218, IOS Press, Amsterdam, The Netherlands, 2007.