Discrete time stochastic Petri box calculus with immediate multiactions

Igor V. Tarasyuk[1] Hermenegilda Macià[2] Valentín Valero[2]

[1]A.P. Ershov Institute of Informatics Systems SB RAS, Novosibirsk, Russia

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

London, 17th September

Index

1 Introduction

2 Syntax

3 Operational semantics

4 Denotational semantics

5 Performance evaluation

6 Case study: shared memory system

7 Conclusions and future work

Summary

- 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.

Summary

- 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.

Summary

- 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.

Summary

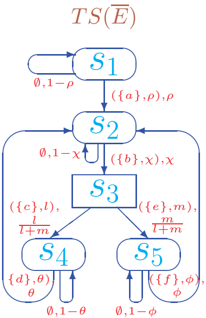
- We propose discrete time stochastic Petri Box Calculus extended with immediate multiactions, called dtsiPBC.
- The step operational semantics is constructed via labeled probabilistic transition systems.
- The denotational semantics is defined via labeled discrete time stochastic Petri nets with immediate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

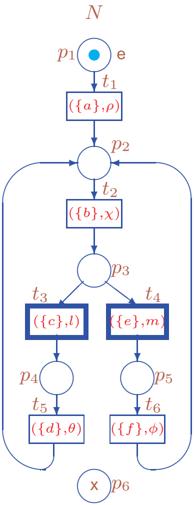
Summary

- 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.

Example:

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





1 Introduction

2 Syntax

3 Operational semantics

4 Denotational semantics

5 Performance evaluation

6 Case study: shared memory system

7 Conclusions and future work

Stochastic and immediate multiactions

- *stochastic multiaction* is a pair (α, ρ), where α is a multiaction and $\rho \in (0; 1)$ is the *probability* of the multiaction α. These probabilities are used to calculate the probabilities of state changes (steps) at discrete time moments.
- *immediate multiaction* is a pair (α, l), where α is a multiaction and $l \in \{1, 2, 3, \ldots\}$ is the non-zero *weight* of the multiaction α.
- 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.

Stochastic and immediate multiactions

- *stochastic multiaction* is a pair (α, ρ), where α is a multiaction and $\rho \in (0; 1)$ is the *probability* of the multiaction α. These probabilities are used to calculate the probabilities of state changes (steps) at discrete time moments.
- *immediate multiaction* is a pair (α, l), where α is a multiaction and $l \in \{1, 2, 3, \ldots\}$ is the non-zero *weight* of the multiaction α.
- 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.

Stochastic and immediate multiactions

- *stochastic multiaction* is a pair (α, ρ), where α is a multiaction and $\rho \in (0; 1)$ is the *probability* of the multiaction α. These probabilities are used to calculate the probabilities of state changes (steps) at discrete time moments.
- *immediate multiaction* is a pair (α, l), where α is a multiaction and $l \in \{1, 2, 3, \ldots\}$ is the non-zero *weight* of the multiaction α.
- 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.

Regular static expressions

Definition

Let $(\alpha, \kappa) \in \mathcal{SIL}$, and $a \in Act$. A regular static expression of dtsiPBC is defined by the following syntax:

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

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

Dynamic expressions

Dynamic expressions specify process states and are obtained from static ones which are annotated with upper or lower bars and specify active components of the system at the current time instant.

Definition

Let $E \in StatExpr,\ a \in Act$. Dynamic expressions are defined as follows:

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

$\overline{E}$ denotes the *initial*, and $\underline{E}$ denotes the *final* state of the process.

The *underlying static expression* of a dynamic one is obtained by removing all the upper and lower bars from it.

1 Introduction

2 Syntax

3 Operational semantics

4 Denotational semantics

5 Performance evaluation

6 Case study: shared memory system

7 Conclusions and future work

Inaction Rules

instantaneous structural transformations

Let $E, F, K \in RegStatExpr$, $G, H, \widetilde{G}, \widetilde{H} \in RegDynExpr$ and $a \in Act$.

Inaction rules for overlined and underlined regular static expressions

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

Inaction rules for arbitrary regular dynamic expressions

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

Initial and final dynamic expressions

An *operative regular dynamic expression* G: no inaction rule can be applied to it.

G and G' are *structurally equivalent*, $G \approx G'$, if they can be reached each from other by applying inaction rules in a forward or backward direction.

G is an *initial* dynamic expression, $init(G)$, if $\exists E \in RegStatExpr \; G \in [\overline{E}]_\approx$.

G is a *final* dynamic expression, $final(G)$, if $\exists E \in RegStatExpr \; G \in [\underline{E}]_\approx$.

Action and empty loop rules

- Action rules with immediate multiactions: execution of non-empty multisets of immediate multiactions. They define instantaneous dynamic expression transformations due to the execution of non-empty multisets of immediate multiactions.
- Action rules with stochastic multiactions: execution of non-empty multisets of stochastic multiactions, they are time consuming, they take one time unit in each step and it mades dynamic expression tranformations.
- Empty loop rule: execution of the empty multiset of activities at a time step, which is used to capture a delay of one time unit at any state when no immediate multiactions are executable. No dynamic expression transformations.

Action and empty loop rules

- Action rules with immediate multiactions: execution of non-empty multisets of immediate multiactions. They define instantaneous dynamic expression transformations due to the execution of non-empty multisets of immediate multiactions.
- Action rules with stochastic multiactions: execution of non-empty multisets of stochastic multiactions, they are time consuming, they take one time unit in each step and it mades dynamic expression tranformations.
- Empty loop rule: execution of the empty multiset of activities at a time step, which is used to capture a delay of one time unit at any state when no immediate multiactions are executable. No dynamic expression transformations.

Action and empty loop rules

- Action rules with immediate multiactions: execution of non-empty multisets of immediate multiactions. They define instantaneous dynamic expression transformations due to the execution of non-empty multisets of immediate multiactions.
- Action rules with stochastic multiactions: execution of non-empty multisets of stochastic multiactions, they are time consuming, they take one time unit in each step and it mades dynamic expression tranformations.
- Empty loop rule: execution of the empty multiset of activities at a time step, which is used to capture a delay of one time unit at any state when no immediate multiactions are executable. No dynamic expression transformations.

$Can(G)$: set of all sets of activities which can be executed from G

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

1. If $final(G)$ then $Can(G) = \emptyset$.
2. If $G = \overline{(\alpha, \kappa)}$ then $Can(G) = \{\{(\alpha, \kappa)\}\}$.
3. If $\Upsilon \in Can(G)$ then
   - $\Upsilon \in Can(G \circ E)$, $\Upsilon \in Can(E \circ G)$ $(\circ \in \{;, \Box\})$,
   - $\Upsilon \in Can(G \| H)$, $\Upsilon \in Can(H \| G)$,
   - $f(\Upsilon) \in Can(G[f])$,
   - $\Upsilon \in Can(G \text{ sy } a)$, $\Upsilon \in Can(G \text{ rs } a)$(when $a, \hat{a} \notin \mathcal{A}(\Upsilon)$),
   - $\Upsilon \in Can([G * E * F])$, $\Upsilon \in Can([E * G * F])$, $\Upsilon \in Can([E * F * G])$
4. If $\Upsilon \in Can(G)$ and $\Xi \in Can(H)$ then $\Upsilon + \Xi \in Can(G \| H)$
5. If $\Upsilon \in Can(G \text{ sy } a)$ and $(\alpha, \kappa), (\beta, \lambda) \in \Upsilon$ s.t. $a \in \alpha$, $\hat{a} \in \beta$ then

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

• $(\Upsilon + \{(\alpha \oplus_a \beta, \kappa + \lambda)\}) \setminus \{(\alpha, \kappa), (\beta, \lambda)\} \in Can(G \text{ sy } a)$, if $\kappa, \lambda \in \mathbb{N} \setminus \{0\}$

$Can(G)$: set of all sets of activities which can be executed from G

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

1. If $final(G)$ then $Can(G) = \emptyset$.
2. If $G = \overline{(\alpha, \kappa)}$ then $Can(G) = \{\{(\alpha, \kappa)\}\}$.
3. If $\Upsilon \in Can(G)$ then
   - $\Upsilon \in Can(G \circ E)$, $\Upsilon \in Can(E \circ G)$ $(\circ \in \{;, \Box\})$,
   - $\Upsilon \in Can(G \| H)$, $\Upsilon \in Can(H \| G)$,
   - $f(\Upsilon) \in Can(G[f])$,
   - $\Upsilon \in Can(G \text{ sy } a)$, $\Upsilon \in Can(G \text{ rs } a)$(when $a, \hat{a} \notin \mathcal{A}(\Upsilon)$),
   - $\Upsilon \in Can([G * E * F])$, $\Upsilon \in Can([E * G * F])$, $\Upsilon \in Can([E * F * G])$
4. If $\Upsilon \in Can(G)$ and $\Xi \in Can(H)$ then $\Upsilon + \Xi \in Can(G \| H)$
5. If $\Upsilon \in Can(G \text{ sy } a)$ and $(\alpha, \kappa), (\beta, \lambda) \in \Upsilon$ s.t. $a \in \alpha$, $\hat{a} \in \beta$ then
   - $(\Upsilon + \{(\alpha \oplus_a \beta, \kappa \cdot \lambda)\}) \setminus \{(\alpha, \kappa), (\beta, \lambda)\} \in Can(G \text{ sy } a)$, if $\kappa, \lambda \in (0; 1)$
   - $(\Upsilon + \{(\alpha \oplus_a \beta, \kappa + \lambda)\}) \setminus \{(\alpha, \kappa), (\beta, \lambda)\} \in Can(G \text{ sy } a)$, if $\kappa, \lambda \in \mathbb{N} \setminus \{0\}$

$Can(G)$: set of all sets of activities which can be executed from G

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

1. If $final(G)$ then $Can(G) = \emptyset$.
2. If $G = \overline{(\alpha,\kappa)}$ then $Can(G) = \{\{(\alpha,\kappa)\}\}$.
3. If $\Upsilon \in Can(G)$ then
   - $\Upsilon \in Can(G \circ E)$, $\Upsilon \in Can(E \circ G)$ $(\circ \in \{;, \Box\})$,
   - $\Upsilon \in Can(G \| H)$, $\Upsilon \in Can(H \| G)$,
   - $f(\Upsilon) \in Can(G[f])$,
   - $\Upsilon \in Can(G \text{ sy } a)$, $\Upsilon \in Can(G \text{ rs } a)$(when $a, \hat{a} \notin \mathcal{A}(\Upsilon)$),
   - $\Upsilon \in Can([G * E * F])$, $\Upsilon \in Can([E * G * F])$, $\Upsilon \in Can([E * F * G])$
4. If $\Upsilon \in Can(G)$ and $\Xi \in Can(H)$ then $\Upsilon + \Xi \in Can(G \| H)$
5. If $\Upsilon \in Can(G \text{ sy } a)$ and $(\alpha,\kappa), (\beta,\lambda) \in \Upsilon$ s.t. $a \in \alpha$, $\hat{a} \in \beta$ then

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

$tang(G)$, $vanish(G)$

- G is *tangible*, $tang(G)$, if $Can(G)$ contains only multisets of stochastic multiactions. Stochastic multiactions are only executable from tangible ones.
- G is *vanishing*, $vanish(G)$, if there are immediate multiactions in the multisets from $Can(G)$, hence, there are non-empty multisets of immediate multiactions in $Can(G)$: Immediate multiactions are only executable from vanishing operative dynamic expressions. No stochastic multiactions can be executed from a vanishing operative dynamic expression G, even if $Can(G)$ contains sets of stochastic multiactions.
- Immediate multiactions have a priority over stochastic ones.

Let $(\alpha, \rho), (\beta, \chi) \in \mathcal{SL}$, $(\alpha, l), (\beta, m) \in \mathcal{IL}$ and $(\alpha, \kappa) \in \mathcal{SIL}$.
$E, F \in RegStatExpr$, $G, H \in OpRegDynExpr$, $\widetilde{G}, \widetilde{H} \in RegDynExpr$ and $a \in Act$.

The names of the action rules with immediate multiactions have suffix '**i**'.

$tang(G)$, $vanish(G)$

- G is *tangible*, $tang(G)$, if $Can(G)$ contains only multisets of stochastic multiactions. Stochastic multiactions are only executable from tangible ones.
- G is *vanishing*, $vanish(G)$, if there are immediate multiactions in the multisets from $Can(G)$, hence, there are non-empty multisets of immediate multiactions in $Can(G)$: Immediate multiactions are only executable from vanishing operative dynamic expressions. No stochastic multiactions can be executed from a vanishing operative dynamic expression G, even if $Can(G)$ contains sets of stochastic multiactions.
- Immediate multiactions have a priority over stochastic ones.

Let $(\alpha,\rho),(\beta,\chi)\in\mathcal{SL}$, $(\alpha,l),(\beta,m)\in\mathcal{IL}$ and $(\alpha,\kappa)\in\mathcal{SIL}$.
$E,F\in RegStatExpr$, $G,H\in OpRegDynExpr$, $\widetilde{G},\widetilde{H}\in RegDynExpr$ and $a\in Act$.

The names of the action rules with immediate multiactions have suffix '**i**'.

$tang(G)$, $vanish(G)$

- G is *tangible*, $tang(G)$, if $Can(G)$ contains only multisets of stochastic multiactions. Stochastic multiactions are only executable from tangible ones.
- G is *vanishing*, $vanish(G)$, if there are immediate multiactions in the multisets from $Can(G)$, hence, there are non-empty multisets of immediate multiactions in $Can(G)$: Immediate multiactions are only executable from vanishing operative dynamic expressions. No stochastic multiactions can be executed from a vanishing operative dynamic expression G, even if $Can(G)$ contains sets of stochastic multiactions.
- Immediate multiactions have a priority over stochastic ones.

Let $(\alpha,\rho),(\beta,\chi) \in \mathcal{SL}$, $(\alpha,l),(\beta,m) \in \mathcal{IL}$ and $(\alpha,\kappa) \in \mathcal{SIL}$.
$E, F \in RegStatExpr$, $G, H \in OpRegDynExpr$, $\widetilde{G}, \widetilde{H} \in RegDynExpr$ and $a \in Act$.

The names of the action rules with immediate multiactions have suffix '**i**'.

$tang(G)$, $vanish(G)$

- G is *tangible*, $tang(G)$, if $Can(G)$ contains only multisets of stochastic multiactions. Stochastic multiactions are only executable from tangible ones.
- G is *vanishing*, $vanish(G)$, if there are immediate multiactions in the multisets from $Can(G)$, hence, there are non-empty multisets of immediate multiactions in $Can(G)$: Immediate multiactions are only executable from vanishing operative dynamic expressions. No stochastic multiactions can be executed from a vanishing operative dynamic expression G, even if $Can(G)$ contains sets of stochastic multiactions.
- Immediate multiactions have a priority over stochastic ones.

Let $(\alpha,\rho),(\beta,\chi)\in\mathcal{SL}$, $(\alpha,l),(\beta,m)\in\mathcal{IL}$ and $(\alpha,\kappa)\in\mathcal{SIL}$.
$E,F\in RegStatExpr$, $G,H\in OpRegDynExpr$, $\widetilde{G},\widetilde{H}\in RegDynExpr$ and $a\in Act$.

The names of the action rules with immediate multiactions have suffix '**i**'.

$tang(G)$, $vanish(G)$

- G is *tangible*, $tang(G)$, if $Can(G)$ contains only multisets of stochastic multiactions. Stochastic multiactions are only executable from tangible ones.
- G is *vanishing*, $vanish(G)$, if there are immediate multiactions in the multisets from $Can(G)$, hence, there are non-empty multisets of immediate multiactions in $Can(G)$: Immediate multiactions are only executable from vanishing operative dynamic expressions. No stochastic multiactions can be executed from a vanishing operative dynamic expression G, even if $Can(G)$ contains sets of stochastic multiactions.
- Immediate multiactions have a priority over stochastic ones.

Let $(\alpha,\rho),(\beta,\chi) \in \mathcal{SL}$, $(\alpha,l),(\beta,m) \in \mathcal{IL}$ and $(\alpha,\kappa) \in \mathcal{SIL}$.
$E, F \in RegStatExpr$, $G, H \in OpRegDynExpr$, $\widetilde{G}, \widetilde{H} \in RegDynExpr$ and $a \in Act$.

The names of the action rules with immediate multiactions have suffix '**i**'.

Action and empty loop rules

$\textbf{El}\ \dfrac{tang(G)}{G \xrightarrow{\emptyset} G}$

$\textbf{B}\ \overline{(\alpha,\kappa)} \xrightarrow{\{(\alpha,\kappa)\}} \underline{(\alpha,\kappa)}$

$\textbf{S}\ \dfrac{G \xrightarrow{\Upsilon} \widetilde{G}}{G;E \xrightarrow{\Upsilon} \widetilde{G};E \quad E;G \xrightarrow{\Upsilon} E;\widetilde{G}}$

$\textbf{C}\ \dfrac{G \xrightarrow{\Gamma} \widetilde{G},\ \neg init(G) \vee (init(G) \wedge tang(\overline{E}))}{G[]E \xrightarrow{\Gamma} \widetilde{G}[]E \quad E[]G \xrightarrow{\Gamma} E[]\widetilde{G}}$

$\textbf{Ci}\ \dfrac{G \xrightarrow{I} \widetilde{G}}{G[]E \xrightarrow{I} \widetilde{G}[]E \quad E[]G \xrightarrow{I} E[]\widetilde{G}}$

$\textbf{P1}\ \dfrac{G \xrightarrow{\Gamma} \widetilde{G},\ tang(H)}{G\|H \xrightarrow{\Gamma} \widetilde{G}\|H \quad H\|G \xrightarrow{\Gamma} H\|\widetilde{G}}$

$\textbf{P1i}\ \dfrac{G \xrightarrow{I} \widetilde{G}}{G\|H \xrightarrow{I} \widetilde{G}\|H \quad H\|G \xrightarrow{I} H\|\widetilde{G}}$

$\textbf{P2}\ \dfrac{G \xrightarrow{\Gamma} \widetilde{G},\ H \xrightarrow{\Delta} \widetilde{H},\ tang(G) \wedge tang(H)}{G\|H \xrightarrow{\Gamma+\Delta} \widetilde{G}\|\widetilde{H}}$

$\textbf{P2i}\ \dfrac{G \xrightarrow{I} \widetilde{G},\ H \xrightarrow{J} \widetilde{H}}{G\|H \xrightarrow{I+J} \widetilde{G}\|\widetilde{H}}$

$\textbf{L}\ \dfrac{G \xrightarrow{\Upsilon} \widetilde{G}}{G[f] \xrightarrow{f(\Upsilon)} \widetilde{G}[f]}$

$\textbf{Rs}\ \dfrac{G \xrightarrow{\Upsilon} \widetilde{G},\ a,\hat{a} \notin \mathcal{A}(\Upsilon)}{G \text{ rs } a \xrightarrow{\Upsilon} \widetilde{G} \text{ rs } a}$

$\textbf{I1}\ \dfrac{G \xrightarrow{\Upsilon} \widetilde{G}}{[G*E*F] \xrightarrow{\Upsilon} [\widetilde{G}*E*F]}$

$\textbf{I2}\ \dfrac{G \xrightarrow{\Gamma} \widetilde{G},\ \neg init(G) \vee (init(G) \wedge tang(\overline{F}))}{[E*G*F] \xrightarrow{\Gamma} [E*\widetilde{G}*F]}$

$\textbf{I2i}\ \dfrac{G \xrightarrow{I} \widetilde{G}}{[E*G*F] \xrightarrow{I} [E*\widetilde{G}*F]}$

$\textbf{I3}\ \dfrac{G \xrightarrow{\Gamma} \widetilde{G},\ \neg init(G) \vee (init(G) \wedge tang(\overline{F}))}{[E*F*G] \xrightarrow{\Gamma} [E*F*\widetilde{G}]}$

$\textbf{I3i}\ \dfrac{G \xrightarrow{I} \widetilde{G}}{[E*F*G] \xrightarrow{I} [E*F*\widetilde{G}]}$

$\textbf{Sy1}\ \dfrac{G \xrightarrow{\Upsilon} \widetilde{G}}{G \text{ sy } a \xrightarrow{\Upsilon} \widetilde{G} \text{ sy } a}$

$\textbf{Sy2}\ \dfrac{G \text{ sy } a \xrightarrow{\Gamma' + \{(\alpha,\rho)\} + \{(\beta,\chi)\}} \widetilde{G} \text{ sy } a,\ a \in \alpha,\ \hat{a} \in \beta,\ tang(G \text{ sy } a)}{G \text{ sy } a \xrightarrow{\Gamma' + \{(\alpha \oplus_a \beta,\ \rho \cdot \chi)\}} \widetilde{G} \text{ sy } a}$

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

$DR(G)$

Definition

The *derivation set* $DR(G)$ of a dynamic expression G is the minimal set:

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

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

The set of *all multisets of activities executable from s* is

$$Exec(s) = \{\Upsilon \mid \exists H \in s\ \exists \widetilde{H}\ H \xrightarrow{\Upsilon} \widetilde{H}\}$$

$DR(G) = DR_T(G) \cup DR_V(G)$

$DR(G)$

Definition

The *derivation set* $DR(G)$ of a dynamic expression G is the minimal set:

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

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

The set of *all multisets of activities executable from s* is

$$Exec(s) = \{\Upsilon \mid \exists H \in s\ \exists \widetilde{H}\ H \xrightarrow{\Upsilon} \widetilde{H}\}$$

$DR(G) = DR_T(G) \cup DR_V(G)$

Probabilities

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

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

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

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

The *probability to execute the multiset of activities Υ in s*:

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

The *probability to move from s to s' by executing any multiset of activities*:

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

Probabilities

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

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

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

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

The *probability to execute the multiset of activities* Υ *in* s:

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

The *probability to move from* s *to* s' *by executing any multiset of activities*:

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

Probabilities

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

$$PF(\Upsilon, s) = \begin{cases} \prod_{(\alpha,\rho)\in\Upsilon} \rho \cdot \prod_{\{\{(\beta,\chi)\}\in Exec(s) \mid (\beta,\chi)\not\in\Upsilon\}} (1-\chi), & s \in DR_T(G) \\ \sum_{(\alpha,l)\in\Upsilon} l, & s \in DR_V(G) \end{cases}$$

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

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

The *probability to execute the multiset of activities Υ in s*:

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

The *probability to move from s to s' by executing any multiset of activities*:

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

$TS(G)$:(labeled probabilistic) transition system

Definition

$TS(G) = (S_G, L_G, \mathcal{T}_G, s_G)$, where

- the set of *states* is $S_G = DR(G)$;
- the set of *labels* is $L_G \subseteq \mathbb{N}_f^{\mathcal{SIL}} \times (0; 1]$;
- 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 $s_G = [G]_\approx$.

1 Introduction

2 Syntax

3 Operational semantics

4 Denotational semantics

5 Performance evaluation

6 Case study: shared memory system

7 Conclusions and future work

LDTSIPN

Definition

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

- P_N and $T_N = Ts_N \uplus Ti_N$ are finite sets of *places* and *stochastic and immediate transitions*,
  s.t. $P_N \cup T_N \neq \emptyset$ and $P_N \cap T_N = \emptyset$;
- $W_N : (P_N \times T_N) \cup (T_N \times P_N) \to \mathbb{N}$ is the *arc weight* function;
- $\Omega_N : T_N \to (0;1) \cup (\mathbb{N} \setminus \{0\})$ is the *transition probability and weight* function;
- $L_N : T_N \to \mathcal{L}$ is the *transition labeling* function;
- $M_N \in \mathbb{N}_f^{P_N}$ is the *initial marking*.

Concurrent transition firings at discrete time moments. LDTSIPNs have *step* semantics. Immediate transitions always fire first, if they can. The associated probabilities in the firings are defined in the same way that in the operational semantics.

LDTSIPN

Definition

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

- P_N and $T_N = Ts_N \uplus Ti_N$ are finite sets of *places* and *stochastic and immediate transitions*,
  s.t. $P_N \cup T_N \neq \emptyset$ and $P_N \cap T_N = \emptyset$;
- $W_N : (P_N \times T_N) \cup (T_N \times P_N) \to \mathbb{N}$ is the *arc weight* function;
- $\Omega_N : T_N \to (0; 1) \cup (\mathbb{N} \setminus \{0\})$ is the *transition probability and weight* function;
- $L_N : T_N \to \mathcal{L}$ is the *transition labeling* function;
- $M_N \in \mathbb{N}_f^{P_N}$ is the *initial marking*.

Concurrent transition firings at discrete time moments. LDTSIPNs have *step* semantics. Immediate transitions always fire first, if they can. The associated probabilities in the firings are defined in the same way that in the operational semantics.

dtsi-boxes

A *discrete time stochastic and immediate Petri box (dtsi-box)* is $N = (P_N, T_N, W_N, \Lambda_N)$ is a DTSIPN where:
Λ_N is the *place and transition labeling* function s.t.

- $\Lambda_N|_{P_N} : P_N \to \{\mathrm{e}, \mathrm{i}, \mathrm{x}\}$ (it specifies *entry, internal* and *exit* places);
- $\Lambda_N|_{T_N} : T_N \to \{\varrho \mid \varrho \subseteq \mathbf{N}_f^{\mathcal{SIL}} \times \mathcal{SIL}\}$ (it associates transitions with the *relabeling relations*).

Moreover,

- $\forall t \in T_N \ {}^\bullet t \neq \emptyset \neq t^\bullet$.
- For the set of *entry* places of N, ${}^\circ N = \{p \in P_N \mid \Lambda_N(p) = \mathrm{e}\}$, and the set of *exit* places of N, $N^\circ = \{p \in P_N \mid \Lambda_N(p) = \mathrm{x}\}$, it holds: ${}^\circ N \neq \emptyset \neq N^\circ$ and ${}^\bullet({}^\circ N) = \emptyset = (N^\circ)^\bullet$.

A dtsi-box is *plain* if $\forall t \in T_N \ \Lambda_N(t) \in \mathcal{SL}$, i.e., $\Lambda_N(t)$ is the constant relabeling.
A *marked plain dtsi-box* is a pair (N, M_N), where N is a plain dtsi-box.

dtsi-boxes

A *discrete time stochastic and immediate Petri box (dtsi-box)* is $N = (P_N, T_N, W_N, \Lambda_N)$ is a DTSIPN where:
Λ_N is the *place and transition labeling* function s.t.

- $\Lambda_N|_{P_N} : P_N \to \{\mathrm{e}, \mathrm{i}, \mathrm{x}\}$ (it specifies *entry, internal* and *exit* places);
- $\Lambda_N|_{T_N} : T_N \to \{\varrho \mid \varrho \subseteq \mathbf{N}_f^{\mathcal{SIL}} \times \mathcal{SIL}\}$ (it associates transitions with the *relabeling relations*).

Moreover,

- $\forall t \in T_N \ {}^\bullet t \neq \emptyset \neq t^\bullet$.
- For the set of *entry* places of N, ${}^\circ N = \{p \in P_N \mid \Lambda_N(p) = \mathrm{e}\}$, and the set of *exit* places of N, $N^\circ = \{p \in P_N \mid \Lambda_N(p) = \mathrm{x}\}$, it holds: ${}^\circ N \neq \emptyset \neq N^\circ$ and ${}^\bullet({}^\circ N) = \emptyset = (N^\circ)^\bullet$.

A dtsi-box is *plain* if $\forall t \in T_N \ \Lambda_N(t) \in \mathcal{SL}$, i.e., $\Lambda_N(t)$ is the constant relabeling.
A *marked plain dtsi-box* is a pair (N, M_N), where N is a plain dtsi-box.

dtsi-boxes

A *discrete time stochastic and immediate Petri box (dtsi-box)* is $N = (P_N, T_N, W_N, \Lambda_N)$ is a DTSIPN where:
Λ_N is the *place and transition labeling* function s.t.

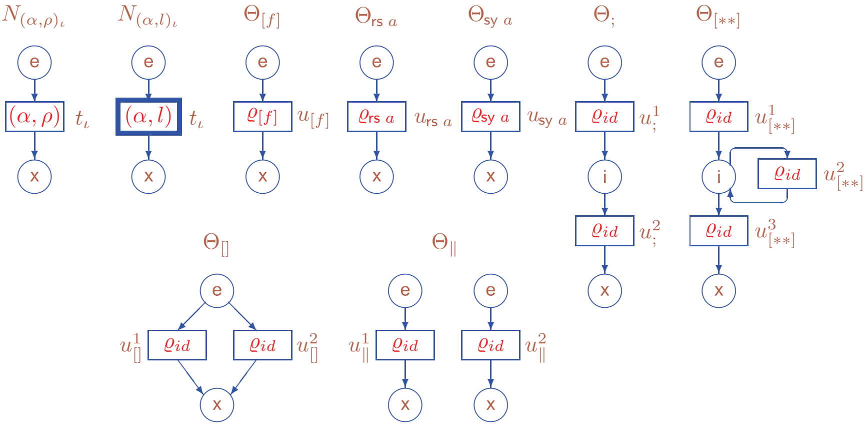
- $\Lambda_N|_{P_N} : P_N \to \{\mathrm{e}, \mathrm{i}, \mathrm{x}\}$ (it specifies *entry, internal* and *exit* places);
- $\Lambda_N|_{T_N} : T_N \to \{\varrho \mid \varrho \subseteq \boldsymbol{N}_f^{\mathcal{SIL}} \times \mathcal{SIL}\}$ (it associates transitions with the *relabeling relations*).

Moreover,

- $\forall t \in T_N \ {}^\bullet t \neq \emptyset \neq t^\bullet$.
- For the set of *entry* places of N, ${}^\circ N = \{p \in P_N \mid \Lambda_N(p) = \mathrm{e}\}$, and the set of *exit* places of N, $N^\circ = \{p \in P_N \mid \Lambda_N(p) = \mathrm{x}\}$, it holds: ${}^\circ N \neq \emptyset \neq N^\circ$ and ${}^\bullet({}^\circ N) = \emptyset = (N^\circ)^\bullet$.

A dtsi-box is *plain* if $\forall t \in T_N \ \Lambda_N(t) \in \mathcal{SL}$, i.e., $\Lambda_N(t)$ is the constant relabeling.
A *marked plain dtsi-box* is a pair (N, M_N), where N is a plain dtsi-box.

plain and operator dtsi-boxes



Algebra of dtsi-boxes

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 plain dtsi-boxes:

- $Box_{dtsi}((\alpha, \kappa)_\iota) = N_{(\alpha,\kappa)_\iota}$;
- $Box_{dtsi}(E \circ F) = \Theta_\circ(Box_{dtsi}(E), Box_{dtsi}(F))$, $\circ \in \{;, [], \|\}$;
- $Box_{dtsi}(E[f]) = \Theta_{[f]}(Box_{dtsi}(E))$;
- $Box_{dtsi}(E \circ a) = \Theta_{\circ a}(Box_{dtsi}(E))$, $\circ \in \{\mathrm{rs}, \mathrm{sy}\}$;
- $Box_{dtsi}([E * F * K]) = \Theta_{[**]}(Box_{dtsi}(E), Box_{dtsi}(F), Box_{dtsi}(K))$.

Theorem

For any static expression E

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

Algebra of dtsi-boxes

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 plain dtsi-boxes:

- $Box_{dtsi}((\alpha, \kappa)_\iota) = N_{(\alpha,\kappa)_\iota}$;
- $Box_{dtsi}(E \circ F) = \Theta_\circ(Box_{dtsi}(E), Box_{dtsi}(F))$, $\circ \in \{;, [], \|\}$;
- $Box_{dtsi}(E[f]) = \Theta_{[f]}(Box_{dtsi}(E))$;
- $Box_{dtsi}(E \circ a) = \Theta_{\circ a}(Box_{dtsi}(E))$, $\circ \in \{\mathrm{rs}, \mathrm{sy}\}$;
- $Box_{dtsi}([E * F * K]) = \Theta_{[**]}(Box_{dtsi}(E), Box_{dtsi}(F), Box_{dtsi}(K))$.

Theorem

For any static expression E

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

1 Introduction

2 Syntax

3 Operational semantics

4 Denotational semantics

5 Performance evaluation

6 Case study: shared memory system

7 Conclusions and future work

$SMC(G)$

For a dynamic expression G, a discrete random variable is associated with every tangible state from $DR(G)$. The random values (residence time in the tangible states) are geometrically distributed:
the probability to stay in the tangible state $s \in DR(G)$ for $k-1$ moments and leave it at the moment $k \geq 1$ is

$$PM(s,s)^{k-1}(1 - PM(s,s))$$

- The *average sojourn time in the state s* is

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

The stochastic process associated with a dynamic expression G: the *underlying semi-Markov chain (SMC)* of G, $SMC(G)$.

$SMC(G)$

For a dynamic expression G, a discrete random variable is associated with every tangible state from $DR(G)$. The random values (residence time in the tangible states) are geometrically distributed:
the probability to stay in the tangible state $s \in DR(G)$ for $k-1$ moments and leave it at the moment $k \geq 1$ is

$$PM(s,s)^{k-1}(1 - PM(s,s))$$

- The *average sojourn time in the state s* is

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

The stochastic process associated with a dynamic expression G: the *underlying semi-Markov chain (SMC)* of G, $SMC(G)$.

$EDTMC(G)$

$SMC(G)$ can be analyzed by extracting the *embedded (absorbing) discrete time Markov chain (EDTMC)* of G, $EDTMC(G)$.

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

$$PM^*(s, \tilde{s}) = \left\{ \begin{array}{ll} \frac{PM(s,\tilde{s})}{1-PM(s,s)}, & s \rightarrow s; \\ PM(s, \tilde{s}), & \text{otherwise}; \end{array} \right\}$$

Definition

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 $s \rightarrow \tilde{s}$ and $s \neq \tilde{s}$, where $\mathcal{P} = PM^*(s, \tilde{s})$.

$EDTMC(G)$

$SMC(G)$ can be analyzed by extracting the *embedded (absorbing) discrete time Markov chain (EDTMC)* of G, $EDTMC(G)$.

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

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

Definition

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

$EDTMC(G)$

$SMC(G)$ can be analyzed by extracting the *embedded (absorbing) discrete time Markov chain (EDTMC)* of G, $EDTMC(G)$.

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

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

Definition

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

SMC

Theorem

For any static expression E

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

$SMC(G)$ can be also analyzed by removing the vanish states considering the reduced discrete time Markov chain of G.

SMC

Theorem

For any static expression E

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

$SMC(G)$ can be also analyzed by removing the vanish states considering the reduced discrete time Markov chain of G.

1 Introduction

2 Syntax

3 Operational semantics

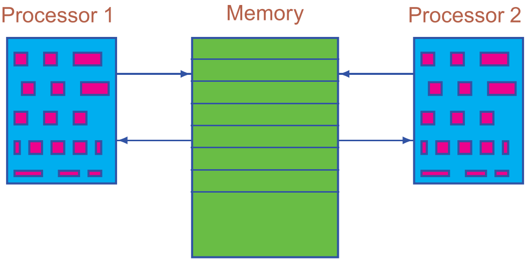
4 Denotational semantics

5 Performance evaluation

6 Case study: shared memory system

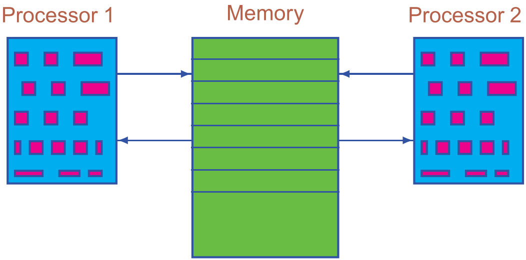
7 Conclusions and future work

two processors accessing a common shared memory



- After activation of the system, two processors are active, and the common memory is available. Each processor can request an access to the memory after which the instantaneous decision is made.
- When the decision is made in favour of a processor, it starts an acquisition of the memory, and another processor waits until the former one ends its operations, and the system returns to the state with both active processors and the available memory.

two processors accessing a common shared memory



- After activation of the system, two processors are active, and the common memory is available. Each processor can request an access to the memory after which the instantaneous decision is made.
- When the decision is made in favour of a processor, it starts an acquisition of the memory, and another processor waits until the former one ends its operations, and the system returns to the state with both active processors and the available memory.

The static expression

- a corresponds to the system activation.
- r_i $(1 \leq i \leq 2)$ represent the common memory request of processor i.
- d_i correspond to the instantaneous decision on the memory allocation in favour of the processor i.
- m_i represent the common memory access of processor i.
- The other actions are used for communication purpose only.

$$P_1 = [(\{x_1\}, \tfrac{1}{2}) * ((\{r_1\}, \tfrac{1}{2}); (\{d_1, y_1\}, 1); (\{m_1, z_1\}, \tfrac{1}{2})) * \mathrm{Stop}]$$
$$P_2 = [(\{x_2\}, \tfrac{1}{2}) * ((\{r_2\}, \tfrac{1}{2}); (\{d_2, y_2\}, 1); (\{m_2, z_2\}, \tfrac{1}{2})) * \mathrm{Stop}]$$
$$M = [(\{a, \widehat{x_1}, \widehat{x_2}\}, \tfrac{1}{2}) * (((\{\widehat{y_1}\}, 1); (\{\widehat{z_1}\}, \tfrac{1}{2}))[]((\{\widehat{y_2}\}, 1); (\{\widehat{z_2}\}, \tfrac{1}{2}))) * \mathrm{Stop}]$$

$$E = (P_1 \| P_2 \| M)$$
$$\mathrm{sy}\ x_1\ \mathrm{sy}\ x_2\ \mathrm{sy}\ y_1\ \mathrm{sy}\ y_2\ \mathrm{sy}\ z_1\ \mathrm{sy}\ z_2\ \mathrm{rs}\ x_1\ \mathrm{rs}\ x_2\ \mathrm{rs}\ y_1\ \mathrm{rs}\ y_2\ \mathrm{rs}\ z_1\ \mathrm{rs}\ z_2$$

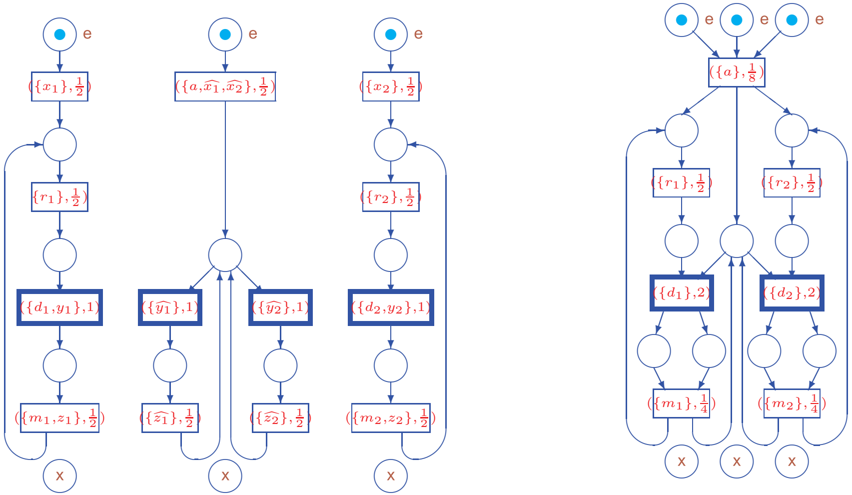
The static expression

- a corresponds to the system activation.
- r_i $(1 \leq i \leq 2)$ represent the common memory request of processor i.
- d_i correspond to the instantaneous decision on the memory allocation in favour of the processor i.
- m_i represent the common memory access of processor i.
- The other actions are used for communication purpose only.

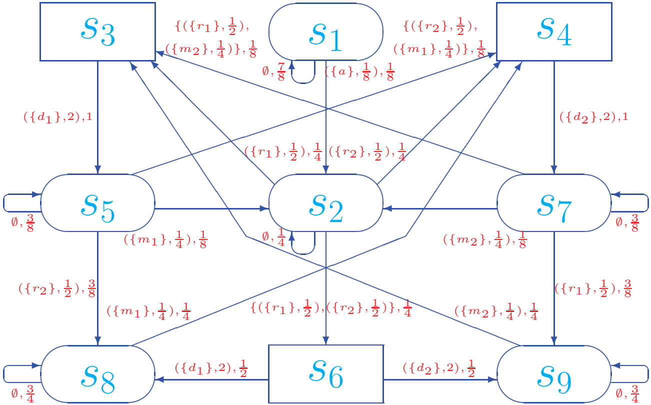
$$P_1 = [(\{x_1\}, \tfrac{1}{2}) * ((\{r_1\}, \tfrac{1}{2}); (\{d_1, y_1\}, 1); (\{m_1, z_1\}, \tfrac{1}{2})) * \mathrm{Stop}]$$
$$P_2 = [(\{x_2\}, \tfrac{1}{2}) * ((\{r_2\}, \tfrac{1}{2}); (\{d_2, y_2\}, 1); (\{m_2, z_2\}, \tfrac{1}{2})) * \mathrm{Stop}]$$
$$M = [(\{a, \widehat{x_1}, \widehat{x_2}\}, \tfrac{1}{2}) * (((\{\widehat{y_1}\}, 1); (\{\widehat{z_1}\}, \tfrac{1}{2}))[]((\{\widehat{y_2}\}, 1); (\{\widehat{z_2}\}, \tfrac{1}{2}))) * \mathrm{Stop}]$$

$$E = (P_1 \| P_2 \| M) \;\mathrm{sy}\; x_1 \;\mathrm{sy}\; x_2 \;\mathrm{sy}\; y_1 \;\mathrm{sy}\; y_2 \;\mathrm{sy}\; z_1 \;\mathrm{sy}\; z_2 \;\mathrm{rs}\; x_1 \;\mathrm{rs}\; x_2 \;\mathrm{rs}\; y_1 \;\mathrm{rs}\; y_2 \;\mathrm{rs}\; z_1 \;\mathrm{rs}\; z_2$$

dtsi-box

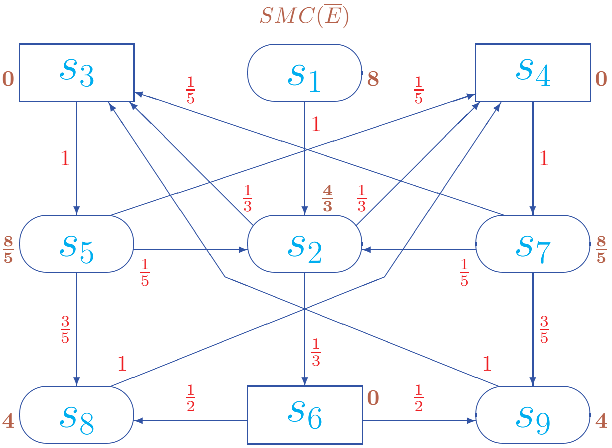

e
e
e
$(\{x_1\},\frac{1}{2})$
$(\{a,\widehat{x_1},\widehat{x_2}\},\frac{1}{2})$
$(\{x_2\},\frac{1}{2})$
$(\{r_1\},\frac{1}{2})$
$(\{r_2\},\frac{1}{2})$
$(\{d_1,y_1\},1)$
$(\{\widehat{y_1}\},1)$
$(\{\widehat{y_2}\},1)$
$(\{d_2,y_2\},1)$
$(\{m_1,z_1\},\frac{1}{2})$
$(\{\widehat{z_1}\},\frac{1}{2})$
$(\{\widehat{z_2}\},\frac{1}{2})$
$(\{m_2,z_2\},\frac{1}{2})$
x
x
x
e
e
e
$(\{a\},\frac{1}{8})$
$(\{r_1\},\frac{1}{2})$
$(\{r_2\},\frac{1}{2})$
$(\{d_1\},2)$
$(\{d_2\},2)$
$(\{m_1\},\frac{1}{4})$
$(\{m_2\},\frac{1}{4})$
x
x
x

$TS(\overline{E})$



s_1: the initial state, s_2: the system is activated and the memory is not requested, s_3: the memory is requested by the Processor 1, s_4: the memory is requested by the Processor 2, s_5: the memory is allocated to the Processor 1, s_6: the memory is requested by two processors, s_7: the memory is allocated to the Processor 2, s_8: the memory is allocated to the Processor 1 and the memory is requested by the Processor 2, s_9: the memory is allocated to the Processor 2 and the memory is requested by the Processor 1.

$SMC(\overline{E})$



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

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

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

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

long term analysis

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

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

The steady-state PMF ψ^* weighted by $SJ = \left(8, \frac{4}{3}, 0, 0, \frac{8}{5}, 0, \frac{8}{5}, 4, 4\right)$:

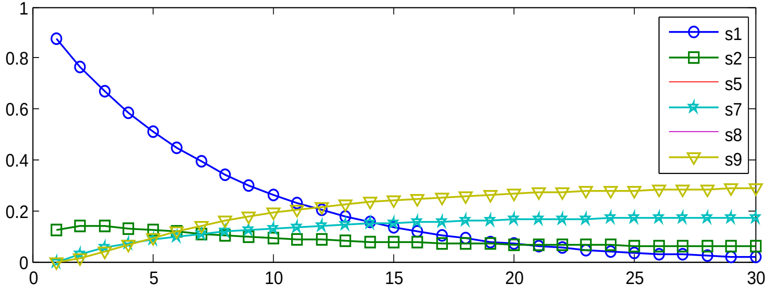
$$\left(0, \frac{1}{11}, 0, 0, \frac{3}{11}, 0, \frac{3}{11}, \frac{5}{11}, \frac{5}{11}\right).$$

We normalize the steady-state weighted PMF dividing it by the sum of its components $\psi^* SJ^T = \frac{17}{11}$.

The steady-state PMF for $SMC(\overline{E})$:

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

Transient probabilities from RDTMC



Note that the corresponding values coincide for s_5, s_7 as well as for s_8, s_9.

Performance indices

- The average recurrence time in the state s_2, where no processor requests the memory, 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 that the memory is available is $\varphi_2 + \varphi_3 + \varphi_4 + \varphi_6 = \frac{1}{17} + 0 + 0 + 0 = \frac{1}{17}$.

  The steady-state probability that the memory is used (i.e., not available), 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.

  The *rate with which the shared memory necessity emerges* coincides with the rate of leaving s_2, calculated as $\frac{\varphi_2}{SJ_2} = \frac{1}{17} \cdot \frac{3}{4} = \frac{3}{68}$.

Performance indices

- The average recurrence time in the state s_2, where no processor requests the memory, 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 that the memory is available is $\varphi_2 + \varphi_3 + \varphi_4 + \varphi_6 = \frac{1}{17} + 0 + 0 + 0 = \frac{1}{17}$.

  The steady-state probability that the memory is used (i.e., not available), 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.

  The *rate with which the shared memory necessity emerges* coincides with the rate of leaving s_2, calculated as $\frac{\varphi_2}{SJ_2} = \frac{1}{17} \cdot \frac{3}{4} = \frac{3}{68}$.

Performance indices

- The average recurrence time in the state s_2, where no processor requests the memory, 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 that the memory is available is $\varphi_2 + \varphi_3 + \varphi_4 + \varphi_6 = \frac{1}{17} + 0 + 0 + 0 = \frac{1}{17}$.

  The steady-state probability that the memory is used (i.e., not available), 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.

  The *rate with which the shared memory necessity emerges* coincides with the rate of leaving s_2, calculated as $\frac{\varphi_2}{SJ_2} = \frac{1}{17} \cdot \frac{3}{4} = \frac{3}{68}$.

- The common memory request of the first processor $(\{r_1\}, \frac{1}{2})$ is only possible from the states s_2, s_7.

  The request probability in each of the states is the sum of the execution probabilities for all multisets of activities containing $(\{r_1\}, \frac{1}{2})$.

  The *steady-state probability of the shared memory request from the first processor* is

$$\varphi_2 \sum_{\{\Upsilon \mid (\{r_1\}, \frac{1}{2}) \in \Upsilon\}} PT(\Upsilon, s_2) + \varphi_7 \sum_{\{\Upsilon \mid (\{r_1\}, \frac{1}{2}) \in \Upsilon\}} PT(\Upsilon, s_7) =$$

$$= \frac{1}{17}\left(\frac{1}{4} + \frac{1}{4}\right) + \frac{3}{17}\left(\frac{3}{8} + \frac{1}{8}\right) = \frac{2}{17}$$

1 Introduction

2 Syntax

3 Operational semantics

4 Denotational semantics

5 Performance evaluation

6 Case study: shared memory system

7 Conclusions and future work

Conclusions

- A discrete time stochastic and immediate extension *dtsiPBC* of finite *PBC* enriched with iteration.
- The step operational semantics based on labeled probabilistic transition systems.
- The denotational semantics in terms of a subclass of LDTSIPNs.
- A consistency of both semantics.
- A method of performance evaluation based on underlying SMCs.
- A case study: the shared memory system.

Conclusions

- A discrete time stochastic and immediate extension *dtsiPBC* of finite *PBC* enriched with iteration.
- The step operational semantics based on labeled probabilistic transition systems.
- The denotational semantics in terms of a subclass of LDTSIPNs.
- A consistency of both semantics.
- A method of performance evaluation based on underlying SMCs.
- A case study: the shared memory system.

Conclusions

- A discrete time stochastic and immediate extension *dtsiPBC* of finite *PBC* enriched with iteration.
- The step operational semantics based on labeled probabilistic transition systems.
- The denotational semantics in terms of a subclass of LDTSIPNs.
- A consistency of both semantics.
- A method of performance evaluation based on underlying SMCs.
- A case study: the shared memory system.

Conclusions

- A discrete time stochastic and immediate extension *dtsiPBC* of finite *PBC* enriched with iteration.
- The step operational semantics based on labeled probabilistic transition systems.
- The denotational semantics in terms of a subclass of LDTSIPNs.
- A consistency of both semantics.
- A method of performance evaluation based on underlying SMCs.
- A case study: the shared memory system.

Conclusions

- A discrete time stochastic and immediate extension *dtsiPBC* of finite *PBC* enriched with iteration.
- The step operational semantics based on labeled probabilistic transition systems.
- The denotational semantics in terms of a subclass of LDTSIPNs.
- A consistency of both semantics.
- A method of performance evaluation based on underlying SMCs.
- A case study: the shared memory system.

Conclusions

- A discrete time stochastic and immediate extension *dtsiPBC* of finite *PBC* enriched with iteration.
- The step operational semantics based on labeled probabilistic transition systems.
- The denotational semantics in terms of a subclass of LDTSIPNs.
- A consistency of both semantics.
- A method of performance evaluation based on underlying SMCs.
- A case study: the shared memory system.

Future work

- Constructing a congruence relation: the equivalence that withstands application of the algebraic operations.
- Introducing the deterministically timed multiactions with fixed time delays
- Extending the syntax with recursion operator.

Future work

- Constructing a congruence relation: the equivalence that withstands application of the algebraic operations.
- Introducing the deterministically timed multiactions with fixed time delays
- Extending the syntax with recursion operator.

Future work

- Constructing a congruence relation: the equivalence that withstands application of the algebraic operations.
- Introducing the deterministically timed multiactions with fixed time delays
- Extending the syntax with recursion operator.

Thank you for your attention!

The slides can be downloaded from Internet:
http://itar.iis.nsk.su/files/itar/pages/pasm12sld.pdf