UNIVERSITY OF TWENTE.

Formal Methods & Tools.

Efficient Modelling and Generation of Probabilistic Automata as well as Markov Automata

Mark Timmer

June 29, 2012

Joint work with Joost-Pieter Katoen, Jaco van de Pol, and Mariëlle Stoelinga

The context: probabilistic model checking

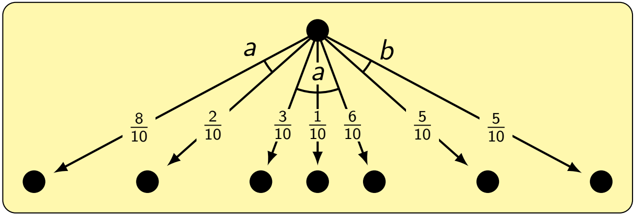
Probabilistic model checking:

- Verifying quantitative properties,
- Using a probabilistic model (e.g., a probabilistic automaton)

The context: probabilistic model checking

Probabilistic model checking:

- Verifying quantitative properties,
- Using a probabilistic model (e.g., a probabilistic automaton)

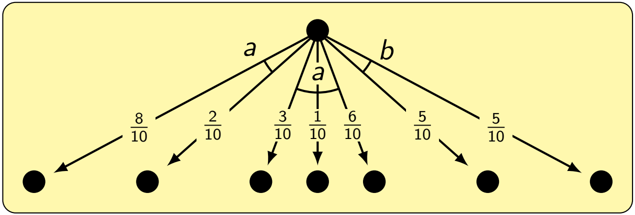


- Non-deterministically choose one of the three transitions
- Probabilistically choose the next state

The context: probabilistic model checking

Probabilistic model checking:

- Verifying quantitative properties,
- Using a probabilistic model (e.g., a probabilistic automaton)

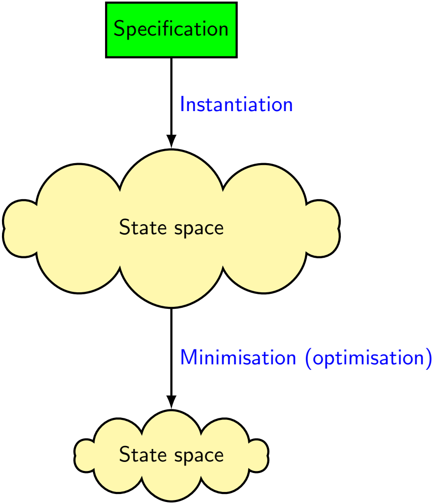


- Non-deterministically choose one of the three transitions
- Probabilistically choose the next state

Limitations of previous approaches:

- Susceptible to the state space explosion problem
- Restricted treatment of data

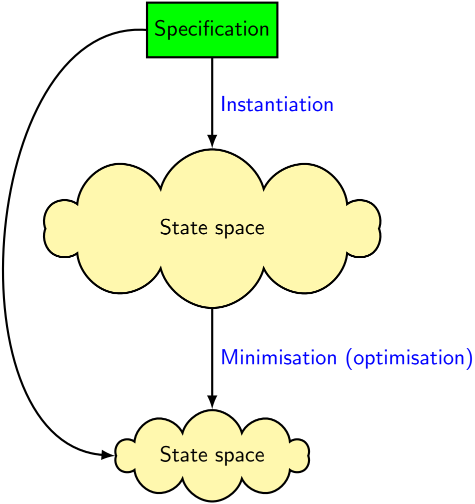
Combating the state space explosion



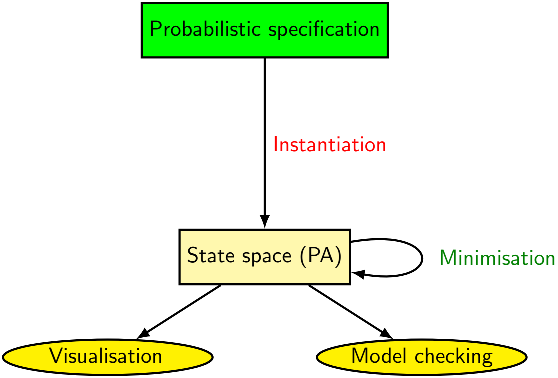
Combating the state space explosion

Optimised instantiation

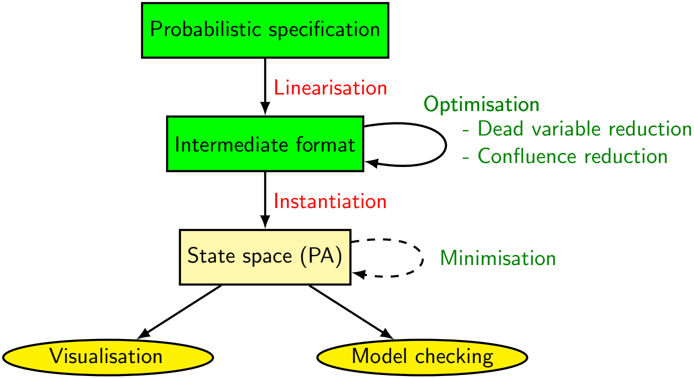
- Dead variable reduction
- Confluence reduction



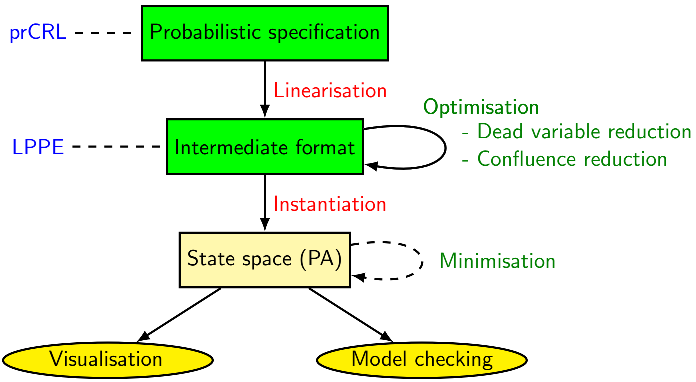
Overview of our approach



Overview of our approach

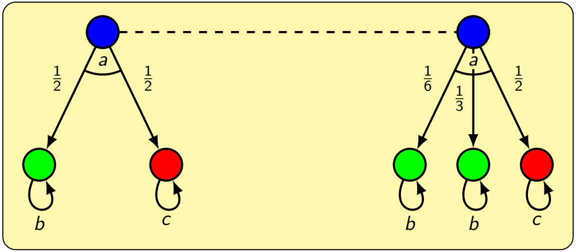


Overview of our approach



Strong bisimulation for Probabilistic Automata

Mimic behaviour with equal probabilities:



Contents

A process algebra with data and probability: prCRL

Specification language prCRL:

- Based on μCRL (so data), with additional probabilistic choice
- Semantics defined in terms of probabilistic automata
- Minimal set of operators to facilitate formal manipulation
- Syntactic sugar easily definable

A process algebra with data and probability: prCRL

Specification language prCRL:

- Based on μCRL (so data), with additional probabilistic choice
- Semantics defined in terms of probabilistic automata
- Minimal set of operators to facilitate formal manipulation
- Syntactic sugar easily definable

The grammar of prCRL process terms

Process terms in prCRL are obtained by the following grammar:

$$p ::= Y(\boldsymbol{t}) \mid c \Rightarrow p \mid p + p \mid \sum_{x:D} p \mid a(\boldsymbol{t}) \sum_{x:D}\!\!\bullet\, f : p$$

Process equations and processes

A process equation is something of the form $X(\boldsymbol{g} : \boldsymbol{G}) = p$.

An example specification

Sending an arbitrary natural number

$$X(\text{active} : \text{Bool}) =$$

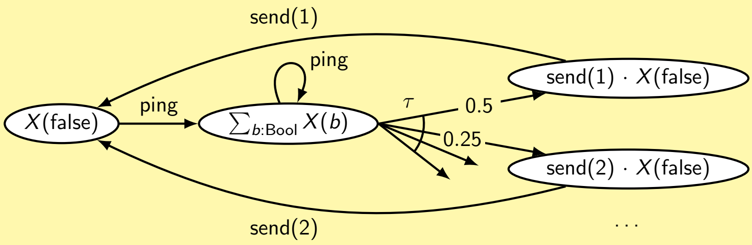
$$\text{not}(\text{active}) \Rightarrow \text{ping} \cdot \sum_{b:\text{Bool}} X(b)$$

$$+ \text{active} \Rightarrow \tau \sum_{n:\mathbb{N}^{>0}}\!\!\bullet\; \tfrac{1}{2^n} : \left(\text{send}(n) \cdot X(\text{false})\right)$$

An example specification

Sending an arbitrary natural number

$$
\begin{aligned}
X(\mathrm{active} : \mathrm{Bool}) = \\
&\mathrm{not(active)} \Rightarrow \mathrm{ping} \cdot \sum_{b:\mathrm{Bool}} X(b) \\
+\ \mathrm{active} &\Rightarrow \tau \sum_{n:\mathbb{N}^{>0}} \frac{1}{2^n} : \Big(\mathrm{send}(n) \cdot X(\mathrm{false})\Big)
\end{aligned}
$$



Composability using extended prCRL

For composability we introduce extended prCRL. It extends prCRL by parallel composition, encapsulation, hiding and renaming.

Composability using extended prCRL

For composability we introduce extended prCRL. It extends prCRL by parallel composition, encapsulation, hiding and renaming.

$$X(n : \{1,2\}) = \text{write}_X(n) \cdot X(n) + \text{choose} \sum_{n' : \{1,2\}} \frac{1}{2} : X(n')$$

$$Y(m : \{1,2\}) = \text{write}_Y(m) \cdot Y(m) + \text{choose}' \sum_{m' : \{1,2\}} \frac{1}{2} : Y(m')$$

Composability using extended prCRL

For composability we introduce extended prCRL. It extends prCRL by parallel composition, encapsulation, hiding and renaming.

$$X(n : \{1,2\}) = \text{write}_X(n) \cdot X(n) + \text{choose} \sum_{n' : \{1,2\}} \frac{1}{2} : X(n')$$

$$Y(m : \{1,2\}) = \text{write}_Y(m) \cdot Y(m) + \text{choose}' \sum_{m' : \{1,2\}} \frac{1}{2} : Y(m')$$

$$Z = (X(1) \,||\, Y(2))$$

Composability using extended prCRL

For composability we introduce extended prCRL. It extends prCRL by parallel composition, encapsulation, hiding and renaming.

$$X(n : \{1,2\}) = \text{write}_X(n) \cdot X(n) + \text{choose} \sum_{n' : \{1,2\}} \frac{1}{2} : X(n')$$

$$Y(m : \{1,2\}) = \text{write}_Y(m) \cdot Y(m) + \text{choose}' \sum_{m' : \{1,2\}} \frac{1}{2} : Y(m')$$

$$Z = \qquad (X(1) \,||\, Y(2))$$

$$\gamma(\text{choose}, \text{choose}') = \text{chooseTogether}$$

Composability using extended prCRL

For composability we introduce extended prCRL. It extends prCRL by parallel composition, encapsulation, hiding and renaming.

$$X(n:\{1,2\}) = \text{write}_X(n) \cdot X(n) + \text{choose} \sum_{n':\{1,2\}} \frac{1}{2} : X(n')$$

$$Y(m:\{1,2\}) = \text{write}_Y(m) \cdot Y(m) + \text{choose}' \sum_{m':\{1,2\}} \frac{1}{2} : Y(m')$$

$$Z = \partial_{\{\text{choose},\text{choose}'\}}(X(1) \mid\mid Y(2))$$

$$\gamma(\text{choose}, \text{choose}') = \text{chooseTogether}$$

Composability using extended prCRL

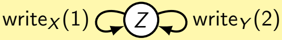
For composability we introduce extended prCRL. It extends prCRL by parallel composition, encapsulation, hiding and renaming.

$$X(n : \{1,2\}) = \text{write}_X(n) \cdot X(n) + \text{choose} \sum_{n' : \{1,2\}} \frac{1}{2} : X(n')$$

$$Y(m : \{1,2\}) = \text{write}_Y(m) \cdot Y(m) + \text{choose}' \sum_{m' : \{1,2\}} \frac{1}{2} : Y(m')$$

$$Z = \partial_{\{\text{choose}, \text{choose}'\}}(X(1) \,||\, Y(2))$$

$$\gamma(\text{choose}, \text{choose}') = \text{chooseTogether}$$



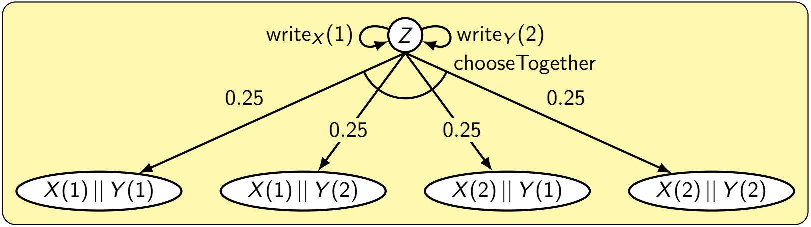
Composability using extended prCRL

For composability we introduce extended prCRL. It extends prCRL by parallel composition, encapsulation, hiding and renaming.

$$X(n : \{1,2\}) = \text{write}_X(n) \cdot X(n) + \text{choose} \sum_{n' : \{1,2\}} \frac{1}{2} : X(n')$$

$$Y(m : \{1,2\}) = \text{write}_Y(m) \cdot Y(m) + \text{choose}' \sum_{m' : \{1,2\}} \frac{1}{2} : Y(m')$$

$$Z = \partial_{\{\text{choose},\text{choose}'\}}(X(1) \mid\mid Y(2))$$
$$\gamma(\text{choose}, \text{choose}') = \text{chooseTogether}$$



A linear format for prCRL: the LPPE

LPPEs are a subset of prCRL specifications:

$$X(\boldsymbol{g} : \boldsymbol{G}) = \sum_{\boldsymbol{d_1} : \boldsymbol{D_1}} c_1 \Rightarrow a_1(b_1) \sum\!\!\!\bullet_{\boldsymbol{e_1} : \boldsymbol{E_1}} f_1 : X(\boldsymbol{n_1})$$
$$\dots$$
$$+ \sum_{\boldsymbol{d_k} : \boldsymbol{D_k}} c_k \Rightarrow a_k(b_k) \sum\!\!\!\bullet_{\boldsymbol{e_k} : \boldsymbol{E_k}} f_k : X(\boldsymbol{n_k})$$

A linear format for prCRL: the LPPE

LPPEs are a subset of prCRL specifications:

$$X(\boldsymbol{g} : \boldsymbol{G}) = \sum_{\boldsymbol{d_1} : \boldsymbol{D_1}} c_1 \Rightarrow a_1(b_1) \sum\!\!\!\!\bullet_{\boldsymbol{e_1} : \boldsymbol{E_1}} f_1 : X(\boldsymbol{n_1})$$
$$\dots$$
$$+ \sum_{\boldsymbol{d_k} : \boldsymbol{D_k}} c_k \Rightarrow a_k(b_k) \sum\!\!\!\!\bullet_{\boldsymbol{e_k} : \boldsymbol{E_k}} f_k : X(\boldsymbol{n_k})$$

Advantages of using LPPEs instead of prCRL specifications:

- Easy state space generation
- Straight-forward parallel composition
- **Symbolic optimisations enabled at the language level**

A linear format for prCRL: the LPPE

LPPEs are a subset of prCRL specifications:

$$
\begin{aligned}
X(\boldsymbol{g} : \boldsymbol{G}) = & \sum_{\boldsymbol{d_1} : \boldsymbol{D_1}} c_1 \Rightarrow a_1(b_1) \sum_{\bullet \, \boldsymbol{e_1} : \boldsymbol{E_1}} f_1 : X(\boldsymbol{n_1}) \\
& \dots \\
+ & \sum_{\boldsymbol{d_k} : \boldsymbol{D_k}} c_k \Rightarrow a_k(b_k) \sum_{\bullet \, \boldsymbol{e_k} : \boldsymbol{E_k}} f_k : X(\boldsymbol{n_k})
\end{aligned}
$$

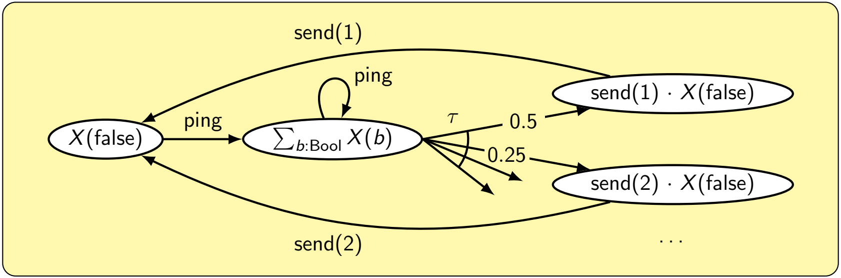
Advantages of using LPPEs instead of prCRL specifications:

- Easy state space generation
- Straight-forward parallel composition
- **Symbolic optimisations enabled at the language level**

Theorem

Every specification (without unguarded recursion) can be linearised to an LPPE, preserving strong probabilistic bisimulation.

Linear Probabilistic Process Equations – an example



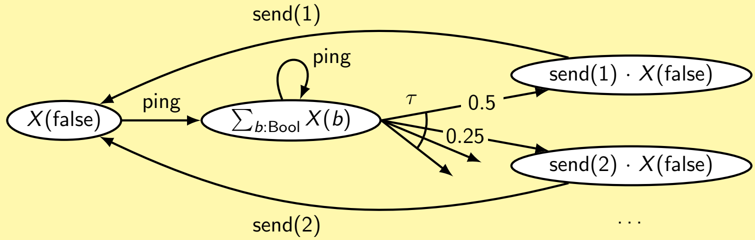
Specification in prCRL

$$X(\text{active} : \text{Bool}) =$$

$$\text{not}(\text{active}) \Rightarrow \text{ping} \cdot \sum_{b:\text{Bool}} X(b)$$

$$+ \text{ active} \Rightarrow \tau \sum_{n:\mathbb{N}^{>0}}\!\!\bullet\ \frac{1}{2^n} : \text{send}(n) \cdot X(\text{false})$$

Linear Probabilistic Process Equations – an example



Specification in prCRL

$$X(\text{active} : \text{Bool}) =$$
$$\text{not}(\text{active}) \Rightarrow \text{ping} \cdot \sum_{b:\text{Bool}} X(b)$$
$$+ \text{ active} \Rightarrow \tau \sum_{n:\mathbb{N}^{>0}}\!\!\bullet\; \frac{1}{2^n} : \text{send}(n) \cdot X(\text{false})$$

Specification in LPPE

$$X(pc : \{1..3\}, n : \mathbb{N}^{\geq 0}) =$$
$$+\ pc = 1 \Rightarrow \text{ping} \cdot X(2, 1)$$
$$+\ pc = 2 \Rightarrow \text{ping} \cdot X(2, 1)$$
$$+\ pc = 2 \Rightarrow \tau \sum_{n:\mathbb{N}^{>0}}\!\!\bullet\; \frac{1}{2^n} : X(3, n)$$
$$+\ pc = 3 \Rightarrow \text{send}(n) \cdot X(1, 1)$$

Linearisation: a simple example without data

Consider the following prCRL specification:

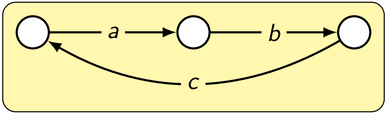
$$X = a \cdot b \cdot c \cdot X$$

Linearisation: a simple example without data

Consider the following prCRL specification:

$$X = a \cdot b \cdot c \cdot X$$

The control flow of X is given by:

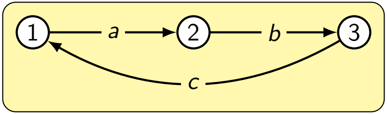


Linearisation: a simple example without data

Consider the following prCRL specification:

$$X = a \cdot b \cdot c \cdot X$$

The control flow of X is given by:

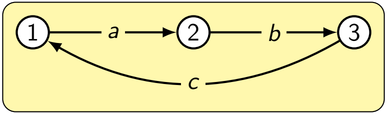


Linearisation: a simple example without data

Consider the following prCRL specification:

$$X = a \cdot b \cdot c \cdot X$$

The control flow of X is given by:



The corresponding LPPE (initialised with pc = 1):

$$\begin{aligned} Y(pc\colon \{1,2,3\}) = & \\ & pc = 1 \Rightarrow a \cdot Y(2) \\ + & pc = 2 \Rightarrow b \cdot Y(3) \\ + & pc = 3 \Rightarrow c \cdot Y(1) \end{aligned}$$

Linearisation: a more complicated example with data

Consider the following prCRL specification:

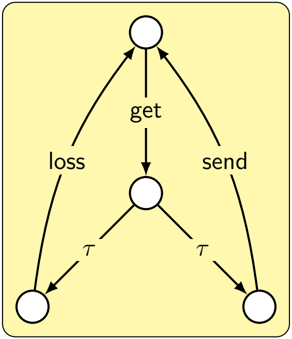
$$X = \sum_{d:D} \text{get}(d) \cdot (\tau \cdot \text{loss} \cdot X + \tau \cdot \text{send}(d) \cdot X)$$

Linearisation: a more complicated example with data

Consider the following prCRL specification:

$$X = \sum_{d:D} \mathrm{get}(d) \cdot (\tau \cdot \mathrm{loss} \cdot X + \tau \cdot \mathrm{send}(d) \cdot X)$$

Control flow:

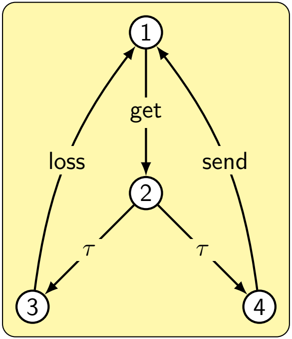


Linearisation: a more complicated example with data

Consider the following prCRL specification:

$$X = \sum_{d:D} \mathrm{get}(d) \cdot (\tau \cdot \mathrm{loss} \cdot X + \tau \cdot \mathrm{send}(d) \cdot X)$$

Control flow:

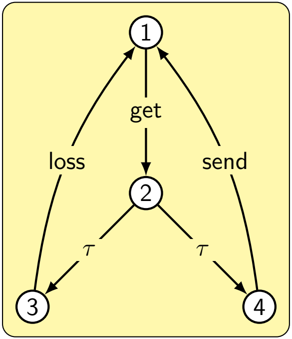


Linearisation: a more complicated example with data

Consider the following prCRL specification:

$$X = \sum_{d:D} \mathrm{get}(d) \cdot (\tau \cdot \mathrm{loss} \cdot X + \tau \cdot \mathrm{send}(d) \cdot X)$$

Control flow:



LPPE:

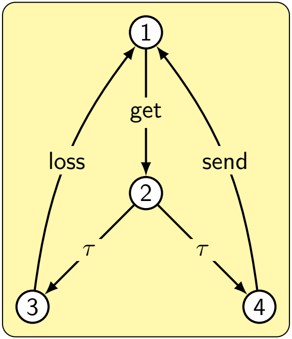
$$\begin{aligned}
Y(pc\colon \{1,2,3,4\}, x\colon D) = \\
\textstyle\sum_{d:D} \quad & pc = 1 \Rightarrow \mathrm{get}(d) \cdot Y(2,d) \\
+ \quad & pc = 2 \Rightarrow \tau \cdot Y(3,x) \\
+ \quad & pc = 2 \Rightarrow \tau \cdot Y(4,x) \\
+ \quad & pc = 3 \Rightarrow \mathrm{loss} \cdot Y(1,x) \\
+ \quad & pc = 4 \Rightarrow \mathrm{send}(x) \cdot Y(1,x)
\end{aligned}$$

Linearisation: a more complicated example with data

Consider the following prCRL specification:

$$X = \sum_{d:D} \text{get}(d) \cdot (\tau \cdot \text{loss} \cdot X + \tau \cdot \text{send}(d) \cdot X)$$

Control flow:



LPPE:

$$
\begin{aligned}
&Y(pc\colon \{1,2,3,4\}, x\colon D) = \\
&\quad \textstyle\sum_{d:D} \quad pc = 1 \Rightarrow \text{get}(d) \cdot Y(2, d) \\
&\quad + \qquad pc = 2 \Rightarrow \tau \cdot Y(3, x) \\
&\quad + \qquad pc = 2 \Rightarrow \tau \cdot Y(4, x) \\
&\quad + \qquad pc = 3 \Rightarrow \text{loss} \cdot Y(1, x) \\
&\quad + \qquad pc = 4 \Rightarrow \text{send}(x) \cdot Y(1, x)
\end{aligned}
$$

Initial process: $Y(1, d_1)$.

Linearisation: a more algorithmic approach

Consider the following prCRL specification:

$$X(d : D) = \sum_{e:D} a(d+e) \sum_{f:D}\!\!\!\!\!\bullet\;\; \frac{1}{|D|} : \left(c(e) \cdot c(f) \cdot X(5) + c(e + f) \cdot X(5) \right)$$

Linearisation: a more algorithmic approach

Consider the following prCRL specification:

$$X(d : D) = \sum_{e:D} a(d+e) \sum_{f:D} \frac{1}{|D|} : \left(c(e) \cdot c(f) \cdot X(5) + c(e+f) \cdot X(5)\right)$$

---

1 $X_1(d : D, e : D, f : D) =$
$\sum_{e:D} a(d+e) \sum_{f:D} \frac{1}{|D|} : (c(e) \cdot c(f) \cdot X(5) + c(e+f) \cdot X(5))$

Linearisation: a more algorithmic approach

Consider the following prCRL specification:

$$X(d:D) = \sum_{e:D} a(d{+}e) \sum\!\!\!\bullet_{f:D} \frac{1}{|D|} : \left(c(e) \cdot c(f) \cdot X(5) + c(e+f) \cdot X(5) \right)$$

---

1 $X_1(d:D, e:D, f:D) =$

$\sum_{e:D} a(d+e) \sum\!\!\!\bullet_{f:D} \frac{1}{|D|} : \; (c(e) \cdot c(f) \cdot X(5) + c(e+f) \cdot X(5))$

Linearisation: a more algorithmic approach

Consider the following prCRL specification:

$$X(d : D) = \sum_{e:D} a(d{+}e) \sum_{f:D} \frac{1}{|D|} : \Big(c(e) \cdot c(f) \cdot X(5) + c(e + f) \cdot X(5)\Big)$$

---

1 $X_1(d : D, e : D, f : D) =$
$\sum_{e:D} a(d + e) \sum_{f:D} \frac{1}{|D|} : \; (c(e) \cdot c(f) \cdot X(5) + c(e + f) \cdot X(5))$

---

2 $X_1(d : D, e : D, f : D) = \sum_{e:D} a(d + e) \sum_{f:D} \frac{1}{|D|} : X_2(d, e, f)$

Linearisation: a more algorithmic approach

Consider the following prCRL specification:

$$X(d:D) = \sum_{e:D} a(d{+}e) \sum_{f:D}^{\bullet} \frac{1}{|D|} : \Big(c(e) \cdot c(f) \cdot X(5) + c(e+f) \cdot X(5)\Big)$$

---

1 $X_1(d:D, e:D, f:D) =$

$\sum_{e:D} a(d+e) \sum^{\bullet}_{f:D} \frac{1}{|D|} : \; (c(e) \cdot c(f) \cdot X(5) + c(e+f) \cdot X(5))$

---

2 $X_1(d:D, e:D, f:D) = \sum_{e:D} a(d+e) \sum^{\bullet}_{f:D} \frac{1}{|D|} : X_2(d, e, f)$

$X_2(d:D, e:D, f:D) = c(e) \cdot c(f) \cdot X(5) + c(e+f) \cdot X(5)$

Linearisation: a more algorithmic approach

Consider the following prCRL specification:

$$X(d : D) = \sum_{e:D} a(d{+}e) \sum_{f:D}^{\bullet} \frac{1}{|D|} : \Big(c(e) \cdot c(f) \cdot X(5) + c(e+f) \cdot X(5)\Big)$$

---

1 $X_1(d : D, e : D, f : D) =$
$\quad\sum_{e:D} a(d+e) \sum^{\bullet}_{f:D} \frac{1}{|D|} : (c(e) \cdot c(f) \cdot X(5) + c(e+f) \cdot X(5))$

---

2 $X_1(d : D, e : D, f : D) = \sum_{e:D} a(d+e) \sum^{\bullet}_{f:D} \frac{1}{|D|} : X_2(d, e, f)$
$X_2(d : D, e : D, f : D) = c(e) \cdot c(f) \cdot X(5) + c(e+f) \cdot X(5)$

Linearisation: a more algorithmic approach

Consider the following prCRL specification:

$$X(d:D) = \sum_{e:D} a(d+e) \sum_{f:D} \frac{1}{|D|} : \Big(c(e) \cdot c(f) \cdot X(5) + c(e+f) \cdot X(5) \Big)$$

---

1 $X_1(d:D, e:D, f:D) =$

$$\sum_{e:D} a(d+e) \sum_{f:D} \frac{1}{|D|} : \; (c(e) \cdot c(f) \cdot X(5) + c(e+f) \cdot X(5))$$

---

2 $X_1(d:D, e:D, f:D) = \sum_{e:D} a(d+e) \sum_{f:D} \frac{1}{|D|} : X_2(d, e, f)$

$X_2(d:D, e:D, f:D) = c(e) \cdot c(f) \cdot X(5) + c(e+f) \cdot X(5)$

---

3 $X_1(d:D, e:D, f:D) = \sum_{e:D} a(d+e) \sum_{f:D} \frac{1}{|D|} : X_2(d, e, f)$

Linearisation: a more algorithmic approach

Consider the following prCRL specification:

$$X(d:D) = \sum_{e:D} a(d+e) \sum_{f:D} \frac{1}{|D|} : \Big(c(e) \cdot c(f) \cdot X(5) + c(e+f) \cdot X(5) \Big)$$

---

1 $X_1(d:D, e:D, f:D) =$
$\sum_{e:D} a(d+e) \sum_{f:D} \frac{1}{|D|} : \ (c(e) \cdot c(f) \cdot X(5) + c(e+f) \cdot X(5))$

---

2 $X_1(d:D, e:D, f:D) = \sum_{e:D} a(d+e) \sum_{f:D} \frac{1}{|D|} : X_2(d, e, f)$
$X_2(d:D, e:D, f:D) = c(e) \cdot c(f) \cdot X(5) + c(e+f) \cdot X(5)$

---

3 $X_1(d:D, e:D, f:D) = \sum_{e:D} a(d+e) \sum_{f:D} \frac{1}{|D|} : X_2(d, e, f)$
$X_2(d:D, e:D, f:D) = c(e) \cdot X_3(d, e, f) + c(e+f) \cdot X_1(5, e, f)$

Linearisation: a more algorithmic approach

Consider the following prCRL specification:

$$X(d : D) = \sum_{e:D} a(d+e) \sum_{f:D} \frac{1}{|D|} : \Big(c(e) \cdot c(f) \cdot X(5) + c(e+f) \cdot X(5) \Big)$$

---

1 $X_1(d : D, e : D, f : D) =$
$\quad \sum_{e:D} a(d+e) \sum_{f:D} \frac{1}{|D|} : (c(e) \cdot c(f) \cdot X(5) + c(e+f) \cdot X(5))$

---

2 $X_1(d : D, e : D, f : D) = \sum_{e:D} a(d+e) \sum_{f:D} \frac{1}{|D|} : X_2(d, e, f)$
$X_2(d : D, e : D, f : D) = c(e) \cdot c(f) \cdot X(5) + c(e+f) \cdot X(5)$

---

3 $X_1(d : D, e : D, f : D) = \sum_{e:D} a(d+e) \sum_{f:D} \frac{1}{|D|} : X_2(d, e, f)$
$X_2(d : D, e : D, f : D) = c(e) \cdot X_3(d, e, f) + c(e+f) \cdot X_1(5, e, f)$
$X_3(d : D, e : D, f : D) = c(f) \cdot X(5)$

Linearisation: a more algorithmic approach

Consider the following prCRL specification:

$$X(d:D) = \sum_{e:D} a(d{+}e) \sum_{f:D} \frac{1}{|D|} : \left(c(e) \cdot c(f) \cdot X(5) + c(e+f) \cdot X(5) \right)$$

---

1 $X_1(d:D, e:D, f:D) =$
$\sum_{e:D} a(d+e) \sum_{f:D} \frac{1}{|D|} : \ (c(e) \cdot c(f) \cdot X(5) + c(e+f) \cdot X(5))$

---

2 $X_1(d:D, e:D, f:D) = \sum_{e:D} a(d+e) \sum_{f:D} \frac{1}{|D|} : X_2(d, e, f)$
$X_2(d:D, e:D, f:D) = c(e) \cdot c(f) \cdot X(5) + c(e+f) \cdot X(5)$

---

3 $X_1(d:D, e:D, f:D) = \sum_{e:D} a(d+e) \sum_{f:D} \frac{1}{|D|} : X_2(d, e, f)$
$X_2(d:D, e:D, f:D) = c(e) \cdot X_3(d, e, f) + c(e+f) \cdot X_1(5, e, f)$
$X_3(d:D, e:D, f:D) = c(f) \cdot X(5)$

---

4 $X_1(d:D, e:D, f:D) = \sum_{e:D} a(d+e) \sum_{f:D} \frac{1}{|D|} \cdot X_2(d, e, f)$
$X_2(d:D, e:D, f:D) = c(e) \cdot X_3(d, e, f) + c(e+f) \cdot X_1(5, e, f)$

Linearisation: a more algorithmic approach

Consider the following prCRL specification:

$$X(d : D) = \sum_{e:D} a(d+e) \sum_{f:D}^{\bullet} \frac{1}{|D|} : \Big(c(e) \cdot c(f) \cdot X(5) + c(e+f) \cdot X(5) \Big)$$

---

1 $X_1(d : D, e : D, f : D) =$
$\qquad \sum_{e:D} a(d+e) \sum^{\bullet}_{f:D} \frac{1}{|D|} : \ (c(e) \cdot c(f) \cdot X(5) + c(e+f) \cdot X(5))$

---

2 $X_1(d : D, e : D, f : D) = \sum_{e:D} a(d+e) \sum^{\bullet}_{f:D} \frac{1}{|D|} : X_2(d, e, f)$
$\quad X_2(d : D, e : D, f : D) = c(e) \cdot c(f) \cdot X(5) + c(e+f) \cdot X(5)$

---

3 $X_1(d : D, e : D, f : D) = \sum_{e:D} a(d+e) \sum^{\bullet}_{f:D} \frac{1}{|D|} : X_2(d, e, f)$
$\quad X_2(d : D, e : D, f : D) = c(e) \cdot X_3(d, e, f) + c(e+f) \cdot X_1(5, e, f)$
$\quad X_3(d : D, e : D, f : D) = c(f) \cdot X(5)$

---

4 $X_1(d : D, e : D, f : D) = \sum_{e:D} a(d+e) \sum^{\bullet}_{f:D} \frac{1}{|D|} \cdot X_2(d, e, f)$
$\quad X_2(d : D, e : D, f : D) = c(e) \cdot X_3(d, e, f) + c(e+f) \cdot X_1(5, e, f)$

Linearisation: a more algorithmic approach

Consider the following prCRL specification:

$$X(d : D) = \sum_{e:D} a(d{+}e) \sum_{f:D} \frac{1}{|D|} : \Big(c(e) \cdot c(f) \cdot X(5) + c(e+f) \cdot X(5) \Big)$$

---

1 $X_1(d : D, e : D, f : D) =$
$\sum_{e:D} a(d+e) \sum_{f:D} \frac{1}{|D|} : \ (c(e) \cdot c(f) \cdot X(5) + c(e+f) \cdot X(5))$

---

2 $X_1(d : D, e : D, f : D) = \sum_{e:D} a(d+e) \sum_{f:D} \frac{1}{|D|} : X_2(d, e, f)$
$X_2(d : D, e : D, f : D) = c(e) \cdot c(f) \cdot X(5) + c(e+f) \cdot X(5)$

---

3 $X_1(d : D, e : D, f : D) = \sum_{e:D} a(d+e) \sum_{f:D} \frac{1}{|D|} : X_2(d, e, f)$
$X_2(d : D, e : D, f : D) = c(e) \cdot X_3(d, e, f) + c(e+f) \cdot X_1(5, e, f)$
$X_3(d : D, e : D, f : D) = c(f) \cdot X(5)$

---

4 $X_1(d : D, e : D, f : D) = \sum_{e:D} a(d+e) \sum_{f:D} \frac{1}{|D|} \cdot X_2(d, e, f)$
$X_2(d : D, e : D, f : D) = c(e) \cdot X_3(d, e, f) + c(e+f) \cdot X_1(5, e, f)$
$X_3(d : D, e : D, f : D) = c(f) \cdot X_1(5, e, f)$

Linearisation: a more algorithmic approach

Consider the following prCRL specification:

$$X(d:D) = \sum_{e:D} a(d{+}e) \sum_{f:D}^{\bullet} \frac{1}{|D|} : \Big(c(e) \cdot c(f) \cdot X(5) + c(e+f) \cdot X(5)\Big)$$

---

4 $X_1(d:D, e:D, f:D) = \sum_{e:D} a(d+e) \sum^{\bullet}_{f:D} \frac{1}{|D|} \cdot X_2(d,e,f)$

$X_2(d:D, e:D, f:D) = c(e) \cdot X_3(d,e,f) + c(e+f) \cdot X_1(5,e,f)$

$X_3(d:D, e:D, f:D) = c(f) \cdot X_1(5,e,f)$

Linearisation: a more algorithmic approach

Consider the following prCRL specification:

$$X(d:D) = \sum_{e:D} a(d{+}e) \sum_{f:D}\!\!\bullet\ \tfrac{1}{|D|}: \left(c(e)\cdot c(f)\cdot X(5) + c(e+f)\cdot X(5)\right)$$

---

4 $X_1(d:D, e:D, f:D) = \sum_{e:D} a(d+e) \sum\!\!\bullet_{f:D} \frac{1}{|D|} \cdot X_2(d,e,f)$

$X_2(d:D, e:D, f:D) = c(e)\cdot X_3(d,e,f) + c(e+f)\cdot X_1(5,e,f)$

$X_3(d:D, e:D, f:D) = c(f)\cdot X_1(5,e,f)$

$$\begin{aligned} X(\mathrm{pc}:\{1,2,3\}, d:D, e:D, f:D) = {} & \\ & \mathrm{pc}=1 \Rightarrow \sum_{e:D} a(d+e) \sum_{f:D}\!\!\bullet\ \tfrac{1}{|D|}\cdot X(2,d,e,f) \\ & + \mathrm{pc}=2 \Rightarrow c(e)\cdot X(3,d,e,f) \\ & + \mathrm{pc}=2 \Rightarrow c(e+f)\cdot X(1,5,e,f) \\ & + \mathrm{pc}=3 \Rightarrow c(f)\cdot X(1,5,e,f) \end{aligned}$$

Linearisation

In general, we always linearise in two steps:

1. Transform the specification to intermediate regular form (IRF) (every process is a summation of single-action terms)
2. Merge all processes into one big process by introducing a program counter

In the first step, global parameters are introduced to remember the values of bound variables.

Reductions techniques for LPPEs

1. LPPE simplification techniques
    - Constant elimination
    - Summation elimination
    - Expression simplification

Reductions techniques for LPPEs

1. LPPE simplification techniques
   - Constant elimination
   - Summation elimination
   - Expression simplification
2. State space reduction techniques
   - Dead variable reduction
   - Confluence reduction

Reductions techniques for LPPEs

1. LPPE simplification techniques
   - Constant elimination
   - Summation elimination
   - Expression simplification
2. State space reduction techniques
   - Dead variable reduction
   - Confluence reduction

$X(id : Id) = print(id) \cdot X(id)$

init $X(Mark)$

$\rightarrow$

$X = print(Mark) \cdot X$

init X

Reductions techniques for LPPEs

1. LPPE simplification techniques
   - Constant elimination
   - Summation elimination
   - Expression simplification
2. State space reduction techniques
   - Dead variable reduction
   - Confluence reduction

$X = \sum_{d:\{1,2,3\}} d = 2 \Rightarrow send(d) \cdot X$

init X

$\rightarrow$

$X = send(2) \cdot X$

init X

Reductions techniques for LPPEs

1. LPPE simplification techniques
   - Constant elimination
   - Summation elimination
   - Expression simplification
2. State space reduction techniques
   - Dead variable reduction
   - Confluence reduction

$X = (3 = 1 + 2 \vee x > 5) \Rightarrow beep \cdot Y$ $\rightarrow$ $X = beep \cdot Y$

Reductions techniques for LPPEs

1. LPPE simplification techniques
    - Constant elimination
    - Summation elimination
    - Expression simplification
2. State space reduction techniques
    - Dead variable reduction
    - Confluence reduction

- Deduce the control flow of an LPPE
- Examine relevance (liveness) of variables
- Reset dead variables

Reductions techniques for LPPEs

1. LPPE simplification techniques
   - Constant elimination
   - Summation elimination
   - Expression simplification
2. State space reduction techniques
   - Dead variable reduction
   - Confluence reduction

- Detect confluent internal transitions
- Give these transitions priority

Intermediate summary

What you heard so far

- We developed the process algebra prCRL, incorporating both data and probability;
- We defined a normal form for prCRL, the LPPE; starting point for symbolic optimisations and easy state space generation;
- We provided a linearisation algorithm to transform prCRL specifications to LPPEs, proved it correct and implemented it;
- We developed several reduction techniques for LPPEs that preserve strong/branching probabilistic bisimulation.

Contents

The overall goal: efficient and expressive modelling

Specifying systems with

- Nondeterminism ⟵ LTSs
- Probability ⟵ DTMCs
- Stochastic timing ⟵ CTMCs

The overall goal: efficient and expressive modelling

Specifying systems with

- Nondeterminism
- Probability
- Stochastic timing

Nondeterminism and Probability → Probabilistic Automata (PAs)

The overall goal: efficient and expressive modelling

Specifying systems with

- Nondeterminism
- Probability
- Stochastic timing

Interactive Markov Chains (IMCs) (Nondeterminism, Stochastic timing)

The overall goal: efficient and expressive modelling

Specifying systems with

- Nondeterminism
- Probability
- Stochastic timing

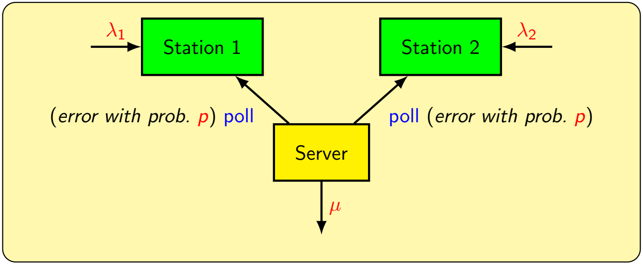
Markov Automata (MAs)

The overall goal: efficient and expressive modelling

Specifying systems with

- Nondeterminism
- Probability
- Stochastic timing

Markov Automata (MAs)

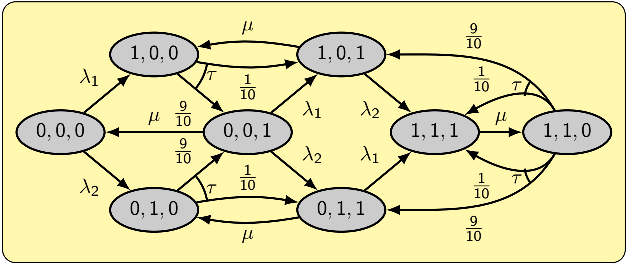


The overall goal: efficient and expressive modelling

Specifying systems with

- Nondeterminism
- Probability
- Stochastic timing

Markov Automata (MAs)

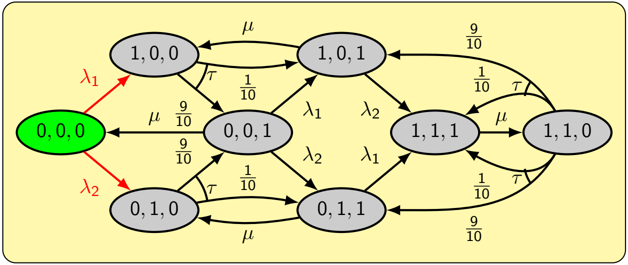


The overall goal: efficient and expressive modelling

Specifying systems with

- Nondeterminism
- Probability
- Stochastic timing

Markov Automata (MAs)

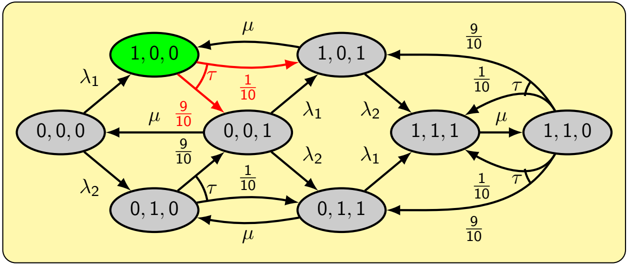


The overall goal: efficient and expressive modelling

Specifying systems with

- Nondeterminism
- Probability
- Stochastic timing

Markov Automata (MAs)

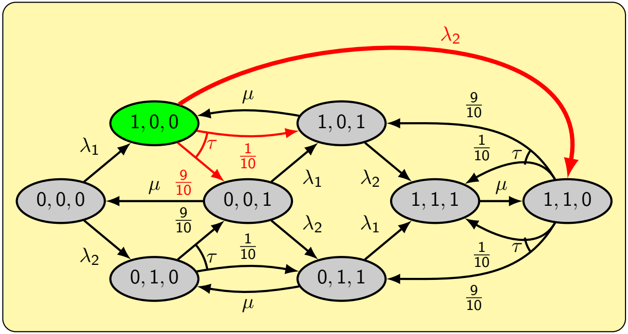


The overall goal: efficient and expressive modelling

Specifying systems with

- Nondeterminism
- Probability
- Stochastic timing

Markov Automata (MAs)

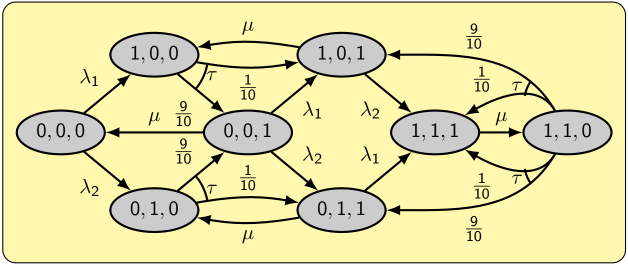


The overall goal: efficient and expressive modelling

Specifying systems with

- Nondeterminism
- Probability
- Stochastic timing

Markov Automata (MAs)

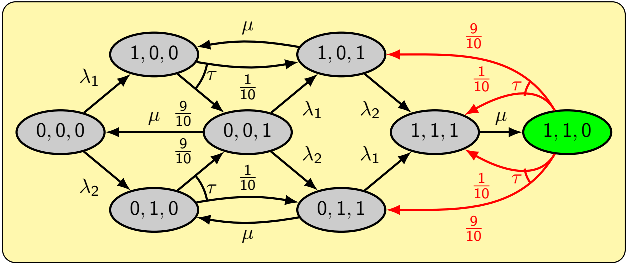


The overall goal: efficient and expressive modelling

Specifying systems with

- Nondeterminism
- Probability
- Stochastic timing

Markov Automata (MAs)

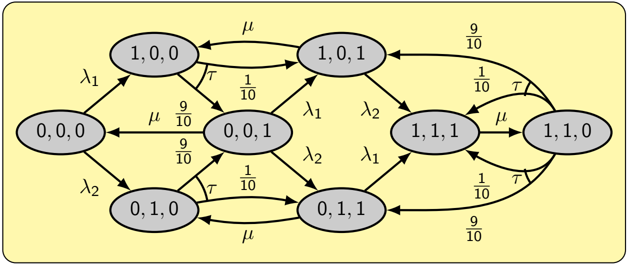


The overall goal: efficient and expressive modelling

Specifying systems with

- Nondeterminism
- Probability
- Stochastic timing

Markov Automata (MAs)



Observed limitations:

- No easy process-algebraic modelling language with data
- Susceptible to the state space explosion problem

Approach: extending and reusing

PA	$\rightarrow$	MA

Approach: extending and reusing

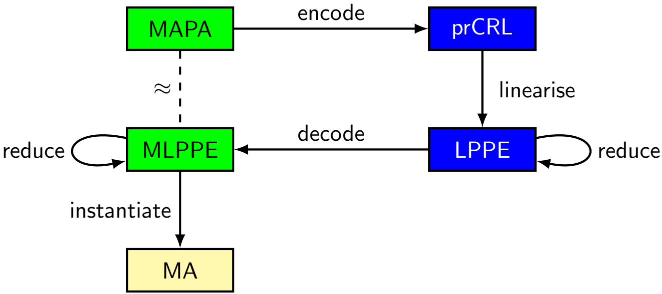
PA	$\rightarrow$	MA	
prCRL	$\rightarrow$	MAPA	(Markov Automata Process Algebra)

Approach: extending and reusing

PA	$\rightarrow$	MA	
prCRL	$\rightarrow$	MAPA	(Markov Automata Process Algebra)
LPPE	$\rightarrow$	MLPPE	(Markovian LPPE)

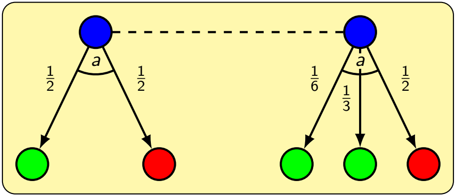
Approach: extending and reusing

PA	$\rightarrow$	MA	
prCRL	$\rightarrow$	MAPA	(Markov Automata Process Algebra)
LPPE	$\rightarrow$	MLPPE	(Markovian LPPE)



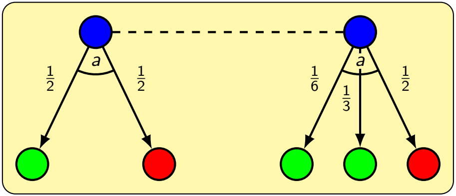
Strong bisimulation for Markov automata

Mimic interactive behaviour:

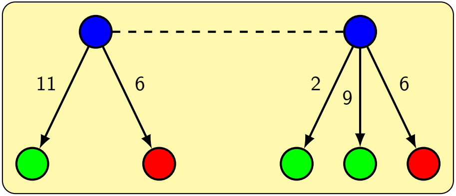


Strong bisimulation for Markov automata

Mimic interactive behaviour:

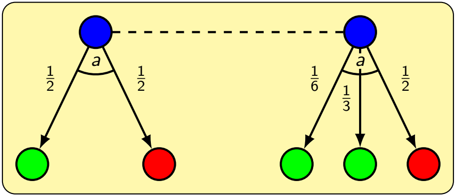


Mimic Markovian behaviour:

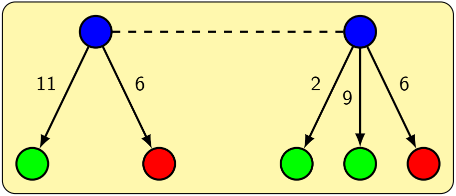


Strong bisimulation for Markov automata

Mimic interactive behaviour:



Mimic Markovian behaviour:



(If a state enables a τ-transition, all rates are ignored.)

A process algebra with data for MAs: MAPA

Specification language MAPA:

- Based on prCRL: data and probabilistic choice
- Additional feature: Markovian rates
- Semantics defined in terms of Markov automata
- Minimal set of operators to facilitate formal manipulation
- Syntactic sugar easily definable

A process algebra with data for MAs: MAPA

Specification language MAPA:

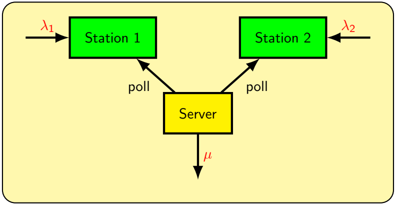
- Based on prCRL: data and probabilistic choice
- Additional feature: Markovian rates
- Semantics defined in terms of Markov automata
- Minimal set of operators to facilitate formal manipulation
- Syntactic sugar easily definable

The grammar of MAPA

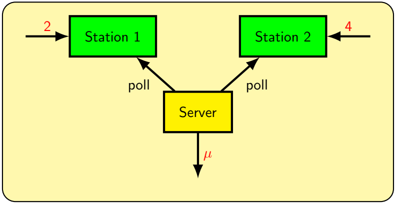
Process terms in MAPA are obtained by the following grammar:

$$p ::= Y(\boldsymbol{t}) \mid c \Rightarrow p \mid p + p \mid \sum_{\boldsymbol{x}:\boldsymbol{D}} p \mid a(\boldsymbol{t}) \sum\!\!\!\!\bullet_{\boldsymbol{x}:\boldsymbol{D}} f : p \mid (\lambda) \cdot p$$

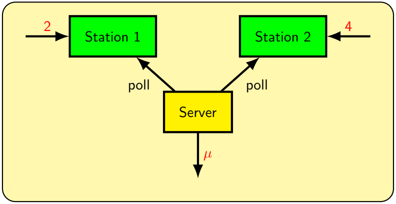
An example specification



An example specification

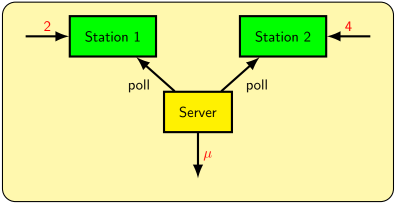


An example specification



- There are 10 types of jobs
- The type of job that arrives is chosen nondeterministically
- Service time depends on job type (hence, we need queues)

An example specification



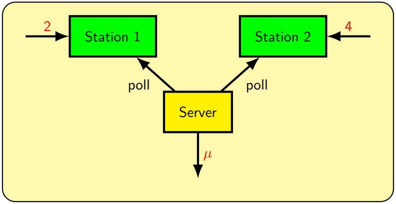
- There are 10 types of jobs
- The type of job that arrives is chosen nondeterministically
- Service time depends on job type (hence, we need queues)

The specification of the stations:

type *Jobs* $= \{1, \ldots, 10\}$

Station$(i : \{1,2\}, q : \text{Queue})$
$= \mathbf{notFull}(q) \quad \Rightarrow (2i) \cdot \sum_{j:Jobs} arrive(j).Station(i, \mathbf{enqueue}(q, j))$

An example specification



- There are 10 types of jobs
- The type of job that arrives is chosen nondeterministically
- Service time depends on job type (hence, we need queues)

The specification of the stations:

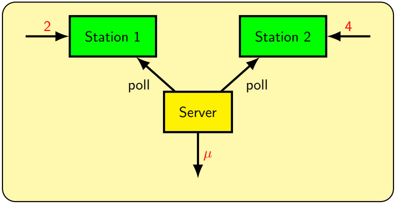
type $\mathit{Jobs} = \{1, \ldots, 10\}$

$\mathit{Station}(i : \{1,2\}, q : \text{Queue})$

$= \mathbf{notFull}(q) \Rightarrow (2i) \cdot \sum_{j:\mathit{Jobs}} \mathit{arrive}(j).\mathit{Station}(i, \mathbf{enqueue}(q, j))$

$+ \mathbf{notEmpty}(q) \Rightarrow \mathit{deliver}(i, \mathbf{head}(q)) \sum_{k \in \{1,9\}} \frac{k}{10} : k = 1 \Rightarrow \mathit{Station}(i, q)$
$+ k = 9 \Rightarrow \mathit{Station}(i, \mathbf{tail}(q))$

An example specification



- There are 10 types of jobs
- The type of job that arrives is chosen nondeterministically
- Service time depends on job type (hence, we need queues)

The specification of the stations:

type $Jobs = \{1, \ldots, 10\}$

$Station(i : \{1, 2\}, q : \text{Queue})$

$= \mathbf{notFull}(q) \quad \Rightarrow (2i) \cdot \sum_{j:Jobs} arrive(j).Station(i, \mathbf{enqueue}(q, j))$

$+ \mathbf{notEmpty}(q) \Rightarrow deliver(i, \mathbf{head}(q))(\frac{1}{10} : Station(i, q) \oplus \frac{9}{10} : Station(i, \mathbf{tail}(q))$

Derivation-based operational semantics

$$\textsc{MarkovPrefix}\ \frac{-}{(\lambda)\cdot p \xrightarrow{\lambda} p} \qquad \textsc{SumLeft}\ \frac{p \xrightarrow{a} p'}{p + q \xrightarrow{a} p'}$$

Derivation-based operational semantics

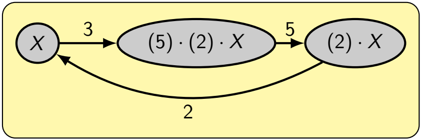
$$\textsc{MarkovPrefix}\ \frac{-}{(\lambda) \cdot p \xrightarrow{\lambda} p} \qquad \textsc{SumLeft}\ \frac{p \xrightarrow{a} p'}{p + q \xrightarrow{a} p'}$$

$$\boxed{X = (3) \cdot (5) \cdot (2) \cdot X}$$

Derivation-based operational semantics

$$\textsc{MarkovPrefix}\ \frac{-}{(\lambda)\cdot p \xrightarrow{\lambda} p} \qquad \textsc{SumLeft}\ \frac{p \xrightarrow{a} p'}{p+q \xrightarrow{a} p'}$$

$$X = (3)\cdot(5)\cdot(2)\cdot X$$

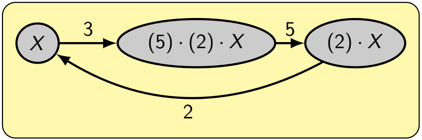


Derivation-based operational semantics

$$\textsc{MarkovPrefix}\ \frac{-}{(\lambda)\cdot p \xrightarrow{\lambda} p} \qquad \textsc{SumLeft}\ \frac{p \xrightarrow{a} p'}{p + q \xrightarrow{a} p'}$$

$$X = (3)\cdot(5)\cdot(2)\cdot X$$

$$X = (3)\cdot(5)\cdot X + c\cdot X$$

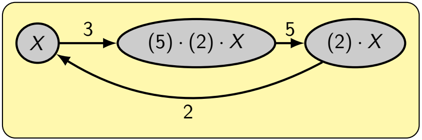


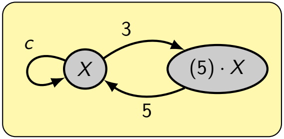
Derivation-based operational semantics

$$\textsc{MarkovPrefix}\ \frac{-}{(\lambda) \cdot p \xrightarrow{\lambda} p} \qquad \textsc{SumLeft}\ \frac{p \xrightarrow{a} p'}{p + q \xrightarrow{a} p'}$$

$$X = (3) \cdot (5) \cdot (2) \cdot X$$

$$X = (3) \cdot (5) \cdot X + c \cdot X$$





Derivation-based operational semantics

$$\textsc{MarkovPrefix}\ \frac{-}{(\lambda) \cdot p \xrightarrow{\lambda} p} \qquad \textsc{SumLeft}\ \frac{p \xrightarrow{a} p'}{p + q \xrightarrow{a} p'}$$

Derivation-based operational semantics

$$\textsc{MarkovPrefix}\ \frac{-}{(\lambda) \cdot p \xrightarrow{\lambda} p} \qquad \textsc{SumLeft}\ \frac{p \xrightarrow{a} p'}{p + q \xrightarrow{a} p'}$$

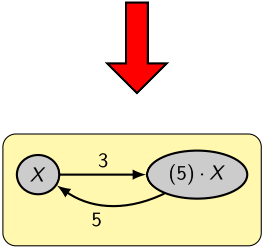
$$\boxed{X = (3) \cdot (5) \cdot X + (3) \cdot (5) \cdot X}$$

Derivation-based operational semantics

$$\textsc{MarkovPrefix}\ \frac{-}{(\lambda)\cdot p \xrightarrow{\lambda} p} \qquad \textsc{SumLeft}\ \frac{p \xrightarrow{a} p'}{p + q \xrightarrow{a} p'}$$

$$X = (3)\cdot(5)\cdot X + (3)\cdot(5)\cdot X$$

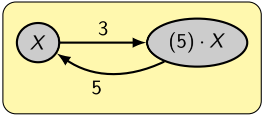
This is not right!



Derivation-based operational semantics

$$\textsc{MarkovPrefix}\ \frac{-}{(\lambda)\cdot p \xrightarrow{\lambda} p} \qquad \textsc{SumLeft}\ \frac{p \xrightarrow{a} p'}{p + q \xrightarrow{a} p'}$$

$$X = (3)\cdot(5)\cdot X + (3)\cdot(5)\cdot X$$



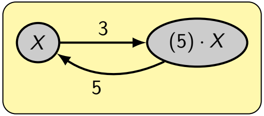
This is not right!

As a solution, we look at derivations:

Derivation-based operational semantics

$$\textsc{MarkovPrefix}\ \frac{-}{(\lambda)\cdot p \xrightarrow{\lambda}_{MP} p} \qquad \textsc{SumLeft}\ \frac{p \xrightarrow{a}_{D} p'}{p+q \xrightarrow{a}_{SL+D} p'}$$

$$X = (3)\cdot(5)\cdot X + (3)\cdot(5)\cdot X$$



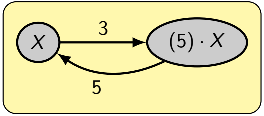
This is not right!

As a solution, we look at derivations:

Derivation-based operational semantics

$$\textsc{MarkovPrefix}\ \frac{-}{(\lambda)\cdot p \xrightarrow{\lambda}_{MP} p} \qquad \textsc{SumLeft}\ \frac{p \xrightarrow{a}_{D} p'}{p + q \xrightarrow{a}_{SL+D} p'}$$

$$X = (3)\cdot(5)\cdot X + (3)\cdot(5)\cdot X$$



This is not right!

As a solution, we look at derivations:

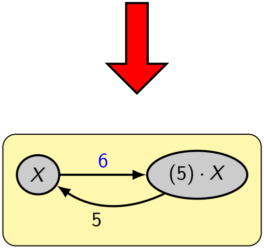
$$X \xrightarrow{3}_{\langle SL,MP\rangle} (5)\cdot X$$
$$X \xrightarrow{3}_{\langle SR,MP\rangle} (5)\cdot X$$

Hence, the total rate from X to $(5)\cdot X$ is $3+3=6$.

Derivation-based operational semantics

$$\textsc{MarkovPrefix}\ \frac{-}{(\lambda)\cdot p \xrightarrow{\lambda}_{MP} p} \qquad \textsc{SumLeft}\ \frac{p \xrightarrow{a}_{D} p'}{p + q \xrightarrow{a}_{SL+D} p'}$$

$$X = (3)\cdot(5)\cdot X + (3)\cdot(5)\cdot X$$



This is not right!

As a solution, we look at derivations:

$$X \xrightarrow{3}_{\langle SL,MP\rangle} (5)\cdot X$$
$$X \xrightarrow{3}_{\langle SR,MP\rangle} (5)\cdot X$$

Hence, the total rate from X to $(5)\cdot X$ is $3+3=6$.

MLPPEs

We defined a special format for MAPA, the MLPPE:

$$X(\boldsymbol{g} : \boldsymbol{G}) = \sum_{i \in I} \sum_{\boldsymbol{d_i} : \boldsymbol{D_i}} c_i \Rightarrow a_i(\boldsymbol{b_i}) \sum_{\bullet\, \boldsymbol{e_i} : \boldsymbol{E_i}} f_i : X(\boldsymbol{n_i})$$

$$+ \sum_{j \in J} \sum_{\boldsymbol{d_j} : \boldsymbol{D_j}} c_j \Rightarrow (\lambda_j) \cdot X(\boldsymbol{n_j})$$

MLPPEs

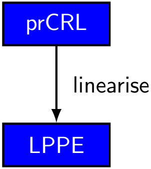
We defined a special format for MAPA, the MLPPE:

$$X(\boldsymbol{g} : \boldsymbol{G}) = \sum_{i \in I} \sum_{\boldsymbol{d_i} : \boldsymbol{D_i}} c_i \Rightarrow a_i(\boldsymbol{b_i}) \sum_{\bullet \, \boldsymbol{e_i} : \boldsymbol{E_i}} f_i : X(\boldsymbol{n_i})$$
$$+ \sum_{j \in J} \sum_{\boldsymbol{d_j} : \boldsymbol{D_j}} c_j \Rightarrow (\lambda_j) \cdot X(\boldsymbol{n_j})$$

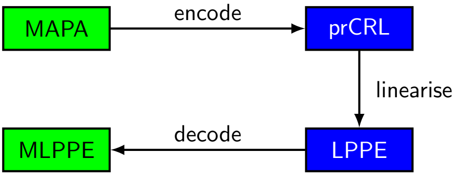
Advantages of using MLPPEs instead of MAPA specifications:

- Easy state space generation
- Straight-forward parallel composition
- **Symbolic optimisations enabled at the language level**

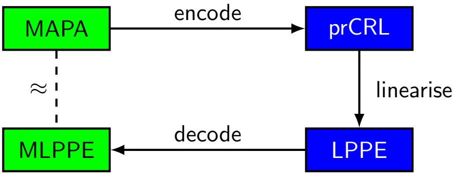
Encoding into prCRL



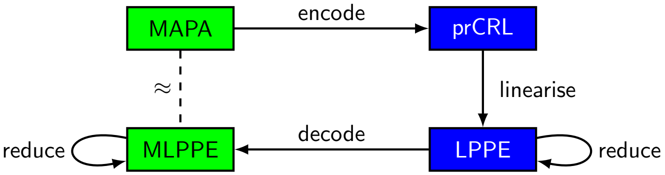
Encoding into prCRL



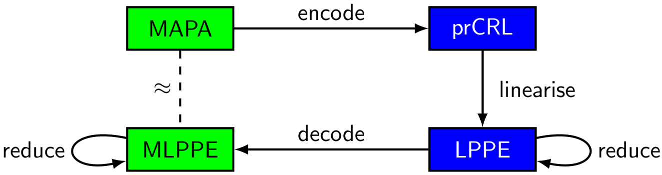
Encoding into prCRL



Encoding into prCRL

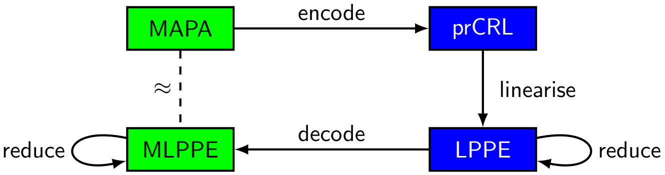


Encoding into prCRL



Basic idea: encode a rate λ as action rate(λ).

Encoding into prCRL

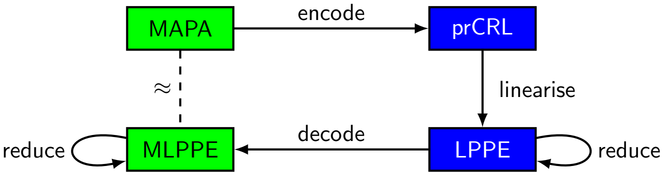


Basic idea: encode a rate λ as action rate(λ).

Problem:

Bisimulation-preserving reductions on prCRL might change MAPA behaviour

Encoding into prCRL



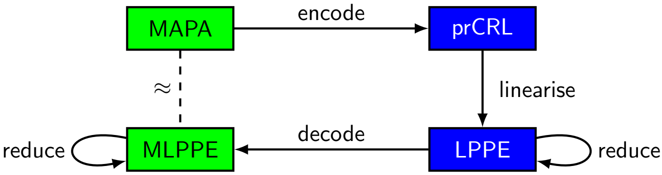
Basic idea: encode a rate λ as action rate(λ).

Problem:

Bisimulation-preserving reductions on prCRL might change MAPA behaviour

$$(\lambda) \cdot p + (\lambda) \cdot p$$

Encoding into prCRL



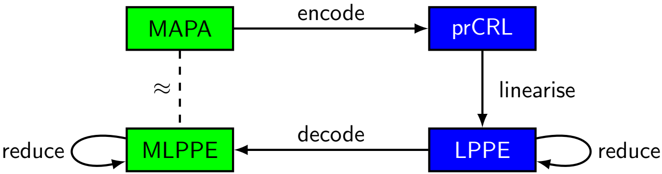
Basic idea: encode a rate λ as action rate(λ).

Problem:

Bisimulation-preserving reductions on prCRL might change MAPA behaviour

$$(\lambda) \cdot p + (\lambda) \cdot p \;\Rightarrow\; \text{rate}(\lambda) \cdot p + \text{rate}(\lambda) \cdot p$$

Encoding into prCRL



Basic idea: encode a rate λ as action rate(λ).

Problem:

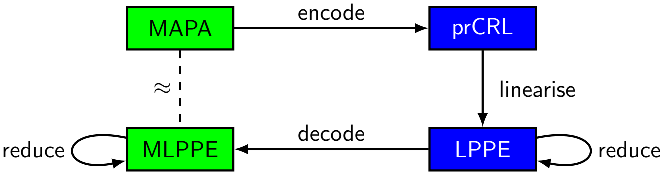
Bisimulation-preserving reductions on prCRL might change MAPA behaviour

$$(\lambda) \cdot p + (\lambda) \cdot p \;\Rightarrow\; \text{rate}(\lambda) \cdot p + \text{rate}(\lambda) \cdot p$$

$$\approx_{\text{PA}}$$

$$\text{rate}(\lambda) \cdot p$$

Encoding into prCRL



Basic idea: encode a rate λ as action rate(λ).

Problem:

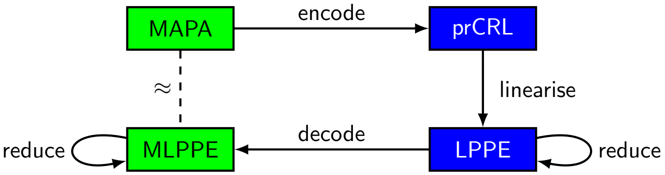
Bisimulation-preserving reductions on prCRL might change MAPA behaviour

$$(\lambda) \cdot p + (\lambda) \cdot p \;\Rightarrow\; \text{rate}(\lambda) \cdot p + \text{rate}(\lambda) \cdot p$$

$$\approx_{\text{PA}}$$

$$(\lambda) \cdot p \;\Leftarrow\; \text{rate}(\lambda) \cdot p$$

Encoding into prCRL



Basic idea: encode a rate λ as action rate(λ).

Problem:

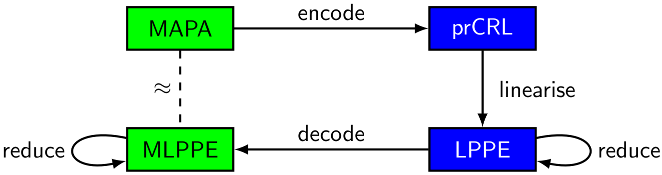
Bisimulation-preserving reductions on prCRL might change MAPA behaviour

$$(\lambda) \cdot p + (\lambda) \cdot p \;\Rightarrow\; \text{rate}(\lambda) \cdot p + \text{rate}(\lambda) \cdot p$$

$$\not\approx_{\text{MA}} \qquad\qquad \approx_{\text{PA}}$$

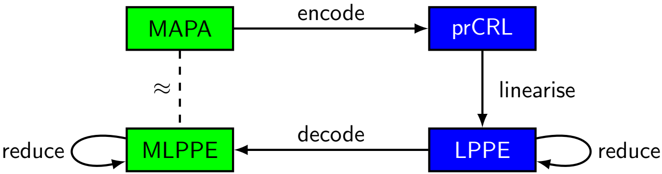
$$(\lambda) \cdot p \;\Leftarrow\; \text{rate}(\lambda) \cdot p$$

Encoding into prCRL



Possible solution: encode a rate λ as action rate$_i(\lambda)$.

Encoding into prCRL

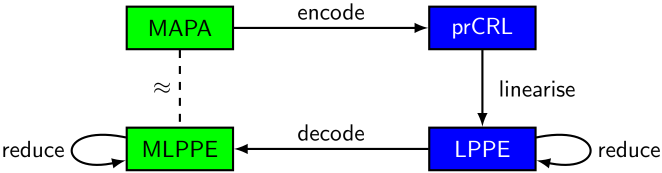


Possible solution: encode a rate λ as action $\text{rate}_i(\lambda)$.

Problem:

This still doesn't work. . .

Encoding into prCRL



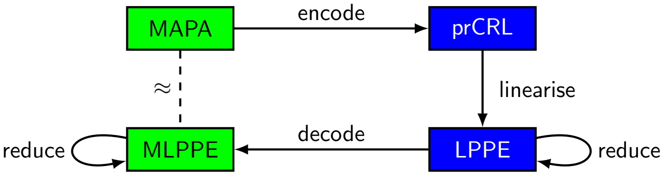
Possible solution: encode a rate λ as action $\text{rate}_i(\lambda)$.

Problem:

This still doesn't work...

$$(\lambda) \cdot p + (\lambda) \cdot p$$

Encoding into prCRL



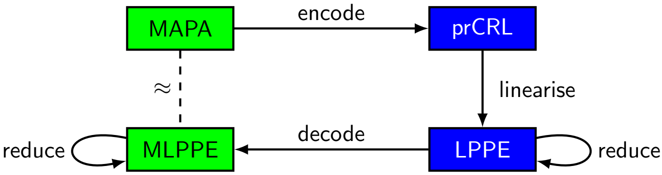
Possible solution: encode a rate λ as action $\text{rate}_i(\lambda)$.

Problem:

This still doesn't work. . .

$$(\lambda) \cdot p + (\lambda) \cdot p \;\Rightarrow\; \text{rate}_1(\lambda) \cdot p + \text{rate}_2(\lambda) \cdot p$$

Encoding into prCRL



Possible solution: encode a rate λ as action $\text{rate}_i(\lambda)$.

Problem:

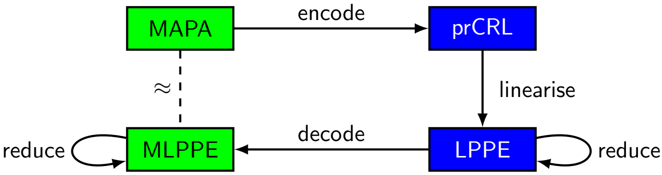
This still doesn't work...

$$(\lambda) \cdot p + (\lambda) \cdot p \;\Rightarrow\; \text{rate}_1(\lambda) \cdot p + \text{rate}_2(\lambda) \cdot p$$

$$\approx_{\text{PA}}$$

$$\text{rate}_1(\lambda) \cdot p + \text{rate}_2(\lambda) \cdot p + \text{rate}_2(\lambda) \cdot p$$

Encoding into prCRL



Possible solution: encode a rate λ as action $\text{rate}_i(\lambda)$.

Problem:

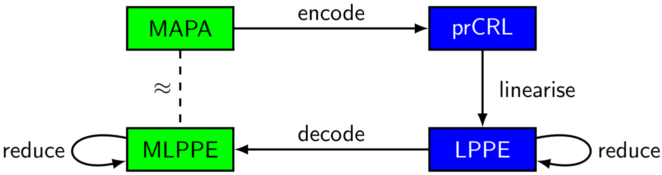
This still doesn't work...

$$(\lambda) \cdot p + (\lambda) \cdot p \;\Rightarrow\; \text{rate}_1(\lambda) \cdot p + \text{rate}_2(\lambda) \cdot p$$

$$\approx_{\text{PA}}$$

$$(\lambda) \cdot p + (\lambda) \cdot p + (\lambda) \cdot p \;\Leftarrow\; \text{rate}_1(\lambda) \cdot p + \text{rate}_2(\lambda) \cdot p + \text{rate}_2(\lambda) \cdot p$$

Encoding into prCRL



Possible solution: encode a rate λ as action $\text{rate}_i(\lambda)$.

Problem:

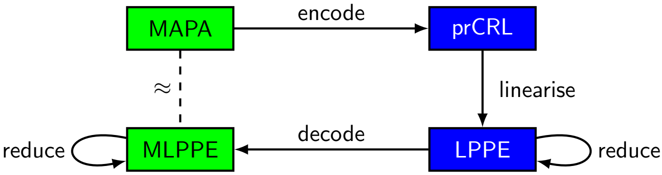
This still doesn't work...

$$(\lambda) \cdot p + (\lambda) \cdot p \;\Rightarrow\; \text{rate}_1(\lambda) \cdot p + \text{rate}_2(\lambda) \cdot p$$

$$\not\approx_{\text{MA}} \qquad\qquad \approx_{\text{PA}}$$

$$(\lambda) \cdot p + (\lambda) \cdot p + (\lambda) \cdot p \;\Leftarrow\; \text{rate}_1(\lambda) \cdot p + \text{rate}_2(\lambda) \cdot p + \text{rate}_2(\lambda) \cdot p$$

Encoding into prCRL



Possible solution: encode a rate λ as action rate$_i(\lambda)$.

Problem:

This still doesn't work. . .

$$(\lambda) \cdot p + (\lambda) \cdot p \;\Rightarrow\; \text{rate}_1(\lambda) \cdot p + \text{rate}_2(\lambda) \cdot p$$

$$\not\approx_{MA} \qquad\qquad \approx_{PA}$$

$$(\lambda) \cdot p + (\lambda) \cdot p + (\lambda) \cdot p \;\Leftarrow\; \text{rate}_1(\lambda) \cdot p + \text{rate}_2(\lambda) \cdot p + \text{rate}_2(\lambda) \cdot p$$

Stronger equivalence on prCRL specifications needed!

Derivation-preserving bisimulation

Two prCRL terms are derivation-preserving bisimulation if

- There is a strong bisimulation relation R containing them

Derivation-preserving bisimulation

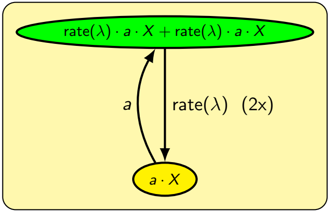
Two prCRL terms are derivation-preserving bisimulation if

- There is a strong bisimulation relation R containing them
- Every bisimilar pair (p, p') has the same number of rate(λ) derivations to every equivalence class $[r]_R$.

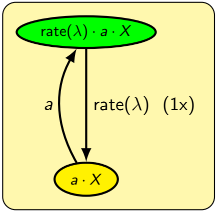
Derivation-preserving bisimulation

Two prCRL terms are derivation-preserving bisimulation if

- There is a strong bisimulation relation R containing them
- Every bisimilar pair (p, p') has the same number of rate(λ) derivations to every equivalence class $[r]_R$.



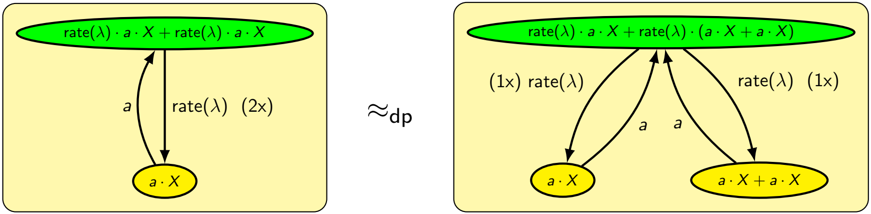
$\not\approx_{\text{dp}}$



Derivation-preserving bisimulation

Two prCRL terms are derivation-preserving bisimulation if

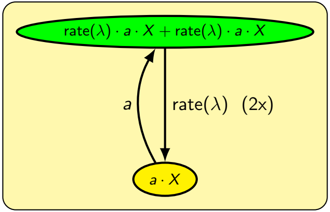
- There is a strong bisimulation relation R containing them
- Every bisimilar pair (p, p') has the same number of rate(λ) derivations to every equivalence class $[r]_R$.



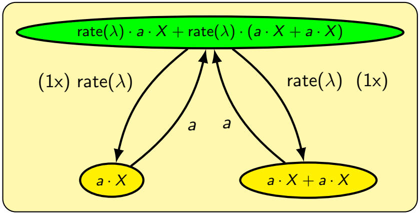
Derivation-preserving bisimulation

Two prCRL terms are derivation-preserving bisimulation if

- There is a strong bisimulation relation R containing them
- Every bisimilar pair (p, p') has the same number of rate(λ) derivations to every equivalence class $[r]_R$.



$\approx_{\text{dp}}$



Proposition

Derivation-preserving bisimulation is a congruence for prCRL.

Derivation-preserving bisimulation: important results

Theorem

Given a derivation-preserving prCRL transformation f,

$$\text{decode}(f(\text{encode}(M))) \approx M$$

for every MAPA specification M.

Derivation-preserving bisimulation: important results

Theorem

Given a derivation-preserving prCRL transformation f,

$$\text{decode}(f(\text{encode}(M))) \approx M$$

for every MAPA specification M.

This enables many techniques from the PA world to be generalised trivially to the MA world!

Derivation-preserving bisimulation: important results

Theorem

Given a derivation-preserving prCRL transformation f,

$$\text{decode}(f(\text{encode}(M))) \approx M$$

for every MAPA specification M.

This enables many techniques from the PA world to be generalised trivially to the MA world!

Corollary

The linearisation procedure of prCRL can be reused for MAPA.

Generalising existing reduction techniques

Existing reduction techniques that preserve derivations:

- Constant elimination
- Expression simplification
- Dead variable reduction

Generalising existing reduction techniques

Existing reduction techniques that preserve derivations:

- Constant elimination
- Expression simplification
- Dead variable reduction

Generalising existing reduction techniques

Existing reduction techniques that preserve derivations:

- Constant elimination
- Expression simplification
- Dead variable reduction

$$\text{deadVarRed} = \text{decode} \circ \text{deadVarRedOld} \circ \text{encode}$$

Novel reduction techniques

New reduction techniques for MAPA:

- Maximal progress reduction
- Summation elimination

Novel reduction techniques

New reduction techniques for MAPA:

- Maximal progress reduction
- Summation elimination

$X = \tau \cdot X + (5) \cdot X$ $\rightarrow$ $X = \tau \cdot X$

Novel reduction techniques

New reduction techniques for MAPA:

- Maximal progress reduction
- Summation elimination

$$X = \sum_{d:\{1,2,3\}} d = 2 \Rightarrow send(d) \cdot X$$

$$Y = \sum_{d:\{1,2,3\}} (5) \cdot Y$$

$\rightarrow$

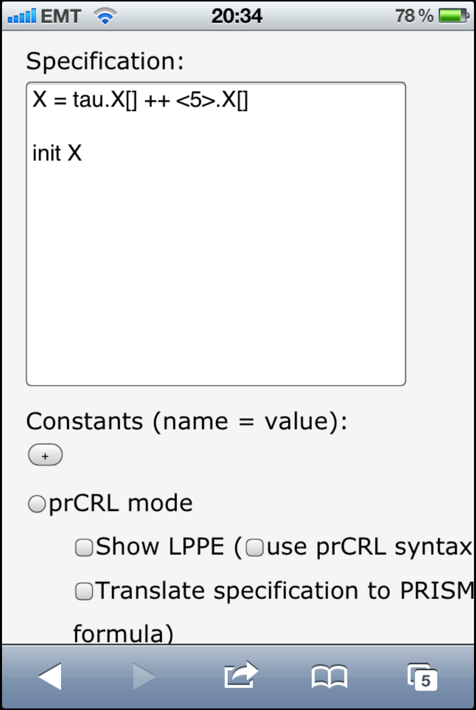
$$X = send(2) \cdot X$$

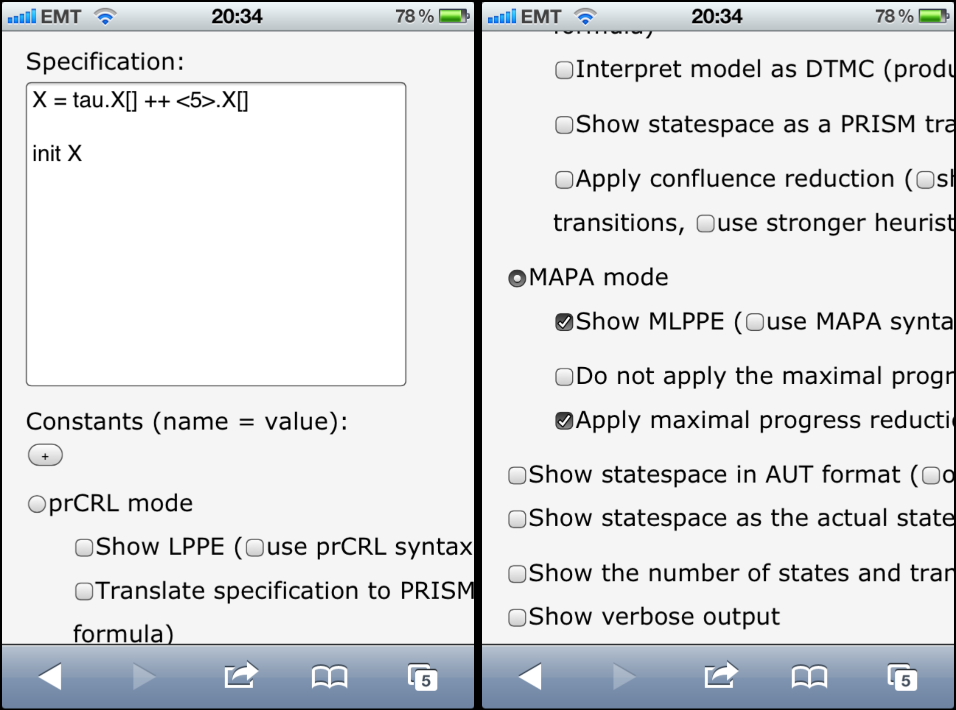
$$Y = (15) \cdot Y$$

Implementation and Case Study

Implementation in SCOOP:

- Programmed in Haskell
- Stand-alone and web-based interface
- Linearisation, optimisation, state space generation


EMT
20:34
78 %
Specification:
X = tau.X[] ++ <5>.X[]
init X
Constants (name = value):
+
prCRL mode
Show LPPE (use prCRL syntax
Translate specification to PRISM
formula)
5


EMT
20:34
78 %
Specification:
X = tau.X[] ++ <5>.X[]
init X
Constants (name = value):
+
prCRL mode
Show LPPE (use prCRL syntax
Translate specification to PRISM
formula)
EMT
20:34
78 %
Interpret model as DTMC (produ
Show statespace as a PRISM tra
Apply confluence reduction (sh
transitions, use stronger heurist
MAPA mode
Show MLPPE (use MAPA synta
Do not apply the maximal progr
Apply maximal progress reducti
Show statespace in AUT format (o
Show statespace as the actual state
Show the number of states and tran
Show verbose output
5
5

EMT 20:35 78 %

Apply unused variable reduction

☐ Apply dead variable reduction

☐ Apply transition merging

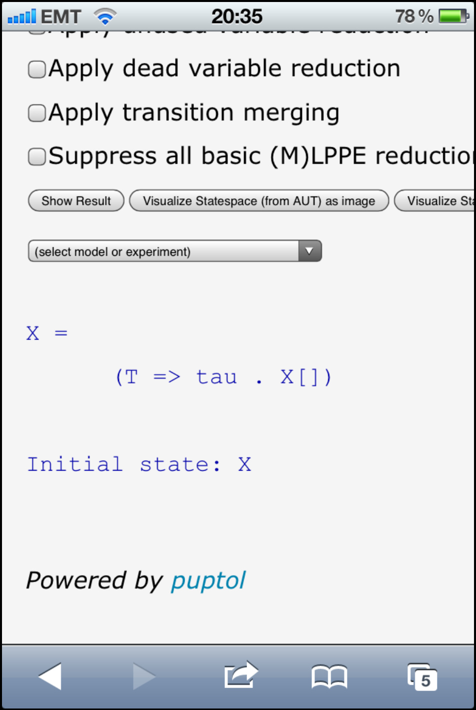
☐ Suppress all basic (M)LPPE reductio

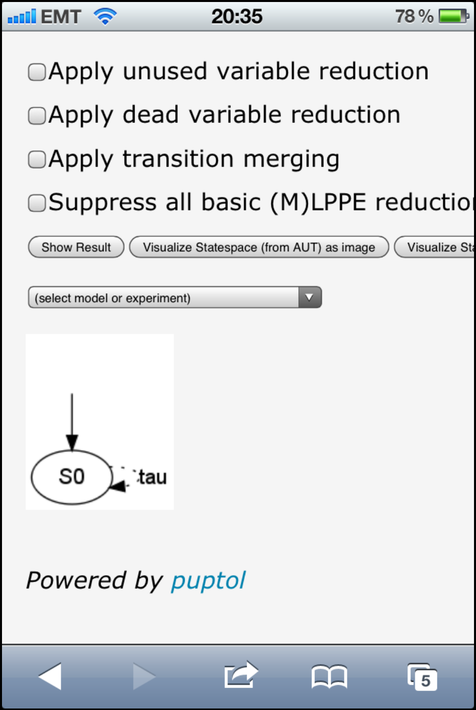
Show Result | Visualize Statespace (from AUT) as image | Visualize St

(select model or experiment)

```
X =
       (T => tau . X[])


Initial state: X
```

Powered by puptol


EMT
20:35
78 %
Apply dead variable reduction
Apply transition merging
Suppress all basic (M)LPPE reductio
Show Result
Visualize Statespace (from AUT) as image
Visualize Sta
(select model or experiment)
X =
(T => tau . X[])
Initial state: X
Powered by puptol
5


EMT
20:35
78 %
Apply unused variable reduction
Apply dead variable reduction
Apply transition merging
Suppress all basic (M)LPPE reductio
Show Result
Visualize Statespace (from AUT) as image
Visualize Sta
(select model or experiment)
S0
tau
Powered by puptol
5

Implementation and Case Study

Implementation in SCOOP:

- Programmed in Haskell
- Stand-alone and web-based interface
- Linearisation, optimisation, state space generation

	Original				Reduced				
Spec.	States	Trans.	MLPPE	Time	States	Trans.	MLPPE	Time	Red.
queue-3-5	316,058	581,892	15 / 335	87.4	218,714	484,548	8 / 224	20.7	76%
queue-3-6	1,005,699	1,874,138	15 / 335	323.3	670,294	1,538,733	8 / 224	64.7	80%
queue-3-6'	1,005,699	1,874,138	15 / 335	319.5	74	108	5 / 170	0.0	100%
queue-5-2	27,659	47,130	15 / 335	4.3	23,690	43,161	8 / 224	1.9	56%
queue-5-3	1,191,738	2,116,304	15 / 335	235.8	926,746	1,851,312	8 / 224	84.2	64%
queue-5-3'	1,191,738	2,116,304	15 / 335	233.2	170	256	5 / 170	0.0	100%
queue-25-1	3,330	5,256	15 / 335	0.5	3,330	5,256	8 / 224	0.4	20%
queue-100-1	50,805	81,006	15 / 335	8.9	50,805	81,006	8 / 224	6.6	26%
mutex-3-2	17,352	40,200	27 / 3,540	12.3	10,560	25,392	12 / 2,190	4.6	63%
mutex-3-4	129,112	320,136	27 / 3,540	95.8	70,744	169,128	12 / 2,190	30.3	68%
mutex-3-6	425,528	1,137,048	27 / 3,540	330.8	224,000	534,624	12 / 2,190	99.0	70%
mutex-4-1	27,701	80,516	36 / 5,872	33.0	20,025	62,876	16 / 3,632	13.5	59%
mutex-4-2	360,768	1,035,584	36 / 5,872	435.9	218,624	671,328	16 / 3,632	145.5	67%
mutex-4-3	1,711,141	5,015,692	36 / 5,872	2,108.0	958,921	2,923,300	16 / 3,632	644.3	69%
mutex-5-1	294,882	1,051,775	45 / 8,780	549.7	218,717	841,750	20 / 5,430	216.6	61%

Table: State space generation using SCOOP.

Implementation and Case Study

Implementation in SCOOP:

- Programmed in Haskell
- Stand-alone and web-based interface
- Linearisation, optimisation, state space generation

Spec.	Original States	Original Trans.	Original MLPPE	Original Time	Reduced States	Reduced Trans.	Reduced MLPPE	Reduced Time	Red.
queue-3-5	316,058	581,892	15 / 335	87.4	218,714	484,548	8 / 224	20.7	76%
queue-3-6	1,005,699	1,874,138	15 / 335	323.3	670,294	1,538,733	8 / 224	64.7	80%
queue-3-6'	1,005,699	1,874,138	15 / 335	319.5	74	108	5 / 170	0.0	100%
queue-5-2	27,659	47,130	15 / 335	4.3	23,690	43,161	8 / 224	1.9	56%
queue-5-3	1,191,738	2,116,304	15 / 335	235.8	926,746	1,851,312	8 / 224	84.2	64%
queue-5-3'	1,191,738	2,116,304	15 / 335	233.2	170	256	5 / 170	0.0	100%
queue-25-1	3,330	5,256	15 / 335	0.5	3,330	5,256	8 / 224	0.4	20%
queue-100-1	50,805	81,006	15 / 335	8.9	50,805	81,006	8 / 224	6.6	26%
mutex-3-2	17,352	40,200	27 / 3,540	12.3	10,560	25,392	12 / 2,190	4.6	63%
mutex-3-4	129,112	320,136	27 / 3,540	95.8	70,744	169,128	12 / 2,190	30.3	68%
mutex-3-6	425,528	1,137,048	27 / 3,540	330.8	224,000	534,624	12 / 2,190	99.0	70%
mutex-4-1	27,701	80,516	36 / 5,872	33.0	20,025	62,876	16 / 3,632	13.5	59%
mutex-4-2	360,768	1,035,584	36 / 5,872	435.9	218,624	671,328	16 / 3,632	145.5	67%
mutex-4-3	1,711,141	5,015,692	36 / 5,872	2,108.0	958,921	2,923,300	16 / 3,632	644.3	69%
mutex-5-1	294,882	1,051,775	45 / 8,780	549.7	218,717	841,750	20 / 5,430	216.6	61%

Table: State space generation using SCOOP.

Implementation and Case Study

Implementation in SCOOP:

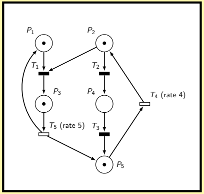
- Programmed in Haskell
- Stand-alone and web-based interface
- Linearisation, optimisation, state space generation

	Original				Reduced				
Spec.	States	Trans.	MLPPE	Time	States	Trans.	MLPPE	Time	Red.
queue-3-5	316,058	581,892	15 / 335	87.4	218,714	484,548	8 / 224	20.7	76%
queue-3-6	1,005,699	1,874,138	15 / 335	323.3	670,294	1,538,733	8 / 224	64.7	80%
queue-3-6'	1,005,699	1,874,138	15 / 335	319.5	74	108	5 / 170	0.0	100%
queue-5-2	27,659	47,130	15 / 335	4.3	23,690	43,161	8 / 224	1.9	56%
queue-5-3	1,191,738	2,116,304	15 / 335	235.8	926,746	1,851,312	8 / 224	84.2	64%
queue-5-3'	1,191,738	2,116,304	15 / 335	233.2	170	256	5 / 170	0.0	100%
queue-25-1	3,330	5,256	15 / 335	0.5	3,330	5,256	8 / 224	0.4	20%
queue-100-1	50,805	81,006	15 / 335	8.9	50,805	81,006	8 / 224	6.6	26%
mutex-3-2	17,352	40,200	27 / 3,540	12.3	10,560	25,392	12 / 2,190	4.6	63%
mutex-3-4	129,112	320,136	27 / 3,540	95.8	70,744	169,128	12 / 2,190	30.3	68%
mutex-3-6	425,528	1,137,048	27 / 3,540	330.8	224,000	534,624	12 / 2,190	99.0	70%
mutex-4-1	27,701	80,516	36 / 5,872	33.0	20,025	62,876	16 / 3,632	13.5	59%
mutex-4-2	360,768	1,035,584	36 / 5,872	435.9	218,624	671,328	16 / 3,632	145.5	67%
mutex-4-3	1,711,141	5,015,692	36 / 5,872	2,108.0	958,921	2,923,300	16 / 3,632	644.3	69%
mutex-5-1	294,882	1,051,775	45 / 8,780	549.7	218,717	841,750	20 / 5,430	216.6	61%

Table: State space generation using SCOOP.

GSPN analysis

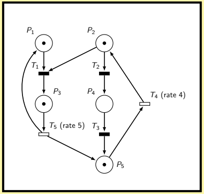
GSPN (PNML)



`reach P1 = 1 & P5 = 2`

GSPN analysis

GSPN (PNML)




```
reach P1 = 1 & P5 = 2
```

GEMMA

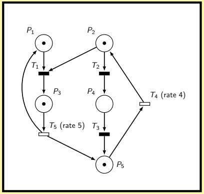
MAPA

```
GSPN(P1:ℕ,P2:ℕ,P3:ℕ,
P4:ℕ,P5:ℕ) =

P2 >= 1 => T2 .
    GSPN[P2--, P4++]
+ P5 >= 1 => (4.0) .
    GSPN[P2++, P5--]
+ ...

init GSPN(1,1,1,0,1)
```

```
reach P1 = 1 & P5 = 2
```

GSPN analysis

GSPN (PNML)



`reach P1 = 1 & P5 = 2`

GEMMA

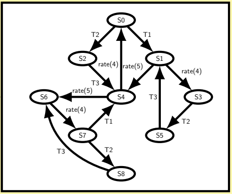
MAPA

```
GSPN(P1:ℕ,P2:ℕ,P3:ℕ,
P4:ℕ,P5:ℕ) =

P2 >= 1 => T2 .
    GSPN[P2--, P4++]
+ P5 >= 1 => (4.0) .
    GSPN[P2++, P5--]
+ ...

init GSPN(1,1,1,0,1)
```

`reach P1 = 1 & P5 = 2`

SCOOP

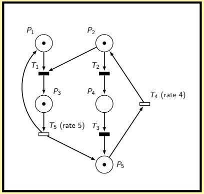
MA



`#GOALS S4`

GSPN analysis

GSPN (PNML)



`reach P1 = 1 & P5 = 2`

GEMMA

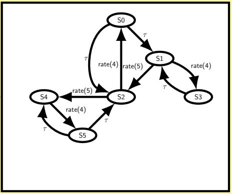
MAPA

```
GSPN(P1:ℕ,P2:ℕ,P3:ℕ,
P4:ℕ,P5:ℕ) =

P2 >= 1 => T2 .
    GSPN[P2--, P4++]
+ P5 >= 1 => (4.0) .
    GSPN[P2++, P5--]
+ ...

init GSPN(1,1,1,0,1)
```

`reach P1 = 1 & P5 = 2`

SCOOP (optimised)

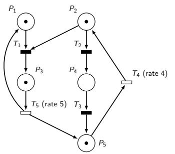
MA



`#GOALS S2`

GSPN analysis

GSPN (PNML)



`reach P1 = 1 & P5 = 2`

GEMMA

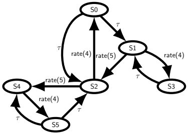
MAPA

```
GSPN(P1:ℕ,P2:ℕ,P3:ℕ,
P4:ℕ,P5:ℕ) =

P2 >= 1 => T2 .
    GSPN[P2--, P4++]
+ P5 >= 1 => (4.0) .
    GSPN[P2++, P5--]
+ ...

init GSPN(1,1,1,0,1)
```

`reach P1 = 1 & P5 = 2`

SCOOP (optimised)

MA



`#GOALS S2`

IMCA

Results

Min. unbounded reach.:	1.0
Max. unbounded reach.:	1.0
Min. expected time:	0.0
Max. expected time:	0.2
Min. LRA:	0.0
Max. LRA:	0.4

`#GOALS S2`

Conclusions and Future Work

Conclusions:

- We introduced a new process-algebraic framework (MAPA) with data for modelling and generating Markov automata
- We introduced the MLPPE for easy state space generation, parallel composition and reduction techniques

Conclusions and Future Work

Conclusions:

- We introduced a new process-algebraic framework (MAPA) with data for modelling and generating Markov automata
- We introduced the MLPPE for easy state space generation, parallel composition and reduction techniques
- We showed an encoding of MAPA into prCRL
- We showed when prCRL techniques can be used safely by encoding, using a novel notion of bisimulation

Conclusions and Future Work

Conclusions:

- We introduced a new process-algebraic framework (MAPA) with data for modelling and generating Markov automata
- We introduced the MLPPE for easy state space generation, parallel composition and reduction techniques
- We showed an encoding of MAPA into prCRL
- We showed when prCRL techniques can be used safely by encoding, using a novel notion of bisimulation
- All our results apply to LTSs, DTMCs, CTMCs, IMCs and PAs
- Model checking of MAs and GSPNs is now possible

Conclusions and Future Work

Conclusions:

- We introduced a new process-algebraic framework (MAPA) with data for modelling and generating Markov automata
- We introduced the MLPPE for easy state space generation, parallel composition and reduction techniques
- We showed an encoding of MAPA into prCRL
- We showed when prCRL techniques can be used safely by encoding, using a novel notion of bisimulation
- All our results apply to LTSs, DTMCs, CTMCs, IMCs and PAs
- Model checking of MAs and GSPNs is now possible

Future Work:

- Generalise confluence reduction to MAs and MAPA

Questions

Questions?

Have a look at `fmt.cs.utwente.nl/~timmer/scoop`