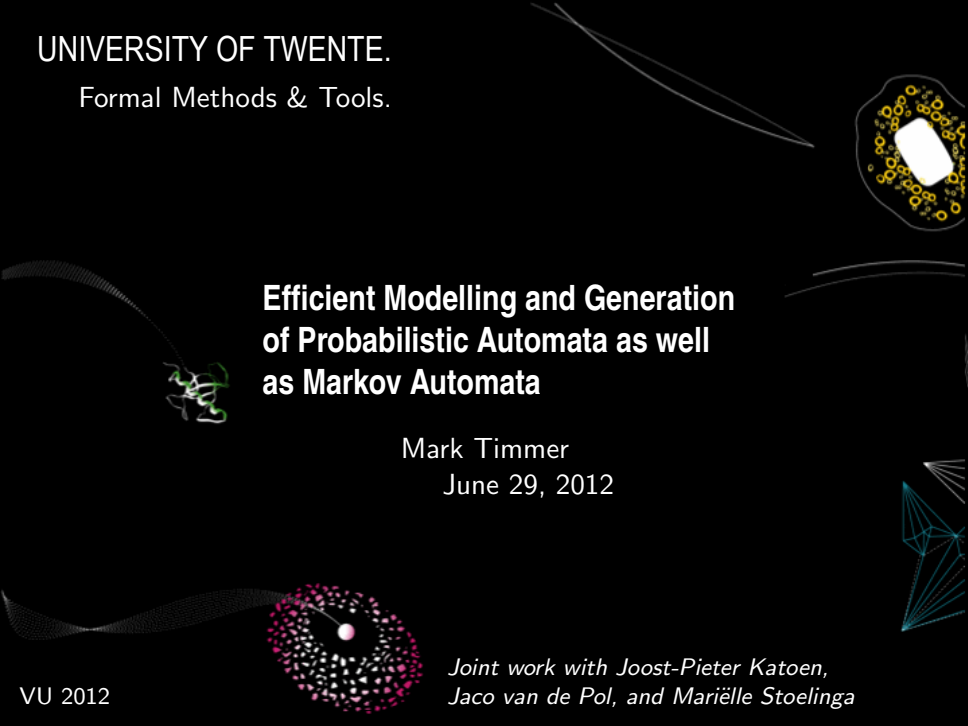


UNIVERSITY OF TWENTE.

Formal Methods & Tools.



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

Mark Timmer

June 29, 2012

The context: probabilistic model checking

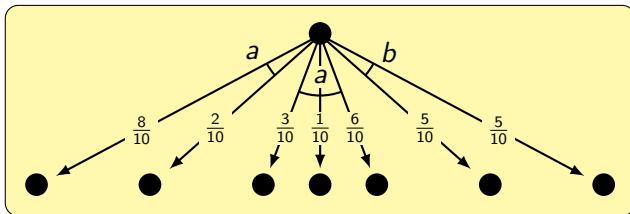
Probabilistic model checking:

- 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)

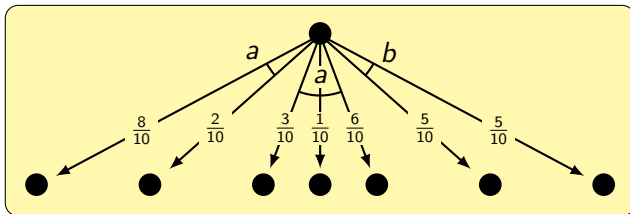


- **Non-deterministically** choose one of the three transitions
- **Probabilistically** choose the next state

The context: probabilistic model checking

Probabilistic model checking:

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

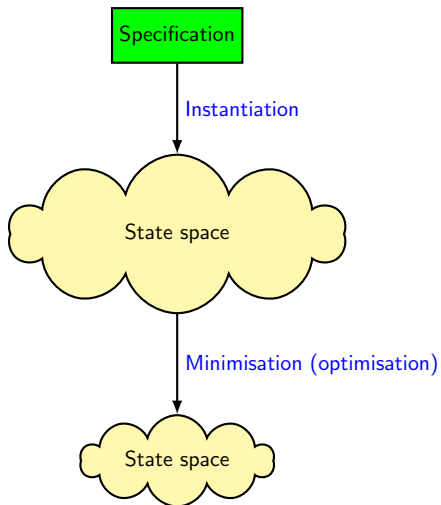


- **Non-deterministically** choose one of the three transitions
- **Probabilistically** choose the next state

Limitations of previous approaches:

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

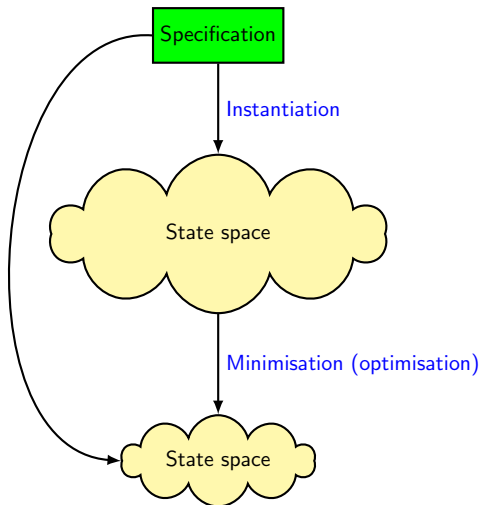
Combating the state space explosion



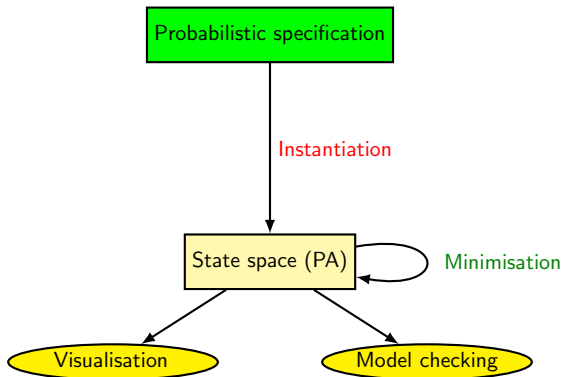
Combating the state space explosion

Optimised instantiation

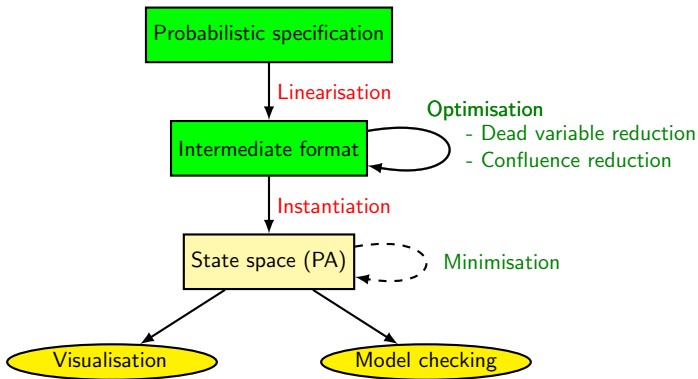
- Dead variable reduction
- Confluence reduction



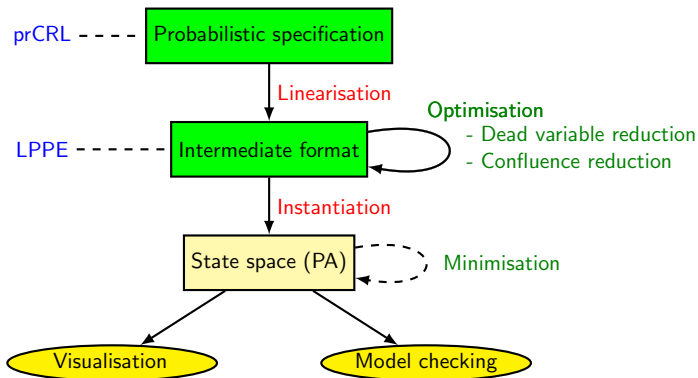
Overview of our approach



Overview of our approach

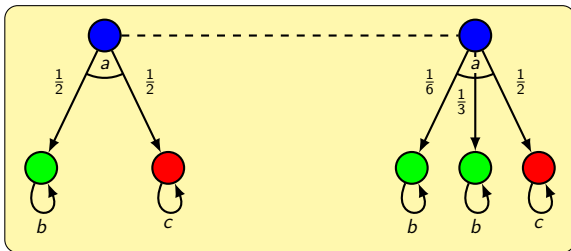


Overview of our approach



Strong bisimulation for Probabilistic Automata

Mimic behaviour with equal probabilities:



Contents

- 1 Introduction
- 2 A process algebra with data and probability: prCRL
- 3 Linearisation: from prCRL to LPPE
- 4 Reduction techniques
- 5 Modelling Markov Automata using MAPA
- 6 Encoding and decoding
- 7 Reduction techniques revisited
- 8 Case study
- 9 Conclusions and Future Work

A process algebra with data and probability: prCRL

Specification language prCRL:

- Based on μ CRL (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 μ CRL (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:

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

Process equations and processes

A **process equation** is something of the form $X(g : G) = p$.

An example specification

Sending an arbitrary natural number

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

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

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

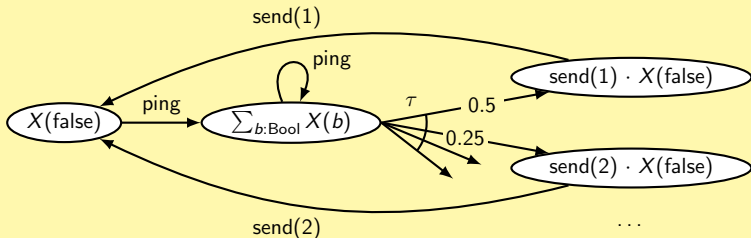
An example specification

Sending an arbitrary natural number

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

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

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



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**.

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

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

$$Z = (X(1) \parallel Y(2))$$

Composability using extended prCRL

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

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

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

$$Z = (X(1) \parallel Y(2))$$

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

Composability using extended prCRL

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

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

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

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

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

Composability using extended prCRL

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

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

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

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

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



Composability using extended prCRL

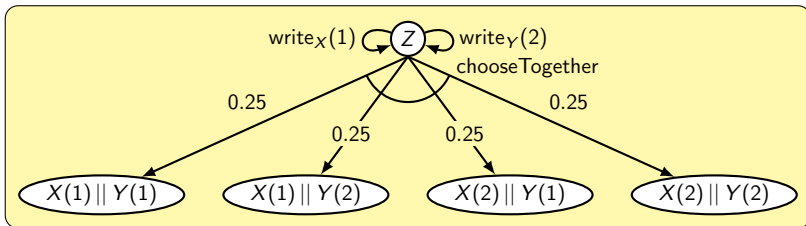
For composability we introduce **extended prCRL**. It extends prCRL by **parallel composition**, **encapsulation**, **hiding** and **renaming**.

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

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

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

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



A linear format for prCRL: the LPPE

LPPEs are a subset of prCRL specifications:

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

A linear format for prCRL: the LPPE

LPPEs are a subset of prCRL specifications:

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

Advantages of using LPPEs instead of prCRL specifications:

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

A linear format for prCRL: the LPPE

LPPEs are a subset of prCRL specifications:

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

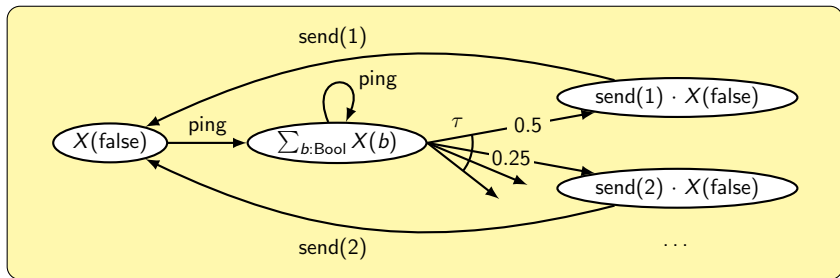
Advantages of using LPPEs instead of prCRL specifications:

- 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



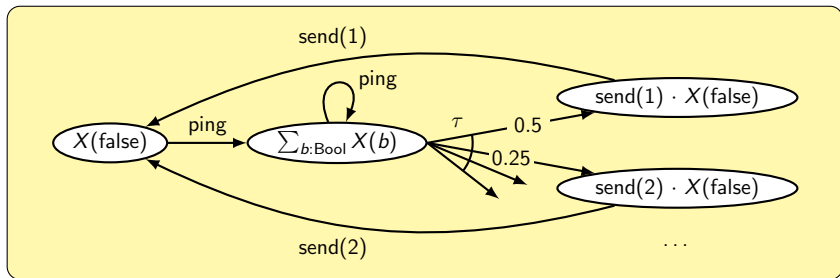
Specification in prCRL

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

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

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

Linear Probabilistic Process Equations – an example



Specification in prCRL

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

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

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

Specification in LPPE

$X(\text{pc} : \{1..3\}, n : \mathbb{N}^{\geq 0}) =$

$$+ \text{pc} = 1 \Rightarrow \text{ping} \cdot X(2, 1)$$

$$+ \text{pc} = 2 \Rightarrow \text{ping} \cdot X(2, 1)$$

$$+ \text{pc} = 2 \Rightarrow \tau \sum_{n:\mathbb{N}^{>0}} \frac{1}{2^n} : X(3, n)$$

$$+ \text{pc} = 3 \Rightarrow \text{send}(n) \cdot X(1, 1)$$

Linearisation: a simple example without data

Consider the following prCRL specification:

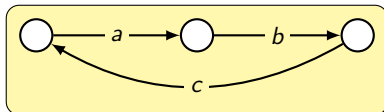
$$X = a \cdot b \cdot c \cdot X$$

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:

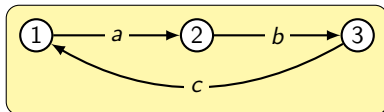


Linearisation: a simple example without data

Consider the following prCRL specification:

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

The control flow of X is given by:

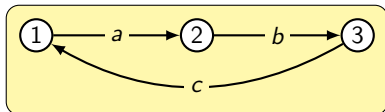


Linearisation: a simple example without data

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 $pc = 1$):

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

Linearisation: a more complicated example with data

Consider the following prCRL specification:

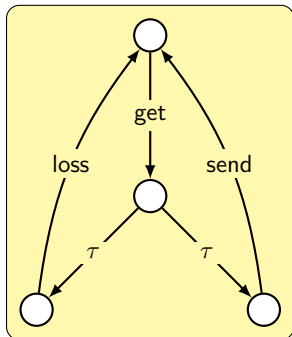
$$X = \sum_{d:D} \text{get}(d) \cdot (\tau \cdot \text{loss} \cdot X + \tau \cdot \text{send}(d) \cdot X)$$

Linearisation: a more complicated example with data

Consider the following prCRL specification:

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

Control flow:

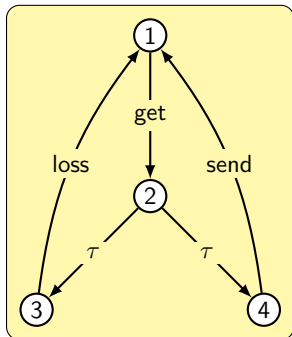


Linearisation: a more complicated example with data

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:

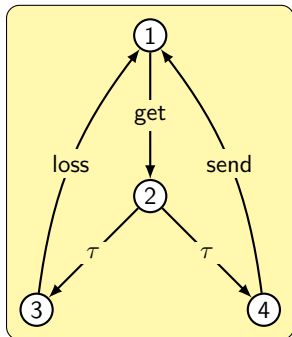


Linearisation: a more complicated example with data

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:

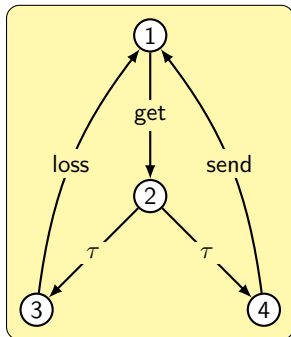
$$\begin{aligned}
 Y(pc: \{1, 2, 3, 4\}, x: D) = & \\
 & \sum_{d:D} pc = 1 \Rightarrow \text{get}(d) \cdot Y(2, d) \\
 + & pc = 2 \Rightarrow \tau \cdot Y(3, x) \\
 + & pc = 2 \Rightarrow \tau \cdot Y(4, x) \\
 + & pc = 3 \Rightarrow \text{loss} \cdot Y(1, x) \\
 + & pc = 4 \Rightarrow \text{send}(x)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

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:

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

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

Linearisation: a more algorithmic approach

Consider the following prCRL specification:

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

Linearisation: a more algorithmic approach

Consider the following prCRL specification:

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

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

Linearisation: a more algorithmic approach

Consider the following prCRL specification:

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

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

Linearisation: a more algorithmic approach

Consider the following prCRL specification:

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

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

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

Linearisation: a more algorithmic approach

Consider the following prCRL specification:

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

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

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

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

Linearisation: a more algorithmic approach

Consider the following prCRL specification:

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

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

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

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

Linearisation: a more algorithmic approach

Consider the following prCRL specification:

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

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

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

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

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

Linearisation: a more algorithmic approach

Consider the following prCRL specification:

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

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

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

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

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

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

Linearisation: a more algorithmic approach

Consider the following prCRL specification:

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

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

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

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

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

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

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

Linearisation: a more algorithmic approach

Consider the following prCRL specification:

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

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

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

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

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

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

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

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

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

Linearisation: a more algorithmic approach

Consider the following prCRL specification:

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

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

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

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

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

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

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

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

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

Linearisation: a more algorithmic approach

Consider the following prCRL specification:

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

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

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

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

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

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

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

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

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

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

Linearisation: a more algorithmic approach

Consider the following prCRL specification:

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

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

Linearisation: a more algorithmic approach

Consider the following prCRL specification:

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

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

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

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 $X(Mark)$

→

$$X = print(Mark) \cdot X$$

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

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

init X

→

$$X = \mathit{send}(2) \cdot X$$

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

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

→

$X = \text{beep} \cdot Y$

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

- 1 Introduction
- 2 A process algebra with data and probability: prCRL
- 3 Linearisation: from prCRL to LPPE
- 4 Reduction techniques
- 5 Modelling Markov Automata using MAPA**
- 6 Encoding and decoding
- 7 Reduction techniques revisited
- 8 Case study
- 9 Conclusions and Future Work

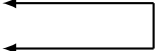
The overall goal: efficient and expressive modelling

Specifying systems with

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

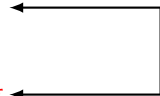
The overall goal: efficient and expressive modelling

Specifying systems with

- Nondeterminism
 - Probability
 - Stochastic timing
- 
- Probabilistic Automata (PAs)

The overall goal: efficient and expressive modelling


Specifying systems with

- Nondeterminism
 - Probability
 - Stochastic timing
- 
- ```
graph LR; A[Nondeterminism] --> B[Interactive Markov Chains (IMCs)]; C[Probability] --> B; D[Stochastic timing] --> B;
```

Interactive Markov Chains (IMCs)

# The overall goal: efficient and expressive modelling

## Specifying systems with

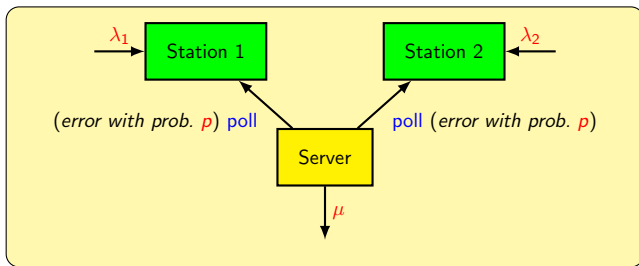
- Nondeterminism
  - Probability
  - Stochastic timing
- 
- Markov Automata (MAs)



# The overall goal: efficient and expressive modelling

## Specifying systems with

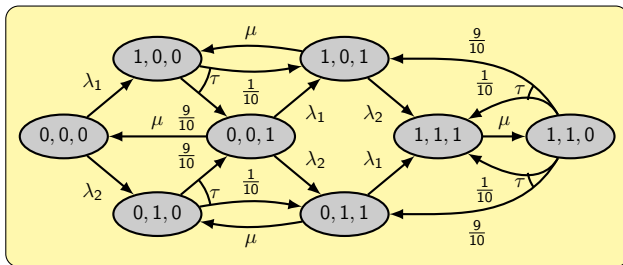
- Nondeterminism
  - Probability
  - Stochastic timing
- ← ← ← Markov Automata (MAs)



# The overall goal: efficient and expressive modelling

## Specifying systems with

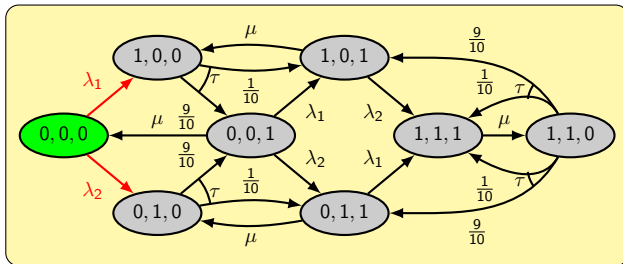
- Nondeterminism
  - Probability
  - Stochastic timing
- ← ← ← Markov Automata (MAs)



# The overall goal: efficient and expressive modelling

## Specifying systems with

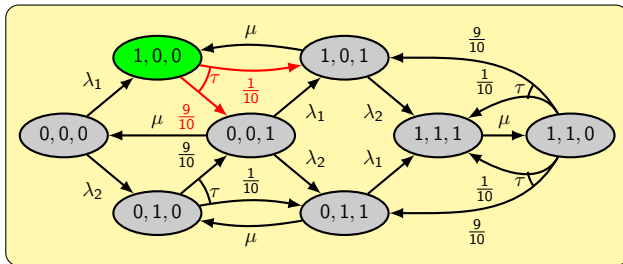
- Nondeterminism
  - Probability
  - Stochastic timing
- ← ← ← Markov Automata (MAs)



# The overall goal: efficient and expressive modelling

## Specifying systems with

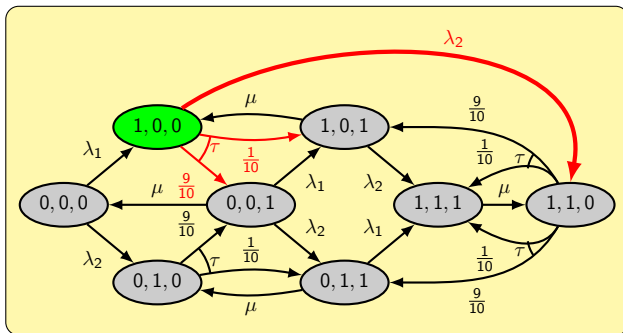
- Nondeterminism
  - Probability
  - Stochastic timing
- ←←← Markov Automata (MAs)



# The overall goal: efficient and expressive modelling

## Specifying systems with

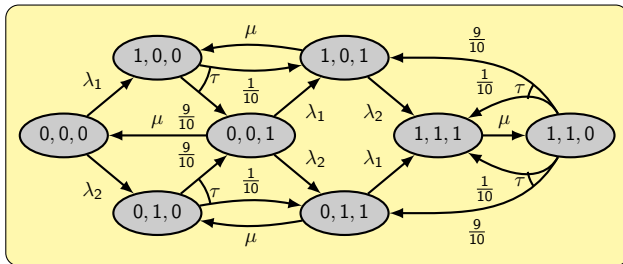
- Nondeterminism
  - Probability
  - Stochastic timing
- ← ← ← Markov Automata (MAs)



# The overall goal: efficient and expressive modelling

## Specifying systems with

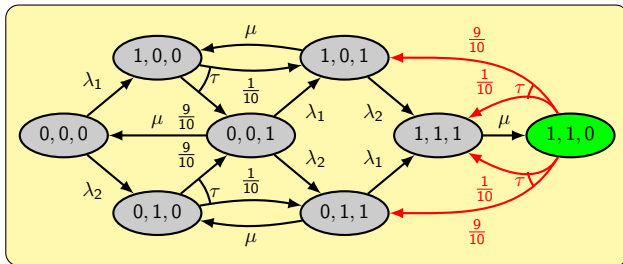
- Nondeterminism
  - Probability
  - Stochastic timing
- ← ← ← Markov Automata (MAs)



# The overall goal: efficient and expressive modelling

## Specifying systems with

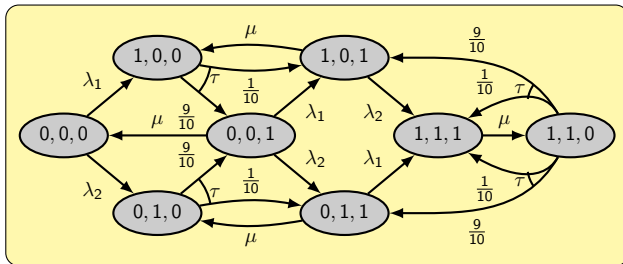
- Nondeterminism
  - Probability
  - Stochastic timing
- ← ← ← Markov Automata (MAs)



# 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 → MA

---

# Approach: extending and reusing

PA → MA

---

prCRL → MAPA (Markov Automata Process Algebra)

# Approach: extending and reusing

PA → MA

---

prCRL → MAPA (Markov Automata Process Algebra)

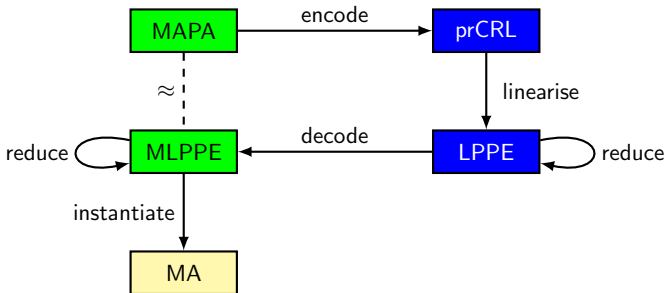
LPPE → MLPPE (Markovian LPPE)

# Approach: extending and reusing

PA → MA

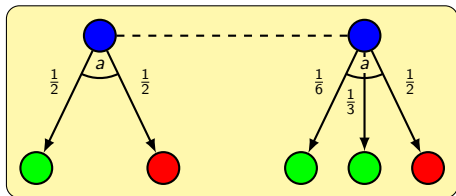
prCRL → MAPA (Markov Automata Process Algebra)

LPPE → MLPPE (Markovian LPPE)



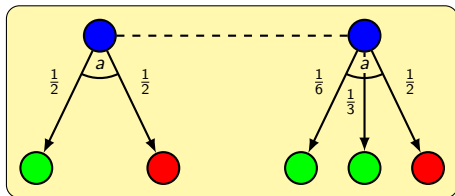
# Strong bisimulation for Markov automata

Mimic interactive behaviour:

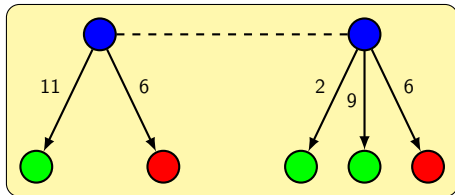


# Strong bisimulation for Markov automata

Mimic interactive behaviour:

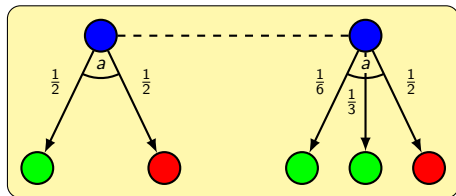


Mimic Markovian behaviour:

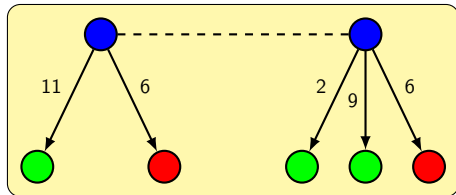


# Strong bisimulation for Markov automata

Mimic interactive behaviour:



Mimic Markovian behaviour:



(If a state enables a  $\tau$ -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:

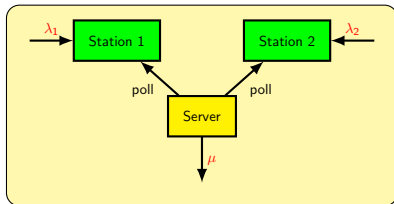
- Based on prCRL: **data** and **probabilistic choice**
- Additional feature: Markovian **rates**
- Semantics defined in terms of **Markov automata**
- Minimal set of operators to facilitate **formal manipulation**
- **Syntactic sugar** easily definable

## The grammar of MAPA

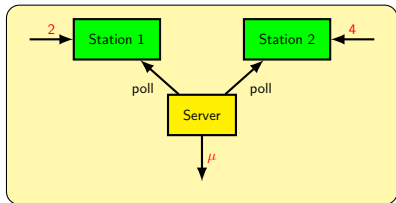
**Process terms** in MAPA are obtained by the following grammar:

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

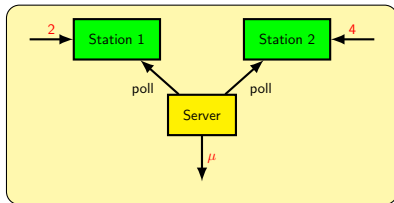
# An example specification



# An example specification

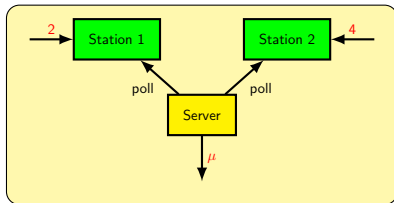


# An example specification



- 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**)

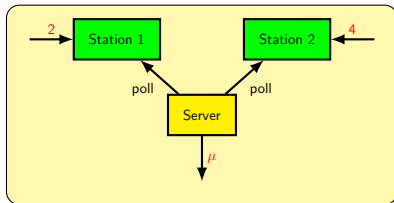
The specification of the stations:

```
type Jobs = {1, ..., 10}
```

```
Station(i : {1, 2}, q : Queue)
```

```
= notFull(q) ⇒ (2i) . ∑j:Jobs arrive(j).Station(i, 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:

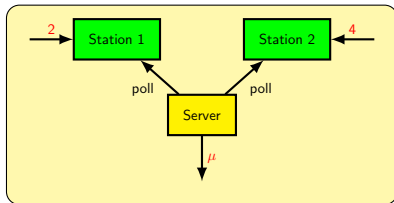
**type**  $Jobs = \{1, \dots, 10\}$

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

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

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

# 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**  $Jobs = \{1, \dots, 10\}$

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

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

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

## Derivation-based operational semantics

$$\text{MARKOVPREFIX} \frac{-}{(\lambda) \cdot p \xrightarrow{\lambda} p} \qquad \text{SUMLEFT} \frac{p \xrightarrow{a} p'}{p + q \xrightarrow{a} p'}$$



## Derivation-based operational semantics

$$\text{MARKOVPREFIX} \frac{-}{(\lambda) \cdot p \xrightarrow{\lambda} p} \quad \text{SUMLEFT} \frac{p \xrightarrow{a} p'}{p + q \xrightarrow{a} p'}$$

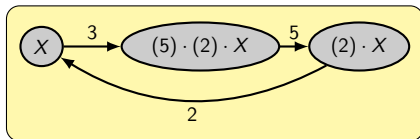
$$X = (3) \cdot (5) \cdot (2) \cdot X$$

## Derivation-based operational semantics

$$\text{MARKOVPREFIX} \frac{-}{(\lambda) \cdot p \xrightarrow{\lambda} p}$$

$$\text{SUMLEFT} \frac{p \xrightarrow{a} p'}{p + q \xrightarrow{a} p'}$$

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



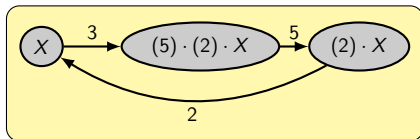
## Derivation-based operational semantics

$$\text{MARKOVPREFIX} \frac{-}{(\lambda) \cdot p \xrightarrow{\lambda} p}$$

$$\text{SUMLEFT} \frac{p \xrightarrow{a} p'}{p + q \xrightarrow{a} p'}$$

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

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



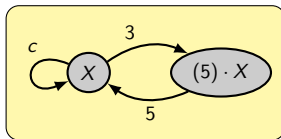
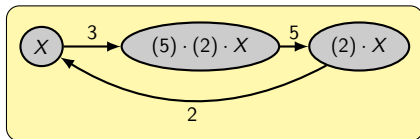
## Derivation-based operational semantics

$$\text{MARKOVPREFIX} \frac{-}{(\lambda) \cdot p \xrightarrow{\lambda} p}$$

$$\text{SUMLEFT} \frac{p \xrightarrow{a} p'}{p + q \xrightarrow{a} p'}$$

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

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



## Derivation-based operational semantics

$$\text{MARKOVPREFIX} \frac{-}{(\lambda) \cdot p \xrightarrow{\lambda} p}$$

$$\text{SUMLEFT} \frac{p \xrightarrow{a} p'}{p + q \xrightarrow{a} p'}$$

## Derivation-based operational semantics

$$\text{MARKOVPREFIX} \frac{-}{(\lambda) \cdot p \xrightarrow{\lambda} p} \quad \text{SUMLEFT} \frac{p \xrightarrow{a} p'}{p + q \xrightarrow{a} p'}$$

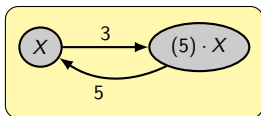
$$X = (3) \cdot (5) \cdot X + (3) \cdot (5) \cdot X$$

## Derivation-based operational semantics

$$\text{MARKOVPREFIX} \frac{-}{(\lambda) \cdot p \xrightarrow{\lambda} p} \quad \text{SUMLEFT} \frac{p \xrightarrow{a} p'}{p + q \xrightarrow{a} p'}$$

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

This is not right!



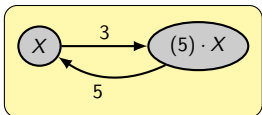
## Derivation-based operational semantics

$$\text{MARKOVPREFIX} \frac{-}{(\lambda) \cdot p \xrightarrow{\lambda} p} \quad \text{SUMLEFT} \frac{p \xrightarrow{a} p'}{p + q \xrightarrow{a} p'}$$

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

This is not right!

As a solution, we look at [derivations](#):





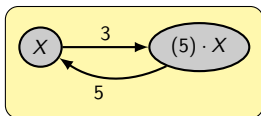
## Derivation-based operational semantics

$$\text{MARKOVPREFIX} \frac{-}{(\lambda) \cdot p \xrightarrow{\lambda} \text{MP } p} \quad \text{SUMLEFT} \frac{p \xrightarrow{a} \text{D } p'}{p + q \xrightarrow{a} \text{SL+D } p'}$$

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

This is not right!

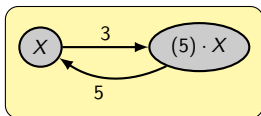
As a solution, we look at [derivations](#):



## Derivation-based operational semantics

$$\text{MARKOVPREFIX} \frac{-}{(\lambda) \cdot p \xrightarrow{\lambda} \text{MP } p} \quad \text{SUMLEFT} \frac{p \xrightarrow{a} \text{D } p'}{p + q \xrightarrow{a} \text{SL+D } p'}$$

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



This is not right!

As a solution, we look at **derivations**:

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

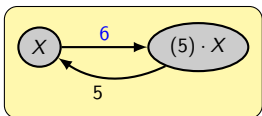
$$X \xrightarrow{3} \langle \text{SR}, \text{MP} \rangle (5) \cdot X$$

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

## Derivation-based operational semantics

$$\text{MARKOVPREFIX} \frac{-}{(\lambda) \cdot p \xrightarrow{\lambda} \text{MP } p} \quad \text{SUMLEFT} \frac{p \xrightarrow{a} \text{D } p'}{p + q \xrightarrow{a} \text{SL+D } p'}$$

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



This is not right!

As a solution, we look at **derivations**:

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

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

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

## MLPPEs

We defined a special format for MAPA, the **MLPPE**:

$$\begin{aligned}
 X(g : G) = & \sum_{i \in I} \sum_{d_i : D_i} c_i \Rightarrow a_i(\mathbf{b}_i) \sum_{e_i : E_i} f_i : X(\mathbf{n}_i) \\
 & + \sum_{j \in J} \sum_{d_j : D_j} c_j \Rightarrow (\lambda_j) \cdot X(\mathbf{n}_j)
 \end{aligned}$$

## MLPPEs

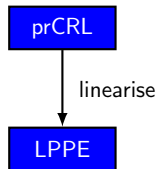
We defined a special format for MAPA, the **MLPPE**:

$$\begin{aligned}
 X(g : \mathbf{G}) &= \sum_{i \in I} \sum_{\mathbf{d}_i : \mathbf{D}_i} c_i \Rightarrow a_i(\mathbf{b}_i) \sum_{\mathbf{e}_i : \mathbf{E}_i} f_i : X(\mathbf{n}_i) \\
 &+ \sum_{j \in J} \sum_{\mathbf{d}_j : \mathbf{D}_j} c_j \Rightarrow (\lambda_j) \cdot X(\mathbf{n}_j)
 \end{aligned}$$

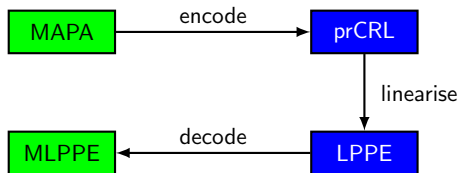
Advantages of using MLPPEs instead of MAPA specifications:

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

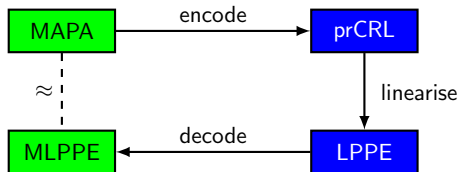
# Encoding into prCRL



# Encoding into prCRL

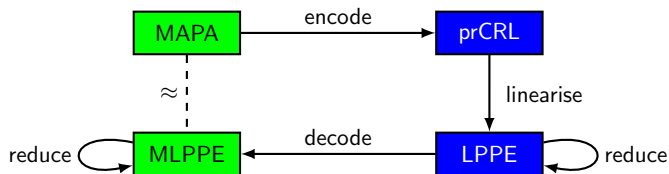


# Encoding into prCRL

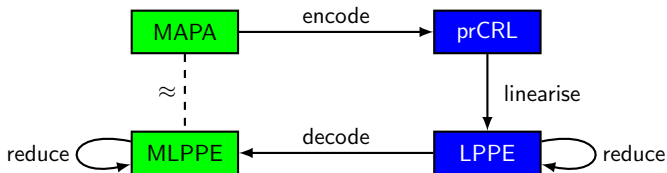




# Encoding into prCRL

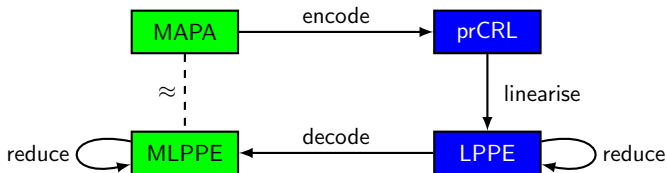


# Encoding into prCRL



Basic idea: encode a **rate**  $\lambda$  as **action rate**( $\lambda$ ).

# Encoding into prCRL

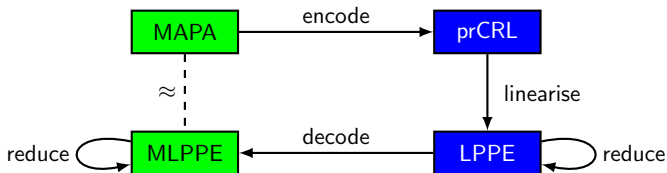


Basic idea: encode a **rate**  $\lambda$  as **action rate**( $\lambda$ ).

Problem:

Bisimulation-preserving reductions on prCRL might change MAPA behaviour

# Encoding into prCRL



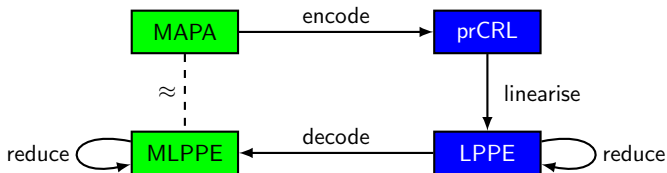
Basic idea: encode a **rate**  $\lambda$  as **action rate**( $\lambda$ ).

Problem:

Bisimulation-preserving reductions on prCRL might change MAPA behaviour

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

# Encoding into prCRL



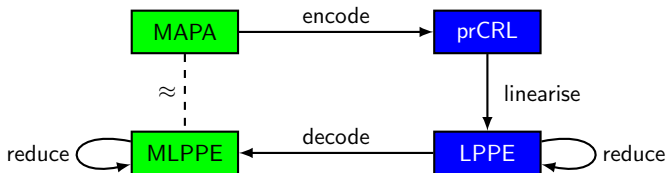
Basic idea: encode a **rate**  $\lambda$  as **action rate**( $\lambda$ ).

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



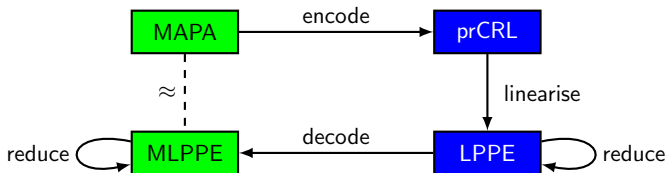
Basic idea: encode a **rate**  $\lambda$  as **action rate**( $\lambda$ ).

Problem:

Bisimulation-preserving reductions on prCRL might change MAPA behaviour

$$\begin{aligned}
 (\lambda) \cdot p + (\lambda) \cdot p &\Rightarrow \text{rate}(\lambda) \cdot p + \text{rate}(\lambda) \cdot p \\
 &\quad \approx_{\text{PA}} \\
 &\quad \text{rate}(\lambda) \cdot p
 \end{aligned}$$

# Encoding into prCRL



Basic idea: encode a **rate**  $\lambda$  as **action rate**( $\lambda$ ).

Problem:

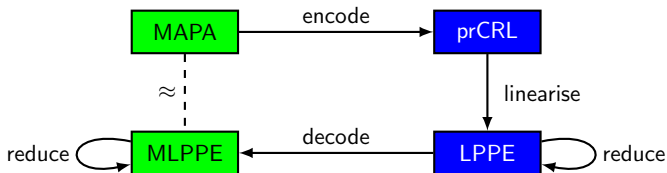
Bisimulation-preserving reductions on prCRL might change MAPA behaviour

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

$\approx_{\text{PA}}$

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

# Encoding into prCRL



Basic idea: encode a **rate**  $\lambda$  as **action rate**( $\lambda$ ).

Problem:

Bisimulation-preserving reductions on prCRL might change MAPA behaviour

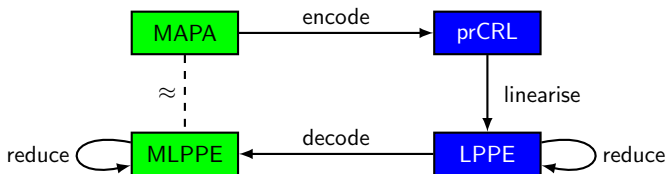
$$(\lambda) \cdot p + (\lambda) \cdot p \Rightarrow \text{rate}(\lambda) \cdot p + \text{rate}(\lambda) \cdot p$$

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

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

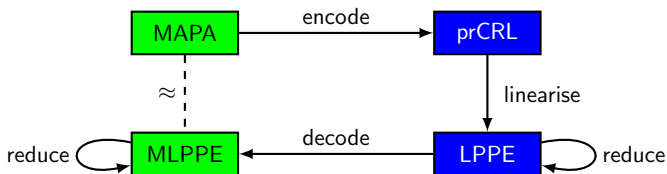


# Encoding into prCRL



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

# Encoding into prCRL

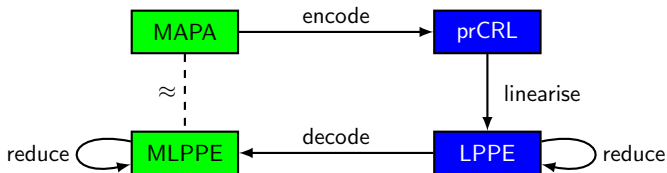


Possible solution: encode a **rate  $\lambda$**  as **action rate $_i(\lambda)$** .

Problem:

This still **doesn't work...**

# Encoding into prCRL



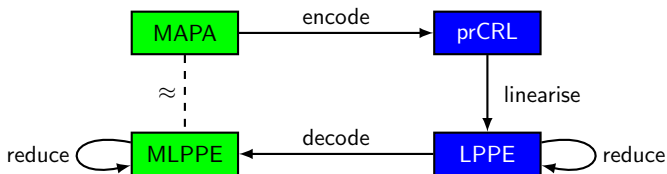
Possible solution: encode a **rate**  $\lambda$  as **action rate**  $j(\lambda)$ .

Problem:

This still **doesn't work**...

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

# Encoding into prCRL



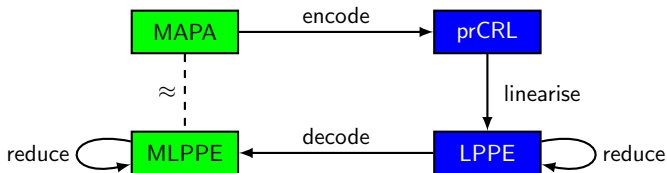
Possible solution: encode a **rate**  $\lambda$  as **action rate**  $i(\lambda)$ .

Problem:

This still **doesn't work**...

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

# Encoding into prCRL



Possible solution: encode a **rate**  $\lambda$  as **action rate**  $i(\lambda)$ .

Problem:

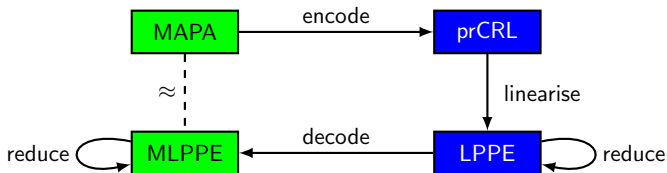
This still **doesn't work**...

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

$\approx_{\text{PA}}$

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

# Encoding into prCRL



Possible solution: encode a **rate**  $\lambda$  as **action rate**  $i(\lambda)$ .

Problem:

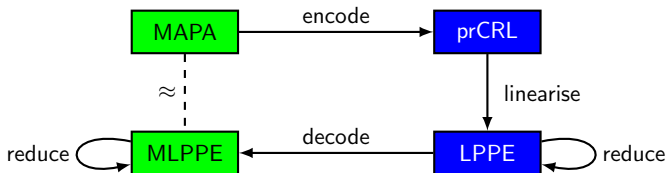
This still **doesn't work**...

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

$\approx_{\text{PA}}$

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

# Encoding into prCRL



Possible solution: encode a **rate**  $\lambda$  as **action rate**  $i(\lambda)$ .

Problem:

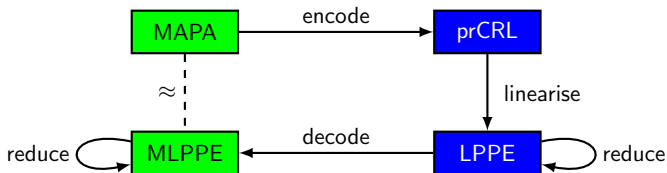
This still **doesn't work**...

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

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

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

# Encoding into prCRL



Possible solution: encode a **rate**  $\lambda$  as **action rate**  $i(\lambda)$ .

Problem:

This still **doesn't work**...

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

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

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

**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

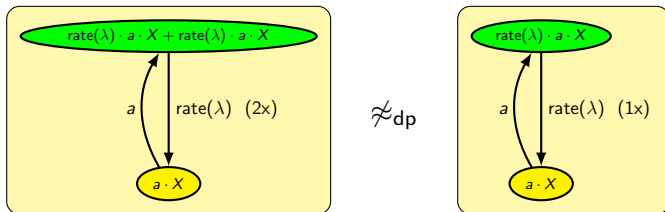
Two prCRL terms are **derivation-preserving bisimulation** if

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

# Derivation-preserving bisimulation

Two prCRL terms are **derivation-preserving bisimulation** if

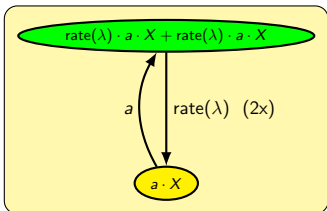
- There is a **strong bisimulation** relation  $R$  containing them
- Every bisimilar pair  $(p, p')$  has the **same number of  $\text{rate}(\lambda)$  derivations to every equivalence class  $[r]_R$** .



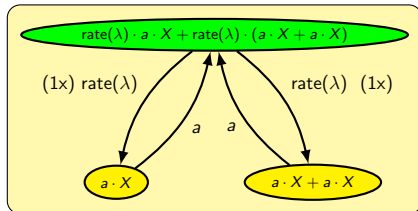
# Derivation-preserving bisimulation

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  $\text{rate}(\lambda)$  derivations to every equivalence class  $[r]_R$** .



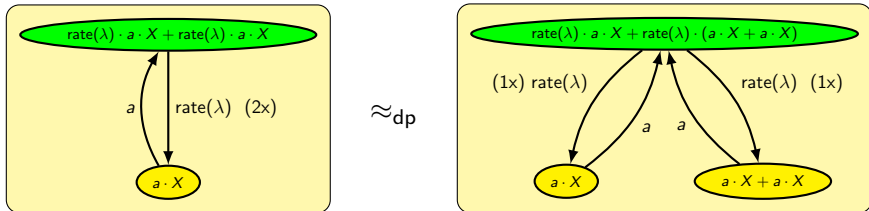
$\approx_{dp}$



# 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  $\text{rate}(\lambda)$  derivations to every equivalence class  $[r]_R$** .



## 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

```

1 abstract = ()
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```



# 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**

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

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

→

$$X = \text{send}(2) \cdot X$$

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

# Implementation and Case Study

## Implementation in SCOOP:

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

## Specification:

```
X = tau.X[] ++ <5>.X[]
```

```
init X
```

## Constants (name = value):



prCRL mode

Show LPPE ( use prCRL syntax)

Translate specification to PRISM

formula)





Specification:

```
X = tau.X[] ++ <5>.X[]
init X
```

Constants (name = value):



prCRL mode

- Show LPPE ( use prCRL syntax)
- Translate specification to PRISM formula)

- 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 reductio

- Show statespace in AUT format ( o
- Show statespace as the actual state
- Show the number of states and tran
- Show verbose output

- Apply dead variable reduction
- Apply transition merging
- Suppress all basic (M)LPPE reduction

Show Result

Visualize Statespace (from AUT) as image

Visualize St

(select model or experiment)

X =

(T => tau . X[])

Initial state: X

Powered by *puptol*

- Apply dead variable reduction
- Apply transition merging
- Suppress all basic (M)LPPE reduction

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

(select model or experiment)

X =

(T => tau . X[])

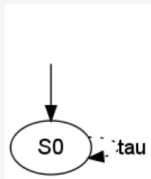
Initial state: X

Powered by *puptol*

- Apply unused variable reduction
- Apply dead variable reduction
- Apply transition merging
- Suppress all basic (M)LPPE reduction

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

(select model or experiment)



Powered by *puptol*

# Implementation and Case Study

## Implementation in SCOOP:

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

| Spec.       | Original  |           |            |         | Reduced |           |            |       | Red. |
|-------------|-----------|-----------|------------|---------|---------|-----------|------------|-------|------|
|             | States    | Trans.    | MLPPE      | Time    | States  | Trans.    | MLPPE      | Time  |      |
| 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      |               |                 |            | Reduced       |               |                |            | Red.       |
|-------------|---------------|---------------|-----------------|------------|---------------|---------------|----------------|------------|------------|
|             | States        | Trans.        | MLPPE           | Time       | States        | Trans.        | MLPPE          | Time       |            |
| 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 | <b>50,805</b> | <b>81,006</b> | <b>15 / 335</b> | <b>8.9</b> | <b>50,805</b> | <b>81,006</b> | <b>8 / 224</b> | <b>6.6</b> | <b>26%</b> |
| 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:

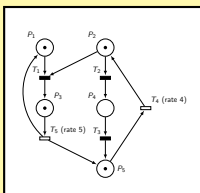
- Programmed in Haskell
- Stand-alone and web-based interface
- Linearisation, optimisation, state space generation

| Spec.       | Original  |           |            |         | Reduced |           |            |       | Red. |
|-------------|-----------|-----------|------------|---------|---------|-----------|------------|-------|------|
|             | States    | Trans.    | MLPPE      | Time    | States  | Trans.    | MLPPE      | Time  |      |
| 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

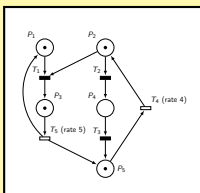
GSPN  
(PNML)



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) =
```

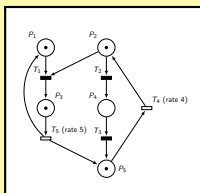
```
P2 >= 1 => T2 .
 GSPN[P2--, P4++]
+ P5 >= 1 => (4.0) .
 GSPN[P2++, P5--]
+ ...
```

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

reach P1 = 1 & P5 = 2



## GSPN analysis

GSPN  
(PNML)

reach P1 = 1 &amp; P5 = 2

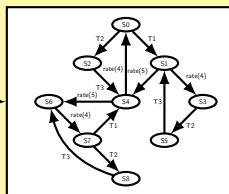
GEMMA

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 &amp; P5 = 2

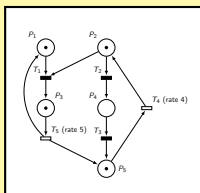
SCOOP



#GOALS S4

MA

## GSPN analysis

GSPN  
(PNML)

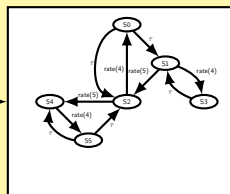
reach P1 = 1 &amp; P5 = 2

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)
```

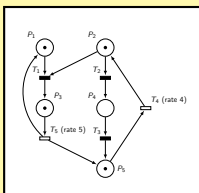
reach P1 = 1 &amp; P5 = 2

SCOOP  
(optimised)

#GOALS S2

MA

## GSPN analysis

GSPN  
(PNML)

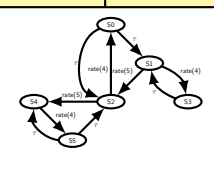
reach P1 = 1 &amp; P5 = 2

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 &amp; P5 = 2

SCOOP  
(optimised)

#GOALS S2

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

Results

MA

# 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`