

UNIVERSITY OF TWENTE.

Formal Methods & Tools.

Efficient Modelling and Generation of Markov Automata

Mark Timmer

October 22, 2012

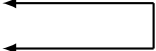
The overall goal: efficient and expressive modelling

Specifying systems with

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

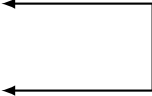
The overall goal: efficient and expressive modelling

Specifying systems with

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

The overall goal: efficient and expressive modelling


Specifying systems with

- Nondeterminism
 - Probability
 - Stochastic timing
- 

Interactive Markov Chains (IMCs)

The overall goal: efficient and expressive modelling

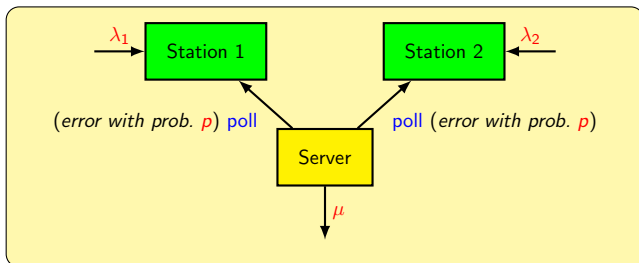
Specifying systems with

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

The overall goal: efficient and expressive modelling

Specifying systems with

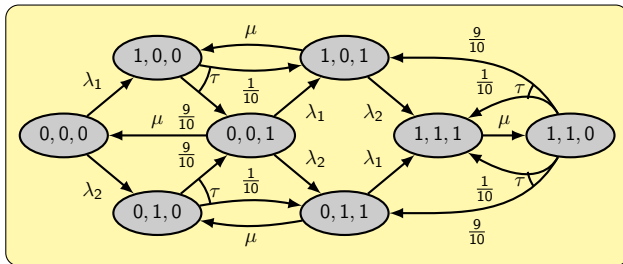
- Nondeterminism
 - Probability
 - Stochastic timing
- Markov Automata (MAs)



The overall goal: efficient and expressive modelling

Specifying systems with

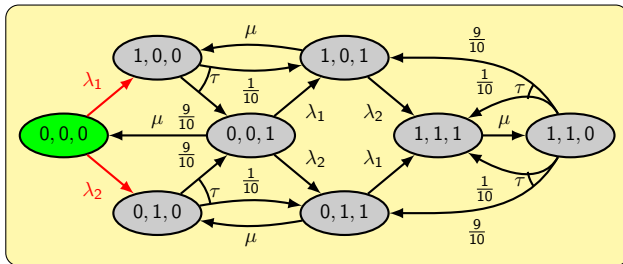
- Nondeterminism
 - Probability
 - Stochastic timing
- Markov Automata (MAs)



The overall goal: efficient and expressive modelling

Specifying systems with

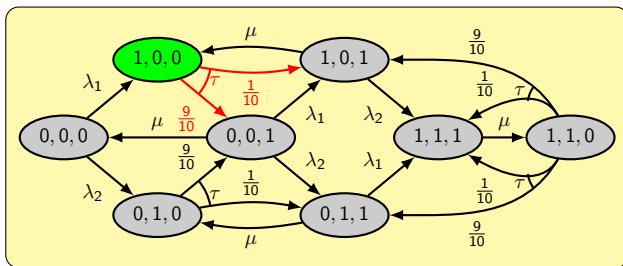
- Nondeterminism
 - Probability
 - Stochastic timing
- ←←← Markov Automata (MAs)



The overall goal: efficient and expressive modelling

Specifying systems with

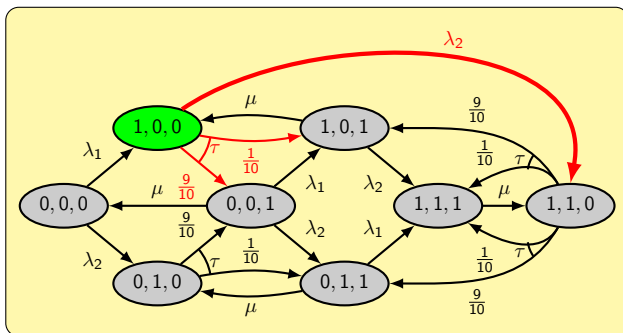
- Nondeterminism
 - Probability
 - Stochastic timing
- Markov Automata (MAs)



The overall goal: efficient and expressive modelling

Specifying systems with

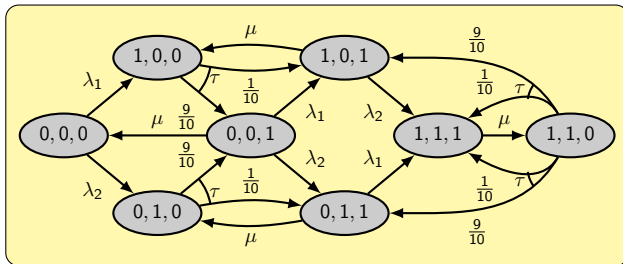
- Nondeterminism
 - Probability
 - Stochastic timing
- ←←←← Markov Automata (MAs)



The overall goal: efficient and expressive modelling

Specifying systems with

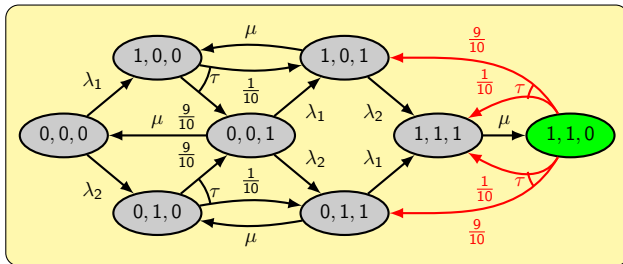
- Nondeterminism
 - Probability
 - Stochastic timing
- Markov Automata (MAs)



The overall goal: efficient and expressive modelling

Specifying systems with

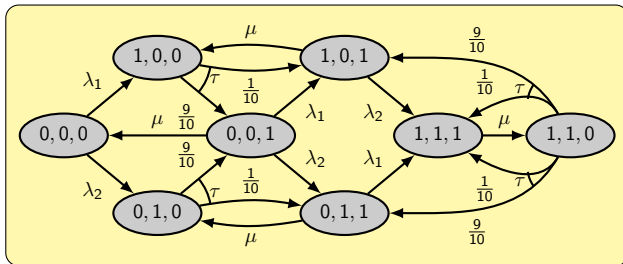
- Nondeterminism
 - Probability
 - Stochastic timing
- ←←← Markov Automata (MAs)



The overall goal: efficient and expressive modelling

Specifying systems with

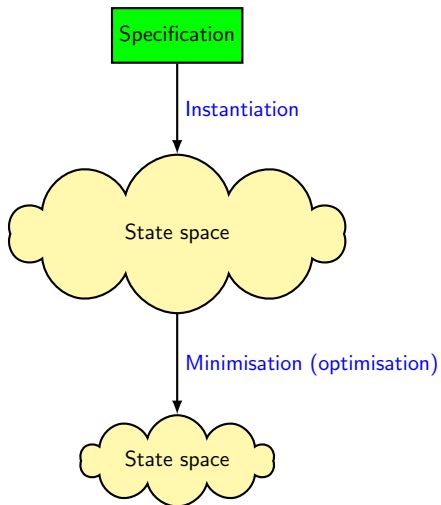
- Nondeterminism
 - Probability
 - Stochastic timing
- ← ← ← Markov Automata (MAs)



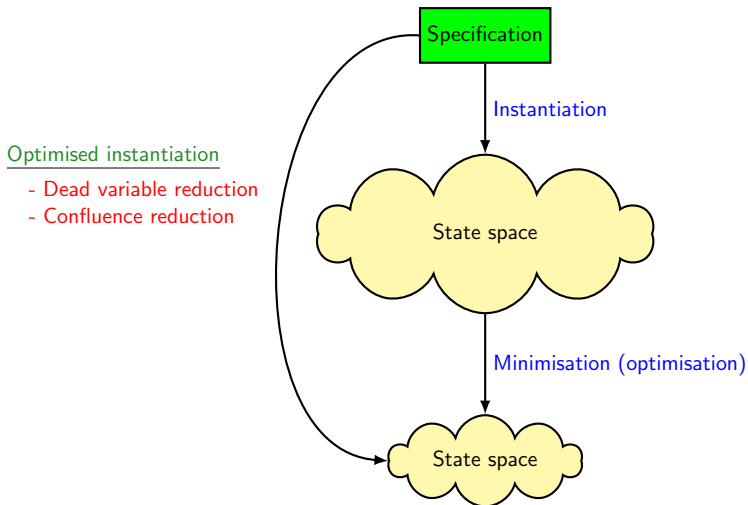
Observed limitations:

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

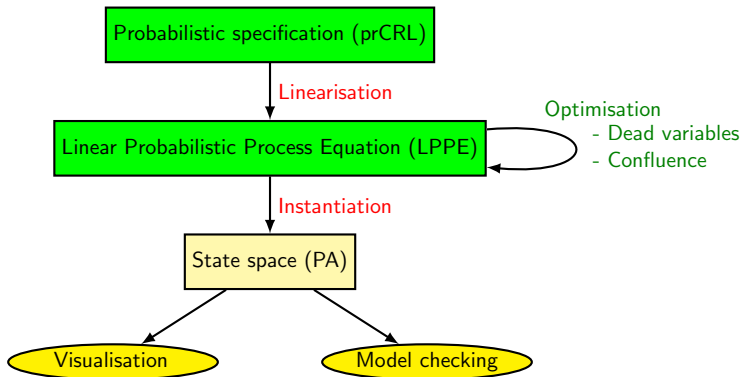
Combating the state space explosion



Combating the state space explosion



Earlier approach in the PA context



Current approach: extending and reusing

PA → MA

Current approach: extending and reusing

PA → MA

prCRL → MAPA (Markov Automata Process Algebra)

Current approach: extending and reusing

PA → MA

prCRL → MAPA (Markov Automata Process Algebra)

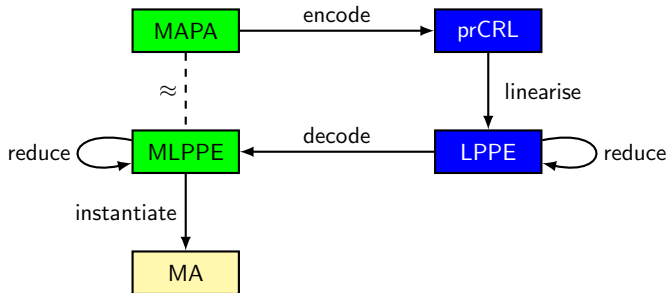
LPPE → MLPPE (Markovian LPPE)

Current approach: extending and reusing

PA → MA

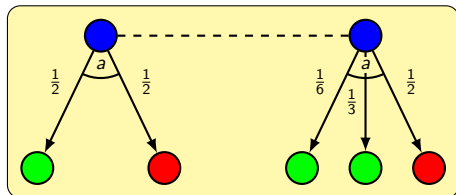
prCRL → MAPA (Markov Automata Process Algebra)

LPPE → MLPPE (Markovian LPPE)



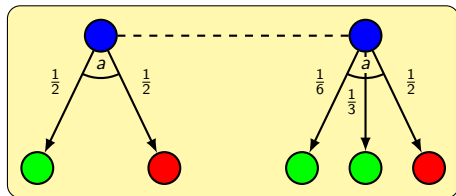
Strong bisimulation for Markov automata

Mimic interactive behaviour:

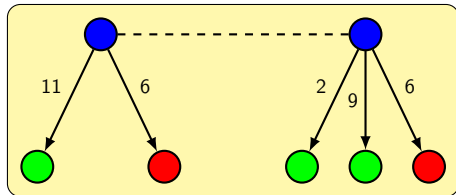


Strong bisimulation for Markov automata

Mimic interactive behaviour:

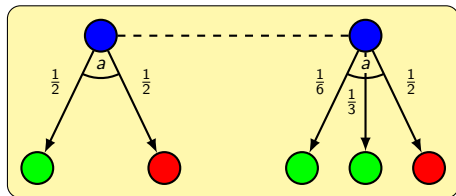


Mimic Markovian behaviour:

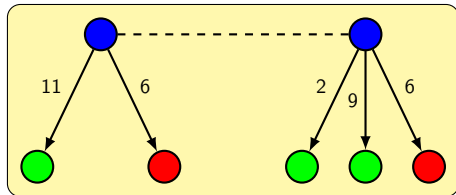


Strong bisimulation for Markov automata

Mimic interactive behaviour:



Mimic Markovian behaviour:



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

Contents

- 1 Introduction
- 2 A process algebra with data for MAs: MAPA
- 3 Encoding and decoding
- 4 Reductions
- 5 Case study
- 6 Conclusions and Future Work

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:

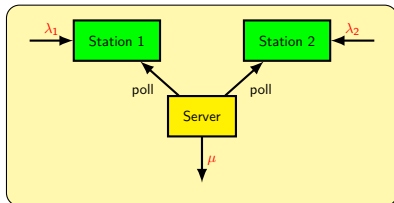
- Based on prCRL: **data** and **probabilistic choice**
- Additional feature: Markovian **rates**
- Semantics defined in terms of **Markov automata**
- Minimal set of operators to facilitate **formal manipulation**
- **Syntactic sugar** easily definable

The grammar of MAPA

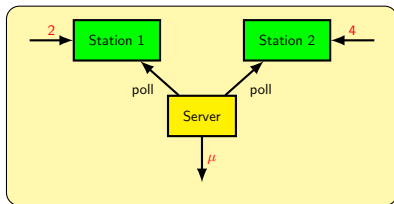
Process terms in MAPA are obtained by the following grammar:

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

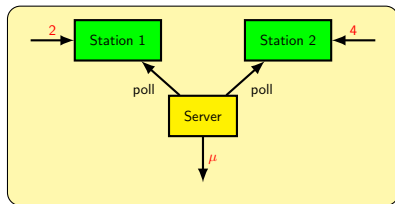
An example specification



An example specification

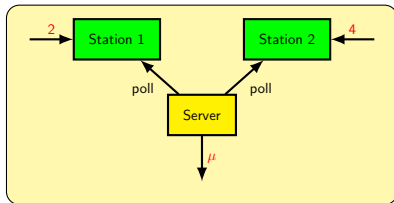


An example specification



- 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**)

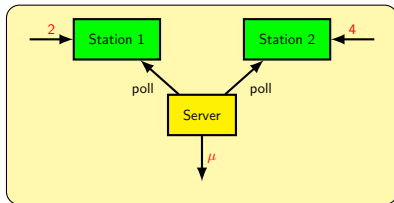
The specification of the stations:

```
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:

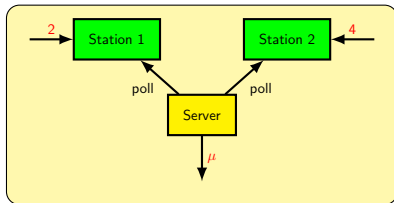
type $Jobs = \{1, \dots, 10\}$

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

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

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

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

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

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

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

MLPPEs

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

$$\begin{aligned}
 X(g : G) &= \sum_{d_1 : D_1} c_1 && \Rightarrow a_1(\mathbf{b}_1) \sum_{e_1 : E_1} f_1 : X(\mathbf{n}_1) \\
 &+ \dots \\
 &+ \sum_{d_m : D_m} c_m && \Rightarrow a_m(\mathbf{b}_m) \sum_{e_m : E_m} f_m : X(\mathbf{n}_m)
 \end{aligned}$$

MLPPEs

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

$$\begin{aligned}
 X(g : G) &= \sum_{d_1 : D_1} c_1 && \Rightarrow a_1(\mathbf{b}_1) \sum_{e_1 : E_1} f_1 : X(\mathbf{n}_1) \\
 &+ \dots \\
 &+ \sum_{d_m : D_m} c_m && \Rightarrow a_m(\mathbf{b}_m) \sum_{e_m : E_m} f_m : X(\mathbf{n}_m) \\
 &+ \sum_{d_{m+1} : D_{m+1}} c_{m+1} && \Rightarrow (\lambda_{m+1}) \cdot X(\mathbf{n}_{m+1}) \\
 &+ \dots \\
 &+ \sum_{d_n : D_n} c_n && \Rightarrow (\lambda_n) \cdot X(\mathbf{n}_n)
 \end{aligned}$$

MLPPEs

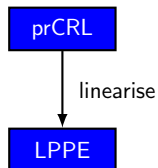
We defined a special format for MAPA, the **MLPPE**:

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

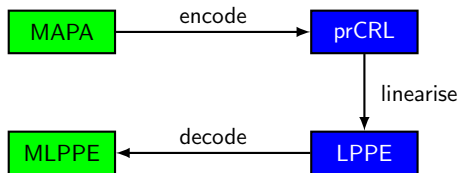
Advantages of using MLPPEs instead of MAPA specifications:

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

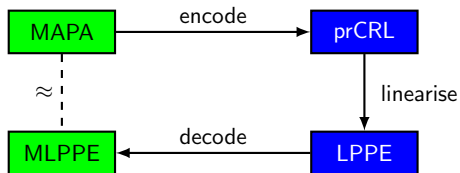
Encoding into prCRL



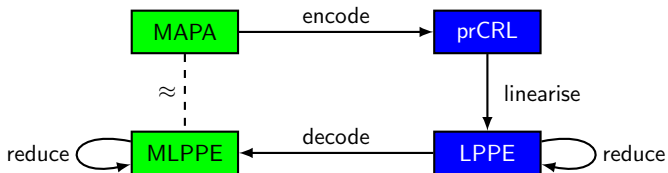
Encoding into prCRL



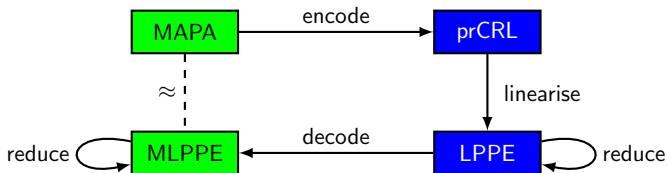
Encoding into prCRL



Encoding into prCRL

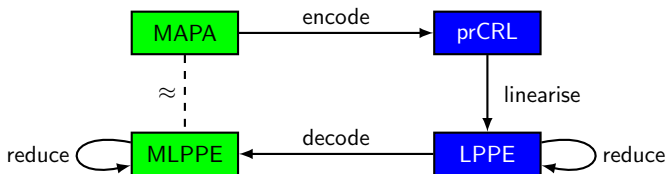


Encoding into prCRL



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

Encoding into prCRL

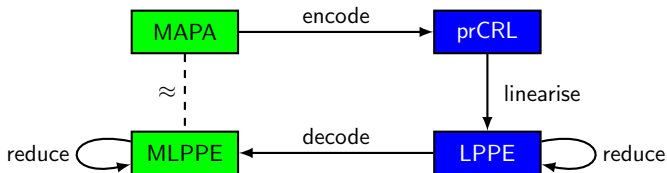


Basic idea: encode a **rate λ** as **action rate(λ)**.

Problem:

Bisimulation-preserving reductions on prCRL might **change MAPA behaviour**

Encoding into prCRL



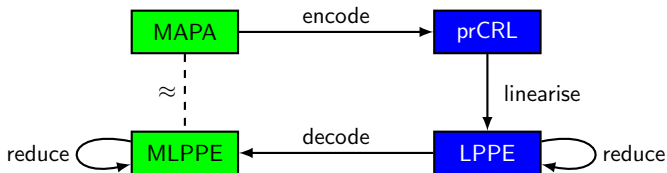
Basic idea: encode a **rate** λ as **action rate**(λ).

Problem:

Bisimulation-preserving reductions on prCRL might change MAPA behaviour

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

Encoding into prCRL



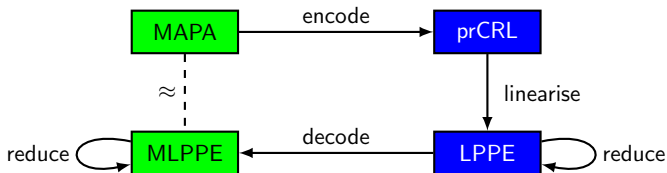
Basic idea: encode a **rate** λ as **action rate**(λ).

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



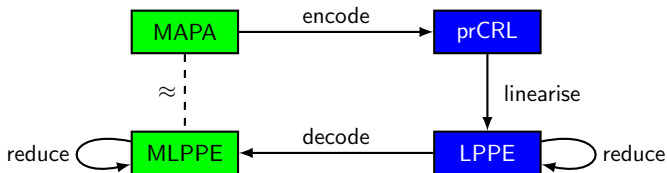
Basic idea: encode a **rate** λ as **action rate**(λ).

Problem:

Bisimulation-preserving reductions on prCRL might change MAPA behaviour

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

Encoding into prCRL



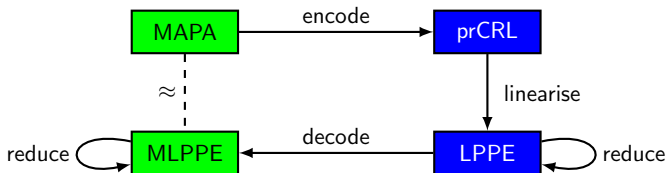
Basic idea: encode a **rate** λ as **action rate**(λ).

Problem:

Bisimulation-preserving reductions on prCRL might **change** MAPA behaviour

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

Encoding into prCRL



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

Problem:

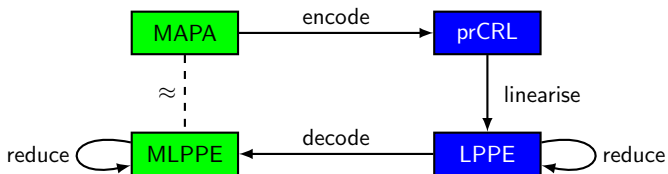
Bisimulation-preserving reductions on prCRL might change MAPA behaviour

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

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

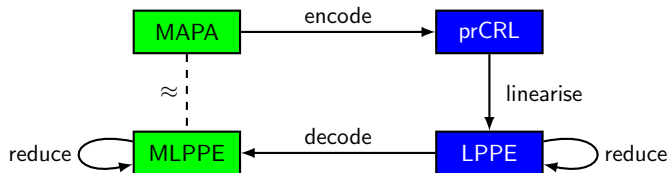
$$(\lambda) \cdot p \Leftarrow \text{rate}(\lambda) \cdot p$$

Encoding into prCRL



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

Encoding into prCRL

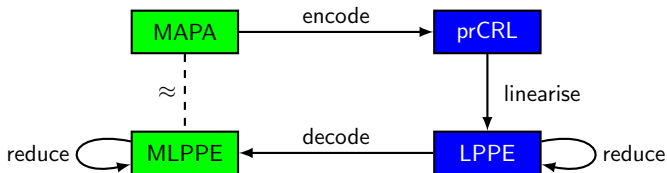


Possible solution: encode a **rate λ** as **action rate $_i(\lambda)$** .

Problem:

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

Encoding into prCRL



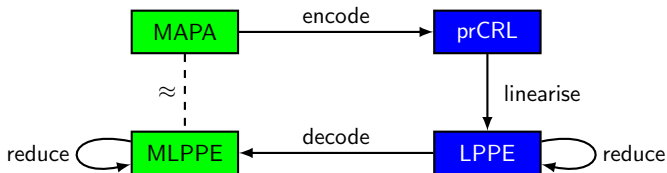
Possible solution: encode a **rate λ** as **action rate $_i(\lambda)$** .

Problem:

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

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

Encoding into prCRL



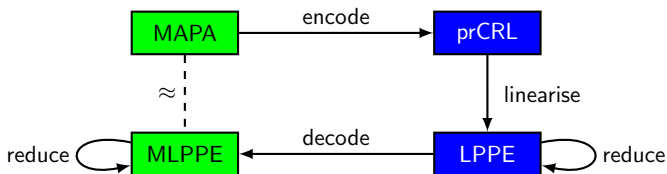
Possible solution: encode a **rate** λ as **action rate** $i(\lambda)$.

Problem:

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

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

Encoding into prCRL



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

Problem:

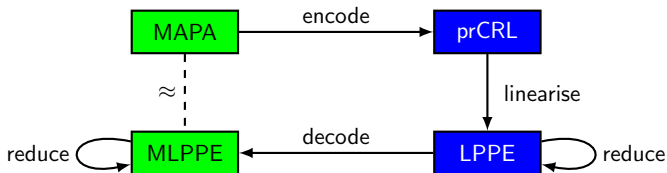
This still **doesn't work**...

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

\approx_{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 rate** $i(\lambda)$.

Problem:

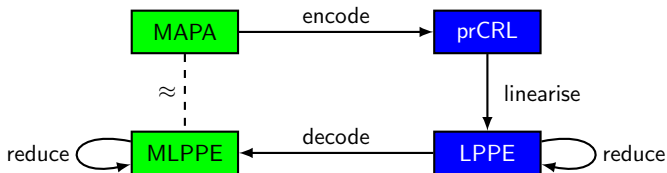
This still **doesn't work**...

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

\approx_{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:

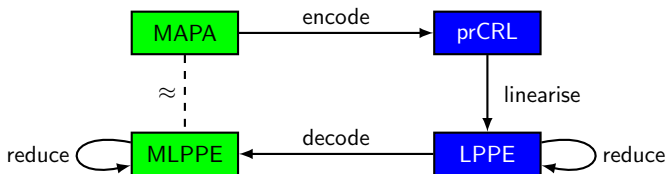
This still **doesn't work**...

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

$\not\approx_{MA}$ \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** λ as **action rate** $i(\lambda)$.

Problem:

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

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

$\not\approx_{MA}$ \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

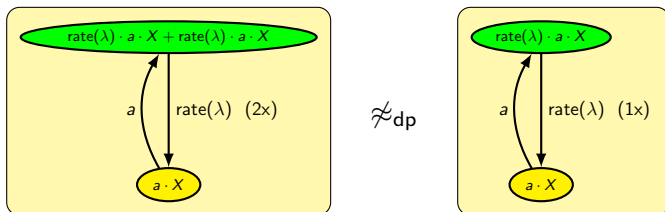
Two prCRL terms are **derivation-preserving bisimulation** if

- 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

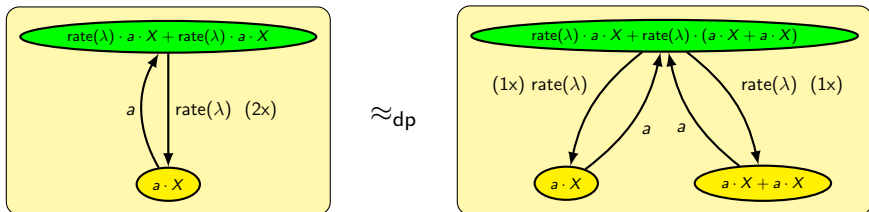
- There is a **strong bisimulation** relation R containing them
- Every bisimilar pair (p, p') has the **same number of $\text{rate}(\lambda)$ derivations to every equivalence class $[r]_R$** .



Derivation-preserving bisimulation

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

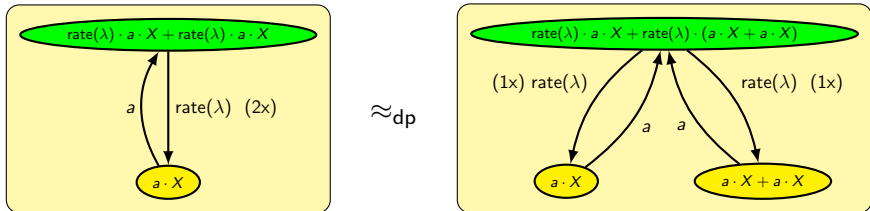
- There is a **strong bisimulation** relation R containing them
- Every bisimilar pair (p, p') has the **same number of $\text{rate}(\lambda)$ derivations to every equivalence class $[r]_R$** .



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

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

```
init X(Mark)
```

→

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

```
init X
```


Generalising existing reduction techniques

Existing reduction techniques that preserve derivations:

- Constant elimination
- Expression simplification
- Dead variable reduction

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

Generalising existing reduction techniques

Existing reduction techniques that preserve derivations:

- Constant elimination
- Expression simplification
- Dead variable reduction

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

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

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[])

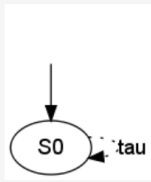
Initial state: X

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

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

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:

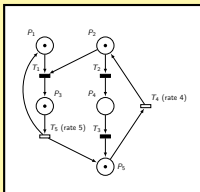
- Programmed in Haskell
- Linearisation, optimisation, state space generation
- Stand-alone and web-based interface

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

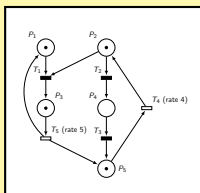
GSPN
(PNML)



reach $P1 = 1$ & $P5 = 2$

GSPN analysis

GSPN
(PNML)



reach P1 = 1 & P5 = 2

GEMMA

MAPA

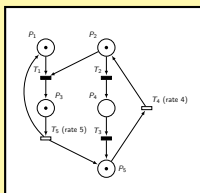
```
GSPN(P1:N,P2:N,P3:N,
P4:N,P5:N) =
P2 >= 1 => T2 .
  GSPN[P2--, P4++]
+ P5 >= 1 => (4.0) .
  GSPN[P2++, P5--]
+ ...

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

reach P1 = 1 & P5 = 2

GSPN analysis

GSPN
(PNML)



reach P1 = 1 & P5 = 2

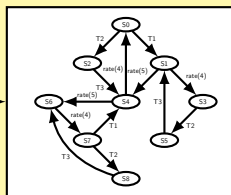
GEMMA

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

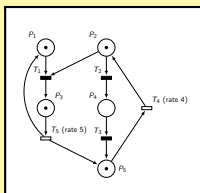


#GOALS S4

MA

GSPN analysis

GSPN
(PNML)



reach P1 = 1 & P5 = 2

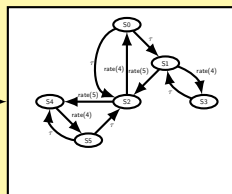
GEMMA

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)

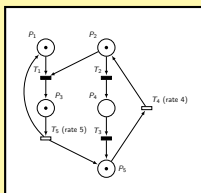


#GOALS S2

MA

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)**

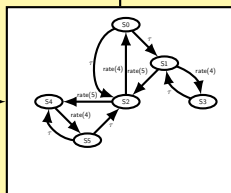
Min. reach. [1, 1.5]: 0.007
Max. reach. [1, 1.5]: 0.930

Min. expected time: 0.0
Max. expected time: 0.2

Min. LRA: 0.0
Max. LRA: 0.4

#GOALS S2

IMCA



#GOALS S2

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**

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**

Future Work:

- Generalise **confluence reduction** to MAs and MAPA

Questions

Questions?

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