UNIVERSITY OF TWENTE.

Formal Methods & Tools.

Efficient Modelling and Generation of Markov Automata

Mark Timmer

April 26, 2012

ROCKS meeting 2012

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

The overall goal: efficient and expressive modelling

Specifying systems with

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

The overall goal: efficient and expressive modelling

Specifying systems with

- Nondeterminism
- Probability
- Stochastic timing

Nondeterminism, Probability ← 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

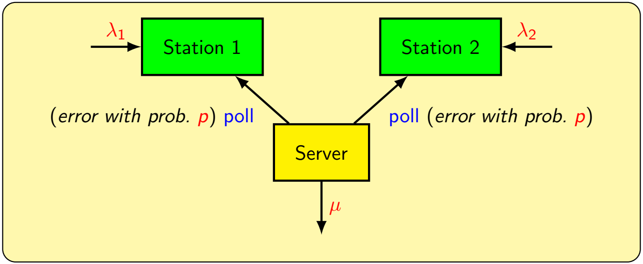
← Markov Automata (MAs)

The overall goal: efficient and expressive modelling

Specifying systems with

- Nondeterminism
- Probability
- Stochastic timing

→ Markov Automata (MAs)

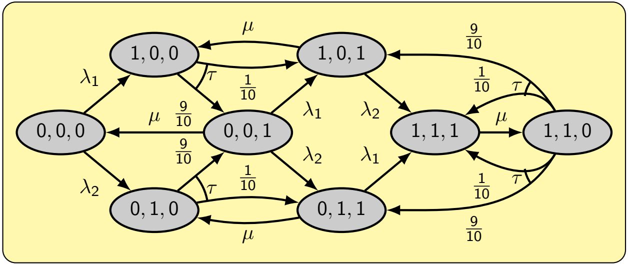


The overall goal: efficient and expressive modelling

Specifying systems with

- Nondeterminism
- Probability
- Stochastic timing

← Markov Automata (MAs)

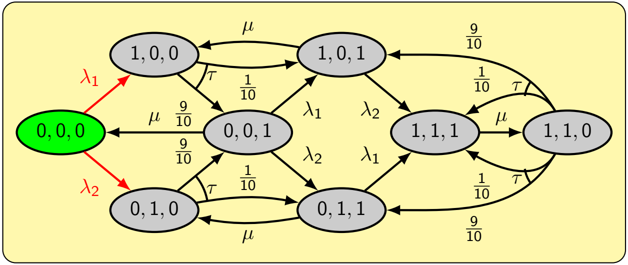


The overall goal: efficient and expressive modelling

Specifying systems with

- Nondeterminism
- Probability
- Stochastic timing

← Markov Automata (MAs)

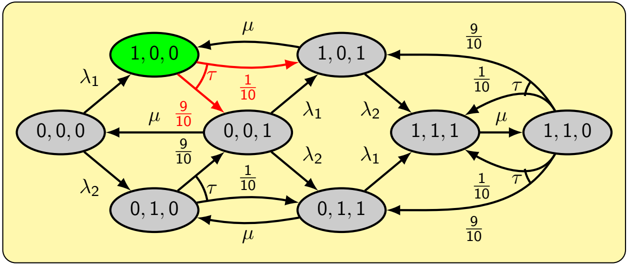


The overall goal: efficient and expressive modelling

Specifying systems with

- Nondeterminism
- Probability
- Stochastic timing

Markov Automata (MAs)

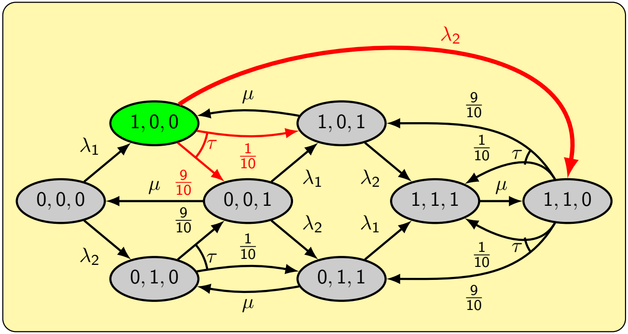


The overall goal: efficient and expressive modelling

Specifying systems with

- Nondeterminism
- Probability
- Stochastic timing

← Markov Automata (MAs)

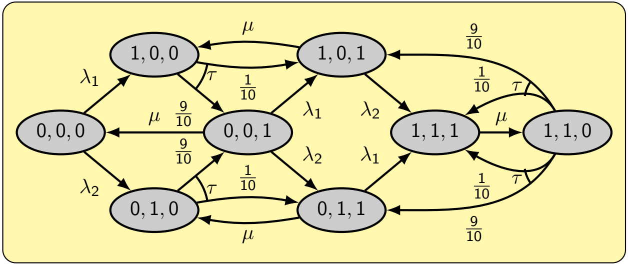


The overall goal: efficient and expressive modelling

Specifying systems with

- Nondeterminism
- Probability
- Stochastic timing

Markov Automata (MAs)

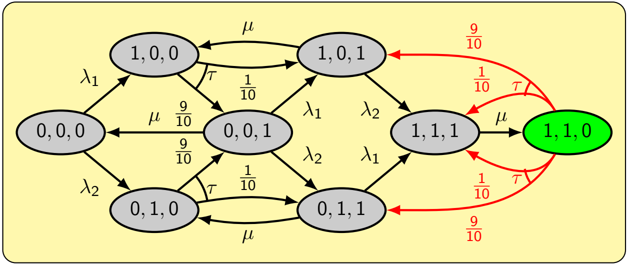


The overall goal: efficient and expressive modelling

Specifying systems with

- Nondeterminism
- Probability
- Stochastic timing

← Markov Automata (MAs)

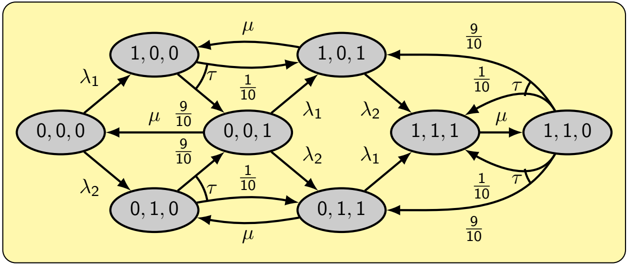


The overall goal: efficient and expressive modelling

Specifying systems with

- Nondeterminism
- Probability
- Stochastic timing

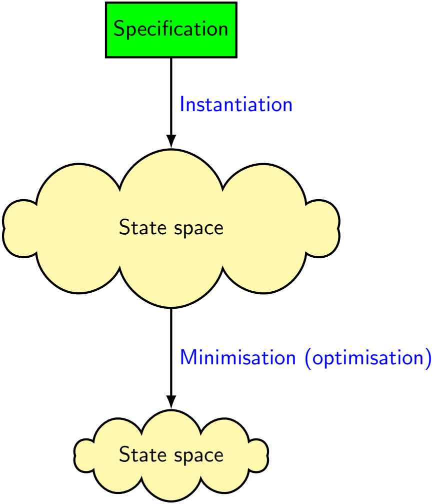
← Markov Automata (MAs)



Observed limitations:

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

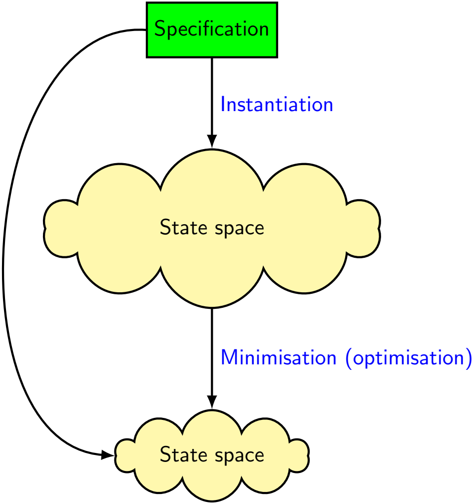
Combating the state space explosion



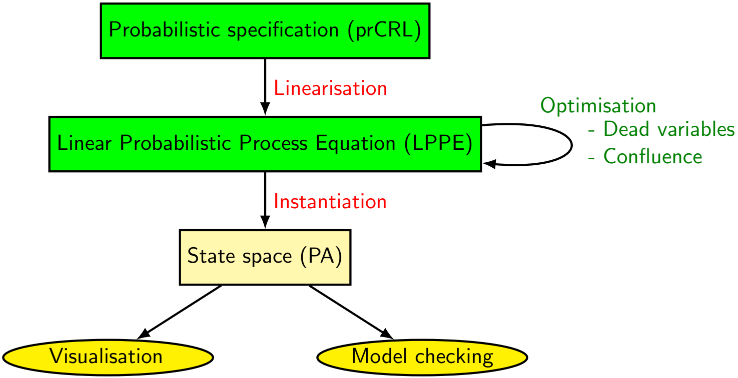
Combating the state space explosion

Optimised instantiation

- Dead variable reduction
- Confluence reduction



Earlier approach in the PA context



Current approach: extending and reusing

PA	$\rightarrow$	MA

Current approach: extending and reusing

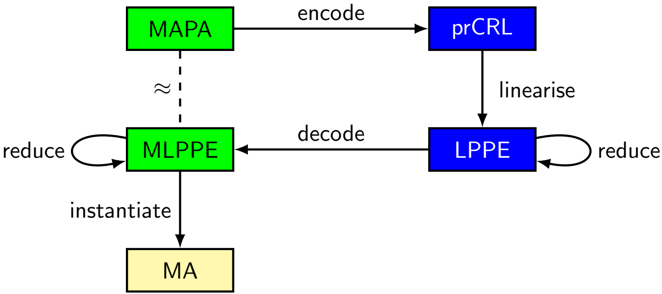
PA	$\rightarrow$	MA	
prCRL	$\rightarrow$	MAPA	(Markov Automata Process Algebra)

Current approach: extending and reusing

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

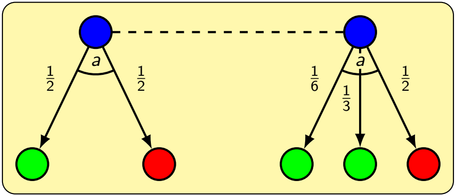
Current approach: extending and reusing

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



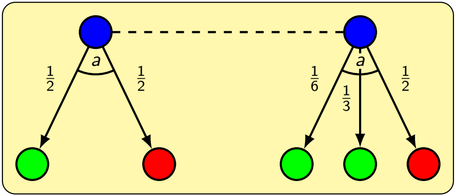
Strong bisimulation for Markov automata

Mimic interactive behaviour:

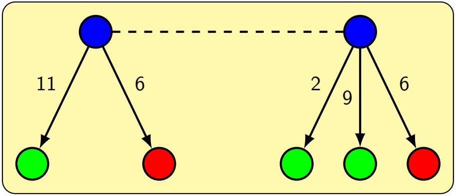


Strong bisimulation for Markov automata

Mimic interactive behaviour:

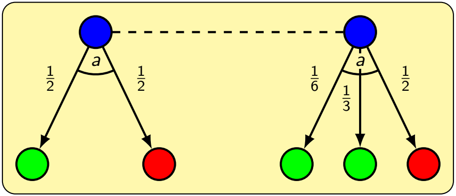


Mimic Markovian behaviour:

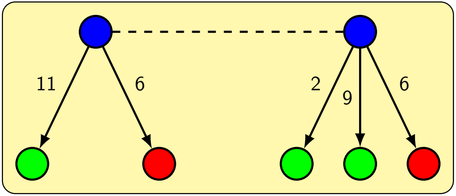


Strong bisimulation for Markov automata

Mimic interactive behaviour:



Mimic Markovian behaviour:



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

Contents

A process algebra with data for MAs: MAPA

Specification language MAPA:

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

A process algebra with data for MAs: MAPA

Specification language MAPA:

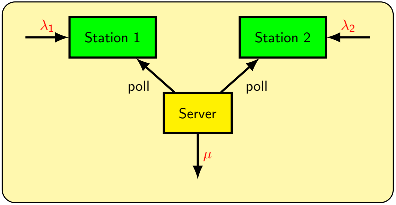
- Based on prCRL: data and probabilistic choice
- Additional feature: Markovian rates
- Semantics defined in terms of Markov automata
- Minimal set of operators to facilitate formal manipulation
- Syntactic sugar easily definable

The grammar of MAPA

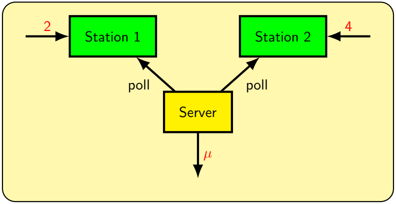
Process terms in MAPA are obtained by the following grammar:

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

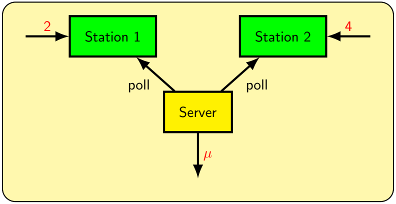
An example specification



An example specification

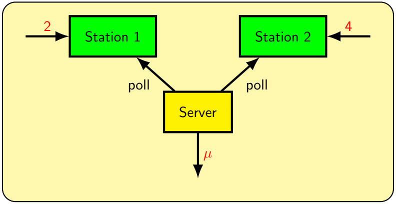


An example specification



- 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



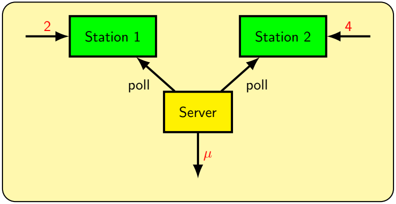
- There are 10 types of jobs
- The type of job that arrives is chosen nondeterministically
- Service time depends on job type (hence, we need queues)

The specification of the stations:

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

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

An example specification



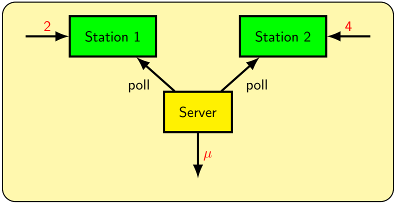
- There are 10 types of jobs
- The type of job that arrives is chosen nondeterministically
- Service time depends on job type (hence, we need queues)

The specification of the stations:

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

$$\begin{aligned} &\mathit{Station}(i : \{1,2\}, q : \text{Queue}) \\ &\quad = \textbf{notFull}(q) \quad \Rightarrow (2i) \cdot \textstyle\sum_{j:\mathit{Jobs}} \mathit{arrive}(j).\mathit{Station}(i, \textbf{enqueue}(q, j)) \\ &\quad + \textbf{notEmpty}(q) \Rightarrow \mathit{deliver}(i, \textbf{head}(q)) \sum_{k \in \{1,9\}} \tfrac{k}{10} : k = 1 \Rightarrow \mathit{Station}(i, q) \\ &\qquad\qquad\qquad\qquad\qquad\qquad\qquad + k = 9 \Rightarrow \mathit{Station}(i, \textbf{tail}(q)) \end{aligned}$$

An example specification



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

The specification of the stations:

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

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

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

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

Derivation-based operational semantics

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

Derivation-based operational semantics

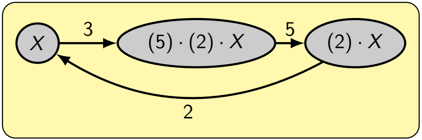
$$\textsc{MarkovPrefix}\ \frac{-}{(\lambda) \cdot p \xrightarrow{\lambda} p} \qquad \textsc{SumLeft}\ \frac{p \xrightarrow{a} p'}{p + q \xrightarrow{a} p'}$$

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

Derivation-based operational semantics

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

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

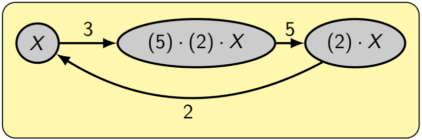


Derivation-based operational semantics

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

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

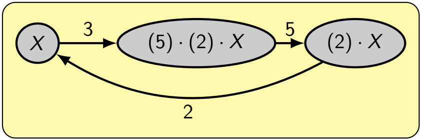
$$X = (3) \cdot (5) \cdot X + c \cdot X$$



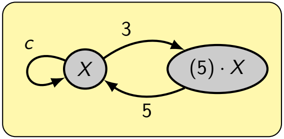
Derivation-based operational semantics

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

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



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



Derivation-based operational semantics

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

Derivation-based operational semantics

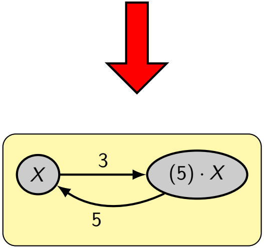
$$\textsc{MarkovPrefix}\ \frac{-}{(\lambda) \cdot p \xrightarrow{\lambda} p} \qquad \textsc{SumLeft}\ \frac{p \xrightarrow{a} p'}{p + q \xrightarrow{a} p'}$$

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

Derivation-based operational semantics

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

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

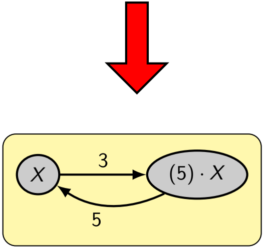


This is not right!

Derivation-based operational semantics

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

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



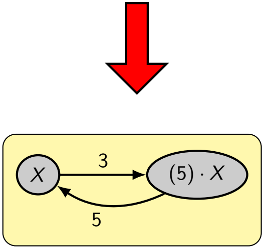
This is not right!

As a solution, we look at derivations:

Derivation-based operational semantics

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

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



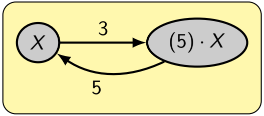
This is not right!

As a solution, we look at derivations:

Derivation-based operational semantics

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

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



This is not right!

As a solution, we look at derivations:

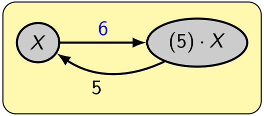
$$X \xrightarrow{3}_{\langle SL,MP\rangle} (5)\cdot X$$
$$X \xrightarrow{3}_{\langle SR,MP\rangle} (5)\cdot X$$

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

Derivation-based operational semantics

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

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



This is not right!

As a solution, we look at derivations:

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

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

MLPPEs

We defined a special format for MAPA, the MLPPE:

$$
\begin{aligned}
X(\boldsymbol{g} : \boldsymbol{G}) = & \sum_{\boldsymbol{d_1} : \boldsymbol{D_1}} c_1 && \Rightarrow a_1(\boldsymbol{b_1}) \sum_{\bullet\, \boldsymbol{e_1} : \boldsymbol{E_1}} f_1 : X(\boldsymbol{n_1}) \\
& + \quad \cdots \\
& + \sum_{\boldsymbol{d_m} : \boldsymbol{D_m}} c_m && \Rightarrow a_m(\boldsymbol{b_m}) \sum_{\bullet\, \boldsymbol{e_m} : \boldsymbol{E_m}} f_m : X(\boldsymbol{n_m})
\end{aligned}
$$

MLPPEs

We defined a special format for MAPA, the MLPPE:

$$
\begin{aligned}
X(\boldsymbol{g} : \boldsymbol{G}) = & \sum_{\boldsymbol{d_1}:\boldsymbol{D_1}} c_1 && \Rightarrow a_1(\boldsymbol{b_1}) \sum\nolimits^{\bullet}_{\boldsymbol{e_1}:\boldsymbol{E_1}} f_1 : X(\boldsymbol{n_1}) \\
+ & \cdots \\
+ & \sum_{\boldsymbol{d_m}:\boldsymbol{D_m}} c_m && \Rightarrow a_m(\boldsymbol{b_m}) \sum\nolimits^{\bullet}_{\boldsymbol{e_m}:\boldsymbol{E_m}} f_m : X(\boldsymbol{n_m}) \\
+ & \sum_{\boldsymbol{d_{m+1}}:\boldsymbol{D_{m+1}}} c_{m+1} && \Rightarrow (\lambda_{m+1}) \cdot X(\boldsymbol{n_{m+1}}) \\
+ & \cdots \\
+ & \sum_{\boldsymbol{d_n}:\boldsymbol{D_n}} c_n && \Rightarrow (\lambda_n) \cdot X(\boldsymbol{n_n})
\end{aligned}
$$

MLPPEs

We defined a special format for MAPA, the MLPPE:

$$
\begin{aligned}
X(\boldsymbol{g} : \boldsymbol{G}) = & \sum_{\boldsymbol{d_1} : \boldsymbol{D_1}} c_1 \Rightarrow a_1(\boldsymbol{b_1}) \sum_{\bullet\, \boldsymbol{e_1} : \boldsymbol{E_1}} f_1 : X(\boldsymbol{n_1}) \\
+ & \cdots \\
+ & \sum_{\boldsymbol{d_m} : \boldsymbol{D_m}} c_m \Rightarrow a_m(\boldsymbol{b_m}) \sum_{\bullet\, \boldsymbol{e_m} : \boldsymbol{E_m}} f_m : X(\boldsymbol{n_m}) \\
+ & \sum_{\boldsymbol{d_{m+1}} : \boldsymbol{D_{m+1}}} c_{m+1} \Rightarrow (\lambda_{m+1}) \cdot X(\boldsymbol{n_{m+1}}) \\
+ & \cdots \\
+ & \sum_{\boldsymbol{d_n} : \boldsymbol{D_n}} c_n \Rightarrow (\lambda_n) \cdot X(\boldsymbol{n_n})
\end{aligned}
$$

MLPPEs

We defined a special format for MAPA, the MLPPE:

$$
\begin{aligned}
X(\boldsymbol{g} : \boldsymbol{G}) = & \sum_{\boldsymbol{d_1} : \boldsymbol{D_1}} c_1 \Rightarrow a_1(\boldsymbol{b_1}) \sum_{\bullet\, \boldsymbol{e_1} : \boldsymbol{E_1}} f_1 : X(\boldsymbol{n_1}) \\
& + \cdots \\
& + \sum_{\boldsymbol{d_m} : \boldsymbol{D_m}} c_m \Rightarrow a_m(\boldsymbol{b_m}) \sum_{\bullet\, \boldsymbol{e_m} : \boldsymbol{E_m}} f_m : X(\boldsymbol{n_m}) \\
& + \sum_{\boldsymbol{d_{m+1}} : \boldsymbol{D_{m+1}}} c_{m+1} \Rightarrow (\lambda_{m+1}) \cdot X(\boldsymbol{n_{m+1}}) \\
& + \cdots \\
& + \sum_{\boldsymbol{d_n} : \boldsymbol{D_n}} c_n \Rightarrow (\lambda_n) \cdot X(\boldsymbol{n_n})
\end{aligned}
$$

MLPPEs

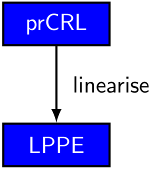
We defined a special format for MAPA, the MLPPE:

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

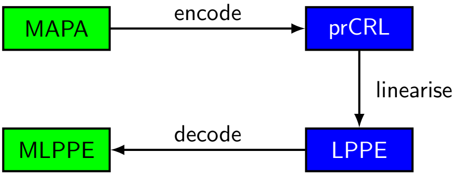
Advantages of using MLPPEs instead of MAPA specifications:

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

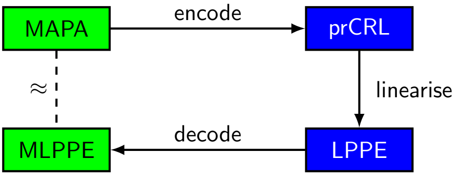
Encoding into prCRL



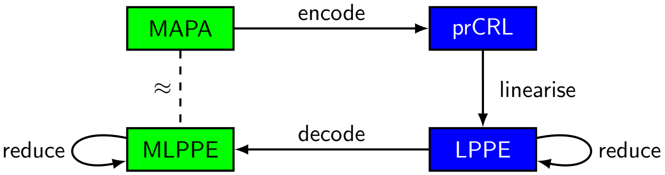
Encoding into prCRL



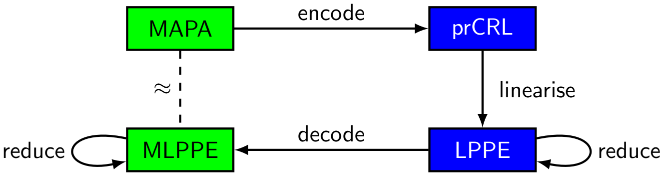
Encoding into prCRL



Encoding into prCRL

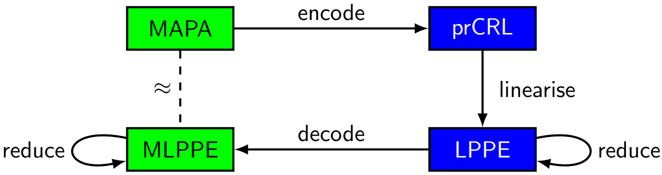


Encoding into prCRL



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

Encoding into prCRL

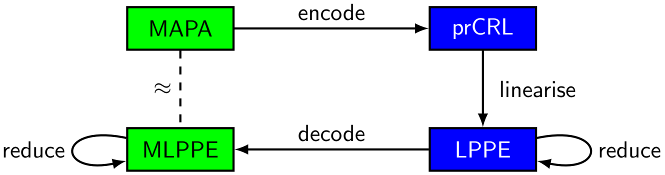


Basic idea: encode a rate λ as action rate(λ).

Problem:

Bisimulation-preserving reductions on prCRL might change MAPA behaviour

Encoding into prCRL



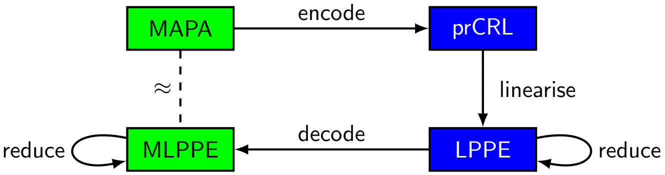
Basic idea: encode a rate λ as action rate(λ).

Problem:

Bisimulation-preserving reductions on prCRL might change MAPA behaviour

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

Encoding into prCRL



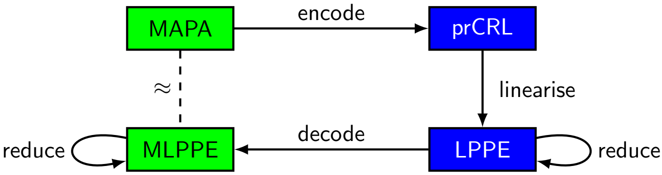
Basic idea: encode a rate λ as action rate(λ).

Problem:

Bisimulation-preserving reductions on prCRL might change MAPA behaviour

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

Encoding into prCRL



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

Problem:

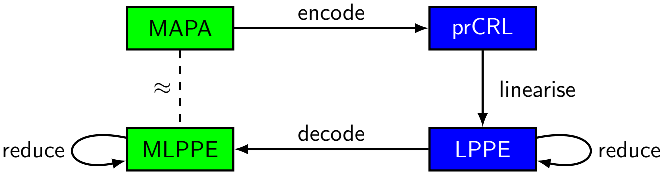
Bisimulation-preserving reductions on prCRL might change MAPA behaviour

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

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

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

Encoding into prCRL



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

Problem:

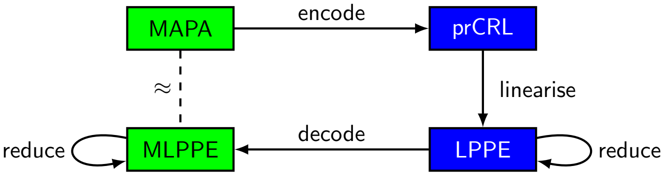
Bisimulation-preserving reductions on prCRL might change MAPA behaviour

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

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

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

Encoding into prCRL



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

Problem:

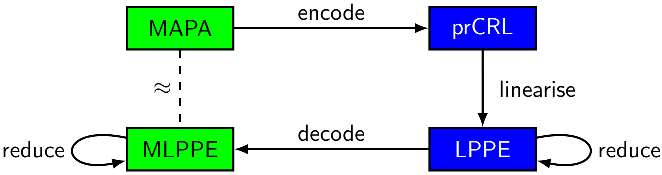
Bisimulation-preserving reductions on prCRL might change MAPA behaviour

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

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

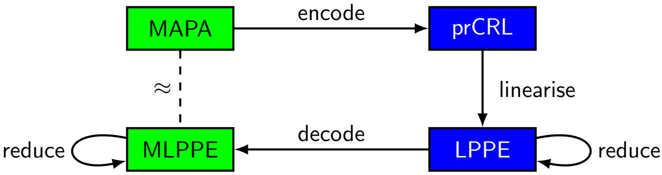
$$(\lambda) \cdot p \;\Leftarrow\; \text{rate}(\lambda) \cdot p$$

Encoding into prCRL



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

Encoding into prCRL

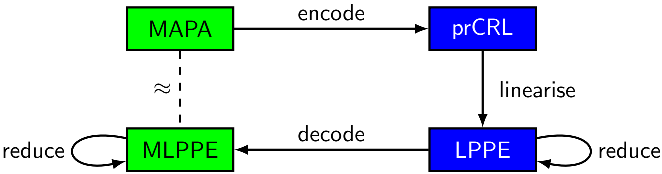


Possible solution: encode a rate λ as action $\text{rate}_i(\lambda)$.

Problem:

This still doesn't work...

Encoding into prCRL



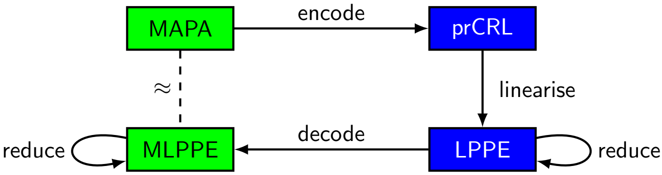
Possible solution: encode a rate λ as action rate$_i(\lambda)$.

Problem:

This still doesn't work. . .

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

Encoding into prCRL



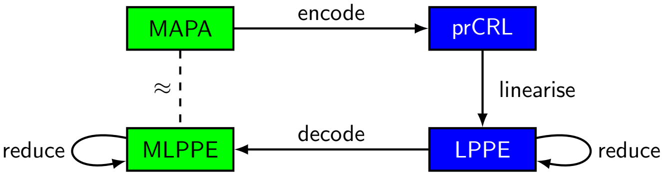
Possible solution: encode a rate λ as action $\text{rate}_i(\lambda)$.

Problem:

This still doesn't work. . .

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

Encoding into prCRL



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

Problem:

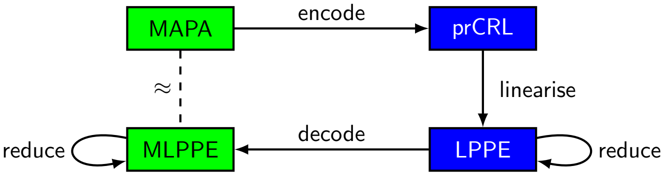
This still doesn't work...

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

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

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

Encoding into prCRL



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

Problem:

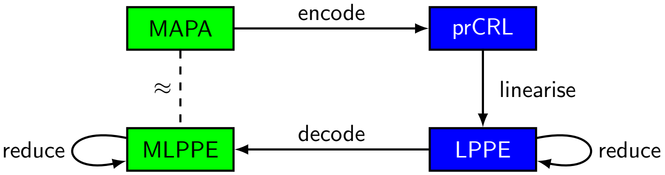
This still doesn't work. . .

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

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

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

Encoding into prCRL



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

Problem:

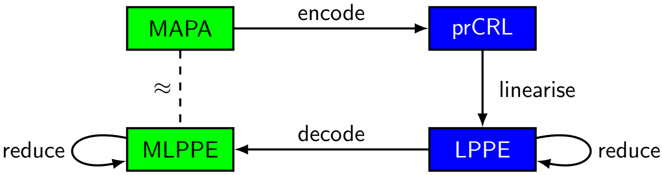
This still doesn't work. . .

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

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

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

Encoding into prCRL



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

Problem:

This still doesn't work...

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

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

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

Stronger equivalence on prCRL specifications needed!

Derivation-preserving bisimulation

Two prCRL terms are derivation-preserving bisimulation if

- There is a strong bisimulation relation R containing them

Derivation-preserving bisimulation

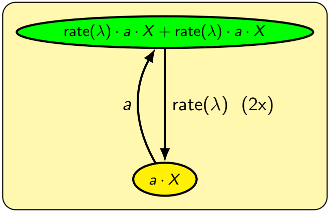
Two prCRL terms are derivation-preserving bisimulation if

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

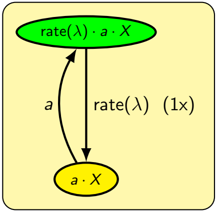
Derivation-preserving bisimulation

Two prCRL terms are derivation-preserving bisimulation if

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



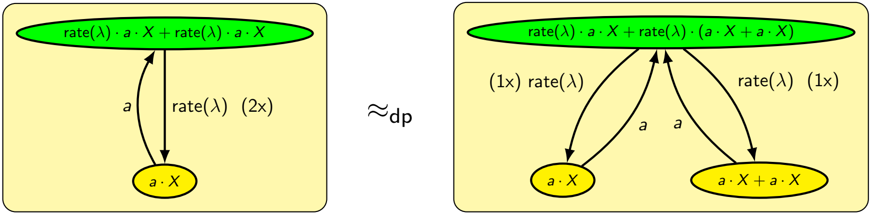
$\not\approx_{\text{dp}}$



Derivation-preserving bisimulation

Two prCRL terms are derivation-preserving bisimulation if

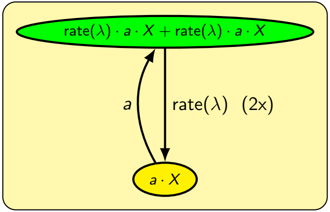
- There is a strong bisimulation relation R containing them
- Every bisimilar pair (p, p') has the same number of rate(λ) derivations to every equivalence class $[r]_R$.



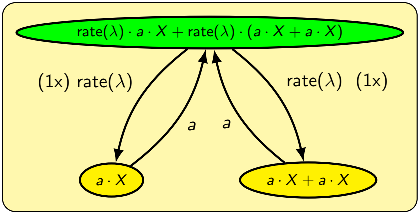
Derivation-preserving bisimulation

Two prCRL terms are derivation-preserving bisimulation if

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



$\approx_{\text{dp}}$



Proposition

Derivation-preserving bisimulation is a congruence for prCRL.

Derivation-preserving bisimulation: important results

Theorem

Given a derivation-preserving prCRL transformation f,

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

for every MAPA specification M.

Derivation-preserving bisimulation: important results

Theorem

Given a derivation-preserving prCRL transformation f,

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

for every MAPA specification M.

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

Derivation-preserving bisimulation: important results

Theorem

Given a derivation-preserving prCRL transformation f,

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

for every MAPA specification M.

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

Corollary

The linearisation procedure of prCRL can be reused for MAPA.

Generalising existing reduction techniques

Existing reduction techniques that preserve derivations:

- Constant elimination
- Expression simplification
- Dead variable reduction

Generalising existing reduction techniques

Existing reduction techniques that preserve derivations:

- Constant elimination
- Expression simplification
- Dead variable reduction

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

init $X(Mark)$

$\rightarrow$

$X = print(Mark) \cdot X$

init X

Generalising existing reduction techniques

Existing reduction techniques that preserve derivations:

- Constant elimination
- Expression simplification
- Dead variable reduction

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

$\rightarrow$

$X = beep \cdot Y$

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

Generalising existing reduction techniques

Implementation of dead variable reduction for prCRL:

```haskell
                                                                ]
                                   | otherwise = []

usedInRelevantNextState :: LPPE -> Int -> Int -> Relevance -> Int -> String -> Bool
usedInRelevantNextState lppe summandNr dk relevant cfp target = or [variableInExpression parameter (getNextState summand l) | (l,cfpp,targett) <- relevant, cfp == cfpp, target == targett]
  where
    summand   = getPSummand lppe summandNr
    parameter = fst ((getLPPEPars lppe)!!dk)

indirectRelevance3 :: DataSpec -> LPPE -> [Int] -> Rules -> BelongsTo -> Relevance -> [Int] -> Relevance
indirectRelevance3 dataspec lppe cfps rules belongsTo relevant [dk, dj] | elem (dk, dj) belongsTo = [(dk, dj, value) | i <- getPSummandNrs lppe,
                                                                                 indirectRelevance3_ dataspec lppe cfps i dk dj rules belongsTo relevant,
                                                                                 value <- [sourceValue dataspec lppe i dj]]
                                   | otherwise = []

indirectRelevance3_ :: DataSpec -> LPPE -> [Int] -> Int -> Int -> Int -> Rules -> BelongsTo -> Relevance -> Bool
indirectRelevance3_ dataspec lppe cfps summandNr dk dj rules belongsTo relevant = valid
  where
    summand    = getPSummand lppe summandNr
    parameters = [0..length (getLPPEPars lppe) - 1]
    valid = or [usedInRelevantNextState2 lppe summandNr dk belongsTo relevant dp (destinationValue dataspec lppe summandNr dp) dj | dp <- cfps, elem (summandNr, dp) rules]

usedInRelevantNextState2 lppe summandNr dk belongsTo relevant cfp target dj = or [variableInExpression parameter (getNextState summand l) | (l,cfpp,targett) <- relevant, cfp == cfpp, target == targett, no
  where
    summand   = getPSummand lppe summandNr
    parameter = fst ((getLPPEPars lppe)!!dk)


transformLPPE :: DataSpec -> LPPE -> Rules -> BelongsTo -> InitialState -> [Int] -> [Int] -> Relevance -> LPPE
transformLPPE dataspec (LPPE name pars summands) rules belongsTo initial dps cfps relevant = LPPE name pars (transformSummands dataspec (LPPE name pars summands) rules belongsTo initial dps cfps relevant :

transformSummands dataspec lppe rules belongsTo initial dps cfps relevant [] i = []
transformSummands dataspec lppe rules belongsTo initial dps cfps relevant (s:ss) i = (transformSummand dataspec lppe rules belongsTo i initial dps cfps relevant s):
                                                                    (transformSummands dataspec lppe rules belongsTo initial dps cfps relevant ss (i+1))

transformSummand dataspec lppe rules belongsTo i initial dps cfps relevant (params, c, a, aps, probChoices, g) = newSummand
  where
    newG       = transformNextStates dataspec lppe rules belongsTo i initial dps cfps relevant g [0..length g - 1]
    newSummand = (params, c, a, aps, probChoices, newG)

transformNextStates dataspec lppe rules belongsTo i initial dps cfps relevant g []     = g
transformNextStates dataspec lppe rules belongsTo i initial dps cfps relevant g (k:ks) | reset && (g!!k) /= Variable (initial!!k) = transformNextStates2 dataspec lppe rules belongsTo i initial dps cfps re
                                                                    | otherwise = transformNextStates dataspec lppe rules belongsTo i initial dps cfps relevant g ks
  where
    newG = (take k g) ++ [Variable (initial!!k)] ++ (drop (k+1) g)
    relevanceInfo = [elem k dps && elem (i, cfp) rules && elem (k,cfp) belongsTo && not(elem (k, cfp, destinationValue dataspec lppe i cfp) relevant) | cfp <- cfps]
    reset = (relevanceInfo /= [] && or relevanceInfo)

transformNextStates2 dataspec lppe rules belongsTo i initial dps cfps relevant newG ks old new = transformNextStates dataspec lppe rules belongsTo i initial dps cfps relevant newG ks

improveInitialState :: DataSpec -> LPPE -> BelongsTo -> Relevance -> InitialState -> InitialState
improveInitialState dataspec lppe belongsTo relevant initial = [improveInitialValue dataspec lppe initial belongsTo relevant parNr (initial!!parNr) | parNr <- [0..length initial - 1]]

improveInitialValue :: DataSpec -> LPPE -> InitialState -> BelongsTo -> Relevance -> Int -> String -> String
improveInitialValue dataspec lppe initial belongsTo relevant parNr initValue | canChange && willChange = newValue
                                                                  | otherwise = initValue
  where
    canChange  = [c | (d,c) <- belongsTo, d == parNr, not(elem (d,c,initial!!c) relevant)] /= []
    willChange = shouldChange dataspec lppe parNr
    newValue   = getNewValue dataspec lppe parNr


shouldChange :: DataSpec -> LPPE -> Int -> Bool
shouldChange dataspec lppe i = [s | s <- getPSummands lppe, changedToConstant dataspec (getNextState s i)] /= []

changedToConstant dataspec (Variable v) = stringIsConstant dataspec v
changedToConstant dataspec _            = False

getNewValue :: DataSpec -> LPPE -> Int -> String
getNewValue dataspec lppe i = [getValue (getNextState s i) | s <- getPSummands lppe, changedToConstant dataspec (getNextState s i)]!!0

getValue (Variable v) = v
```

Generalising existing reduction techniques

Implementation of dead variable reduction for prCRL:

```
                                    ]
                    | otherwise = []

usedInRelevantNextState :: LPPE -> Int -> Int -> Relevance -> Int -> String -> Bool
usedInRelevantNextState lppe summandNr dk relevant cfp target = or [variableInExpression parameter (getNextState summand l) | (l,cfpp,targett) <- relevant, cfp == cfpp, target == targett]
  where
    summand   = getPSummand lppe summandNr
    parameter = fst ((getLPPEPars lppe)!!dk)

indirectRelevance3 :: DataSpec -> LPPE -> [Int] -> Rules -> BelongsTo -> Relevance -> [Int] -> Relevance
indirectRelevance3 dataspec lppe cfps rules belongsTo relevant [dk, dj] | elem (dk, dj) belongsTo = [(dk, dj, value) | i <- getPSummandNrs lppe,
                                                                          indirectRelevance3_ dataspec lppe cfps i dk dj rules belongsTo relevant,
                                                                          value <- [sourceValue dataspec lppe i dj]]
                    | otherwise = []

indirectRelevance3_ :: DataSpec -> LPPE -> [Int] -> Int -> Int -> Int -> Rules -> BelongsTo -> Relevance -> Bool
indirectRelevance3_ dataspec lppe cfps summandNr dk dj rules belongsTo relevant = valid
  where
    summand    = getPSummand lppe summandNr
    parameters = [0..length (getLPPEPars lppe) - 1]
    valid = or [usedInRelevantNextState2 lppe summandNr dk belongsTo relevant dp (destinationValue dataspec lppe summandNr dp) dj | dp <- cfps, elem (summandNr, dp) rules]

usedInRelevantNextState2 lppe summandNr dk belongsTo relevant cfp target dj = or [variableInExpression parameter (getNextState summand l) | (l,cfpp,targett) <- relevant, cfp == cfpp, target == targett, no
  where
    summand   = getPSummand lppe summandNr
    parameter = fst ((getLPPEPars lppe)!!dk)


transformLPPE :: DataSpec -> LPPE -> Rules -> BelongsTo -> InitialState -> [Int] -> [Int] -> Relevance -> LPPE
transformLPPE dataspec (LPPE name pars summands) rules belongsTo initial dps cfps relevant = LPPE name pars (transformSummands dataspec (LPPE name pars summands) rules belongsTo initial dps cfps relevant :

transformSummands dataspec lppe rules belongsTo initial dps cfps relevant [] i = []
transformSummands dataspec lppe rules belongsTo initial dps cfps relevant (s:ss) i = (transformSummand dataspec lppe rules belongsTo i initial dps cfps relevant s):
                                             (transformSummands dataspec lppe rules belongsTo initial dps cfps relevant ss (i+1))

transformSummand dataspec lppe rules belongsTo i initial dps cfps relevant (params, c, a, aps, probChoices, g) = newSummand
  where
    newG       = transformNextStates dataspec lppe rules belongsTo i initial dps cfps relevant g [0..length g - 1]
    newSummand = (params, c, a, aps, probChoices, newG)

transformNextStates dataspec lppe rules belongsTo i initial dps cfps relevant g []     = g
transformNextStates dataspec lppe rules belongsTo i initial dps cfps relevant g (k:ks) | reset && (g!!k) /= Variable (initial!!k) = transformNextStates2 dataspec lppe rules belongsTo i initial dps cfps re
                                             | otherwise = transformNextStates dataspec lppe rules belongsTo i initial dps cfps relevant g ks
  where
    newG = (take k g) ++ [Variable (initial!!k)] ++ (drop (k+1) g)
    relevanceInfo = [elem k dps && elem (i, cfp) rules && elem (k,cfp) belongsTo && not(elem (k, cfp, destinationValue dataspec lppe i cfp) relevant) | cfp <- cfps]
    reset = (relevanceInfo /= [] && or relevanceInfo)

transformNextStates2 dataspec lppe rules belongsTo i initial dps cfps relevant newG ks old new = transformNextStates dataspec lppe rules belongsTo i initial dps cfps relevant newG ks

improveInitialState :: DataSpec -> LPPE -> BelongsTo -> Relevance -> InitialState -> InitialState
improveInitialState dataspec lppe belongsTo relevant initial = [improveInitialValue dataspec lppe initial belongsTo relevant parNr (initial!!parNr) | parNr <- [0..length initial - 1]]

improveInitialValue :: DataSpec -> LPPE -> InitialState -> BelongsTo -> Relevance -> Int -> String -> String
improveInitialValue dataspec lppe initial belongsTo relevant parNr initValue | canChange && willChange = newValue
                                             | otherwise = initValue
  where
    canChange  = [c | (d,c) <- belongsTo, d == parNr, not(elem (d,c,initial!!c) relevant)] /= []
    willChange = shouldChange dataspec lppe parNr
    newValue   = getNewValue dataspec lppe parNr


shouldChange :: DataSpec -> LPPE -> Int -> Bool
shouldChange dataspec lppe i = [s | s <- getPSummands lppe, changedToConstant dataspec (getNextState s i)] /= []

changedToConstant dataspec (Variable v) = stringIsConstant dataspec v
changedToConstant dataspec _            = False

getNewValue :: DataSpec -> LPPE -> Int -> String
getNewValue dataspec lppe i = [getValue (getNextState s i) | s <- getPSummands lppe, changedToConstant dataspec (getNextState s i)]!!0

getValue (Variable v) = v
```

Implementation of dead variable reduction for MAPA:

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

Novel reduction techniques

New reduction techniques for MAPA:

- Maximal progress reduction
- Summation elimination

Novel reduction techniques

New reduction techniques for MAPA:

- Maximal progress reduction
- Summation elimination

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

Novel reduction techniques

New reduction techniques for MAPA:

- Maximal progress reduction
- Summation elimination

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

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

$\rightarrow$

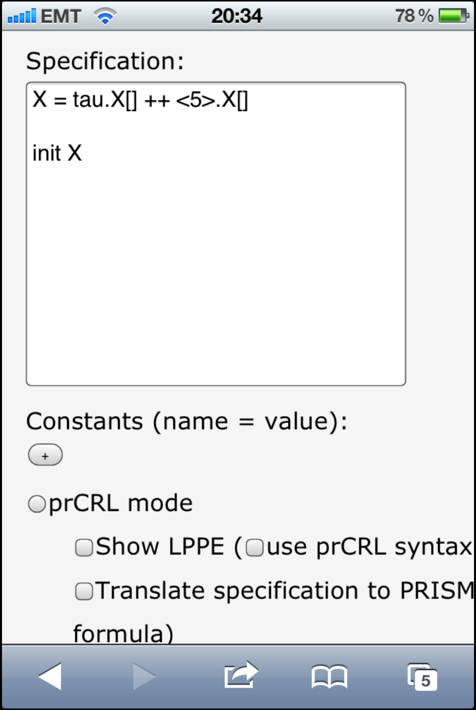
$X = send(2) \cdot X$

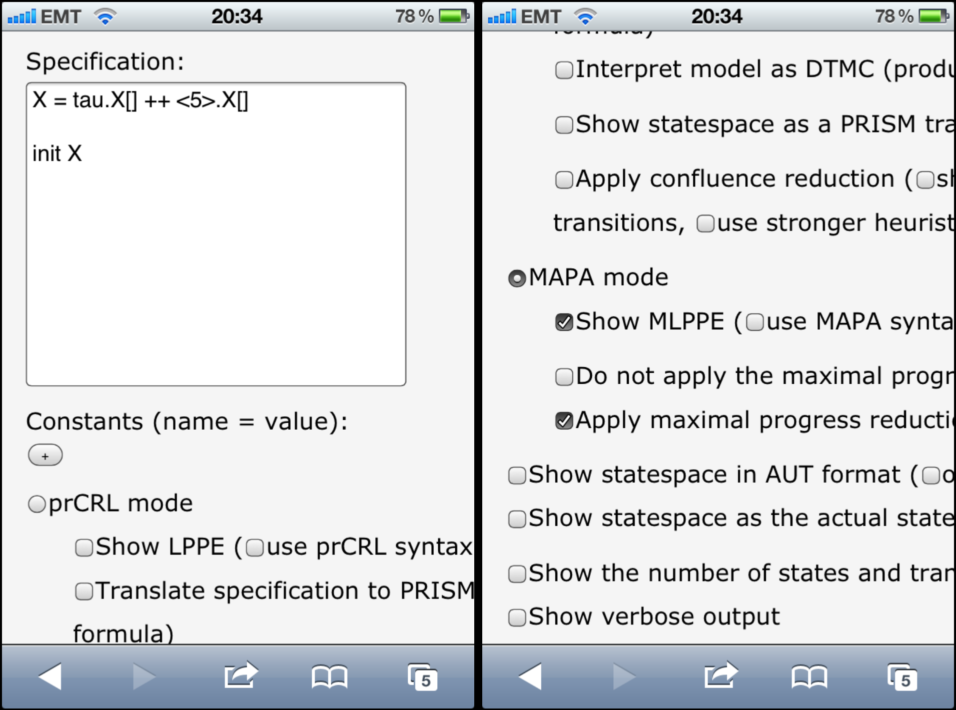
$Y = (15) \cdot Y$

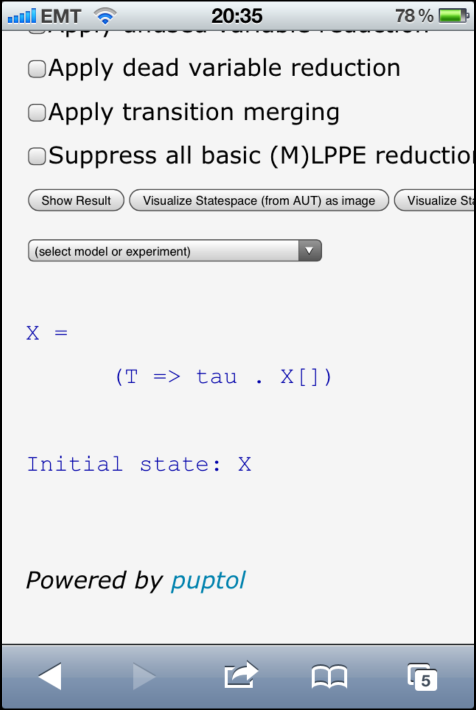
Implementation and Case Study

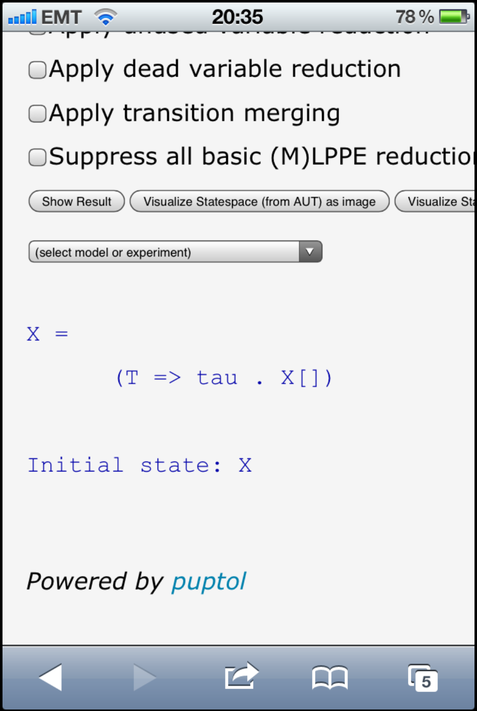
Implementation in SCOOP:

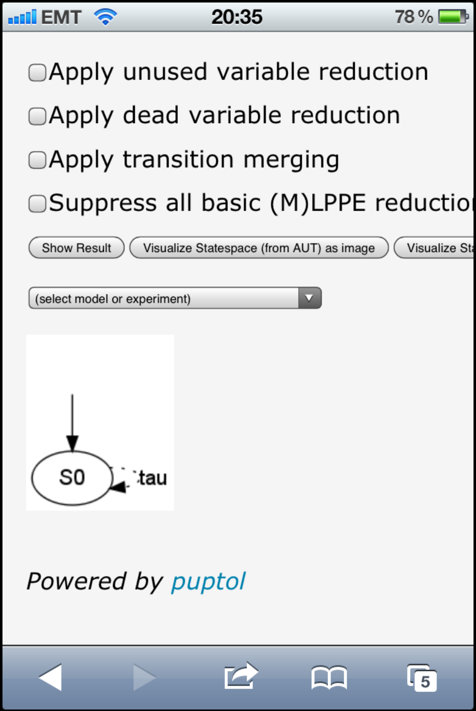
- Programmed in Haskell
- Stand-alone and web-based interface
- Linearisation, optimisation, state space generation


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


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


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


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


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

Implementation and Case Study

Implementation in SCOOP:

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

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

Table: State space generation using SCOOP.

Implementation and Case Study

Implementation in SCOOP:

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

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

Table: State space generation using SCOOP.

Implementation and Case Study

Implementation in SCOOP:

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

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

Table: State space generation using SCOOP.

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
- Link to model checking tools for MAs

Questions

Questions?

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