Symbolic Manipulation of Markov Automata

Jaco van de Pol
September 8, 2012

Joint work with Joost-Pieter Katoen, Mariëlle Stoelinga, and Mark Timmer
Model Driven Design

- Modeling complex interaction, behaviour and data
- High-performance model analysis
- Quantitative modeling and evaluation
Model Driven Design

- Modeling complex interaction, behaviour and data
- High-performance model analysis
- Quantitative modeling and evaluation

Aim
Transfer high-performance scalable analysis to quantitative models
Quantitative Modeling Requirements
Quantitative Modeling Requirements
Specifying systems with

- Nondeterminism → LTS
- Probability → DTMC
- Stochastic timing → CTMC
Specifying systems with

- Nondeterminism
- Probability
- Stochastic timing

Probabilistic Automata (PA, MDP)
Specifying systems with

- Nondeterminism
- Probability
- Stochastic timing

Interactive Markov Chains (IMC)
Markov Automata in the stochastic modeling landscape

Specifying systems with

- Nondeterminism
- Probability
- Stochastic timing

Markov Automata (MA)
Markov Automata in the stochastic modeling landscape

Specifying systems with

- Nondeterminism
- Probability
- Stochastic timing

Markov Automata (MA)

\[ \lambda_1 \rightarrow \text{Station 1} \]

\[ \lambda_2 \rightarrow \text{Station 2} \]

\[ \text{Server} \]

\( (\text{error probability } p) \) poll

\[ \mu \]

poll \( (\text{error probability } p) \)

\[ \tau \]

UNIVERSITY OF TWENTE.

Symbolic Manipulation of Markov Automata

September 8, 2012
Markov Automata in the stochastic modeling landscape

Specifying systems with

- Nondeterminism
- Probability
- Stochastic timing

Markov Automata (MA)
Markov Automata in the stochastic modeling landscape

Specifying systems with

- Nondeterminism
- Probability
- Stochastic timing

Markov Automata (MA)
Markov Automata in the stochastic modeling landscape

Specifying systems with

- Nondeterminism
- Probability
- Stochastic timing

Markov Automata (MA)
Markov Automata in the stochastic modeling landscape

Specifying systems with

- Nondeterminism
- Probability
- Stochastic timing

Markov Automata (MA)
Specifying systems with

- Nondeterminism
- Probability
- Stochastic timing

Markov Automata (MA)
Markov Automata in the stochastic modeling landscape

Specifying systems with

- Nondeterminism
- Probability
- Stochastic timing

Markov Automata (MA)
Overview of our approach – Symbolic Manipulation

- Specification
- Generation
- State space
- Visualization
- Model checking
- Evaluation
- Minimization
Overview of our approach – Symbolic Manipulation

- Specification
- Linearization
- Intermediate format
- Generation
- State space
- Visualization
- Model checking
- Evaluation
- Optimization
  - Static Analysis
  - Confluence reduction
- Minimization
Overview of our approach – Symbolic Manipulation

- Specification
- Linearization
- Optimization
- State space
- Visualization
- Model checking
- Evaluation
- MAPA
- Intermediate format
- Linear Process
- Optimization (Static Analysis, Confluence reduction)
- Minimization
- Markov Automaton
- Generation
Contents

1. Introduction
2. MAPA: Markov-Automatic Process Algebra
3. Linearization
4. Optimization by Static Analysis
5. Confluence Reduction
6. High-performance State Space Generation in LTSmin
7. Implementation in SCOOP and Use Case
8. Conclusions and Future Work
MAPA: a process algebra with data, probabilities, rates

- **ACP** (Algebra of Communicating Processes)
  - Basics: alternation, sequence, recursion .......... (maps to LTS)
  - We also use: parallel composition, encapsulation, hiding

---

The grammar of ACP/μCRL/pCRL/MAPA

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

\[ p ::= Y \mid p + p \mid \text{ } \mid a \mid p \]
MAPA: a process algebra with data, probabilities, rates

- **ACP** (Algebra of Communicating Processes)
  - Basics: alternation, sequence, recursion ........ (maps to LTS)
  - We also use: parallel composition, encapsulation, hiding
- **µCRL** (micro Common Representation Language)
  - Addition: Algebraic Data Types ............... (maps to ∞ LTS)

The grammar of ACP/µCRL/pCRL/MAPA

Process terms in µCRL are obtained by the following grammar:

\[ p ::= Y(t) \mid p + p \mid \sum_{x:D} p \mid c \Rightarrow p \mid a(t) \mid p \]
MAPA: a process algebra with data, probabilities, rates

- **ACP** (Algebra of Communicating Processes)
  - Basics: alternation, sequence, recursion ........ (maps to LTS)
  - We also use: parallel composition, encapsulation, hiding
- **µCRL** (micro Common Representation Language)
  - Addition: Algebraic Data Types ............... (maps to ∞ LTS)
  - Extra: parameters, data choice, conditions

The grammar of ACP/µCRL/prCRL/MAPA

Process terms in µCRL are obtained by the following grammar:

\[
p ::= Y(t) \mid p + p \mid \sum_{x:D} p \mid c \Rightarrow p \mid a(t) \mid p
\]
MAPA: a process algebra with data, probabilities, rates

- **ACP** (Algebra of Communicating Processes)
  - Basics: alternation, sequence, recursion ........ (maps to LTS)
  - We also use: parallel composition, encapsulation, hiding
- **μCRL** (micro Common Representation Language)
  - Addition: Algebraic Data Types ............... (maps to $\infty$ LTS)
  - Extra: parameters, data choice, conditions
- **prCRL**: Probabilistic CRL
  - Addition: probabilistic choice..................(maps to PA)

The grammar of ACP/μCRL/prCRL/MAPA

Process terms in prCRL are obtained by the following grammar:

\[
p ::= Y(t) \mid p + p \mid \sum_{x:D} p \mid c \Rightarrow p \mid a(t)\sum_{x:D} f:p
\]
MAPA: a process algebra with data, probabilities, rates

- **ACP** (Algebra of Communicating Processes)
  - Basics: alternation, sequence, recursion ........ (maps to LTS)
  - We also use: parallel composition, encapsulation, hiding

- **µCRL** (micro Common Representation Language)
  - Addition: Algebraic Data Types ............... (maps to ∞ LTS)
  - Extra: parameters, data choice, conditions

- **prCRL**: Probabilistic CRL
  - Addition: probabilistic choice.................. (maps to PA)

- **MAPA**: Markov-Automatic Process Algebra
  - Addition: rates for stochastic timing ........ (maps to MA)

The grammar of ACP/µCRL/prCRL/MAPA

Process terms in MAPA are obtained by the following grammar:

\[ p ::= Y(t) \mid p + p \mid \sum_{x:D} p \mid c \Rightarrow p \mid a(t)\sum_{x:D} f:p \mid \lambda \cdot p \]
An example specification

- There are 10 different jobs
- The type of job arriving is chosen **nondeterministically**
- Service time depends on job (we store jobs in a **queue**)

```
There are 10 different jobs
The type of job arriving is chosen nondeterministically
Service time depends on job (we store jobs in a queue)
```
An example specification

There are 10 different jobs
- The type of job arriving is chosen nondeterministically
- Service time depends on job (we store jobs in a queue)

The MAPA specification of a station:

\[
\text{type } \text{Jobs} = \{1, \ldots, 10\}
\]

\[
\text{Station}(i : \{1, 2\}, q : \text{Queue}) = \text{notFull}(q) \Rightarrow (2i) \cdot \sum_{j : \text{Jobs}} \text{arrive}(j). \text{Station}(i, \text{enqueue}(q, j))
\]
An example specification

There are 10 different jobs

The type of job arriving is chosen nondeterministically

Service time depends on job (we store jobs in a queue)

The MAPA specification of a station:

\[
\text{type } \text{Jobs} = \{1, \ldots, 10\}
\]

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

\[
= \text{notFull}(q) \Rightarrow (2i) \cdot \sum_{j : \text{Jobs}} \text{arrive}(j).\text{Station}(i, \text{enqueue}(q, j))
\]

\[
+ \text{notEmpty}(q) \Rightarrow \text{deliver}(i, \text{head}(q)) \sum_{k \in \{1, 9\}} \frac{k}{10} : k = 1 \Rightarrow \text{Station}(i, q)
\]

\[
+ k = 9 \Rightarrow \text{Station}(i, \text{tail}(q))
\]
An example specification

There are 10 different jobs
The type of job arriving is chosen nondeterministically
Service time depends on job (we store jobs in a queue)

The MAPA specification of a station:

\[
\text{type } \text{Jobs} = \{1, \ldots, 10\}
\]

\[
\text{Station}(i : \{1, 2\}, q : \text{Queue})
\]
\[
= \text{notFull}(q) \Rightarrow (2i) \cdot \sum_{j : \text{Jobs}} \text{arrive}(j) \cdot \text{Station}(i, \text{enqueue}(q, j))
\]
\[
+ \text{notEmpty}(q) \Rightarrow \text{deliver}(i, \text{head}(q))(\frac{1}{10} : \text{Station}(i, q) \oplus \frac{9}{10} : \text{Station}(i, \text{tail}(q)))
\]
Linear Process Equations

We define an intermediate format for µCRL, the LPE:

\[ X(g : G) = \sum_{d:D_1} c_1 \Rightarrow a_1(t_1)X(n_1) \]

\[ + \cdots \]

\[ + \sum_{d:D_m} c_m \Rightarrow a_m(t_m)X(n_m) \]

- \( g \) is a vector of global state variables, \( d \) are local variables
- \( c, a(t), n \) are the condition, action, and the next state vector
Linear Process Equations

We define an intermediate format for prCRL, the LPPE:

\[ X(g : G) = \sum_{d:D_1} c_1 \implies a_1(t_1) \sum_{e:E_1} f_1 : X(n_1) \]

+ \ldots

+ \sum_{d:D_m} c_m \implies a_m(t_m) \sum_{e:E_m} f_m : X(n_m) \]

- \( g \) is a vector of global state variables, \( d \) are local variables
- \( c, a(t), n \) are the condition, action, and the next state vector
- \( f \) is a probability distribution
We define an intermediate format for MAPA, the MLPPE:

\[
X(g : G) = \sum_{d : D_1} c_1 \Rightarrow a_1(t_1) \sum_{e : E_1} f_1 : X(n_1)
\]

\[+ \cdots\]

\[
+ \sum_{d : D_m} c_m \Rightarrow a_m(t_m) \sum_{e : E_m} f_m : X(n_m)
\]

\[+ \sum_{d : D_{m'}} c_{m'} \Rightarrow \lambda_{m'} \cdot X(n_{m'})\]

\[+ \cdots\]

\[
+ \sum_{d : D_n} c_n \Rightarrow \lambda_n \cdot X(n_n)
\]

- \(g\) is a vector of global state variables, \(d\) are local variables
- \(c, a(t), n\) are the condition, action, and the next state vector
- \(f\) is a probability distribution, and \(\lambda\) indicates the rates
A random number generator

\[ X(\text{active} : \text{Bool}) = \]
\[
\begin{align*}
\text{not(\text{active})} \Rightarrow & \text{ping} \cdot \sum_{b: \text{Bool}} X(b) \\
\text{+ active} \Rightarrow & \tau \sum_{n: \mathbb{N}^+} \frac{1}{2^n} \left( \text{send}(n) \cdot X(\text{false}) \right)
\end{align*}
\]
A random number generator

\[ X(\text{active} : \text{Bool}) = \]

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

\[ + \text{ active } \Rightarrow \tau \sum_{n:\mathbb{N}^+} \frac{1}{2^n} : \left( \text{send}(n) \cdot X(\text{false}) \right) \]
Linear Probabilistic Process Equations – an example

Specification in prCRL

\[ X(\text{active} : \text{Bool}) = \]
\[ \text{not(\text{active})} \Rightarrow \text{ping} \cdot \sum_{b : \text{Bool}} X(b) \]
\[ + \text{active} \Rightarrow \tau \sum_{n : \mathbb{N}^+} \frac{1}{2^n} : \text{send}(n) \cdot X(\text{false}) \]
Linear Probabilistic Process Equations – an example

Specification in prCRL

\[ X(\text{active} : \text{Bool}) = \]
\[ \text{not(active)} \Rightarrow \text{ping} \cdot \sum_{b:\text{Bool}} X(b) \]
\[ + \text{active} \Rightarrow \tau \sum_{n:\mathbb{N}^+} \frac{1}{2^n} : \text{send}(n) \cdot X(\text{false}) \]

Specification in LPPE

\[ X(pc : \{1..3\}, n : \mathbb{N}^+) = \]
\[ + pc = 1 \Rightarrow \text{ping} \cdot X(2, 1) \]
\[ + pc = 2 \Rightarrow \text{ping} \cdot X(2, 1) \]
\[ + pc = 2 \Rightarrow \tau \sum_{n:\mathbb{N}^+} \frac{1}{2^n} : X(3, n) \]
\[ + pc = 3 \Rightarrow \text{send}(n) \cdot X(1, 1) \]
Advantages of Linear Processes

\[ X(g : G) = \sum_{i \in I} \sum_{d_i : D_i} c_i(g, d_i) \Rightarrow a_i(t_i) \sum_{e_i : E_i} f_i(g, d_i, e_i) : X(n_i(g, d_i, e_i)) \]

\[ + \sum_{j \in J} \sum_{d_j : D_j} c_j(g, d_j) \Rightarrow (\lambda_j(g, d_j)) \cdot X(n_j(g, d_j)) \]

Advantages of using Linear Processes:

- Straightforward **state space generation**
- Direct definition of **parallel composition**
- **Symbolic optimizations** enabled at the language level
- Control is encoded in data: apply **automated theorem proving**
Advantages of Linear Processes

\[ X(g : G) = \sum_{i \in I} \sum_{d_i : D_i} c_i(g, d_i) \Rightarrow a_i(t_i) \sum_{e_i : E_i} f_i(g, d_i, e_i) : X(n_i(g, d_i, e_i)) \]

\[ + \sum_{j \in J} \sum_{d_j : D_j} c_j(g, d_j) \Rightarrow (\lambda_j(g, d_j)) \cdot X(n_j(g, d_j)) \]

Advantages of using Linear Processes:

- **Straightforward state space generation**
- **Direct definition of parallel composition**
- **Symbolic optimizations enabled at the language level**
- **Control is encoded in data:** apply automated theorem proving

**Example:** \( \varphi(g) \) is an invariant of \( X \) iff

\[ \bigwedge_{i \in I \cup J} \forall g, d_i. \varphi(g) \land c_i(g, d_i) \Rightarrow \varphi(n_i(g, d_i)) \]
Optimization techniques for Probabilistic LPPE

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

2. State space reduction
   - Dead variable reduction
   - Confluence reduction
Optimization techniques for Probabilistic LPPE

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

2. State space reduction
   - Dead variable reduction
   - Confluence reduction

\[ X(id : ld) = print(id) \cdot X(id) \]
\[ init \ X(Mark) \]

\[ \rightarrow \]

\[ X = print(Mark) \cdot X \]
\[ init \ X \]
Optimization techniques for Probabilistic LPPE

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

2. **State space reduction**
   - Dead variable reduction
   - Confluence reduction

\[ X = \sum_{d \in \{1,2,3\}} d = 2 \Rightarrow \text{send}(d) \cdot X \]

init $X$

\[ X = \text{send}(2) \cdot X \]

init $X$
Optimization techniques for Probabilistic LPPE

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

2. State space reduction
   - Dead variable reduction
   - Confluence reduction

\[ X = (3 = 1 + 2 \lor x > 5) \Rightarrow \text{beep} \cdot Y \]

\[ \Rightarrow \]

\[ X = \text{beep} \cdot Y \]
Optimization techniques for Probabilistic LPPE

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

2. State space reduction
   - Dead variable reduction
   - Confluence reduction

- Deduce the control flow graphs of an LPPE
- Compute the relevant (live) variables at each control location
- Reset dead variables, thereby collapsing states
Optimization techniques for Probabilistic LPPE

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

2. State space reduction
   - Dead variable reduction
   - Confluence reduction

- Detect **confluent** internal transitions
- Give these transitions **priority**
Special optimization techniques for linear MAPA processes:

- Maximal progress reduction
- Summation elimination
Special optimization techniques for linear MAPA processes:

- Maximal progress reduction
- Summation elimination

\[ X = \tau \cdot X + (5) \cdot X \quad \rightarrow \quad X = \tau \cdot X \]
Special optimization techniques for linear MAPA processes:

- Maximal progress reduction
- Summation elimination

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

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

\[
X = send(2) \cdot X
\]

\[
Y = (15) \cdot Y
\]
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:

Some care is necessary:

\[ a \cdot p + \lambda \cdot q = a \cdot p \]  \hspace{1cm} \text{(maximal progress)}

\[ a \cdot p + a \cdot p = a \cdot p \]  \hspace{1cm} \text{(idempotence)}

\[ \lambda \cdot p + \lambda \cdot p = 2\lambda \cdot p \]  \hspace{1cm} \text{(preserve exit rate)}
From Probabilistic systems to Markov Automata

Viewing any rate $\lambda$ just as an action label $\text{rate}(\lambda)$, we get:

$$
\text{encode} : \text{MAPA} \rightarrow \text{prCRL} \\
\text{decode} : \text{prCRL} \rightarrow \text{MAPA}
$$

**Theorem**

*For any prCRL transformation $f$ preserving multiplicities,*

$$
\text{decode}(f(\text{encode}(M))) \approx M
$$
From Probabilistic systems to Markov Automata

Viewing any rate $\lambda$ just as an action label $rate(\lambda)$, we get:

\[
\text{encode} : \text{MAPA} \rightarrow \text{prCRL} \\
\text{decode} : \text{prCRL} \rightarrow \text{MAPA}
\]

**Theorem**

For any prCRL transformation $f$ preserving multiplicities,

\[
\text{decode}(f(\text{encode}(M))) \approx M
\]

Techniques from the PA world *generalise trivially* to the MA world!

**Corollaries**

- The linearization procedure of prCRL can be *reused* for MAPA.
- $\text{deadVarRedMA} = \text{decode} \circ \text{deadVarRedProb} \circ \text{encode}$
Weaker notion: **branching probabilistic bisimulation**
Weaker notion: branching probabilistic bisimulation
Weaker notion: branching probabilistic bisimulation

Probability of green:
\[
\frac{1}{2} \cdot \frac{1}{3} + \frac{1}{2} \cdot \frac{2}{3} = \frac{1}{2}
\]

Probability of red:
\[
\frac{1}{2} \cdot \frac{2}{3} + \frac{1}{2} \cdot \frac{1}{3} = \frac{1}{2}
\]
Weaker notion: **branching probabilistic bisimulation** (based on probabilistic schedulers)
Weaker notion: **branching probabilistic bisimulation**
(based on probabilistic schedulers)

\[
\begin{align*}
\text{Probability of green: } & \quad \frac{1}{2} \cdot \frac{1}{3} + \frac{1}{2} \cdot \frac{2}{3} = \frac{1}{2} \\
\text{Probability of red: } & \quad \frac{1}{2} \cdot \frac{2}{3} + \frac{1}{2} \cdot \frac{1}{3} = \frac{1}{2}
\end{align*}
\]

Reduce the state space modulo branching bisimulation *on-the-fly*?
State space reduction using confluence

**Step 1:** Detect (and mark) confluent $\tau$ transitions.
Step 1: Detect (and mark) confluent $\tau$ transitions.

![Diagram showing state space reduction using confluence](image-url)
Step 1: Detect (and mark) confluent $\tau$ transitions.
State space reduction using confluence

**Step 1:** Detect (and mark) confluent $\tau$ transitions.

**Step 2:** Reduce the state space to representative states in the terminal strongly connected components in the $\tau_c$-graph.
State space reduction using confluence

**Step 1:** Detect (and mark) confluent $\tau$ transitions.

**Step 2:** Reduce the state space to representative states in the terminal strongly connected components in the $\tau_c$-graph.
Confluence: non-probabilistic versus probabilistic

Theorem
States that are connected by confluent $\tau_c$-steps are branching bisimilar.

Strong confluence

Strong probabilistic confluence
Confluence: non-probabilistic versus probabilistic

Strong confluence

Strong probabilistic confluence

Theorem
States that are connected by confluent \( \tau_c \)-steps are branching bisimilar.
Confluence: non-probabilistic versus probabilistic

Strong confluence

Strong probabilistic confluence

Theorem
States that are connected by confluent $\tau_c$-steps are branching bisimilar.
Confluence: non-probabilistic versus probabilistic

**Theorem**

*States that are connected by confluent $\tau$-steps are branching bisimilar.*
Symbolic detection of confluence

\[ X(g : G) = \sum_{d_i : D_i} c_i \Rightarrow \tau \cdot X(n_i) \]

\[ + \sum_{d_j : D_j} c_j \Rightarrow a_j \sum_{e_j : E_j} f_j : X(n_j) \]

Two summands \( i, j \) commute if

\[ i : \tau_c \quad \text{and} \quad j : a \]

\[ j : a \quad \text{and} \quad i : \tau_c \]

UNIVERSITY OF TWENTE.

Symbolic Manipulation of Markov Automata

September 8, 2012 20 / 26
Symbolic detection of confluence

\[ X(g : G) = \sum_{d_i:D_i} c_i \Rightarrow \tau \cdot X(n_i) \]

\[ + \sum_{d_j:D_j} c_j \Rightarrow a_j \sum_{e_j:E_j} f_j \cdot X(n_j) \]

Two summands \( i, j \) commute if

\[ \forall g, d_i, d_j, e_j : (c_i(g, d_i) \land c_j(g, d_j)) \rightarrow \]

\[ \left( \begin{array}{c} \vdots \\ \vdots \end{array} \right) \]
Symbolic detection of confluence

\[ X(g : G) = \sum_{d_i : D_i} c_i \Rightarrow \tau \cdot X(n_i) \]

\[ + \sum_{d_j : D_j} c_j \Rightarrow a_j \sum_{e_j : E_j} f_j \cdot X(n_j) \]

Two summands \( i, j \) commute if

\[ \forall g, d_i, d_j, e_j : (c_i(g, d_i) \land c_j(g, d_j)) \rightarrow \]

\[ c_j(n_i(g, d_i), d_j) \land c_i(n_j(g, d_j, e_j), d_i) \]
Symbolic detection of confluence

\[ X(g : G) = \sum_{d_i : D_i} c_i \Rightarrow \tau \cdot X(n_i) \]
\[ \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad + \sum_{d_j : D_j} c_j \Rightarrow a_j \sum_{e_j : E_j} f_j : X(n_j) \]

Two summands \( i, j \) commute if

\[ \forall g, d_i, d_j, e_j : (c_i(g, d_i) \land c_j(g, d_j)) \rightarrow \]
\[ \left( c_j(n_i(g, d_i), d_j) \land c_i(n_j(g, d_j, e_j), d_i) \right) \land \]
\[ a_j(g, d_j) = a_j(n_i(g, d_i), d_j) \]
Symbolic detection of confluence

\[ X(g : G) = \sum_{d_i : D_i} c_i \Rightarrow \tau \cdot X(n_i) \]

\[ \ldots \]

\[ + \sum_{d_j : D_j} c_j \Rightarrow a_j \sum_{e_j : E_j} f_j \cdot X(n_j) \]

Two summands \( i, j \) commute if

\[ \forall g, d_i, d_j, e_j : (c_i(g, d_i) \land c_j(g, d_j)) \rightarrow \]

\[ \left( c_j(n_i(g, d_i), d_j) \land c_i(n_j(g, d_j, e_j), d_i) \right) \]

\[ \land \quad a_j(g, d_j) = a_j(n_i(g, d_i), d_j) \]

\[ \land \quad n_j(n_i(g, d_i), d_j, e_j) = n_i(n_j(g, d_j, e_j), d_i) \]
Symbolic detection of confluence

\[ X(g : G) = \sum_{d_i : D_i} c_i \Rightarrow \tau \cdot X(n_i) \]

\[ + \sum_{d_j : D_j} c_j \Rightarrow a_j \sum_{e_j : E_j} f_j \cdot X(n_j) \]

Two summands \( i, j \) commute if

\[ \forall g, d_i, d_j, e_j : (c_i(g, d_i) \land c_j(g, d_j)) \rightarrow \]

\[ \begin{align*}
    c_j(n_i(g, d_i), d_j) \land c_i(n_j(g, d_j, e_j), d_i) \\
    \land a_j(g, d_j) = a_j(n_i(g, d_i), d_j) \\
    \land n_j(n_i(g, d_i), d_j, e_j) = n_i(n_j(g, d_j, e_j), d_i) \\
    \land f_j(g, d_j, e_j) = f_j(n_i(g, d_i), d_j, e_j)
\end{align*} \]
Symbolic detection of confluence

\[ X(g : G) = \sum_{d_i : D_i} c_i \Rightarrow \tau \cdot X(n_i) \]

\[ + \sum_{d_j : D_j} c_j \Rightarrow a_j \sum_{e_j : E_j} f_j : X(n_j) \]

Two summands \( i, j \) commute if

\[ \forall g, d_i, d_j, e_j : (c_i(g, d_i) \land c_j(g, d_j)) \rightarrow \]

\[ \begin{align*}
& c_j(n_i(g, d_i), d_j) \land c_i(n_j(g, d_j, e_j), d_i) \\
& \land a_j(g, d_j) = a_j(n_i(g, d_i), d_j) \\
& \land n_j(n_i(g, d_i), d_j, e_j) = n_i(n_j(g, d_j, e_j), d_i) \\
& \land f_j(g, d_j, e_j) = f_j(n_i(g, d_i), d_j, e_j)
\end{align*} \]

**Theorem**

If summand \( i \) commutes with all summands \( j \), then it generates a strong probabilistic confluent set of \( \tau \)-steps.
PINS interface in LTSmin

- PINS provides a Partitioned Next-State Interface
- Decouples specification languages from backend algorithms
Overview of LTSmin Functionality

LTSmin backend algorithms

- **distributed** (cluster/Grid): generation + minimization
- **multicore** (shared memory): generation + LTL model checking
- **symbolic** (BDD/MDD-based): generation + mu-calculus
Overview of LTSmin Functionality

LTSmin backend algorithms

- **distributed** (cluster/Grid): generation + minimization
- **multicore** (shared memory): generation + LTL model checking
- **symbolic** (BDD/MDD-based): generation + mu-calculus

**NEW**: multicore symbolic operations ............... PDMC 2012
Overview of LTSmin Functionality

**LTSmin backend algorithms**
- **distributed** (cluster/Grid): generation + minimization
- **multicore** (shared memory): generation + LTL model checking
- **symbolic** (BDD/MDD-based): generation + mu-calculus

**NEW:** multicore symbolic operations ............... PDMC 2012

**LTSmin language modules**
- **mCRL2**: action based process algebra
- **Promela**: state based
- **UPPAAL**: timed automata, with zones (external DBM)
**SCOOP provides tool support for MAPA**

- Stand-alone version, and web-based interface
- Available at fmt.cs.utwente.nl/~timmer/scoop
- Supports linearization, optimization, state space generation
- Export facilities to model checkers (PRISM, CDP, IMCA)
- Programmed in Haskell
Specification:

\[ X = \text{tau}.X[] ++ <5>.X[] \]

init X

Constants (name = value):

+ 

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

- MAPA mode
  - Show MLPPE (use MAPA syntax)
  - Do not apply the maximal progress reduction
  - Apply maximal progress reduction

- Show statespace in AUT format
- Show statespace as the actual state
- Show the number of states and transitions
- Show verbose output
X =

(T => tau . X[])

Initial state: X

Powered by puptol
Analysis of Generalized Stochastic Petri Nets (GSPN)

GSPN (PNML)

reach P1 = 1 & P5 = 2
Analysis of Generalized Stochastic Petri Nets (GSPN)

**GSPN (PNML)**

reach \( P_1 = 1 \) & \( P_5 = 2 \)

**GEMMA**

\[
\]
\[
P2 \geq 1 \Rightarrow T2 .
\]
\[
\text{GSPN[P2--, P4++]} +
\]
\[
P5 \geq 1 \Rightarrow (4.0) .
\]
\[
\text{GSPN[P2++, P5--]} +
\]
\[
\text{...}
\]
\[
\text{init GSPN(1,1,1,0,1)}
\]

reach \( P_1 = 1 \) & \( P_5 = 2 \)

**MAPA**
Analysis of Generalized Stochastic Petri Nets (GSPN)

GSPN (PNML)

reach P1 = 1 & P5 = 2

GEMMA

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

MAPA

SCOOOP

#GOALS S4

MA
Analysis of Generalized Stochastic Petri Nets (GSPN)

GSPN (PNML)

reach P1 = 1 & P5 = 2


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

MAPA

GEMMA

SCOOOP (optimised)

MA

#GOALS S2
Analysis of Generalized Stochastic Petri Nets (GSPN)

GSPN (PNML)

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

GEMMA

reach P1 = 1 & P5 = 2

MAPA

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

Results

#GOALS S2

SCOOPE (optimised)

#GOALS S2

IMCA

MA

UNIVERSITY OF TWENTE.
Symbolic Manipulation of Markov Automata
September 8, 2012
24 / 26
Conclusion and Future Work

Scalable model checking for quantitative models

<table>
<thead>
<tr>
<th>feature</th>
<th>Process Algebra</th>
<th>Linear Format</th>
<th>Static Analysis</th>
<th>Confluence Reduction</th>
<th>High-Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>μCRL</td>
<td>LPE</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>probability</td>
<td>prCRL</td>
<td>LPPE</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Markovian</td>
<td>MAPA</td>
<td>MLPPE</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>
### Conclusion and Future Work

#### Scalable model checking for quantitative models

<table>
<thead>
<tr>
<th>feature</th>
<th>Process Algebra</th>
<th>Linear Format</th>
<th>Static Analysis</th>
<th>Confluence Reduction</th>
<th>High-Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>$\mu$CRL</td>
<td>LPE</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>probability</td>
<td>prCRL</td>
<td>LPPE</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Markovian</td>
<td>MAPA</td>
<td>MLPPE</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>

All **MA** results also apply to **LTS**, **DTMC**, **CTMC**, **IMC**, **MDP**, **PA**
Conclusion and Future Work

Scalable model checking for quantitative models

<table>
<thead>
<tr>
<th>feature</th>
<th>Process Algebra</th>
<th>Linear Format</th>
<th>Static Analysis</th>
<th>Confluence Reduction</th>
<th>High-Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>(\mu\text{CRL})</td>
<td>LPE</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>probability</td>
<td>prCRL</td>
<td>LPPE</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Markovian</td>
<td>MAPA</td>
<td>MLPPE</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>

All MA results also apply to LTS, DTMC, CTMC, IMC, MDP, PA

Future Work:
- Generalise confluence reduction to MAs and MAPA
- Link SCOOP to LTSmin’s symbolic and multi-core algorithms
- Performance analysis of LTSmin’s multi-core data structures
Questions? Read on!

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

- Mark Timmer, **SCOOP: A Tool for Symbolic Optimisations Of Probabilistic Processes** ........................................ QEST 2011
- Mark Timmer, Mariëlle Stoelinga & Jaco van de Pol, **Confluence Reduction for Probabilistic Systems** ............... TACAS 2011
- J-P Katoen, Jaco van de Pol, Mariëlle Stoelinga & Mark Timmer, **A linear process-algebraic format with data for probabilistic automata**, ................................................ Theoretical Computer Science, 413(1), 2012
- Mark Timmer, J-P Katoen, Jaco van de Pol & Mariëlle Stoelinga, **Efficient Modelling and Generation of Markov Automata** . CONCUR 2012
Questions? Read on!

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

- Mark Timmer, Mariëlle Stoelinga & Jaco van de Pol, Confluence Reduction for Probabilistic Systems ........... TACAS 2011
- J-P Katoen, Jaco van de Pol, Mariëlle Stoelinga & Mark Timmer, A linear process-algebraic format with data for probabilistic automata, ................................ Theoretical Computer Science, 413(1), 2012
- Mark Timmer, J-P Katoen, Jaco van de Pol & Mariëlle Stoelinga, Efficient Modelling and Generation of Markov Automata .CONCUR 2012

Have another look at fmt.cs.utwente.nl/tools/ltsmin/

- Stefan Blom, Jaco van de Pol & Michael Weber, LTSmin: Distributed and Symbolic Reachability ............... CAV 2010
- Alfons Laarman, Jaco van de Pol & Michael Weber, Multi-Core LTSmin: Marrying Modularity and Scalability ..... NFM 2011
- Tom van Dijk, Alfons Laarman & Jaco van de Pol, Multi-core BDD Operations for Symbolic Reachability ...... PDMC 2012