UNIVERSITY OF TWENTE.

Formal Methods & Tools.

Symbolic reductions of probabilistic models using linear process equations

Mark Timmer

September 16, 2010

Oxford ComLab 2010

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

Contents

Table of Contents

The context: probabilistic model checking

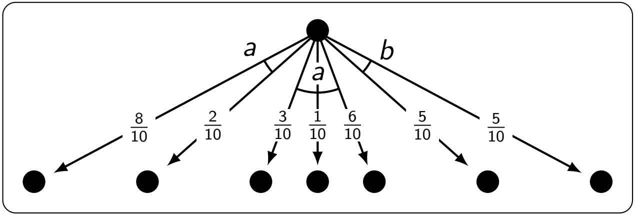
Probabilistic model checking:

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

The context: probabilistic model checking

Probabilistic model checking:

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

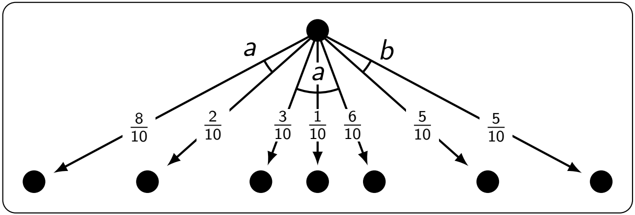


- Non-deterministically choose one of the three transitions
- Probabilistically choose the next state

The context: probabilistic model checking

Probabilistic model checking:

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

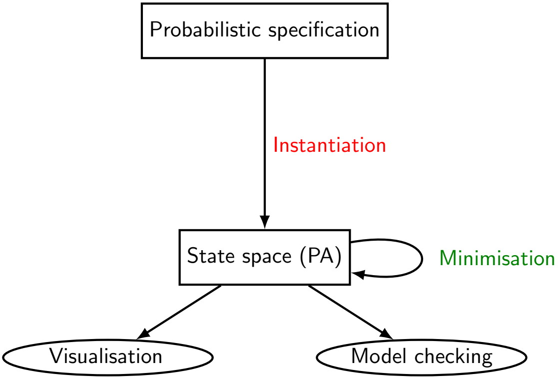


- Non-deterministically choose one of the three transitions
- Probabilistically choose the next state

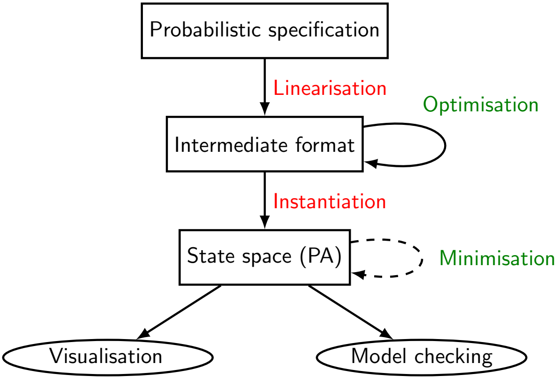
Limitations of previous approaches:

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

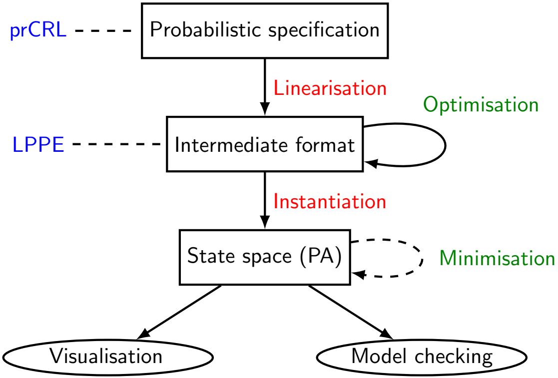
Overview of our approach



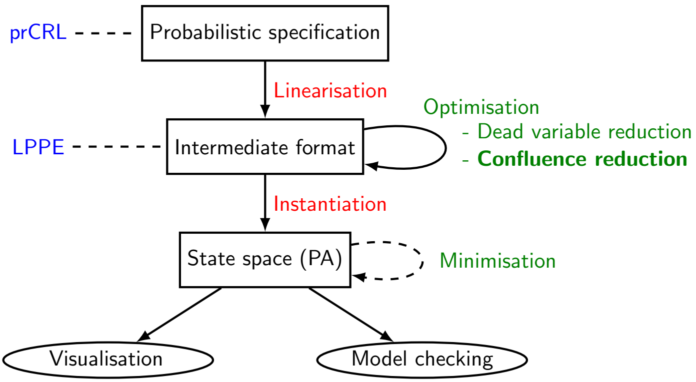
Overview of our approach



Overview of our approach

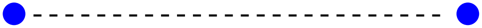


Overview of our approach



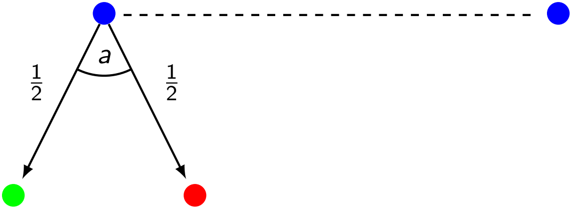
Equivalences: probabilistic bisimulation

Notions of equivalence: strong/branching probabilistic bisimulation



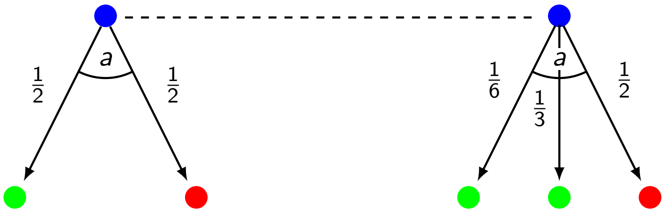
Equivalences: probabilistic bisimulation

Notions of equivalence: strong/branching probabilistic bisimulation



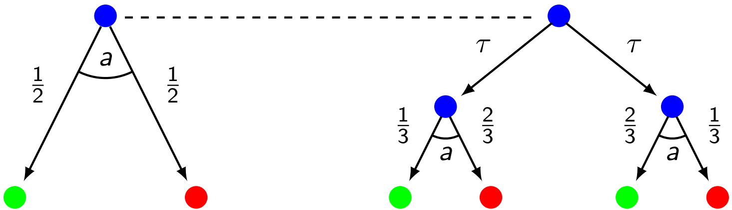
Equivalences: probabilistic bisimulation

Notions of equivalence: strong/branching probabilistic bisimulation



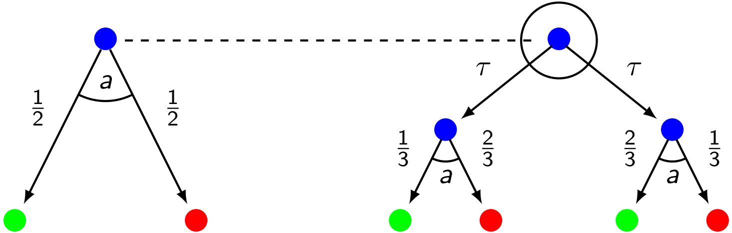
Equivalences: probabilistic bisimulation

Notions of equivalence: strong/branching probabilistic bisimulation



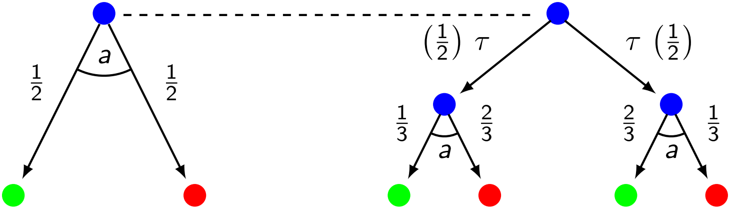
Equivalences: probabilistic bisimulation

Notions of equivalence: strong/branching probabilistic bisimulation



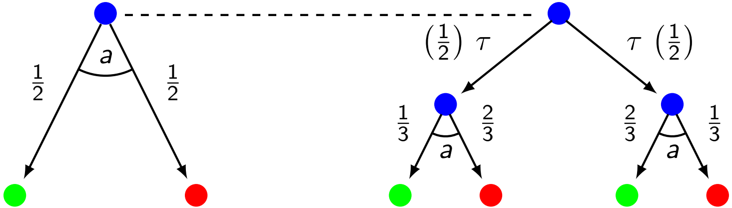
Equivalences: probabilistic bisimulation

Notions of equivalence: strong/branching probabilistic bisimulation



Equivalences: probabilistic bisimulation

Notions of equivalence: strong/branching probabilistic bisimulation



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

Table of Contents

A process algebra with data and probability: prCRL

Specification language prCRL:

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

A process algebra with data and probability: prCRL

Specification language prCRL:

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

The grammar of prCRL process terms

Process terms in prCRL are obtained by the following grammar:

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

Process equations and processes

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

An example specification

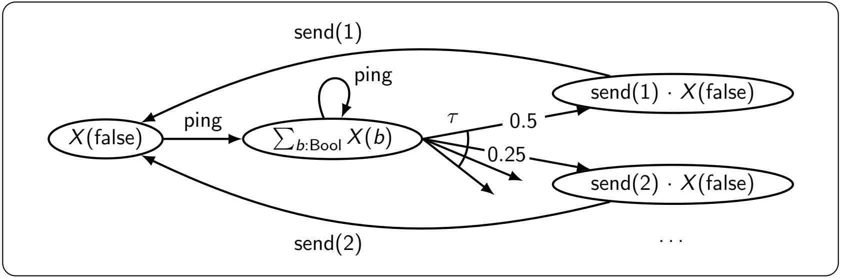
Sending an arbitrary natural number

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

An example specification

Sending an arbitrary natural number

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



Composability using extended prCRL

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

Composability using extended prCRL

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

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

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

Composability using extended prCRL

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

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

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

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

Composability using extended prCRL

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

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

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

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

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

Composability using extended prCRL

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

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

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

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

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

Composability using extended prCRL

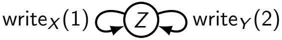
For composability we introduce extended prCRL. It extends prCRL by parallel composition, encapsulation, hiding and renaming.

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

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

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

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



Composability using extended prCRL

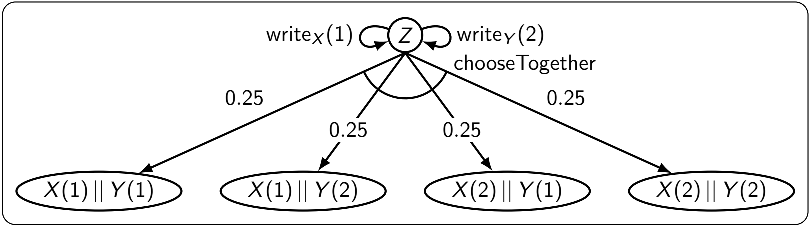
For composability we introduce extended prCRL. It extends prCRL by parallel composition, encapsulation, hiding and renaming.

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

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

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

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



A linear format for prCRL: the LPPE

LPPEs are a subset of prCRL specifications:

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

A linear format for prCRL: the LPPE

LPPEs are a subset of prCRL specifications:

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

Advantages of using LPPEs instead of prCRL specifications:

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

A linear format for prCRL: the LPPE

LPPEs are a subset of prCRL specifications:

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

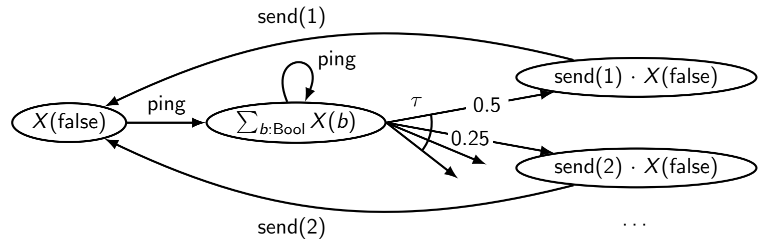
Advantages of using LPPEs instead of prCRL specifications:

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

Theorem

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

Linear Probabilistic Process Equations – an example



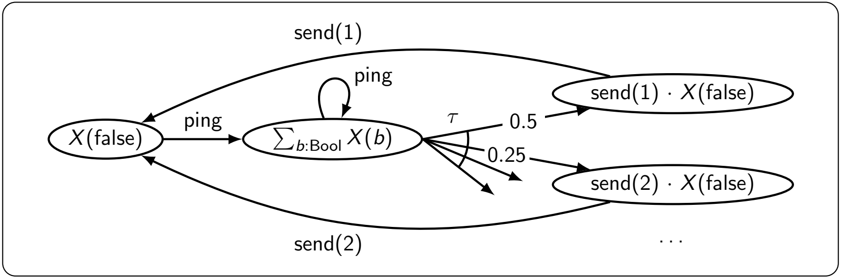
Specification in prCRL

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

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

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

Linear Probabilistic Process Equations – an example



Specification in prCRL

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

Specification in LPPE

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

Table of Contents

Linearisation: a simple example without data

Consider the following prCRL specification:

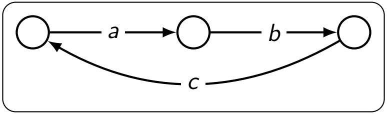
$$X = a \cdot b \cdot c \cdot X$$

Linearisation: a simple example without data

Consider the following prCRL specification:

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

The control flow of X is given by:

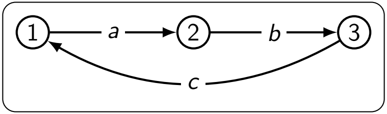


Linearisation: a simple example without data

Consider the following prCRL specification:

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

The control flow of X is given by:

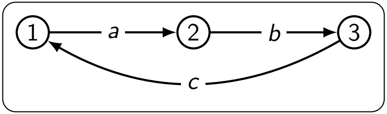


Linearisation: a simple example without data

Consider the following prCRL specification:

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

The control flow of X is given by:



The corresponding LPPE (initialised with pc = 1):

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

Linearisation: a more complicated example with data

Consider the following prCRL specification:

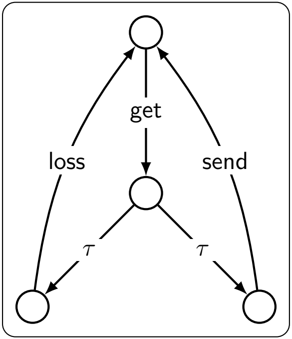
$$X = \sum_{d:D} \mathrm{get}(d) \cdot (\tau \cdot \mathrm{loss} \cdot X + \tau \cdot \mathrm{send}(d) \cdot X)$$

Linearisation: a more complicated example with data

Consider the following prCRL specification:

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

Control flow:

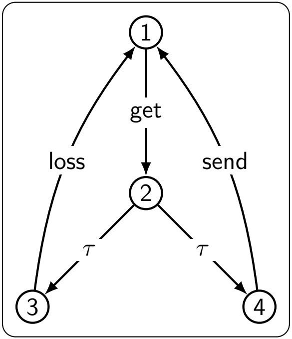


Linearisation: a more complicated example with data

Consider the following prCRL specification:

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

Control flow:

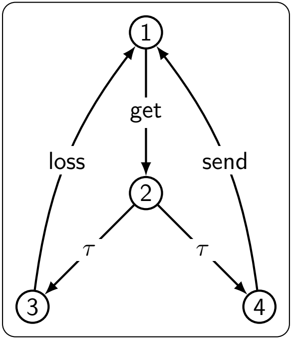


Linearisation: a more complicated example with data

Consider the following prCRL specification:

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

Control flow:



LPPE:

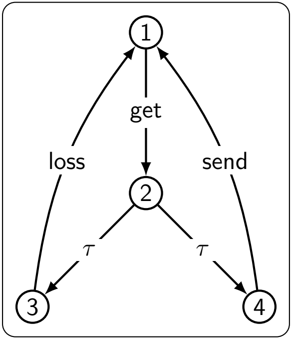
$$
\begin{aligned}
Y(pc\colon \{1,2,3,4\}, x\colon D) = \\
& \textstyle\sum_{d:D} \quad pc = 1 \Rightarrow \text{get}(d) \cdot Y(2, d) \\
+ & \quad pc = 2 \Rightarrow \tau \cdot Y(3, x) \\
+ & \quad pc = 2 \Rightarrow \tau \cdot Y(4, x) \\
+ & \quad pc = 3 \Rightarrow \text{loss} \cdot Y(1, x) \\
+ &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

Linearisation: a more complicated example with data

Consider the following prCRL specification:

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

Control flow:



LPPE:

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

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

Linearisation: a more algorithmic approach

Consider the following prCRL specification:

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

Linearisation: a more algorithmic approach

Consider the following prCRL specification:

$$X(d:D) = \sum_{e:D} a(d+e) \sum\hspace{-0.9em}\bullet_{f:D} \frac{1}{|D|} : \left(c(e) \cdot c(f) \cdot X(5) + c(e+f) \cdot X(5) \right)$$

---

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

$\sum_{e:D} a(d+e) \sum\hspace{-0.9em}\bullet_{f:D} \frac{1}{|D|} : \ (c(e) \cdot c(f) \cdot X(5) + c(e+f) \cdot X(5))$

Linearisation: a more algorithmic approach

Consider the following prCRL specification:

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

---

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

Linearisation: a more algorithmic approach

Consider the following prCRL specification:

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

---

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

---

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

Linearisation: a more algorithmic approach

Consider the following prCRL specification:

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

---

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

---

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

Linearisation: a more algorithmic approach

Consider the following prCRL specification:

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

---

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

---

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

Linearisation: a more algorithmic approach

Consider the following prCRL specification:

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

---

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

---

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

---

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

Linearisation: a more algorithmic approach

Consider the following prCRL specification:

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

---

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

---

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

---

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

Linearisation: a more algorithmic approach

Consider the following prCRL specification:

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

---

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

---

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

---

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

Linearisation: a more algorithmic approach

Consider the following prCRL specification:

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

---

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

---

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

---

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

---

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

Linearisation: a more algorithmic approach

Consider the following prCRL specification:

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

---

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

---

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

---

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

---

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

Linearisation: a more algorithmic approach

Consider the following prCRL specification:

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

---

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

---

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

---

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

---

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

Linearisation: a more algorithmic approach

Consider the following prCRL specification:

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

---

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

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

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

Linearisation: a more algorithmic approach

Consider the following prCRL specification:

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

---

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

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

Linearisation

In general, we always linearise in two steps:

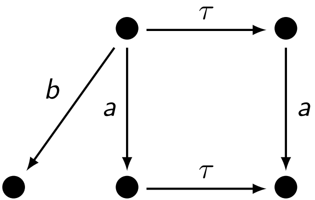
1. Transform the specification to intermediate regular form (IRF) (every process is a summation of single-action terms)
2. Merge all processes into one big process by introducing a program counter

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

Table of Contents

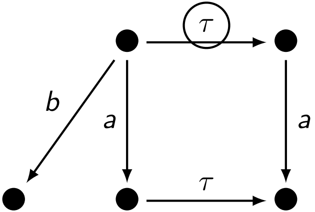
Branching bisimulation preservation by τ-steps

Unobservable τ-steps might disable behaviour...



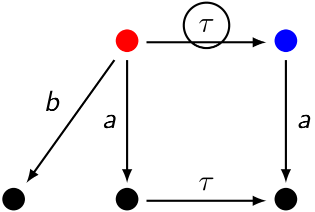
Branching bisimulation preservation by τ-steps

Unobservable τ-steps might disable behaviour. . .



Branching bisimulation preservation by τ-steps

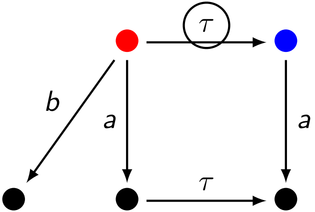
Unobservable τ-steps might disable behaviour. . .



Branching bisimulation preservation by τ-steps

Unobservable τ-steps might disable behaviour...

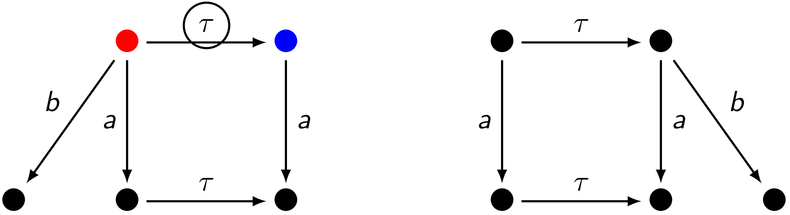
...though often, they connect branching bisimilar states



Branching bisimulation preservation by τ-steps

Unobservable τ-steps might disable behaviour...

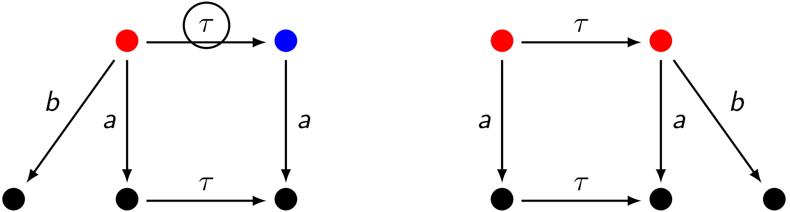
...though often, they connect branching bisimilar states



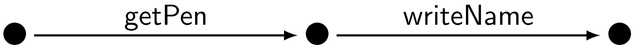
Branching bisimulation preservation by τ-steps

Unobservable τ-steps might disable behaviour...

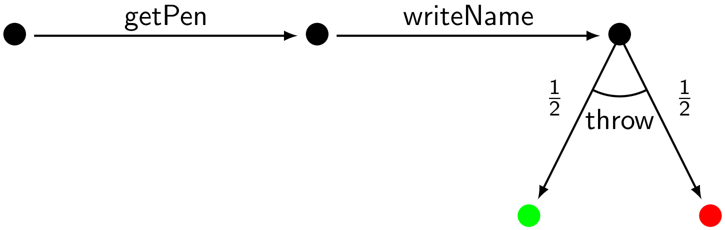
...though often, they connect branching bisimilar states



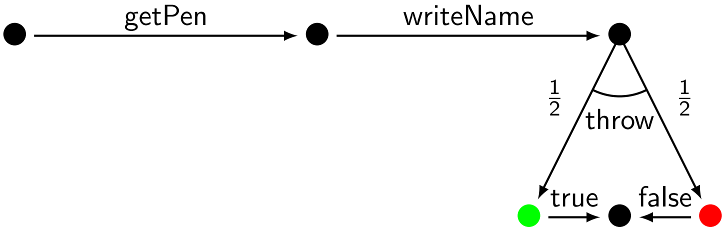
Confluence: an introductory example



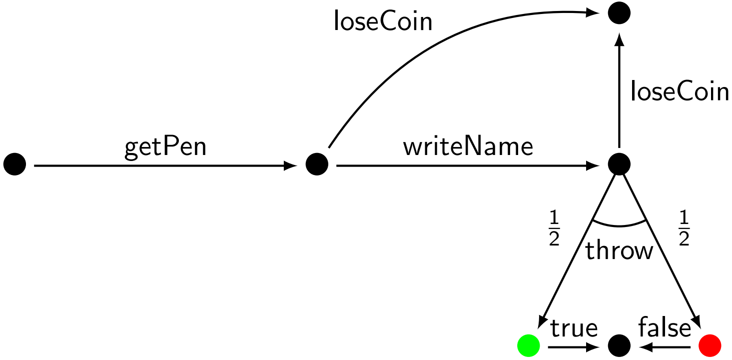
Confluence: an introductory example



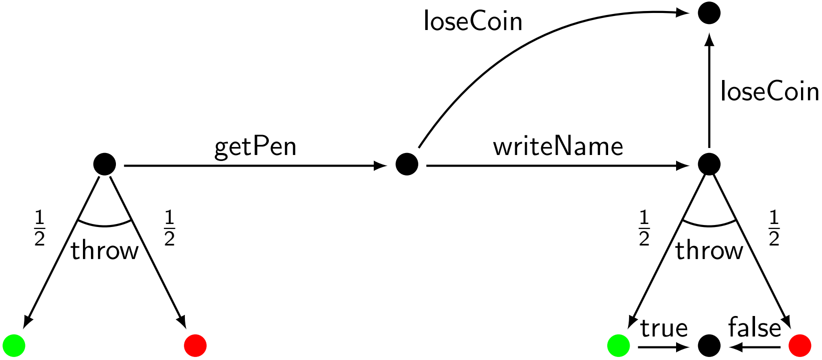
Confluence: an introductory example



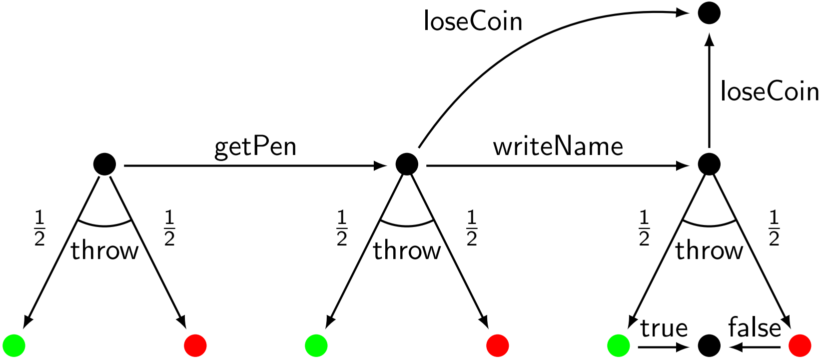
Confluence: an introductory example



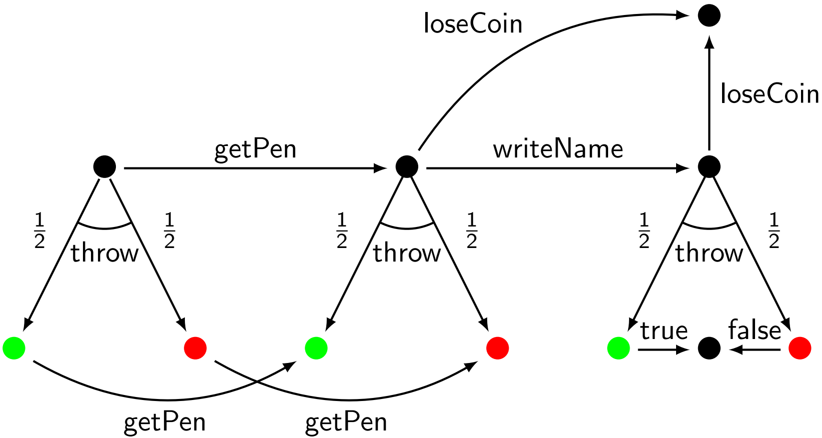
Confluence: an introductory example



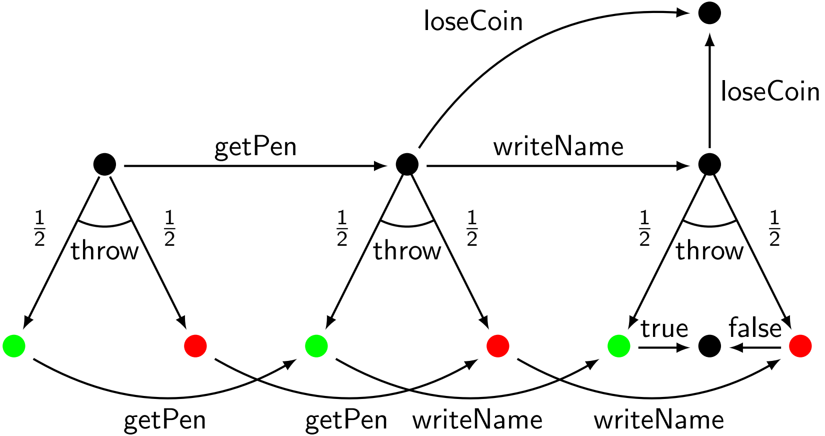
Confluence: an introductory example



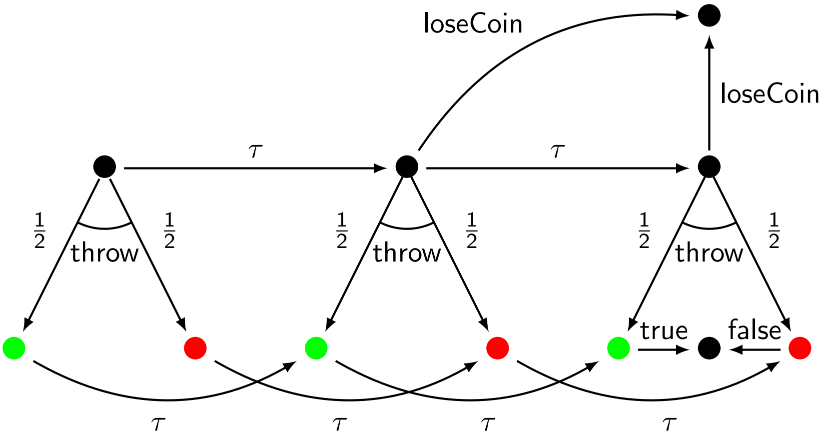
Confluence: an introductory example



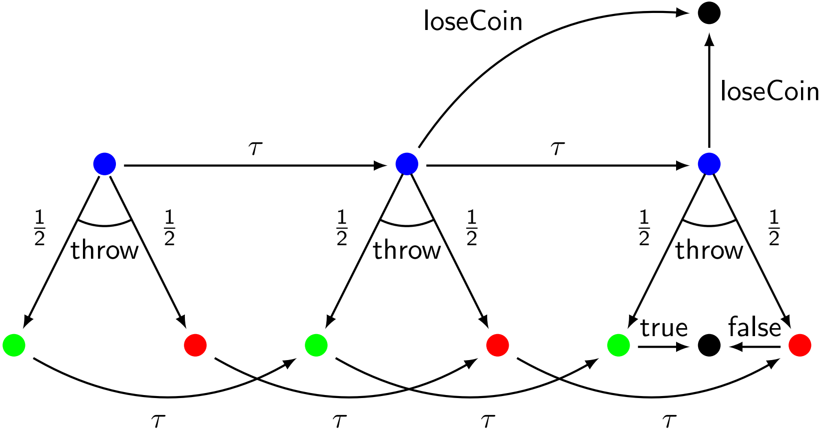
Confluence: an introductory example



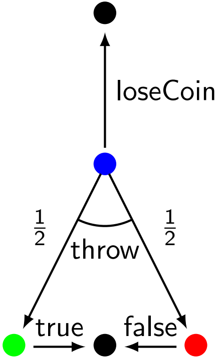
Confluence: an introductory example



Confluence: an introductory example



Confluence: an introductory example



Confluence: non-probabilistic versus probabilistic

Three notions of confluence:

- weak confluence
- confluence
- strong confluence

Confluence: non-probabilistic versus probabilistic

Three notions of confluence:

• weak confluence		• weak probabilistic confluence
• confluence	$\Rightarrow$	• probabilistic confluence
• strong confluence		• strong probabilistic confluence

Confluence: non-probabilistic versus probabilistic

Three notions of confluence:

• weak confluence		• weak probabilistic confluence
• confluence	$\Rightarrow$	• probabilistic confluence
• strong confluence		• strong probabilistic confluence

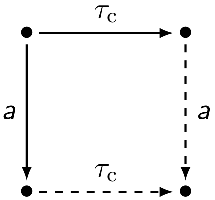
Confluence: non-probabilistic versus probabilistic

Three notions of confluence:

- weak confluence
- confluence
- strong confluence

$\Rightarrow$

- weak probabilistic confluence
- probabilistic confluence
- strong probabilistic confluence



Strong confluence

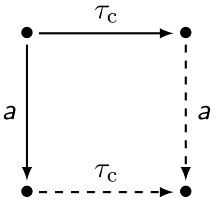
Confluence: non-probabilistic versus probabilistic

Three notions of confluence:

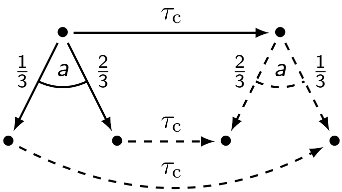
- weak confluence
- confluence
- strong confluence

$\Rightarrow$

- weak probabilistic confluence
- probabilistic confluence
- strong probabilistic confluence

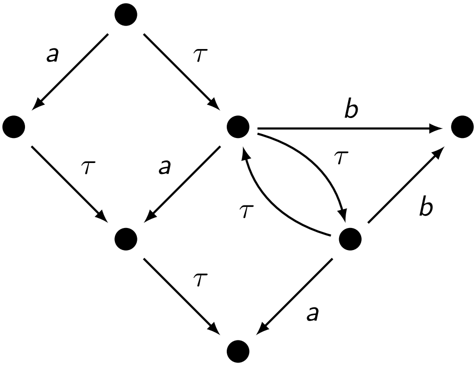


Strong confluence

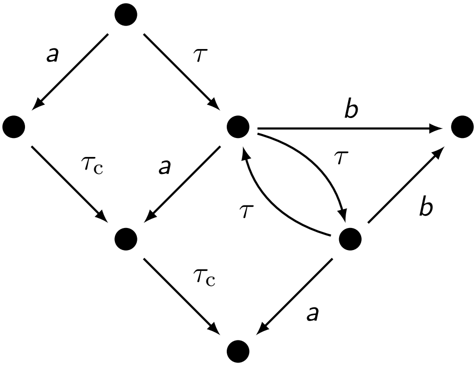


Strong probabilistic confluence

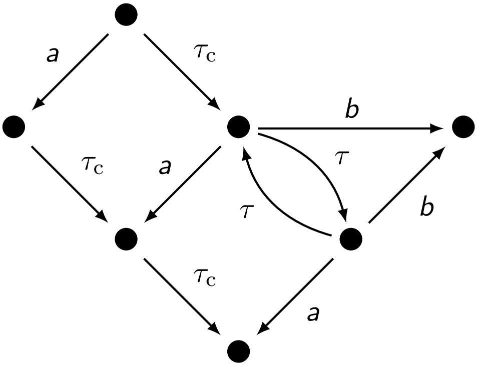
State space reduction using confluence



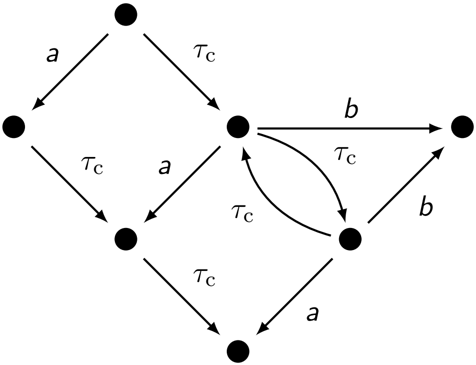
State space reduction using confluence



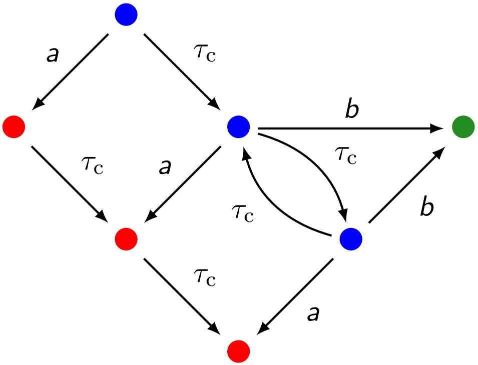
State space reduction using confluence



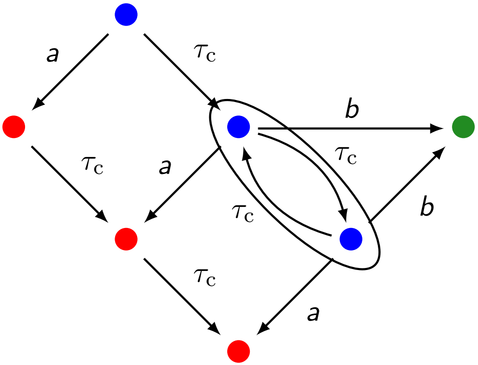
State space reduction using confluence



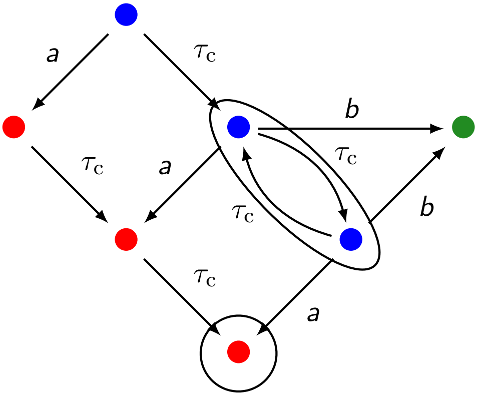
State space reduction using confluence



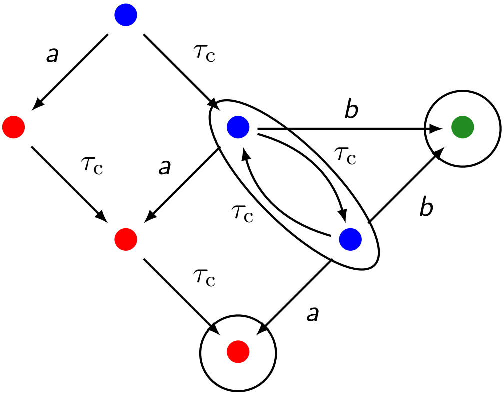
State space reduction using confluence



State space reduction using confluence



State space reduction using confluence



State space reduction using confluence

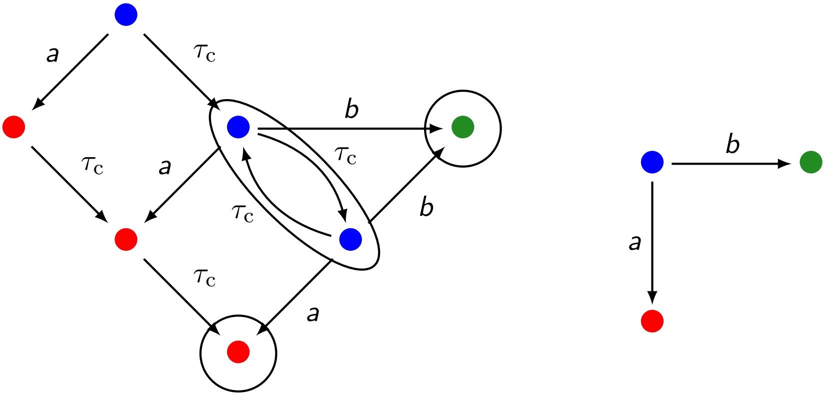


Table of Contents

Detecting confluence: LPPEs

Example specification

$$X(\mathsf{pc} : \{1..2\}, \mathsf{active} : \mathsf{Bool}) =$$
$$\sum_{n:\{1,2,3\}} \mathsf{pc} = 1 \quad \Rightarrow \mathsf{output}(n) \sum_{\mathsf{b}:\mathsf{Bool}}\bullet \tfrac{1}{2} : X(2, \mathsf{b})$$
$$+ \quad \mathsf{pc} = 2 \land \mathsf{active} \Rightarrow \mathsf{beep} \cdot X(1, \mathsf{active})$$

Detecting confluence: LPPEs

Example specification

$$X(\mathrm{pc} : \{1..2\}, \mathrm{active} : \mathrm{Bool}) =$$

$$\sum_{n:\{1,2,3\}} \mathrm{pc} = 1 \quad \Rightarrow \mathrm{output}(n) \sum\!\!\bullet_{\mathrm{b:Bool}} \tfrac{1}{2} : X(2, \mathrm{b})$$

$$+ \quad \mathrm{pc} = 2 \wedge \mathrm{active} \Rightarrow \quad \tau \quad \cdot X(1, \mathrm{active})$$

Detecting confluence: LPPEs

Example specification

$$
\begin{aligned}
&X(\mathrm{pc} : \{1..2\}, \mathrm{active} : \mathrm{Bool}) = \\
&\qquad \sum_{n:\{1,2,3\}} \mathrm{pc} = 1 \qquad \Rightarrow \mathrm{output}(n) \sum_{\mathrm{b:Bool}}\!\!\bullet\ \tfrac{1}{2} : X(2, \mathrm{b}) \\
&\quad + \qquad \mathrm{pc} = 2 \wedge \mathrm{active} \Rightarrow \quad \tau \quad \cdot X(1, \mathrm{active})
\end{aligned}
$$

How to know whether a summand is confluent?

Detecting confluence: LPPEs

Example specification

$$
\begin{aligned}
X(\mathrm{pc} : \{1..2\}, \mathrm{active} : \mathrm{Bool}) = \\
\sum_{n:\{1,2,3\}} \mathrm{pc} = 1 \quad & \Rightarrow \mathrm{output}(n) \sum_{\mathrm{b:Bool}} \tfrac{1}{2} : X(2, \mathrm{b}) \\
+ \quad \mathrm{pc} = 2 \land \mathrm{active} \Rightarrow \quad & \tau \cdot X(1, \mathrm{active})
\end{aligned}
$$

How to know whether a summand is confluent?

- Its action should be τ

Detecting confluence: LPPEs

Example specification

$$X(\text{pc} : \{1..2\}, \text{active} : \text{Bool}) =$$

$$\sum_{n:\{1,2,3\}} \text{pc} = 1 \Rightarrow \text{output}(n) \sum_{\text{b:Bool}} \tfrac{1}{2} : X(2, \text{b})$$

$$+ \quad \text{pc} = 2 \wedge \text{active} \Rightarrow \tau \cdot X(1, \text{active})$$

How to know whether a summand is confluent?

- Its action should be τ
- Its next state should be chosen nonprobabilistically

Detecting confluence: LPPEs

Example specification

$$X(\text{pc} : \{1..2\}, \text{active} : \text{Bool}) =$$
$$\sum_{n:\{1,2,3\}} \text{pc} = 1 \Rightarrow \text{output}(n) \sum_{\text{b:Bool}} \tfrac{1}{2} : X(2, \text{b})$$
$$+ \quad \text{pc} = 2 \land \text{active} \Rightarrow \tau \cdot X(1, \text{active})$$

How to know whether a summand is confluent?

- Its action should be τ
- Its next state should be chosen nonprobabilistically
- It should commute with all the other summands

Detecting confluence: LPPEs

Example specification

$$
\begin{aligned}
& X(\text{pc} : \{1..2\}, \text{active} : \text{Bool}) = \\
& \quad \sum_{n:\{1,2,3\}} \text{pc} = 1 \Rightarrow \text{output}(n) \sum_{\text{b:Bool}} \tfrac{1}{2} : X(2, \text{b}) \\
& + \quad \text{pc} = 2 \wedge \text{active} \Rightarrow \tau \cdot X(1, \text{active})
\end{aligned}
$$

How to know whether a summand is confluent?

- Its action should be τ
- Its next state should be chosen nonprobabilistically
- It should commute with all the other summands
  - Never enabled at the same time
  - Not touching the same variables

Table of Contents

Case study: a leader election protocol

Case study

Leader election protocol à la Itai-Rodeh

- Two processes throw a die
  - *The process with the highest number will be leader*
  - *In case of a tie: throw again*

Case study: a leader election protocol

Case study

Leader election protocol à la Itai-Rodeh

- Two processes throw a die
  - *The process with the highest number will be leader*
  - *In case of a tie: throw again*
- More precisely:
  - *Passive thread: receive value of opponent*
  - *Active thread: roll, send, compare (or block)*

A prCRL model of the leader election protocol

$$
\begin{aligned}
& P(id : \{one, two\}, val : Die, set : Bool) = \\
& \qquad set = false \Rightarrow \sum_{d:Die} \mathit{communicate}(id, other(id), d) \cdot P(id, d, true) \\
& \quad + set = true \Rightarrow \mathit{checkValue}(val) \cdot P(id, val, false) \\
& A(id : \{one, two\}) = \\
& \quad roll(id) \sum_{d:Die} \frac{1}{6} : \overline{\mathit{communicate}}(other(id), id, d) \cdot \sum_{e:Die} \overline{\mathit{checkValue}}(e) \cdot \\
& \qquad \big(\, (d = e \Rightarrow A(id)) \\
& \qquad + (d > e \Rightarrow leader(id) \cdot A(id)) \\
& \qquad + (e > d \Rightarrow follower(id) \cdot A(id))\big) \\
& C(id : \{one, two\}) = P(id, 1, false) \mid\mid A(id) \\
& S = C(one) \mid\mid C(two)
\end{aligned}
$$

Strong bisimulation preserving reductions

In order to obtain reductions, first linearise:

$$\sum_{e21:Die} pc21 = 3 \land pc11 = 1 \land set11 \land val11 = e21 \Rightarrow$$
$$checkValue(val11) \sum_{(k1,k2):\{*\}\times\{*\}}\!\!\!\!\bullet\quad multiply(1.0, 1.0):$$
$$Z(1, id11, val11, false, 1, 4, id21, d21, e21,$$
$$pc12, id12, val12, set12, d12, pc22, id22, d22, e22)$$

Strong bisimulation preserving reductions

In order to obtain reductions, first linearise:

$$\sum_{e21:Die} pc21 = 3 \wedge pc11 = 1 \wedge set11 \wedge val11 = e21 \Rightarrow$$
$$checkValue(val11) \sum_{(k1,k2):\{*\}\times\{*\}}\!\!\!\bullet\; multiply(1.0, 1.0):$$
$$Z(1, id11, val11, false, 1, 4, id21, d21, e21,$$
$$pc12, id12, val12, set12, d12, pc22, id22, d22, e22)$$

Before any reductions:

- 18 parameters
- 14 summands
- 3763 states
- 6158 transitions

Strong bisimulation preserving reductions

In order to obtain reductions, first linearise:

$$pc21 = 3 \wedge \quad set11 \quad \Rightarrow$$

$$checkValue(val11) \sum_{(k1,k2):\{*\}\times\{*\}} 1.0 :$$

$$Z(\quad val11, false, \quad 4, \quad d21, val11,$$
$$val12, set12, \quad pc22, \quad d22, e22)$$

Before any reductions:

- 18 parameters
- 14 summands
- 3763 states
- 6158 transitions

After some reductions:

- 10 parameters
- 12 summands

Strong bisimulation preserving reductions

In order to obtain reductions, first linearise:

$$pc21 = 3 \wedge \quad set11 \quad \Rightarrow$$
$$checkValue(val11) \quad \sum_{(k1,k2):\{*\}\times\{*\}} \bullet \quad 1.0 :$$
$$Z(\quad 1 \quad , false, \quad 4, \quad d21, val11,$$
$$val12, set12, \quad pc22, \quad d22, e22)$$

Before any reductions:

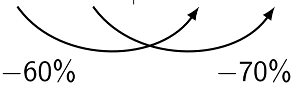
- 18 parameters
- 14 summands
- 3763 states
- 6158 transitions

After some reductions:

- 10 parameters
- 12 summands
- 1693 states (-55%)
- 2438 transitions (-60%)

Branching bisimulation preserving reductions

	Original		Reduced		Running time	
Specification	States	Trans.	States	Trans.	Before	After
`leader`	3763	6158	1399	1922	1.86 sec	0.72 sec
`leaderReduced`	1693	2438	589	722	0.90 sec	0.44 sec
`leader-2-2`	67	94	27	32	0.04 sec	0.65 sec
`leader-2-6`	535	710	199	212	0.36 sec	0.81 sec
`leader-2-36`	18325	23690	6589	6662	516.23 sec	43.11 sec
`leader-3-2`	1018	1815	376	561	1.61 sec	3.81 sec
`leader-3-6`	21664	36519	7936	10233	221.22 sec	44.92 sec



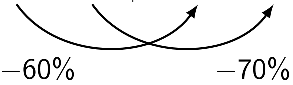
Branching bisimulation preserving reductions

Specification	Original States	Original Trans.	Reduced States	Reduced Trans.	Running time Before	Running time After
`leader`	3763	6158	1399	1922	1.86 sec	0.72 sec
`leaderReduced`	1693	2438	589	722	0.90 sec	0.44 sec
`leader-2-2`	67	94	27	32	0.04 sec	0.65 sec
`leader-2-6`	535	710	199	212	0.36 sec	0.81 sec
`leader-2-36`	18325	23690	6589	6662	516.23 sec	43.11 sec
`leader-3-2`	1018	1815	376	561	1.61 sec	3.81 sec
`leader-3-6`	21664	36519	7936	10233	221.22 sec	44.92 sec

−60% −70%

Branching bisimulation preserving reductions

Specification	Original		Reduced		Running time	
	States	Trans.	States	Trans.	Before	After
`leader`	**3763**	6158	1399	1922	1.86 sec	0.72 sec
`leaderReduced`	1693	2438	589	722	0.90 sec	0.44 sec
`leader-2-2`	67	94	27	32	0.04 sec	0.65 sec
`leader-2-6`	535	710	199	212	0.36 sec	0.81 sec
`leader-2-36`	18325	23690	6589	6662	516.23 sec	43.11 sec
`leader-3-2`	1018	1815	376	561	1.61 sec	3.81 sec
`leader-3-6`	21664	36519	7936	10233	221.22 sec	44.92 sec



Branching bisimulation preserving reductions

	Original		Reduced		Running time	
Specification	States	Trans.	States	Trans.	Before	After
`leader`	**3763**	6158	1399	1922	1.86 sec	0.72 sec
`leaderReduced`	**1693**	2438	589	722	0.90 sec	0.44 sec
`leader-2-2`	67	94	27	32	0.04 sec	0.65 sec
`leader-2-6`	535	710	199	212	0.36 sec	0.81 sec
`leader-2-36`	18325	23690	6589	6662	516.23 sec	43.11 sec
`leader-3-2`	1018	1815	376	561	1.61 sec	3.81 sec
`leader-3-6`	21664	36519	7936	10233	221.22 sec	44.92 sec

−60% −70%

Branching bisimulation preserving reductions

	Original		Reduced		Running time	
Specification	States	Trans.	States	Trans.	Before	After
`leader`	**3763**	6158	1399	1922	1.86 sec	0.72 sec
`leaderReduced`	**1693**	2438	**589**	722	0.90 sec	0.44 sec
`leader-2-2`	67	94	27	32	0.04 sec	0.65 sec
`leader-2-6`	535	710	199	212	0.36 sec	0.81 sec
`leader-2-36`	18325	23690	6589	6662	516.23 sec	43.11 sec
`leader-3-2`	1018	1815	376	561	1.61 sec	3.81 sec
`leader-3-6`	21664	36519	7936	10233	221.22 sec	44.92 sec

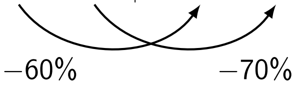


Table of Contents

Conclusions

Conclusions

- We developed the process algebra prCRL, incorporating both data and probability;
- We defined a normal form for prCRL, the LPPE; starting point for symbolic optimisations and easy state space generation;
- We provided a linearisation algorithm to transform prCRL specifications to LPPEs, proved it correct and implemented it;
- We developed three new notions of confluence for PAs that preserve branching probabilistic bisimulation, showed how they can be used for state space reduction, and discussed how to detect then based on an LPPE;
- We illustrated the power of our methods using a case study.

Questions

J. van de Pol and M. Timmer.
State space reduction of linear processes using control flow reconstruction.
In *Proc. of the 7th Int. Symp. on Automated Technology for Verification and Analysis (ATVA)*, volume 5799 of *LNCS*, pages 54–68, 2009.

J.-P. Katoen, J.C. van de Pol, M.I.A. Stoelinga, and M. Timmer.
A linear process-algebraic format for probabilistic systems with data.
In *Proc. of the 10th International Conference on Application of Concurrency to System Design (ACSD)*, pages 213–222. IEEE, 2010.

M. Timmer, M.I.A. Stoelinga, and J.C. van de Pol.
Confluence reduction for probabilistic systems.
In *submission*, 2011.

Questions?