Probabilistic Systems Semantics via Coalgebra

Ana Sokolova

TbiLLC’17 tutorial, part 1
Rigorous methods for engineering of and reasoning about reactive systems
My background

Computer Science

- Theoretical Computer Science
- Formal Methods
- Algebra and Coalgebra
- Security

Data Structures

Concurrency

Memory Management Systems

Real-Time Systems

My background

Ana Sokolova

TbiLLC’17, tutorial, part 1
In this tutorial:

probabilistic systems semantics using (co)algebra
In this tutorial:

probabilistic systems semantics
using (co)algebra
In this tutorial:

probabilistic systems semantics using (co)algebra
In this tutorial:

probabilistic systems semantics using (co)algebra

state-based automata-like models
relations on states
In this tutorial:

probabilistic systems semantics using (co)algebra

- state-based automata-like models
- behaviour semantics
- relations on states
Behaviour semantics are used for verification:

- Behaviour equivalence \( \approx \)

- Behaviour preorder \( \sqsubseteq \)
Behaviour semantics are used for verification:

- Behaviour equivalence $\approx$
- Behaviour preorder $\sqsubseteq$

To identify states with the same behaviour.
Behaviour semantics are used for verification:

- Behaviour equivalence $\approx$
- Behaviour preorder $\sqsubseteq$

(to identify states with the same behaviour)

(to order states according to behaviour)
Behaviour semantics are used for verification:

- Behaviour equivalence $\approx$
- Behaviour preorder $\sqsubseteq$

There are many of them: bisimilarity, trace, ...
Behaviour semantics are used for verification:

- Behaviour equivalence \(\approx\)
- Behaviour preorder \(\sqsubseteq\)

there are many of them: bisimilarity, trace, ...

Sys \(\approx\) Spec

to identify states with the same behaviour
to order states according to behaviour
Behaviour semantics are used for verification:

• **Behaviour equivalence** $\approx$

• **Behaviour preorder** $\sqsubseteq$

there are many of them: bisimilarity, trace, …

Sys $\approx$ Spec

Sys $\sqsubseteq$ Spec

to identify states with the same behaviour

to order states according to behaviour
Plan:

**Part 1.** Modelling probabilistic systems for branching-time semantics

**Part 2.** Traces, linear-time semantics

**Part 3.** Belief-state-transformer semantics via convexity
Plan:

**Part 1.** Modelling probabilistic systems for branching-time semantics

**Part 2.** Traces, linear-time semantics

**Part 3.** Belief-state-transformer semantics via convexity

bisimilarity
Plan:

**Part 1.** Modelling probabilistic systems for branching-time semantics

**Part 2.** Traces, linear-time semantics

**Part 3.** Belief-state-transformer semantics via convexity
Plan:

**Part 1.** Modelling probabilistic systems for branching-time semantics

**Part 2.** Traces, linear-time semantics

**Part 3.** Belief-state-transformer semantics via convexity

- Bisimilarity
- Trace equivalence
- Distribution bisimilarity
Plan:

**Part 1.** Modelling probabilistic systems for branching-time semantics

**Part 2.** Traces, linear-time semantics

**Part 3.** Belief-state-transformer semantics via convexity

all with help of coalgebra
Plan:

**Part 1.** Modelling probabilistic systems for branching-time semantics

**Part 2.** Traces, linear-time semantics

**Part 3.** Belief-state-transformer semantics via convexity

Mathematical framework based on category theory for state-based systems semantics

bisimilarity

distribution bisimilarity

trace equivalence

all with help of coalgebra
Highlights
Highlights

F. Bartels, E. de Vink, A. S. A hierarchy of probabilistic system types TCS’04, CMCS’03
A.S. Coalgebraic analysis of probabilistic systems PhD thesis, TU Eindhoven’05
A. S. Probabilistic systems coalgebraically TCS’11
B. Jacobs, I. Hasuo, A. S. Generic trace semantics via coinduction LMCS’07
B. Jacobs, I. Hasuo, A. S. The microcosm principle and concurrency in coalgebra FoSSaCS’08
A. Silva, A. S. Sound and complete axiomatisation of trace semantics for probabilistic systems MFPS’11
B. Jacobs, A. Silva, A. S. Trace semantics via determinization JSS’15
A. S., H. Woracek Congruences of convex algebras JPAA’15
A. S., H. Woracek Termination in convex sets of distributions CALCO’17
F. Bonchi, A. Silva, A. S. The power of convex algebras CONCUR’17
Highlights

F. Bartels, E. de Vink, A. S. A hierarchy of probabilistic system types TCS’04, CMCS’03
A. S. Coalgebraic analysis of probabilistic systems PhD thesis, TU Eindhoven’05
A. S. Probabilistic systems coalgebraically TCS’11
B. Jacobs, I. Hasuo, A. S. Generic trace semantics via coinduction LMCS’07
B. Jacobs, I. Hasuo, A. S. The microcosm principle and concurrency in coalgebra FoSSaCS’08
A. Silva, A. S. Sound and complete axiomatisation of trace semantics for probabilistic systems MFPS’11
B. Jacobs, A. Silva, A. S. Trace semantics via determinization JSS’15
A. S., H. Woracek Congruences of convex algebras JPAA’15
A. S., H. Woracek Termination in convex sets of distributions CALCO’17
F. Bonchi, A. Silva, A. S. The power of convex algebras CONCUR’17
Highlights

F. Bartels, E. de Vink, A. S. A hierarchy of probabilistic system types TCS’04, CMCS’03
A.S. Coalgebraic analysis of probabilistic systems PhD thesis, TU Eindhoven’05
A. S. Probabilistic systems coalgebraically TCS’11
B. Jacobs, I. Hasuo, A. S. Generic trace semantics via coinduction LMCS’07
B. Jacobs, I. Hasuo, A. S. The microcosm principle and concurrency in coalgebra FoSSaCS’08
A. Silva, A. S. Sound and complete axiomatisation of trace semantics for probabilistic systems MFPS’11
B. Jacobs, A. Silva, A. S. Trace semantics via determinization JSS’15
A. S., H. Woracek Congruences of convex algebras JPAA’15
A. S., H. Woracek Termination in convex sets of distributions CALCO’17
F. Bonchi, A. Silva, A. S. The power of convex algebras CONCUR’17

and some very new work
Joint work with

Erik de Vink  TU/e
Bart Jacobs  Radboud University
Ichiro Hasuo  THE UNIVERSITY OF TOKYO
Harald Woracek  TU WIEN
Alexandra Silva  UCL
Filippo Bonchi  ENSE DE LYON
Part I
Modelling probabilistic systems for branching-time semantics
Part I
Modelling probabilistic systems for branching-time semantics

coalgebraically
Coalgebras

Uniform framework for dynamic transition systems, based on category theory.
Coalgebras

Uniform framework for dynamic transition systems, based on category theory.

\[ X \xrightarrow{c} FX \]
Coalgebras

Uniform framework for dynamic transition systems, based on category theory.

\[ X \xrightarrow{c} FX \]

states
Coalgebras

Uniform framework for dynamic transition systems, based on category theory.

\[ X \xrightarrow{c} FX \]

states

object in the base category \( C \)
Coalgebras

Uniform framework for dynamic transition systems, based on category theory.

\[ X \xrightarrow{c} FX \]

states, behaviour type, object in the base category \( C \)
Coalgebras

Uniform framework for dynamic transition systems, based on category theory.

\[ X \xrightarrow{c} FX \]

- States
- Behaviour type
- Object in the base category \( \mathbf{C} \)
- Functor on the base category \( \mathbf{C} \)
Coalgebras

Uniform framework for dynamic transition systems, based on category theory.

$X \xrightarrow{c} FX$

- States
- Behaviour type
- Object in the base category $\mathcal{C}$
- Functor on the base category $\mathcal{C}$

Form a category too
Coalgebras

Uniform framework for dynamic transition systems, based on category theory.

\[ X \overset{c}{\rightarrow} FX \]

- States: object in the base category $\mathbf{C}$
- Behaviour type: functor on the base category $\mathbf{C}$
- Form a category too: $\text{CoAlg}_\mathbf{C}(F)$

Ana Sokolova

TbiLLC’17, tutorial, part 1
Coalgebras

Uniform framework for dynamic transition systems, based on category theory.

\[ X \xrightarrow{C} FX \]

generic notion of behavioural equivalence
\[ \simeq \]

states

object in the base category \( C \)

behaviour type

functor on the base category \( C \)

form a category too

\( \text{CoAlg}_C(F) \)
The category of F-coalgebras

Objects = coalgebras
Arrows = coalgebra homomorphisms

\[ \text{CoAlg}_C(F) \]
The category of F-coalgebras

Objects = coalgebras

Arrows = coalgebra homomorphisms

\[ X \xrightarrow{c} FX \]
The category of F-coalgebras

Objects = coalgebras

Arrows = coalgebra homomorphisms

\[ X \xrightarrow{c} FX \]

- States
- Behaviour type
- Object in the base category \( \mathcal{C} \)
- Functor on the base category \( \mathcal{C} \)
The category of $F$-coalgebras

Objects = coalgebras

$$X \xrightarrow{c} FX$$

Arrows = coalgebra homomorphisms

$$h : X \rightarrow Y$$

$$(X \xrightarrow{c_X} FX) \xrightarrow{Fh} (FY \xleftarrow{c_Y})$$

CoAlg$_C(F)$

states

behaviour type

object in the base category $C$

functor on the base category $C$
The category of F-coalgebras

Object = coalgebras

Arrows = coalgebra homomorphisms

Objects = coalgebras

Arrows = coalgebra homomorphism

CoAlg(C)(F)
Almost all known probabilistic systems can be modelled as coalgebras on \textbf{Sets} for functors given by the following grammar:

\[
F: = - \mid A \mid \mathcal{D} \mid \mathcal{P} \mid F^A \mid F + F \mid F \circ F \mid F \times F
\]
Modelling discrete probabilistic systems

Almost all known probabilistic systems can be modelled as coalgebras on \textbf{Sets} for functors given by the following grammar:

\[ F : = - \mid A \mid \mathcal{D} \mid \mathcal{P} \mid F^A \mid F + F \mid F \circ F \mid F \times F \]

\[ X \overset{c}{\rightarrow} FX \]
Almost all known probabilistic systems can be modelled as coalgebras on sets for functors given by the following grammar:

\[ F: = \emptyset | A | \mathcal{D} | \mathcal{P} | F^A | F + F | F \circ F | F \times F \]

X \xrightarrow{c} FX
Almost all known probabilistic systems can be modelled as coalgebras on \textbf{Sets} for functors given by the following grammar:

\[
F: = \mathcal{D} | \mathcal{P} | F^{A} | F + F | F \circ F | F \times F
\]

in all cases concrete and coalgebraic bisimilarity (and behavioural equivalence) coincide.
Example
Example

NFA

$$2 \times (\mathcal{P}(-))^A \equiv \mathcal{P}(1 + A \times (-))$$
Modelling discrete probabilistic systems

Probability distribution functor on \textbf{Sets}

\[ \mathcal{D}X = \{ \mu : X \rightarrow [0, 1] \mid \sum_{x \in X} \mu(x) = 1 \} \]

for \( f : X \rightarrow Y \) we have \( \mathcal{D}f : \mathcal{D}X \rightarrow \mathcal{D}Y \) by

\[ \mathcal{D}f(\mu)(y) = \sum_{x \in f^{-1}(y)} \mu(x) = \mu(f^{-1}(y)) \]
Modelling discrete probabilistic systems

Probability distribution functor on Sets

\[ \mathcal{D}X = \{ \mu : X \rightarrow [0, 1] \mid \sum_{x \in X} \mu(x) = 1 \} \]

and its variants

\[ \mathcal{D}_{\leq 1}X = \{ \mu : X \rightarrow [0, 1] \mid \sum_{x \in X} \mu(x) \leq 1 \} \]

\[ \mathcal{D}_fX = \{ \mu : X \rightarrow [0, 1] \mid \sum_{x \in X} \mu(x) = 1, \text{supp}(\mu) \text{ is finite} \} \]
Modelling discrete probabilistic systems

Probability distribution functor on $\textbf{Sets}$

$$\mathcal{D}X = \{\mu : X \to [0, 1] \mid \sum_{x \in X} \mu(x) = 1\}$$

and its variants

$$\mathcal{D}_{\leq 1}X = \{\mu : X \to [0, 1] \mid \sum_{x \in X} \mu(x) \leq 1\}$$

$$\mathcal{D}_fX = \{\mu : X \to [0, 1] \mid \sum_{x \in X} \mu(x) = 1, \text{supp}(\mu) \text{ is finite}\}$$

$\{x \in X \mid \mu(x) \neq 0\}$
Modelling discrete probabilistic systems

Probability distribution functor on \textbf{Sets}

\[ \mathcal{D} X = \{ \mu : X \rightarrow [0, 1] \mid \sum_{x \in X} \mu(x) = 1 \} \]

and its variants

\[ \mathcal{D}_{\leq 1} X = \{ \mu : X \rightarrow [0, 1] \mid \sum_{x \in X} \mu(x) \leq 1 \} \]

\[ \mathcal{D}_f X = \{ \mu : X \rightarrow [0, 1] \mid \sum_{x \in X} \mu(x) = 1, \text{supp}(\mu) \text{ is finite} \} \]
Examples
Examples

\[ 2 \times (\mathcal{P}(-))^A \equiv \mathcal{P} (1 + A \times (-)) \]
Examples

**NFA**

\[ 2 \times (\mathcal{P}(-))^A \cong \mathcal{P} (1 + A \times (-)) \]

**Generative PTS**

\[ \mathcal{D}_{\leq 1} (1 + A \times (-)) \]
Examples

NFA

\[ 2 \times (\mathcal{P}(-))^A \cong \mathcal{P}(1 + A \times (-)) \]

Generative PTS

\[ \mathcal{D}_{\leq 1}(1 + A \times (-)) \]

Simple PA

\[ \mathcal{P}(A \times \mathcal{D}_{\leq 1}(-)) \]
## Expressiveness hierarchy

F. Bartels, A. S., E. de Vink ’03/’04

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MC</strong></td>
<td>$\mathcal{D}$</td>
</tr>
<tr>
<td><strong>DLTS</strong></td>
<td>$(_ + 1)^A$</td>
</tr>
<tr>
<td><strong>LTS</strong></td>
<td>$\mathcal{P}(A \times _) \equiv \mathcal{P}^A$</td>
</tr>
<tr>
<td><strong>React</strong></td>
<td>$(\mathcal{D} + 1)^A$</td>
</tr>
<tr>
<td><strong>Gen</strong></td>
<td>$\mathcal{D}(A \times _) + 1$</td>
</tr>
<tr>
<td><strong>Str</strong></td>
<td>$\mathcal{D} + (A \times _) + 1$</td>
</tr>
<tr>
<td><strong>Alt</strong></td>
<td>$\mathcal{D} + \mathcal{P}(A \times _)$</td>
</tr>
<tr>
<td><strong>Var</strong></td>
<td>$\mathcal{D}(A \times _) + \mathcal{P}(A \times _)$</td>
</tr>
<tr>
<td><strong>SSeg</strong></td>
<td>$\mathcal{P}(A \times \mathcal{D})$</td>
</tr>
<tr>
<td><strong>Seg</strong></td>
<td>$\mathcal{P}\mathcal{D}(A \times _)$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
## Expressiveness hierarchy

**F. Bartels, A. S., E. de Vink ’03/’04**

<table>
<thead>
<tr>
<th>MC</th>
<th>$\mathcal{D}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLTS</td>
<td>$(A + 1)^A$</td>
</tr>
<tr>
<td>LTS</td>
<td>$\mathcal{P}(A \times _ ) \cong \mathcal{P}^A$</td>
</tr>
<tr>
<td>React</td>
<td>$(\mathcal{D} + 1)^A$</td>
</tr>
<tr>
<td>Gen</td>
<td>$\mathcal{D}(A \times _ ) + 1$</td>
</tr>
<tr>
<td>Str</td>
<td>$\mathcal{D} + (A \times _ ) + 1$</td>
</tr>
<tr>
<td>Alt</td>
<td>$\mathcal{D} + \mathcal{P}(A \times _ )$</td>
</tr>
<tr>
<td>Var</td>
<td>$\mathcal{D}(A \times _ ) + \mathcal{P}(A \times _ )$</td>
</tr>
<tr>
<td>SSeg</td>
<td>$\mathcal{P}(A \times \mathcal{D})$</td>
</tr>
<tr>
<td>Seg</td>
<td>$\mathcal{P}\mathcal{D}(A \times _ )$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

**Markov chain**

$$x \rightarrow \mathcal{D}x$$

\[
\begin{align*}
    x_2 & \quad \xrightarrow{1/3} \quad x_1 \\
    x_1 & \quad \xrightarrow{2/3} \quad x_3 \\
    x_3 & \quad \xrightarrow{1} \quad x_3
\end{align*}
\]
# Expressiveness hierarchy

F. Bartels, A. S., E. de Vink ’03/’04

<table>
<thead>
<tr>
<th>MC</th>
<th>DLTS</th>
<th>LTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>LTS</td>
<td>( D ) ((_ + 1)^A )</td>
<td>( P(A \times _) \cong P^A )</td>
</tr>
<tr>
<td>React</td>
<td>((D + 1)^A)</td>
<td>(P^A)</td>
</tr>
<tr>
<td>Gen</td>
<td>(D(A \times _) + 1)</td>
<td>(P^A)</td>
</tr>
<tr>
<td>Str</td>
<td>(D + (A \times _) + 1)</td>
<td>(P^A)</td>
</tr>
<tr>
<td>Alt</td>
<td>(D + P(A \times _))</td>
<td>(P^A)</td>
</tr>
<tr>
<td>Var</td>
<td>(D(A \times _) + P(A \times _))</td>
<td>(P^A)</td>
</tr>
<tr>
<td>SSeg</td>
<td>(P(A \times D))</td>
<td>(P^A)</td>
</tr>
<tr>
<td>Seg</td>
<td>(PD(A \times _))</td>
<td>(P^A)</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

\[ \text{LTS} \quad X \rightarrow P(A \times X) \]

![Diagram](image-url)
# Expressiveness hierarchy

F. Bartels, A. S., E. de Vink ’03/’04

<table>
<thead>
<tr>
<th>MC</th>
<th>DLTS</th>
<th>LTS</th>
<th>React</th>
<th>Gen</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>(_ + 1)^A</td>
<td>P(A × _) ≅ P^A</td>
<td>(D + 1)^A</td>
<td>D(A × _) + 1</td>
<td>D + (A × _) + 1</td>
</tr>
</tbody>
</table>

**Generative system**

\[
X \rightarrow D(A \times X) + 1
\]

\[
\begin{align*}
& a, \frac{3}{4} \quad x_1 \quad a, \frac{1}{4} \\
& b, 1 \quad x_2 \quad x_3 \\
& x_4 \quad x_5 \\
& \ast \quad \ast
\end{align*}
\]
Expressiveness hierarchy

F.Bartels, A.S., E. de Vink ’03/’04

<table>
<thead>
<tr>
<th>MC</th>
<th>$D$</th>
</tr>
</thead>
<tbody>
<tr>
<td>DLTS</td>
<td>$(_ + 1)^A$</td>
</tr>
<tr>
<td>LTS</td>
<td>$P(A \times _) \cong P^A$</td>
</tr>
<tr>
<td>React</td>
<td>$(D + 1)^A$</td>
</tr>
<tr>
<td>Gen</td>
<td>$D(A \times _) + 1$</td>
</tr>
<tr>
<td>Str</td>
<td>$D + (A \times _) + 1$</td>
</tr>
<tr>
<td>Alt</td>
<td>$D + P(A \times _)$</td>
</tr>
<tr>
<td>Var</td>
<td>$D(A \times _) + P(A \times _)$</td>
</tr>
<tr>
<td>SSeg</td>
<td>$P(A \times D)$</td>
</tr>
<tr>
<td>Seg</td>
<td>$P \mathcal{D}(A \times _)$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Simple Segala system (PA)

$X \rightarrow P (A \times \mathcal{D}X)$
Expressiveness hierarchy

F.Bartels, A.S., E. de Vink ’03/’04

behavioural equivalence

translation that preserves and reflects bisimilarity
Behavioural equivalence

\[ X \xrightarrow{c} FX \]

\[ h : X \rightarrow Y \]

\[ F(X) \xrightarrow{Ff} F(Y) \]

\[ \text{CoAlg}_C(F) \]
Behavioural equivalence

\[ h : X \rightarrow Y \]

\[ X \xrightarrow{c} FX \]

\[ \text{CoAlg}_C(F) \]

\[ Ff \]

\[ c_Y \]

\[ c_X \]

\[ F_Y \]
Behavioural equivalence

$X \xrightarrow{c} FX$

$h : X \rightarrow Y$

$X \xrightarrow{f} Y$

$FX \xrightarrow{Ff} FY$

CoAlg$_C(F)$
Kernel bisimulation = kernel of a coalgebra homomorphism
Behavioural equivalence

\[ X \xrightarrow{c} FX \]

Generic notion

\[ \text{Kernel bisimulation} = \ker(h) = \{(x, y) \mid h(x) = h(y)\} \]

Branching-time semantics

\[ \text{CoAlg}_C(F) \]
Behavioural equivalence

Kernel bisimulation = kernel of a coalgebra homomorphism

\[ \ker(h) = \{(x, y) \mid h(x) = h(y)\} \]

Behaviour equivalence = union of all kernel bisimulations

\[ h : X \rightarrow Y \]

\[ X \xrightarrow{c} FX \]

\[ FX \xrightarrow{Ff} FY \]

\[ X \xrightarrow{f} Y \]

\[ c_X \downarrow \]

\[ c_Y \downarrow \]

\[ \text{CoAlg}_C(F) \]

generic notion

branching-time semantics

TbiLLC’17, tutorial, part 1
Behavioural equivalence

Kernel bisimulation $\approx$ kernel of a coalgebra homomorphism

$\ker(h) = \{(x, y) \mid h(x) = h(y)\}$

Behaviour equivalence $\approx$ union of all kernel bisimulations

coincides with “concrete” bisimilarity
Bisimilarity

LTS
Bisimilarity

$R$

LTS

bisimulation
Bisimilarity

LTS

bisimulation

$R$
Bisimilarity

LTS

bisimulation

\[ R \]

\[ a \]
Bisimilarity

\[ \text{LTS} \]

\[ \text{bisimulation} \]

\[ R \]

\[ a \]

\[ \alpha \]
Bisimilarity
Bisimilarity

\( R \) bisimulation

\( a \)

\( R \)

\( a \)

\( R \) bisimulation

\( \sim \) largest bisimulation

LTS
Bisimilarity

\[ x R y \Rightarrow x \xrightarrow{a} x' \Rightarrow \exists y'. y \xrightarrow{a} y' \land x' R y' \]

LTS

transfer condition

\sim \text{ largest bisimulation}
Bisimilarity

\[ x R y \Rightarrow \\
\quad x \xrightarrow{a} x' \Rightarrow \exists y'. y \xrightarrow{a} y' \land x' R y' \]

LTS

\sim \text{ largest bisimulation}

coincides with behavioural equivalence

transfer condition
Bisimilarity

Markov chains
Bisimilarity

$R$
Bisimilarity

Markov chains

bisimulation

$\sim$
Bisimilarity

Markov chains

bisimulation

$R$

$\mu$
Bisimilarity

Markov chains

bisimulation

\[ R \]

\[ \mu \]

\[ \nu \]
Bisimilarity

Markov chains

bisimulation

\[ \mu \stackrel{R}{\Rightarrow} \nu \]
Bisimilarity

Markov chains

bisimulation

$\mu \equiv_R \nu$

lifting of $R$ to distributions
Bisimilarity

Markov chains

bisimulation

lifting of R to distributions

assign the same probability to “R-classes”
Bisimilarity

Markov chains

$\sim$ largest bisimulation

lifting of $R$ to distributions

assign the same probability to “$R$-classes”

$R$

$\mu$

$\equiv_R$

$\nu$
Bisimilarity

Markov chains

\( R \)

bisimulation

\( \sim \) largest bisimulation

transfer condition

\[ x \ R \ y \Rightarrow \]

\[ x \xrightarrow{\mu} \Rightarrow y \xrightarrow{\nu} \land \mu \equiv_R \nu \]
Bisimilarity

$$\sim$$ largest bisimulation

Markov chains

bisimulation

$$x R y \Rightarrow \quad x \rightarrow \mu \Rightarrow y \rightarrow \nu \land \mu \equiv R \nu$$

transfer condition

coincides with behavioural equivalence

$$\equiv_R$$

$$\mu$$

$$\nu$$
Bisimilarity

Markov chains

Markov chains

Bisimulation

transfer condition

\[ x R y \Rightarrow \]

\[ x \rightarrow \mu \Rightarrow y \rightarrow \nu \land \mu \equiv_R \nu \]

\[ \sim \text{ largest bisimulation} \]

coincides with behavioural equivalence

but is trivial

Ana Sokolova

TbiLLC’17, tutorial, part 1
Bisimilarity

Markov chains

transfer condition

\( x \xrightarrow{R} y \Rightarrow \)

\[ x \xrightarrow{\mu} \Rightarrow y \xrightarrow{\nu} \land \mu \equiv_R \nu \]

Bisimulation

\( ~ \) largest bisimulation

for non-trivial behaviour we need labels / termination

coincides with behavioural equivalence

but is trivial

Ana Sokolova
Bisimilarity

Simple Segala systems / simple PA
Bisimilarity

Simple Segala systems / simple PA

bisimulation

$R$
Bisimilarity

Simple Segala systems / simple PA

bisimulation

R

TbiLLC’17, tutorial, part 1
Bisimilarity

Simple Segala systems / simple PA

bisimulation

$R$

$\alpha$

$\mu$
Bisimilarity

Simple Segala systems / simple PA

\[ R \]

\[ a \]

\[ \mu \]

\[ a \]

\[ \nu \]

bisimulation
Bisimilarity

Simple Segala systems / simple PA

bisimulation

\[ \mu \leadsto^R \nu \]

Ana Sokolova
TbiLLC’17, tutorial, part 1
Bisimilarity

Simple Segala systems / simple PA

bisimulation

lifting of $R$ to distributions
Bisimilarity

Simple Segala systems / simple PA

lifting of R to distributions

assign the same probability to “R-classes”

bisimulation
Bisimilarity

Simple Segala systems / simple PA

lifting of $R$ to distributions
assign the same probability to "$R$-classes"

bisimulation

$\mu$ $\equiv_R$ $\sim$

$R$

$\alpha$

$\sim$

largest bisimulation
Bisimilarity

Simple Segala systems / simple PA

bisimulation

transfer condition

\[ x R y \Rightarrow \]
\[ x \overset{a}{\to} \mu \Rightarrow \exists \nu. y \overset{a}{\to} \nu \land \mu \equiv_R \nu \]
Bisimilarity

Simple Segala systems / simple PA

\[ x \xrightarrow{R} y \Rightarrow x \xrightarrow{a} \mu \Rightarrow \exists \nu. y \xrightarrow{a} \nu \land \mu \equiv_R \nu \]

\[ R \approx \] largest bisimulation

transfer condition

coincides with behavioural equivalence
Bisimilarity

Simple Segala systems / simple PA

transfer condition

\[ x \stackrel{R}{\rightarrow} y \Rightarrow \]

\[ x \xrightarrow{a} \mu \Rightarrow \exists \nu. \ y \xrightarrow{a} \nu \land \mu \equiv_R \nu \]

\sim \text{ largest bisimulation}

all concrete bisimilarity notions coincide with behavioural equivalence

coincides with behavioural equivalence
Bisimilarity

F - coalgebras
Bisimilarity

F - coalgebras

bisimulation

$R$
Bisimilarity

F - coalgebras

bisimulation

$R$
Bisimilarity

F - coalgebras

bisimulation

\[ R \]
Bisimilarity

F - coalgebras

bisimulation

\[ \text{Rel}(F)(R) \]

\[ R \]
Bisimilarity

F - coalgebras

bisimulation

$R$

$\text{Rel}(F)(R)$

F-relation lifting of $R$
Bisimilarity

F - coalgebras

bisimulation

$\sim$ largest bisimulation

$R$

$\text{Rel}(F)(R)$

F-relation lifting of $R$
Bisimilarity

~ largest bisimulation

our class of F-coalgebras

bisimulation

\[ R \]

\[ \text{Rel}(F)(R) \]

F-relation lifting of R

Ana Sokolova
Bisimilarity

our class of F-coalgebras

\[ x \ R \ y \Rightarrow c(x) \ Rel(F)(R) \ c(y) \]

\[ \sim \text{ largest bisimulation} \]
Bisimilarity

our class of F-coalgebras

\[ x R y \Rightarrow c(x) \text{ Rel}(F)(R) c(y) \]

\( \sim \) largest bisimulation

coincides with behavioural equivalence

transfer condition

\[ \text{Rel}(F)(R) \]
Bisimilarity

our class of F-coalgebras

\[ x \mathrel{R} y \Rightarrow c(x) \mathrel{\text{Rel}(F)(R)} c(y) \]

\[ \sim \text{ largest bisimulation} \]

\[ \text{provides a modular proof of coincidence} \]

\[ \text{coincides with behavioural equivalence} \]
Expressiveness hierarchy

F. Bartels, A.S., E. de Vink ’03/’04

behavioural equivalence

translation that preserves and reflects bisimilarity
The translation
The translation that preserves and reflects bisimilarity
The translation
that preserves and reflects bisimilarity
behavioural equivalence
The translation

Theorem

For $F$-coalgebras $\rightarrow G$-coalgebras, it suffices to give an injective natural transformation from $F$ to $G$. 

that preserves and reflects bisimilarity

behavioural equivalence
The translation

Theorem

For F-coalgebras \( \rightarrow \) G-coalgebras, it suffices to give an injective natural transformation from F to G.

behavioural equivalence is preserved and reflected

that preserves and reflects bisimilarity

behavioural equivalence
The translation

**Theorem**

For $F$-coalgebras $\rightarrow G$-coalgebras, it suffices to give an injective natural transformation from $F$ to $G$.

- Behavioural equivalence is preserved and reflected.

- If $F$ preserves weak pullbacks then behavioural equivalence coincides with coalgebraic bisimilarity (and so bisimilarity is preserved and reflected).

- That preserves and reflects bisimilarity

**behavioural equivalence**
Example translation
Example translation

that preserves and reflects bisimilarity
Example translation

that preserves and reflects bisimilarity

behavioural equivalence
Example translation

Simple Segala system (PA)

\[ X \rightarrow \mathcal{P}(A \times \mathcal{D}X) \]

that preserves and reflects bisimilarity

behavioural equivalence
Example translation

that preserves and reflects bisimilarity

behavioural equivalence

Simple Segala system (PA)

\[ X \rightarrow \mathcal{P} (A \times \Delta X) \]
Example translation

Simple Segala system (PA)
\[ X \rightarrow \mathcal{P}(A \times \mathcal{D}X) \]

General Segala system (PA)
\[ X \rightarrow \mathcal{P}\mathcal{D}(A \times X) \]

that preserves and reflects bisimilarity

behavioural equivalence
Probabilities are not that special…

• Subsets, multisets, distributions,.. are all instances of the same functor

• For a monoid \((M, +, 0)\) and a subset \(S \subseteq M\)

\[
V_S(X) = \{ \varphi: X \to M \mid \text{supp}(x) \text{ is finite and } \sum_{x \in X} \varphi(x) \in S \}
\]
Probabilities are not that special...

- Subsets, multisets, distributions,.. are all instances of the same functor

- For a monoid \((M, +, 0)\) and a subset \(S \subseteq M\)

\[
V_S(X) = \{\varphi : X \rightarrow M \mid \text{supp}(x) \text{ is finite and } \sum_{x \in X} \varphi(x) \in S\}
\]
Probabilities are not that special...

- Subsets, multisets, distributions,... are all instances of the same functor

For a monoid \((M, +, 0)\) and a subset \(S \subseteq M\)

\[
V_S(X) = \{ \varphi : X \to M \mid \text{supp}(x) \text{ is finite and } \sum_{x \in X} \varphi(x) \in S \}
\]

\[
V_S = \mathcal{P}_f \\
M = (\{0, 1\}, \lor, 0) \\
S = M
\]
Probabilities are not that special…

- Subsets, multisets, distributions,.. are all instances of the **same functor**
- For a monoid \((M, +, 0)\) and a subset \(S \subseteq M\)

\[
V_S(X) = \{\varphi: X \to M \mid \text{supp}(x) \text{ is finite and } \sum_{x \in X} \varphi(x) \in S\}
\]

\[
\begin{align*}
V_S &= \mathcal{P}_f \\
M &= (\{0, 1\}, \lor, 0) \\
S &= M
\end{align*}
\]

\[
\begin{align*}
V_S &= \mathcal{M}_f \\
M &= (\mathbb{N}, +, 0) \\
S &= M
\end{align*}
\]
Probabilities are not that special...

- Subsets, multisets, distributions,.. are all instances of the same functor
- For a monoid \((M, +, 0)\) and a subset \(S \subseteq M\)

\[
V_S(X) = \{ \varphi: X \to M \mid \text{supp}(x) \text{ is finite and } \sum_{x \in X} \varphi(x) \in S \} 
\]

- If \(V_S = \mathcal{P}_f\):
  \[
  M = (\{0, 1\}, \lor, 0) \\
  S = M 
  \]

- If \(V_S = \mathcal{M}_f\):
  \[
  M = (\mathbb{N}, +, 0) \\
  S = M 
  \]

- If \(V_S = \mathcal{D}_f\):
  \[
  M = (\mathbb{R}^+, +, 0) \\
  S = [0, 1] 
  \]
Probabilities are not that special…

- Subsets, multisets, distributions,.. are all instances of the **same functor**
- For a monoid \((M, +, 0)\) and a subset \(S \subseteq M\)

\[
V_S(X) = \{ \varphi: X \to M \mid \text{supp}(x) \text{ is finite and } \sum_{x \in X} \varphi(x) \in S \}
\]

\[
\begin{align*}
V_S &= \mathcal{P}_f \\
M &= \{0, 1\}, \lor, 0 \\
S &= M
\end{align*}
\]

\[
\begin{align*}
V_S &= \mathcal{M}_f \\
M &= (\mathbb{N}, +, 0) \\
S &= M
\end{align*}
\]

\[
\begin{align*}
V_S &= \mathcal{D}_f \\
M &= (\mathbb{R}^+, +, 0) \\
S &= [0, 1]
\end{align*}
\]

**additional structure on** \(M\) **adds structure to** \(V_S\)