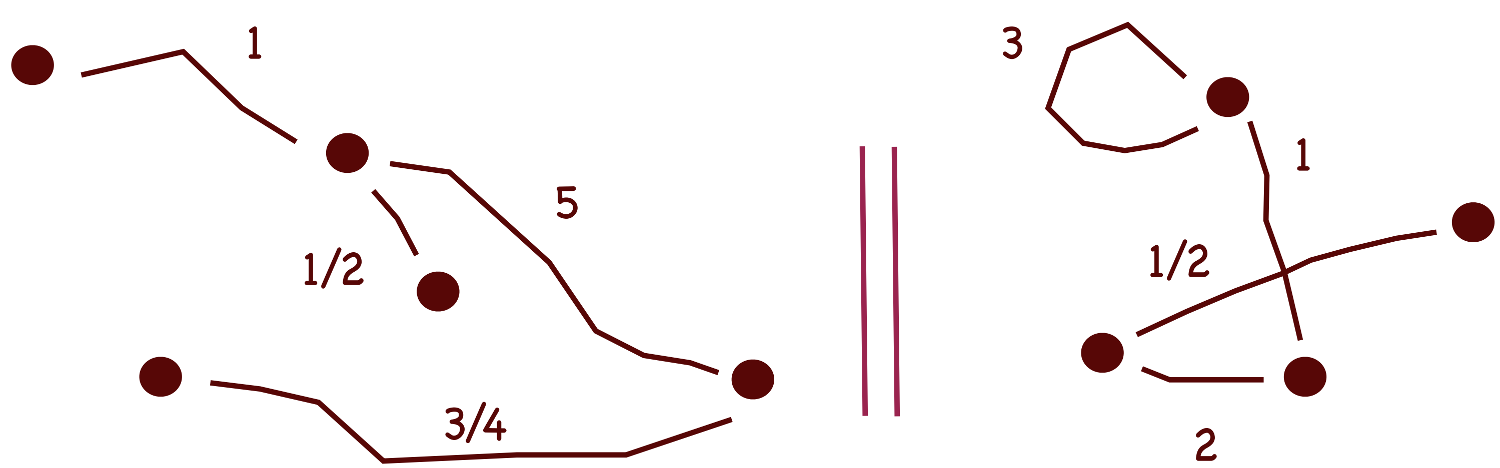
Probabilistic Systems Semantics via Coalgebra

Ana Sokolova
UNIVERSITY of SALZBURG



TbiLLC'17 tutorial, part 2

Plan:

Part 1. Modelling probabilistic systems for branching-time semantics

bisimilarity

Part 2. Traces, linear-time semantics

trace equivalence

Part 3. Belief-state-transformer semantics via convexity

distribution bisimilarity

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

all with help of coalgebra

Plan:

not fully done yet

Part 1. Modelling probabilistic systems for branching-time semantics

bisimilarity

Part 2. Traces, linear-time semantics

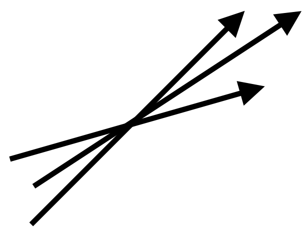
trace equivalence

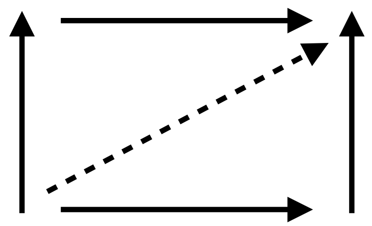
Part 3. Belief-state-transformer semantics via convexity

distribution bisimilarity

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

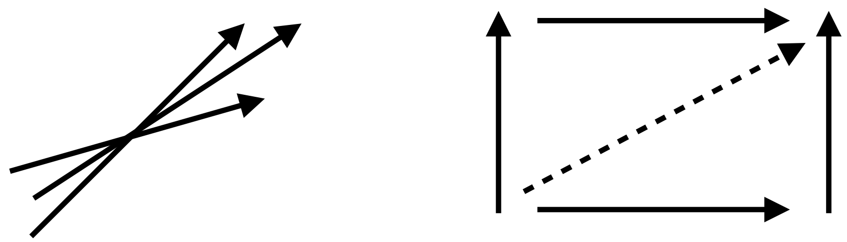
all with help of coalgebra





Discrete probabilistic systems are coalgebras

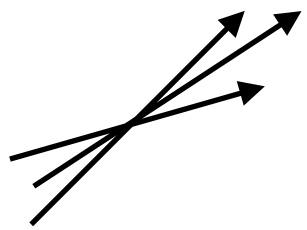
$$X \xrightarrow{c} FX$$

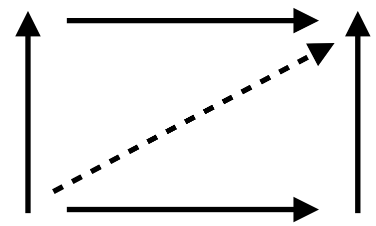


Discrete probabilistic systems are coalgebras

$X \xrightarrow{c} FX$

on **Sets**



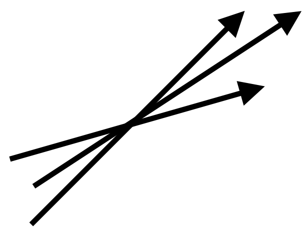


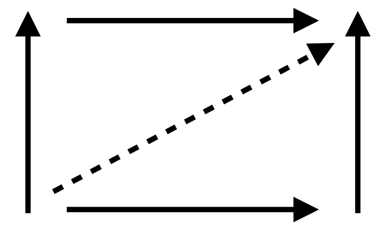
Discrete probabilistic systems are coalgebras

on **Sets**

$$X \xrightarrow{c} FX$$

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





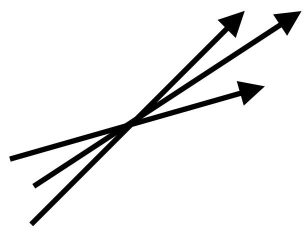
Discrete probabilistic systems are coalgebras

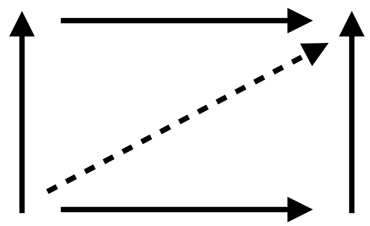
on **Sets**

$X \xrightarrow{c} FX$

probability distribution functor

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





Source Wikipedia, by Diacritica - Own work

Discrete probabilistic systems are coalgebras

$X \xrightarrow{c} FX$

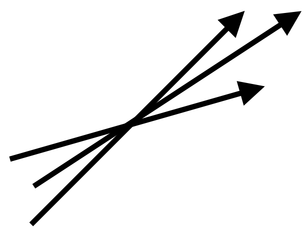
on **Sets**

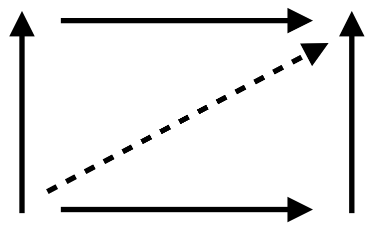
probability distribution functor

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

generic behaviour equivalence $\approx$

branching-time semantics





Source Wikipedia, by Diacritica - Own work

Discrete probabilistic systems are coalgebras

on **Sets**

$X \xrightarrow{c} FX$

probability distribution functor

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

generic behaviour equivalence
$\approx$

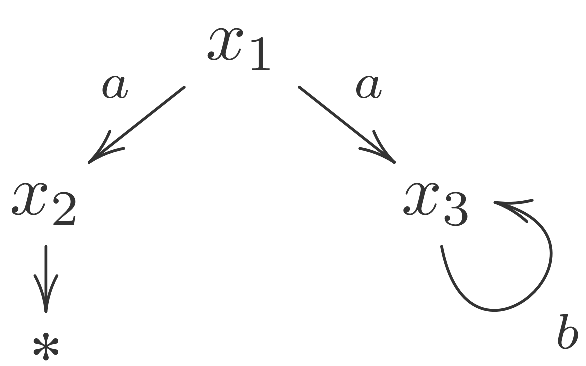
branching-time semantics

coincides with concrete bisimilarity

Examples

NFA

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

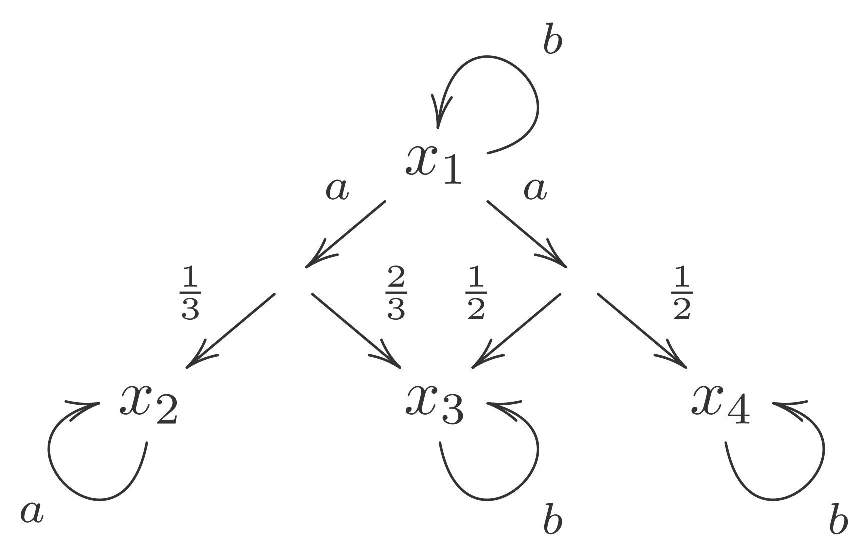


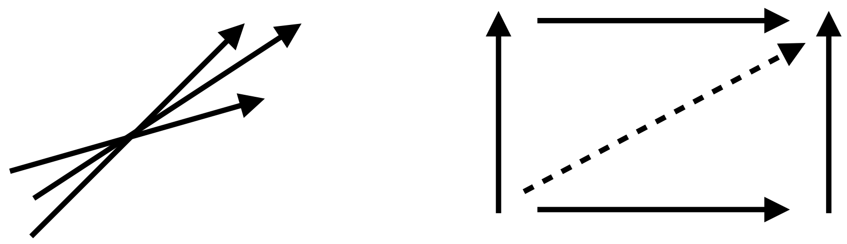
Generative PTS

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

Simple PA

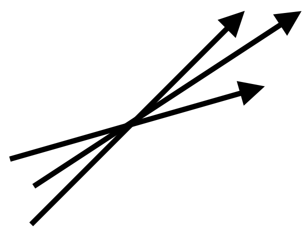
$\mathcal{P}(A \times \mathcal{D}(-))$

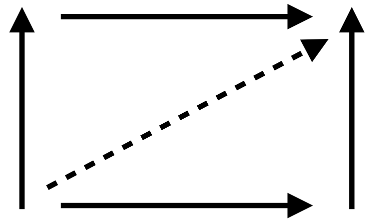




Source Wikipedia, by Diacritica - Own work

Continuous Probabilistic Systems

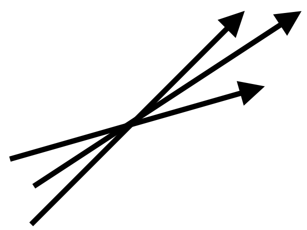


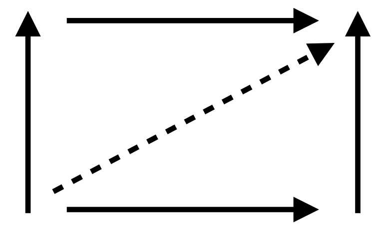


Source Wikipedia, by Diacritica - Own work
CC BY-SA 3.0

Continuous Probabilistic Systems

coalgebraically



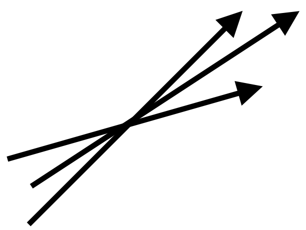


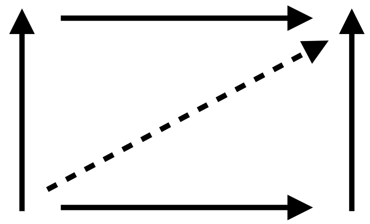
Source Wikipedia, by Diacritica - Own work

Continuous Probabilistic Systems

coalgebraically

live beyond **Sets**





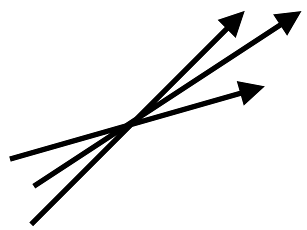
Source Wikipedia, by Diacritica - Own work

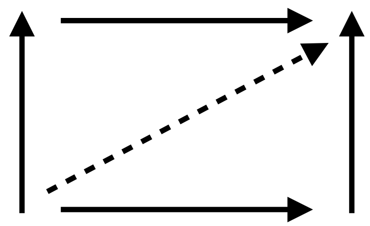
Continuous Probabilistic Systems

coalgebraically

live beyond
Sets

in
Meas





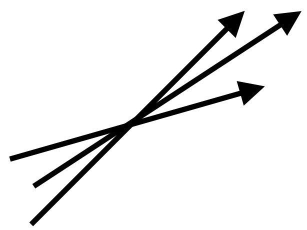
Continuous Probabilistic Systems

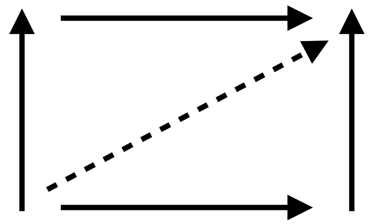
coalgebraically

live beyond **Sets**

in **Meas**

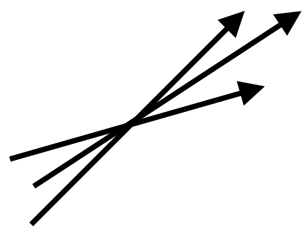
Objects = measurable spaces   Arrows = measurable maps

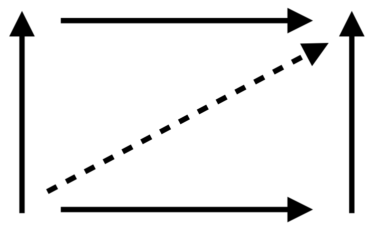




Source Wikipedia, by Diacritica - Own work

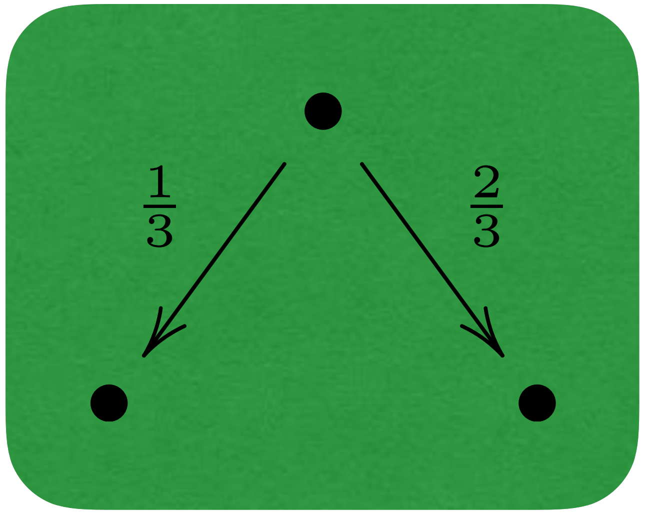
Discrete vs. Continuous Probabilistic Systems

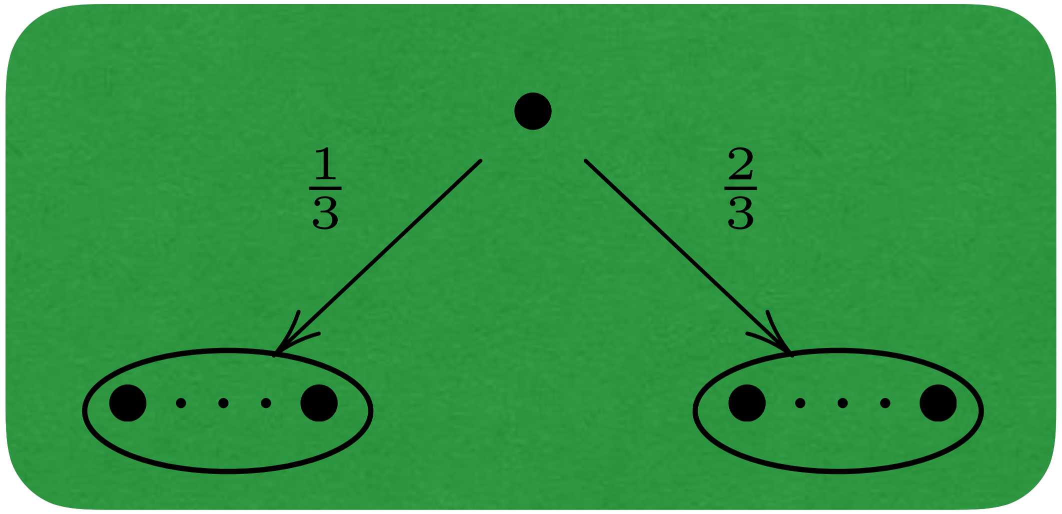


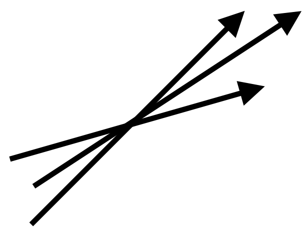


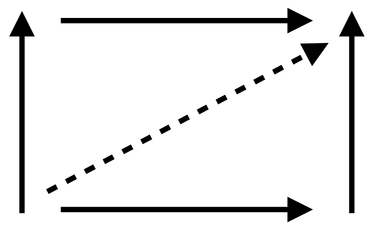
Source Wikipedia, by Diacritica - Own work

Discrete vs. Continuous Probabilistic Systems

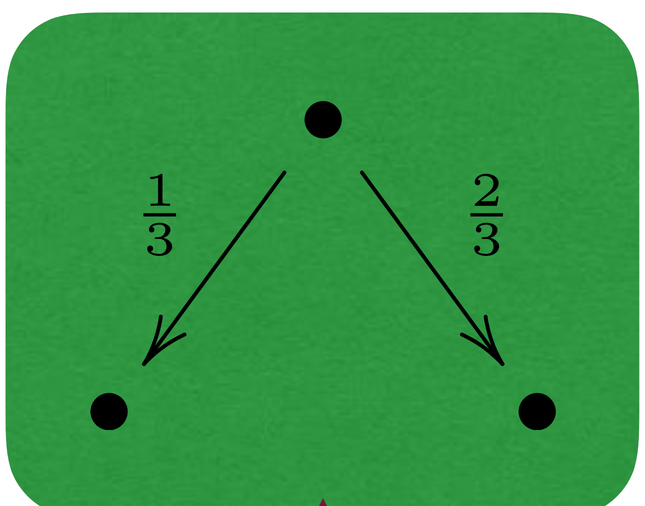




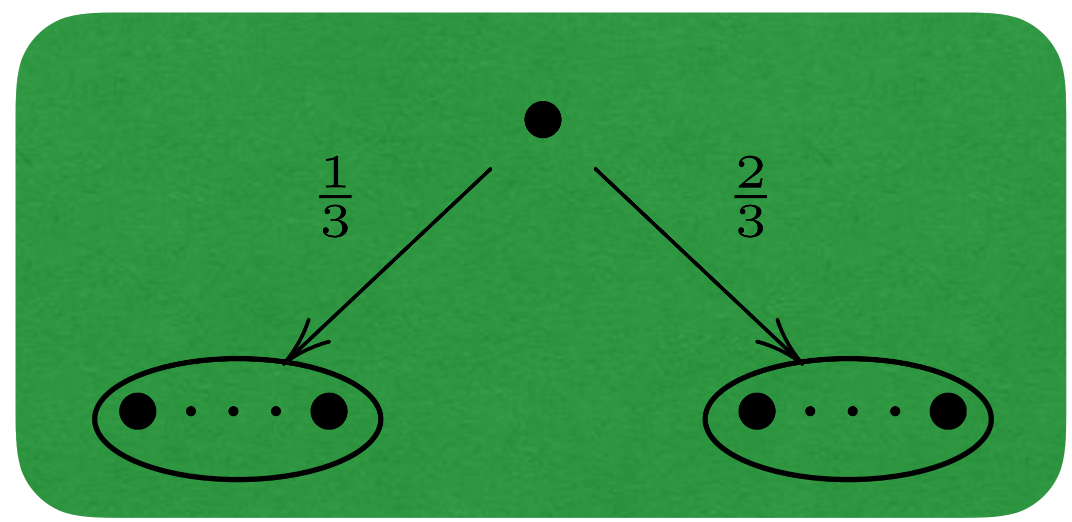


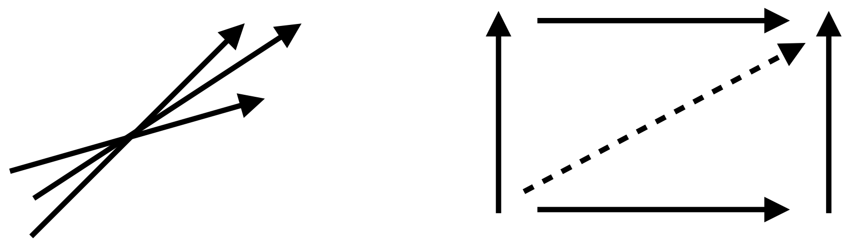


Discrete vs. Continuous Probabilistic Systems



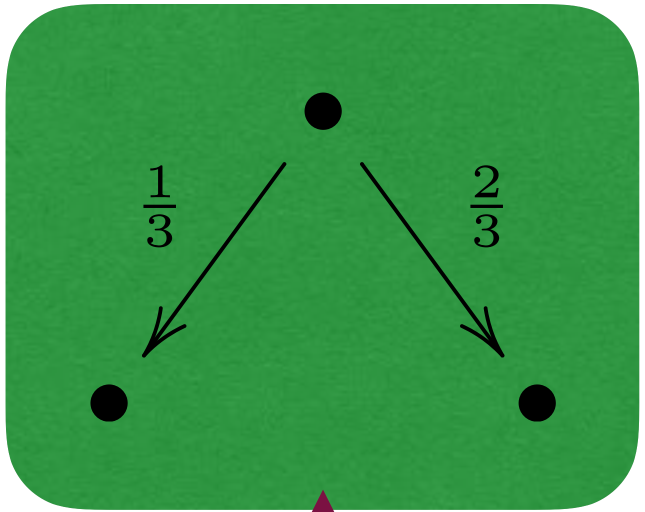
Markov chains



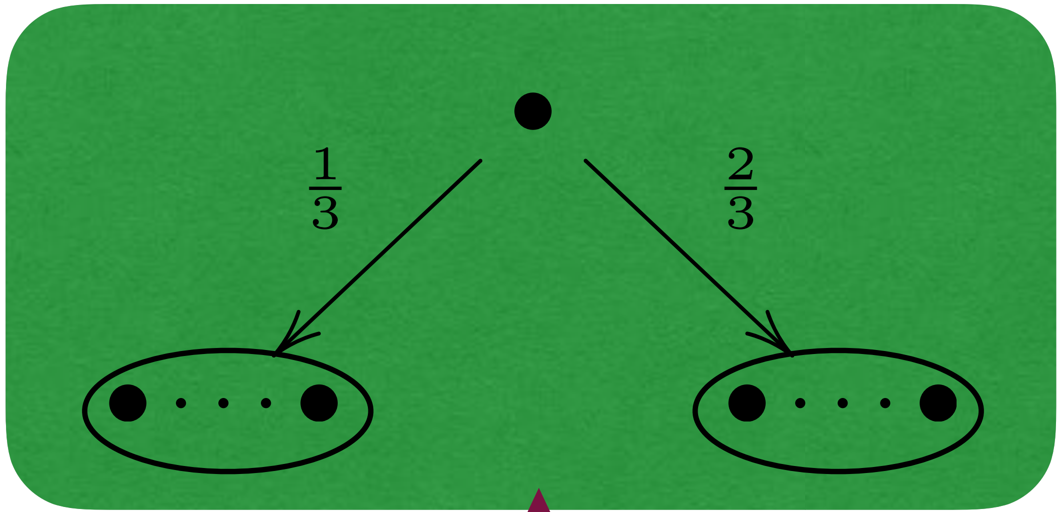


Source Wikipedia, by Diacritica - Own work

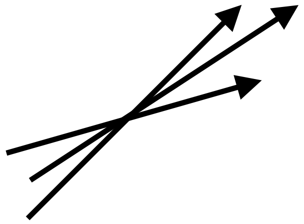
Discrete vs. Continuous Probabilistic Systems

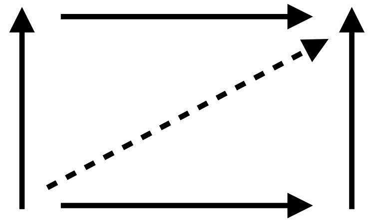


Markov chains



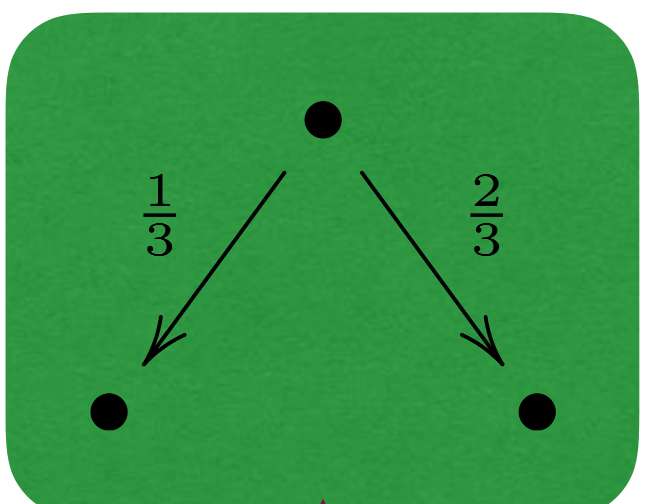
Markov processes

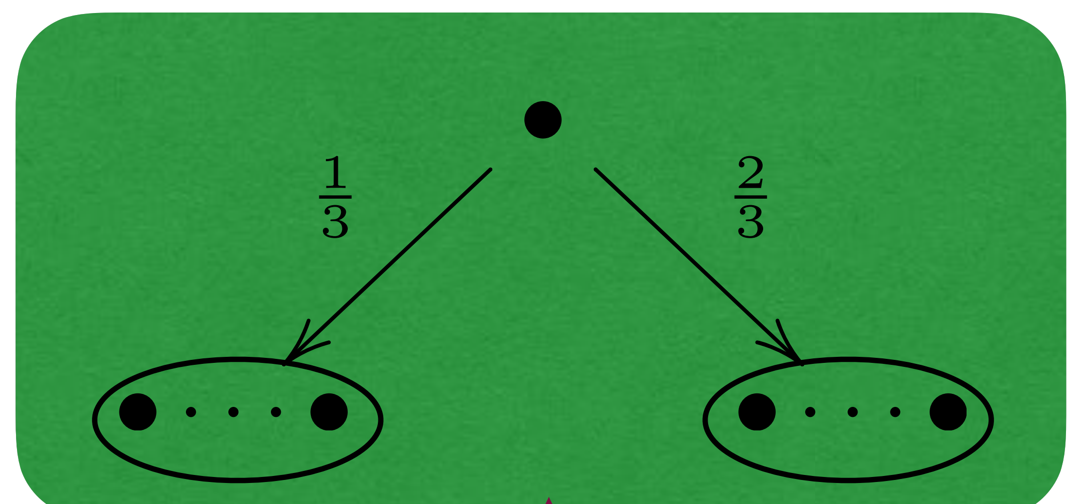




Source Wikipedia, by Diacritica - Own work

Discrete vs. Continuous Probabilistic Systems

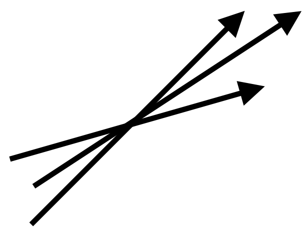


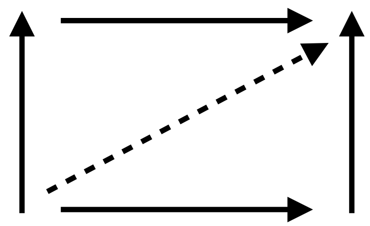


Markov chains

Markov processes

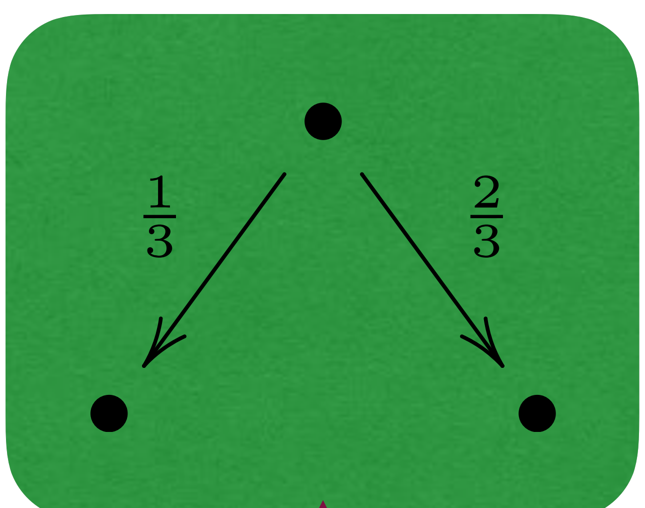
coalgebras of the Giry functor

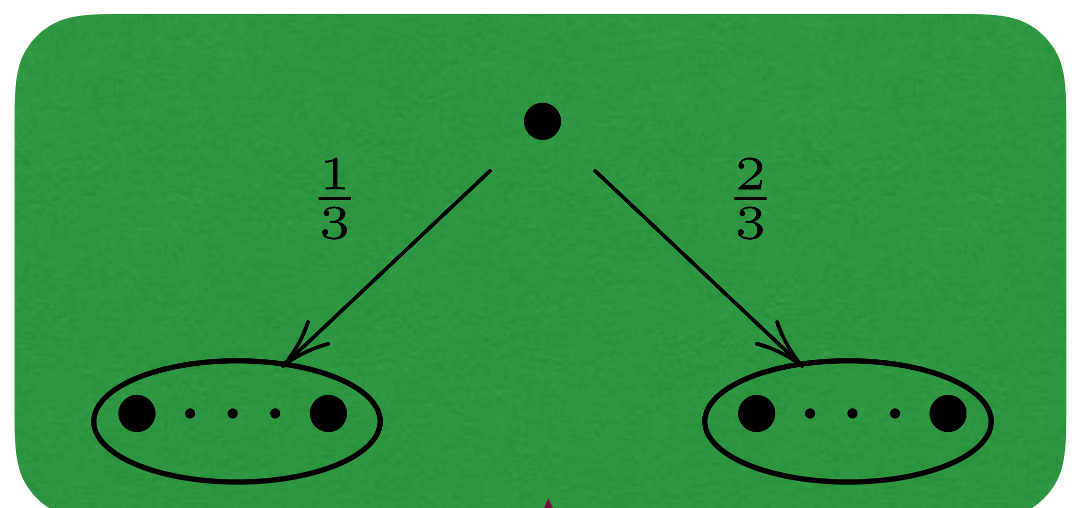




Source Wikipedia, by Diacritica - Own work

Discrete vs. Continuous Probabilistic Systems



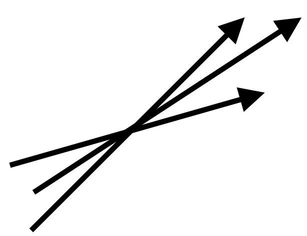


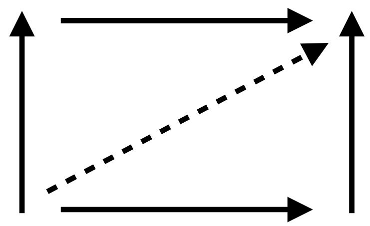
Markov chains

Markov processes

coalgebras of the Giry functor

on **Meas**

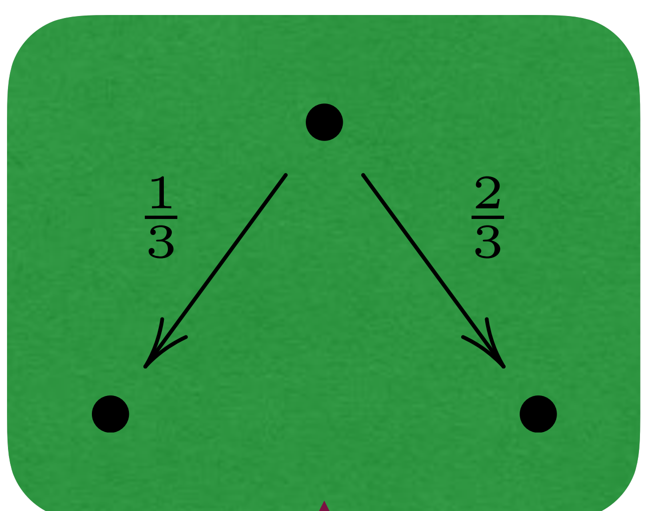


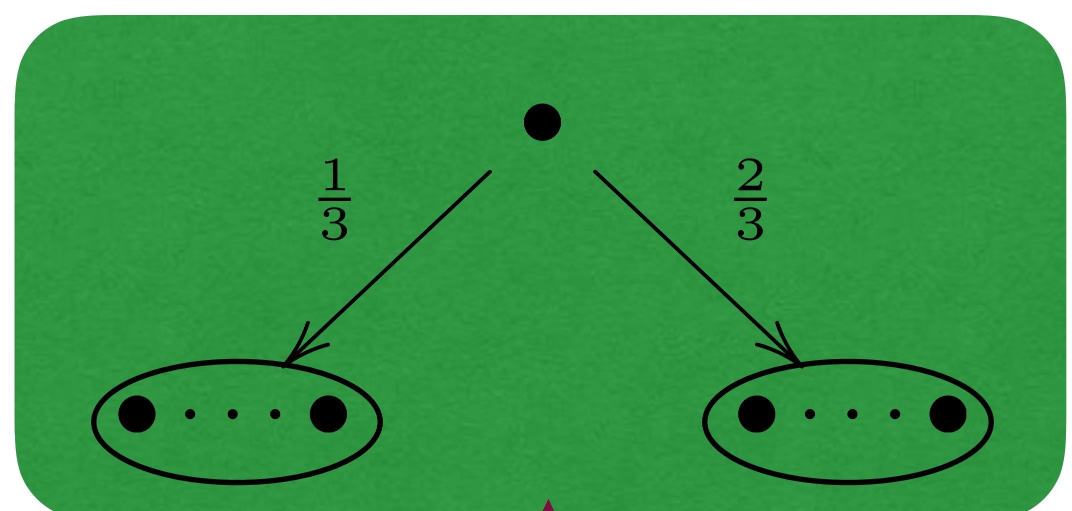


Source Wikipedia, by Diacritica - Own work

Discrete vs. Continuous Probabilistic Systems

more complex
but
analogous results
are possible



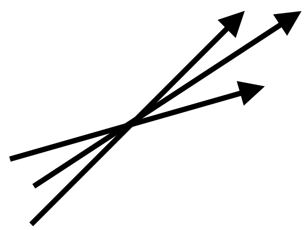


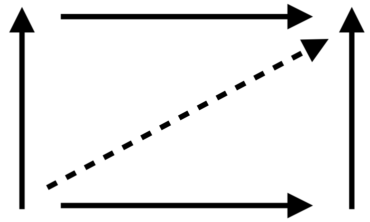
Markov chains

Markov processes

coalgebras
of the
Giry functor

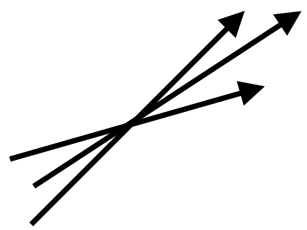
on
Meas

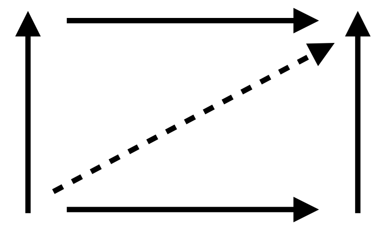




Source Wikipedia, by Diacritica - Own work

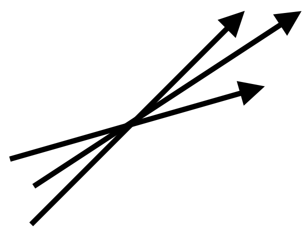
Both discrete and continuous probabilistic systems
are
coalgebras

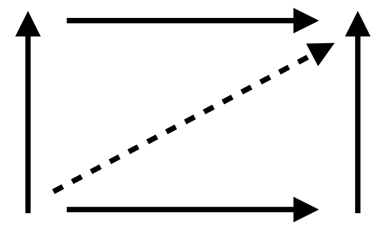




Both discrete and continuous
probabilistic systems
are
coalgebras

$\mathcal{D}$ on
$\mathbf{Sets}$

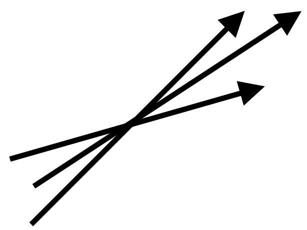


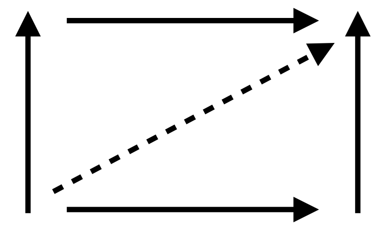


Both discrete and continuous
probabilistic systems
are
coalgebras

$\mathcal{D}$ on
$\mathbf{Sets}$

$\mathcal{G}$ on
$\mathbf{Meas}$





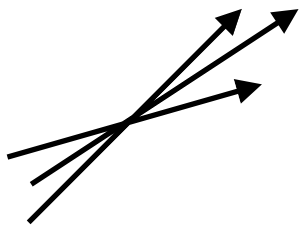
Source Wikipedia, by Diacritica - Own work

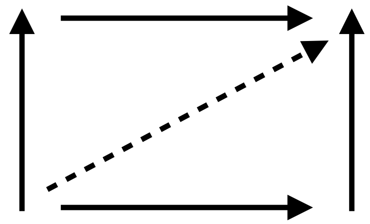
Both discrete and continuous probabilistic systems are coalgebras

$\mathcal{D}$ on
$\mathbf{Sets}$

$\mathcal{G}$ on
$\mathbf{Meas}$

generic notion of behavioural equivalence





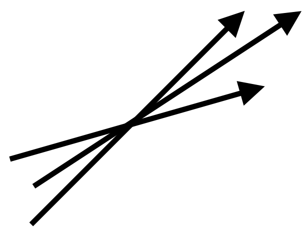
Both discrete and continuous probabilistic systems are coalgebras

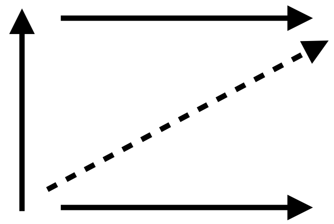
$\mathcal{D}$ on **Sets**

$\mathcal{G}$ on **Meas**

generic notion of behavioural equivalence

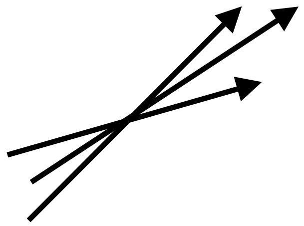
strong, branching-time semantics

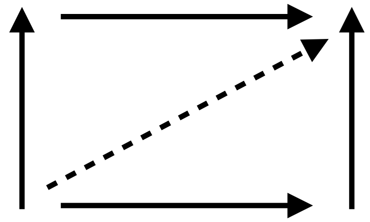




Source Wikipedia, by Diacritica - Own work
CC BY-SA 3.0

Part II
Modelling probabilistic systems for linear-time semantics





Part II

Modelling probabilistic systems
for
linear-time semantics

coalgebraically

Trace semantics

Trace semantics

NFA = LTS + termination

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

Trace semantics

NFA = LTS + termination

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

Are the following systems equivalent?

Trace semantics

NFA = LTS + termination

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

Are the following systems equivalent?

$$\begin{tikzcd}
 & x_1 \arrow[dl, "a"'] \arrow[dr, "a"] & \\
x_2 \arrow[d, "b"'] & & x_3 \arrow[d, "c"] \\
x_4 \arrow[d, "1"'] & & x_5 \arrow[d, "1"] \\
* & & *
\end{tikzcd}
\qquad
\begin{tikzcd}
 & y_1 \arrow[d, "a"'] & \\
 & y_2 \arrow[dl, "b"'] \arrow[dr, "c"] & \\
y_3 \arrow[d] & & y_4 \arrow[d] \\
* & & *
\end{tikzcd}$$

Trace semantics

NFA = LTS + termination

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

top states

Are the following systems equivalent?

$$\begin{tikzcd}[column sep=small]
 & x_1 \arrow[dl, "a"'] \arrow[dr, "a"] & \\
x_2 \arrow[d, "b"'] & & x_3 \arrow[d, "c"] \\
x_4 \arrow[d, "1"'] & & x_5 \arrow[d, "1"] \\
* & & *
\end{tikzcd}
\qquad
\begin{tikzcd}[column sep=small]
 & y_1 \arrow[d, "a"'] & \\
 & y_2 \arrow[dl, "b"'] \arrow[dr, "c"] & \\
y_3 \arrow[d] & & y_4 \arrow[d] \\
* & & *
\end{tikzcd}$$

Trace semantics

NFA = LTS + termination

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

top states

Are the following systems equivalent?

$$\begin{tikzcd}[column sep=small]
 & x_1 \arrow[dl, "a"'] \arrow[dr, "a"] & \\
x_2 \arrow[d, "b"'] & & x_3 \arrow[d, "c"] \\
x_4 \arrow[d, "1"'] & & x_5 \arrow[d, "1"] \\
* & & *
\end{tikzcd}
\qquad
\begin{tikzcd}[column sep=small]
 & y_1 \arrow[d, "a"'] & \\
 & y_2 \arrow[dl, "b"'] \arrow[dr, "c"] & \\
y_3 \arrow[d] & & y_4 \arrow[d] \\
* & & *
\end{tikzcd}$$

- no, they are not wrt. bisimilarity
- yes, they are wrt. trace equivalence

Trace semantics

NFA = LTS + termination

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

top states

Are the following systems equivalent?

$$\begin{tikzcd}
 & x_1 \arrow[dl, "a"'] \arrow[dr, "a"] & \\
x_2 \arrow[d, "b"'] & & x_3 \arrow[d, "c"] \\
x_4 \arrow[d, "1"'] & & x_5 \arrow[d, "1"] \\
* & & *
\end{tikzcd}
\qquad
\begin{tikzcd}
 & y_1 \arrow[d, "a"'] & \\
 & y_2 \arrow[dl, "b"'] \arrow[dr, "c"] & \\
y_3 \arrow[d] & & y_4 \arrow[d] \\
* & & *
\end{tikzcd}$$

- no, they are not wrt. bisimilarity
- yes, they are wrt. trace equivalence

$$\mathrm{tr}(x_1) = \mathrm{tr}(y_1) = \{ab, ac\}$$

Trace semantics

NFA = LTS + termination

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

top states

Are the following systems equivalent?

$$
\begin{tikzcd}[column sep=small]
 & x_1 \arrow[dl, "a"'] \arrow[dr, "a"] & \\
x_2 \arrow[d, "b"'] & & x_3 \arrow[d, "c"] \\
x_4 \arrow[d, "1"'] & & x_5 \arrow[d, "1"] \\
* & & *
\end{tikzcd}
\qquad
\begin{tikzcd}[column sep=small]
 & y_1 \arrow[d, "a"'] & \\
 & y_2 \arrow[dl, "b"'] \arrow[dr, "c"] & \\
y_3 \arrow[d] & & y_4 \arrow[d] \\
* & & *
\end{tikzcd}
$$

$\mathrm{tr}\colon X \to \mathcal{P}(A^*)$

- no, they are not wrt. bisimilarity
- yes, they are wrt. trace equivalence

$\mathrm{tr}(x_1) = \mathrm{tr}(y_1) = \{ab, ac\}$

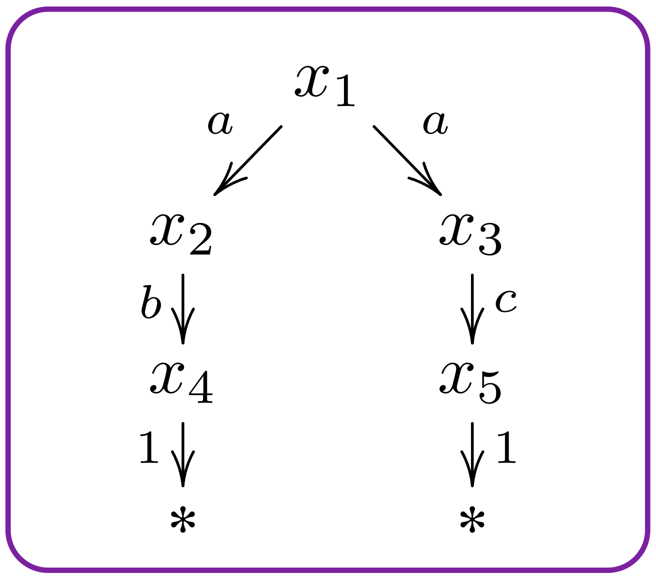
Trace semantics

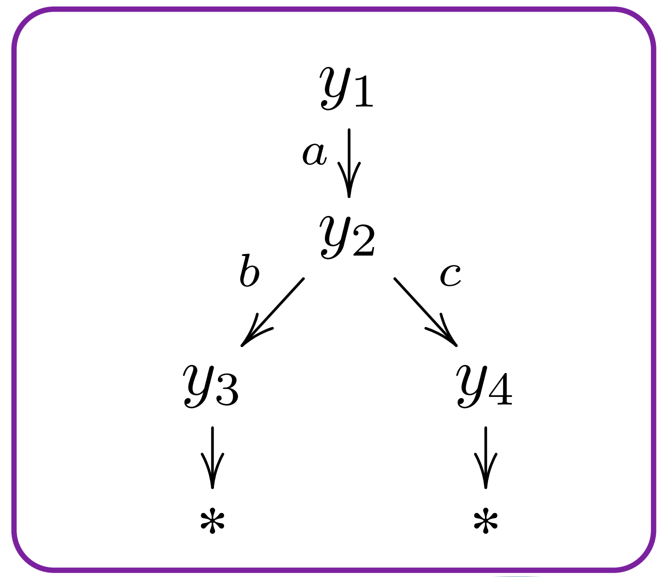
NFA = LTS + termination

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

top states

Are the following systems equivalent?





$\mathrm{tr}\colon X \to \mathcal{P}(A^*)$

- no, they are not wrt. bisimilarity
- yes, they are wrt. trace equivalence

kernel of the trace map

$\mathrm{tr}(x_1) = \mathrm{tr}(y_1) = \{ab, ac\}$

Trace semantics

Trace semantics

Generative PTS

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

Trace semantics

Generative PTS

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

Are the following systems equivalent?

top states

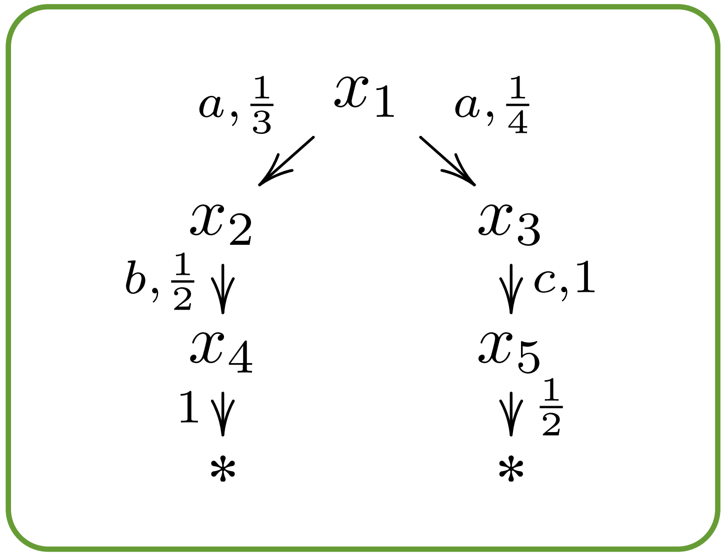
Trace semantics

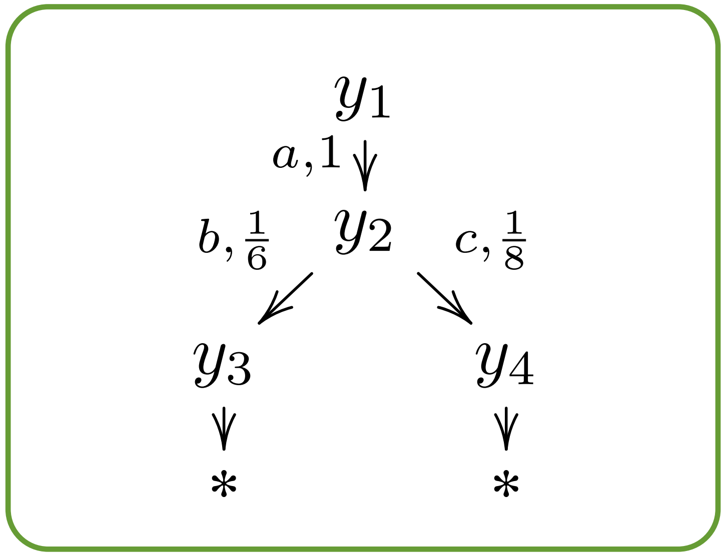
Generative PTS

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

Are the following systems equivalent?

top states





Trace semantics

Generative PTS

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

top states

Are the following systems equivalent?

$$\begin{tikzcd}
 & x_1 \arrow[dl, "{a,\frac{1}{3}}"'] \arrow[dr, "{a,\frac{1}{4}}"] & \\
x_2 \arrow[d, "{b,\frac{1}{2}}"'] & & x_3 \arrow[d, "{c,1}"] \\
x_4 \arrow[d, "1"'] & & x_5 \arrow[d, "\frac{1}{2}"] \\
* & & *
\end{tikzcd}
\qquad
\begin{tikzcd}
 & y_1 \arrow[d, "{a,1}"'] & \\
 & y_2 \arrow[dl, "{b,\frac{1}{6}}"'] \arrow[dr, "{c,\frac{1}{8}}"] & \\
y_3 \arrow[d] & & y_4 \arrow[d] \\
* & & *
\end{tikzcd}$$

- different wrt. bisimilarity
- equivalent wrt. trace equivalence

Trace semantics

Generative PTS

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

top states

Are the following systems equivalent?

$$\begin{tikzcd}
 & x_1 \arrow[dl, "{a, \frac{1}{3}}"'] \arrow[dr, "{a, \frac{1}{4}}"] & \\
x_2 \arrow[d, "{b, \frac{1}{2}}"'] & & x_3 \arrow[d, "{c, 1}"] \\
x_4 \arrow[d, "1"'] & & x_5 \arrow[d, "\frac{1}{2}"] \\
* & & *
\end{tikzcd}
\qquad
\begin{tikzcd}
 & y_1 \arrow[d, "{a, 1}"'] & \\
 & y_2 \arrow[dl, "{b, \frac{1}{6}}"'] \arrow[dr, "{c, \frac{1}{8}}"] & \\
y_3 \arrow[d] & & y_4 \arrow[d] \\
* & & *
\end{tikzcd}$$

- different wrt. bisimilarity
- equivalent wrt. trace equivalence

$$\mathrm{tr}(x_1) = \mathrm{tr}(y_1) = \left(ab \mapsto \frac{1}{6}, ac \mapsto \frac{1}{8}\right)$$

Trace semantics

Generative PTS

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

top states

Are the following systems equivalent?

$$
\begin{tikzcd}
 & x_1 \arrow[dl, "{a,\frac{1}{3}}"'] \arrow[dr, "{a,\frac{1}{4}}"] & \\
x_2 \arrow[d, "{b,\frac{1}{2}}"'] & & x_3 \arrow[d, "{c,1}"] \\
x_4 \arrow[d, "1"'] & & x_5 \arrow[d, "\frac{1}{2}"] \\
* & & *
\end{tikzcd}
\qquad
\begin{tikzcd}
 & y_1 \arrow[d, "{a,1}"'] & \\
 & y_2 \arrow[dl, "{b,\frac{1}{6}}"'] \arrow[dr, "{c,\frac{1}{8}}"] & \\
y_3 \arrow[d] & & y_4 \arrow[d] \\
* & & *
\end{tikzcd}
$$

- different wrt. bisimilarity
- equivalent wrt. trace equivalence

kernel of the trace map

$$\mathrm{tr}(x_1) = \mathrm{tr}(y_1) = \left(ab \mapsto \frac{1}{6}, ac \mapsto \frac{1}{8}\right)$$

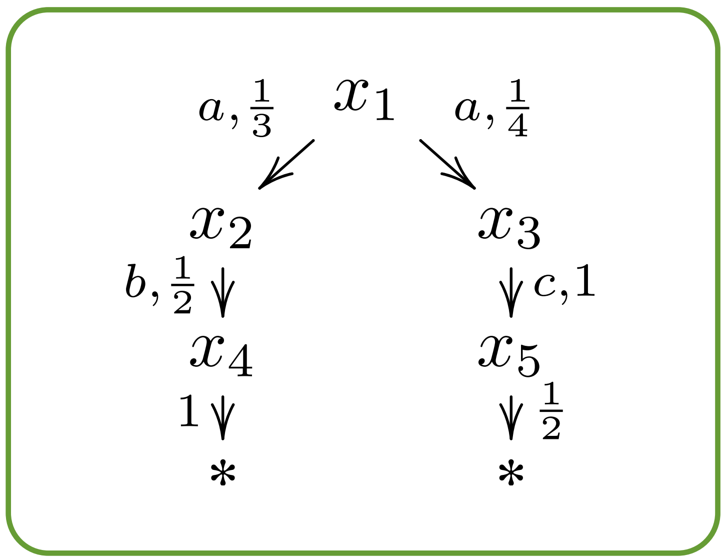
Trace semantics

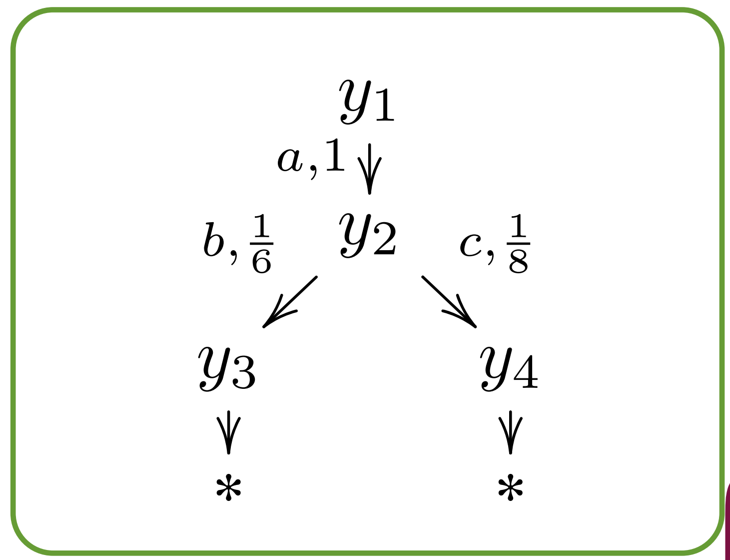
Generative PTS

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

top states

Are the following systems equivalent?





$\mathrm{tr}\colon X \to \mathcal{D}(A^*)$

kernel of the trace map

- different wrt. bisimilarity
- equivalent wrt. trace equivalence

$$\mathrm{tr}(x_1) = \mathrm{tr}(y_1) = \left(ab \mapsto \frac{1}{6}, ac \mapsto \frac{1}{8}\right)$$

Trace semantics coalgebraically?

NFA / LTS

Two ideas:

Trace semantics coalgebraically?

Two ideas: NFA / LTS

(1) unfold branching + transitions on words

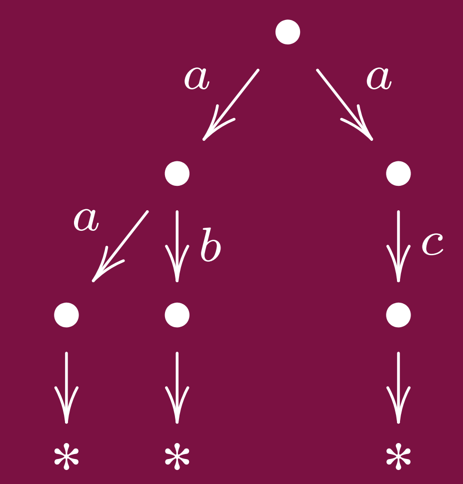
(2) trace = bisimilarity after determinisation

Trace semantics coalgebraically?

NFA / LTS

Two ideas:

(1) unfold branching + transitions on words



(2) trace = bisimilarity after determinisation

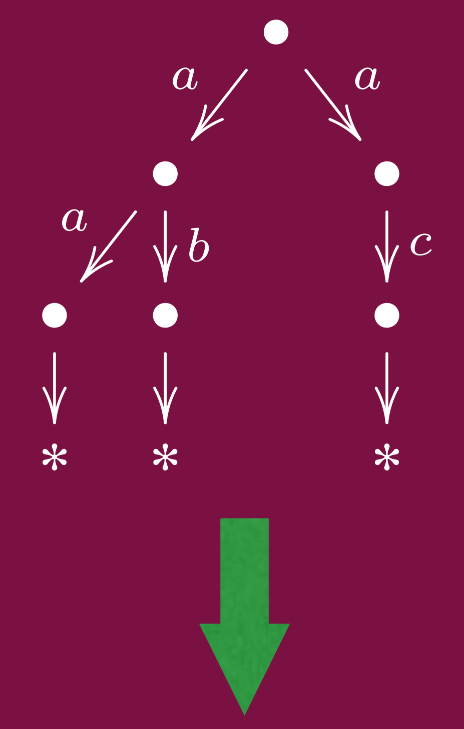
Trace semantics coalgebraically?

NFA / LTS

Two ideas:

(1) unfold branching + transitions on words

(2) trace = bisimilarity after determinisation



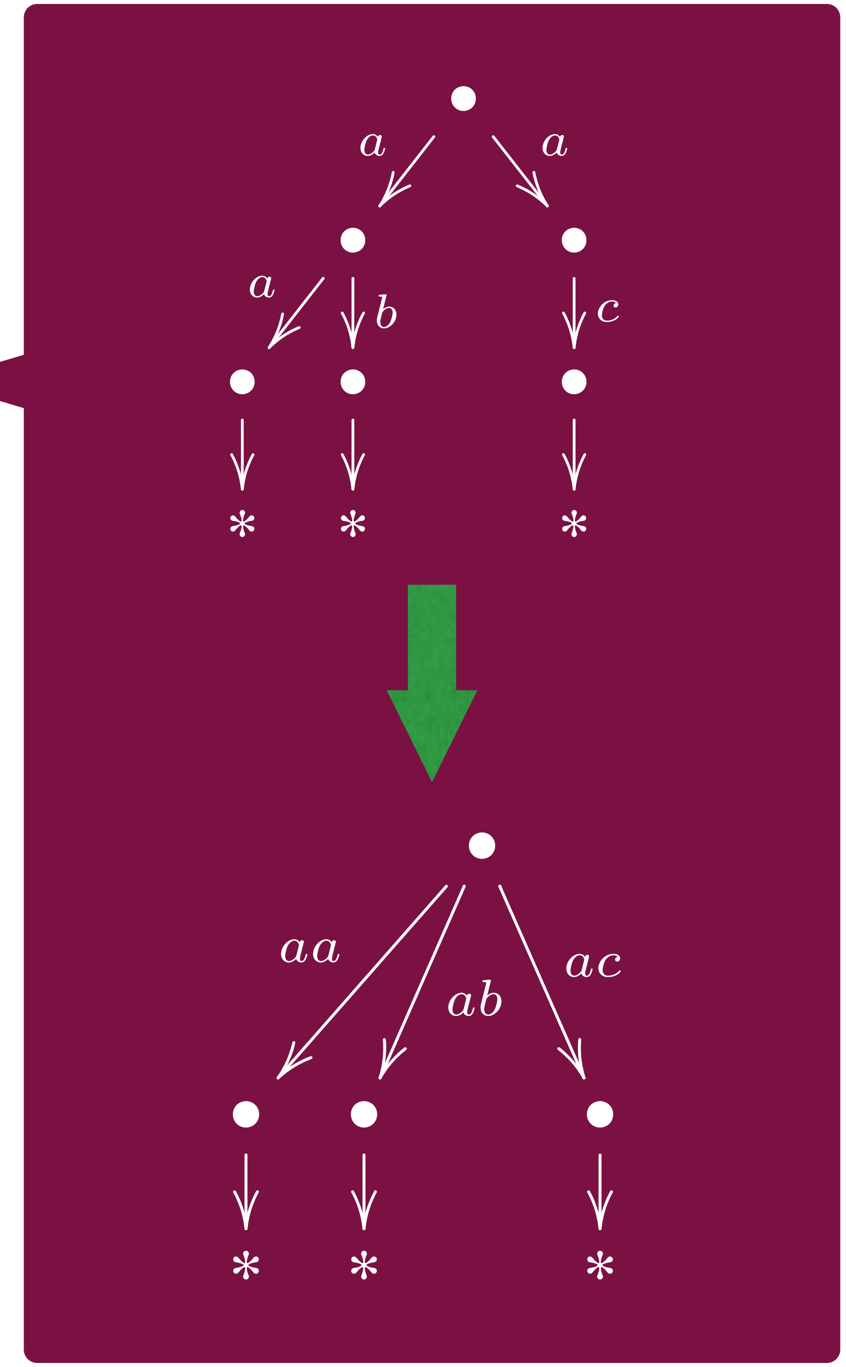
Trace semantics coalgebraically?

NFA / LTS

Two ideas:

(1) unfold branching + transitions on words

(2) trace = bisimilarity after determinisation

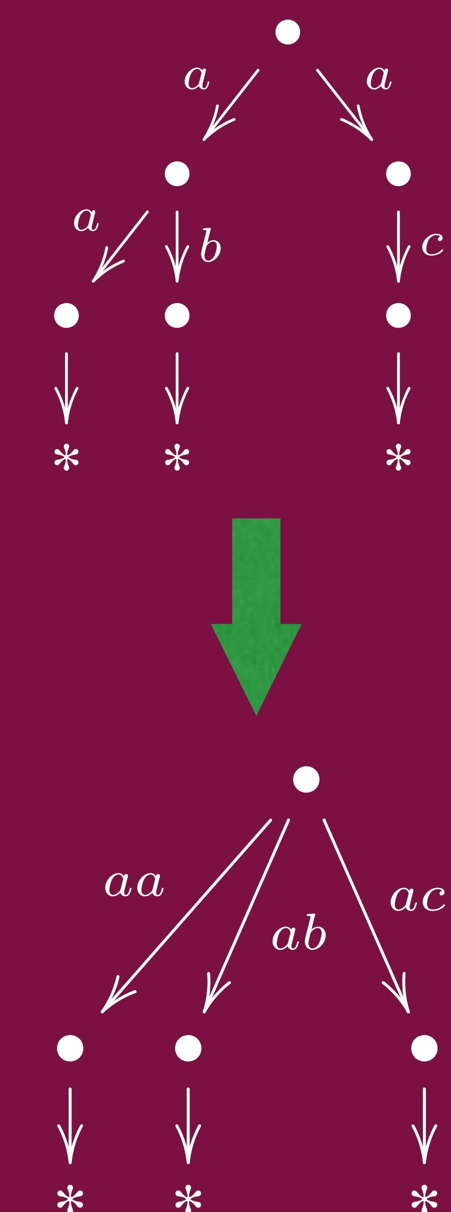


Trace semantics coalgebraically?

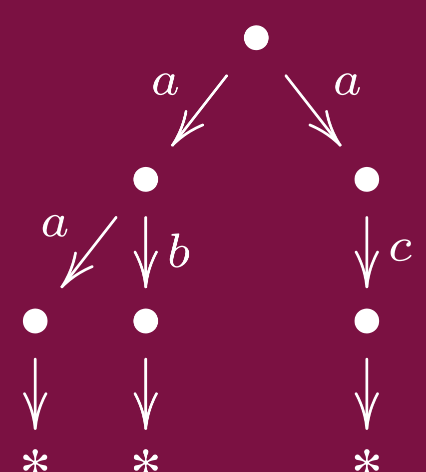
NFA / LTS

Two ideas:

(1) unfold branching + transitions on words



(2) trace = bisimilarity after determinisation

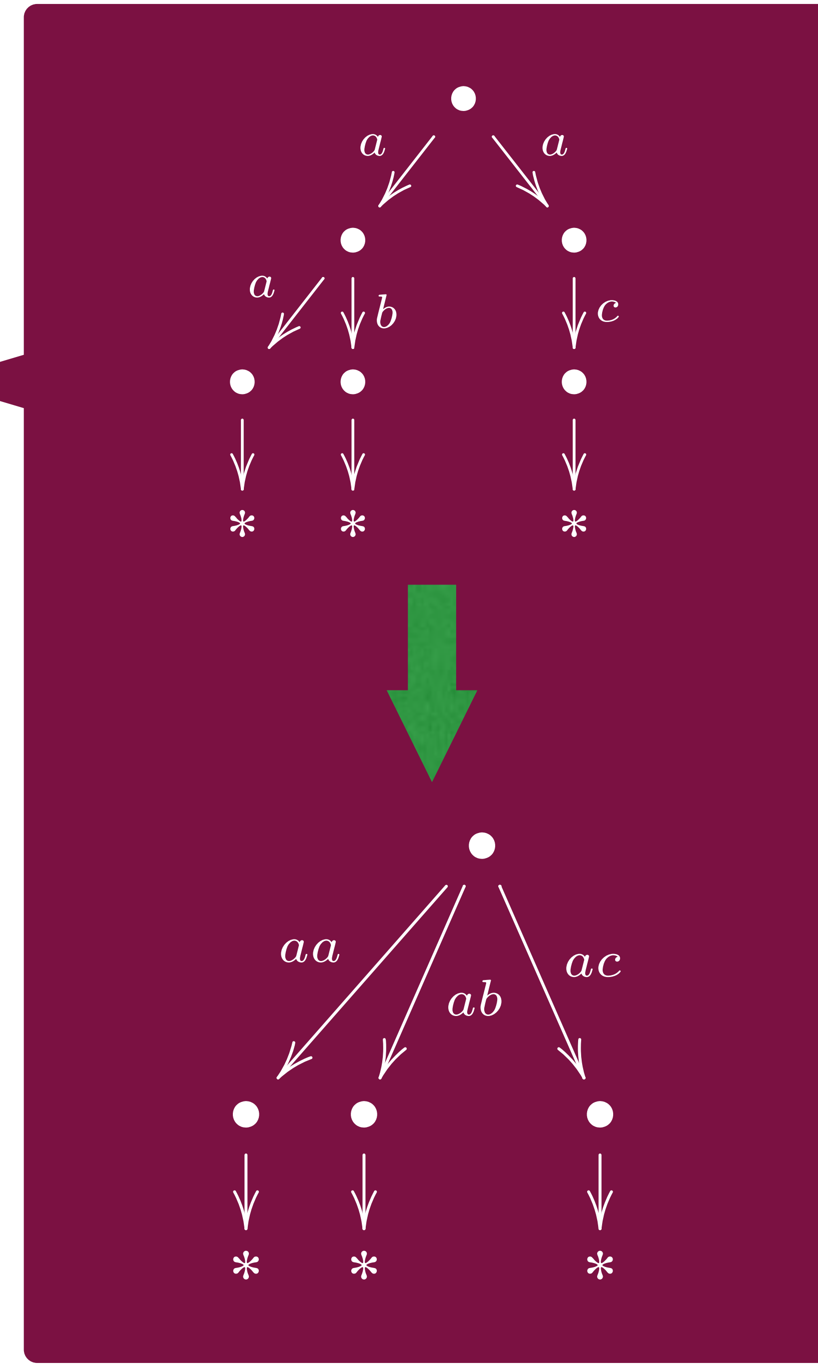


Trace semantics coalgebraically?

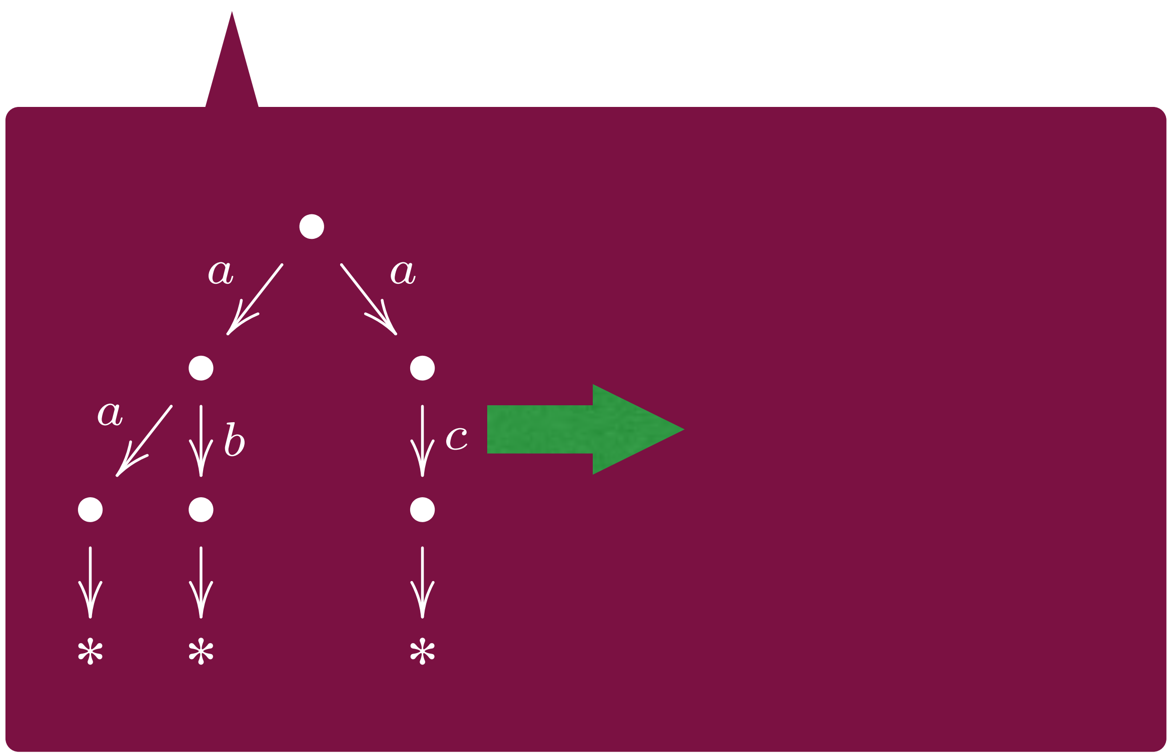
NFA / LTS

Two ideas:

(1) unfold branching + transitions on words



(2) trace = bisimilarity after determinisation

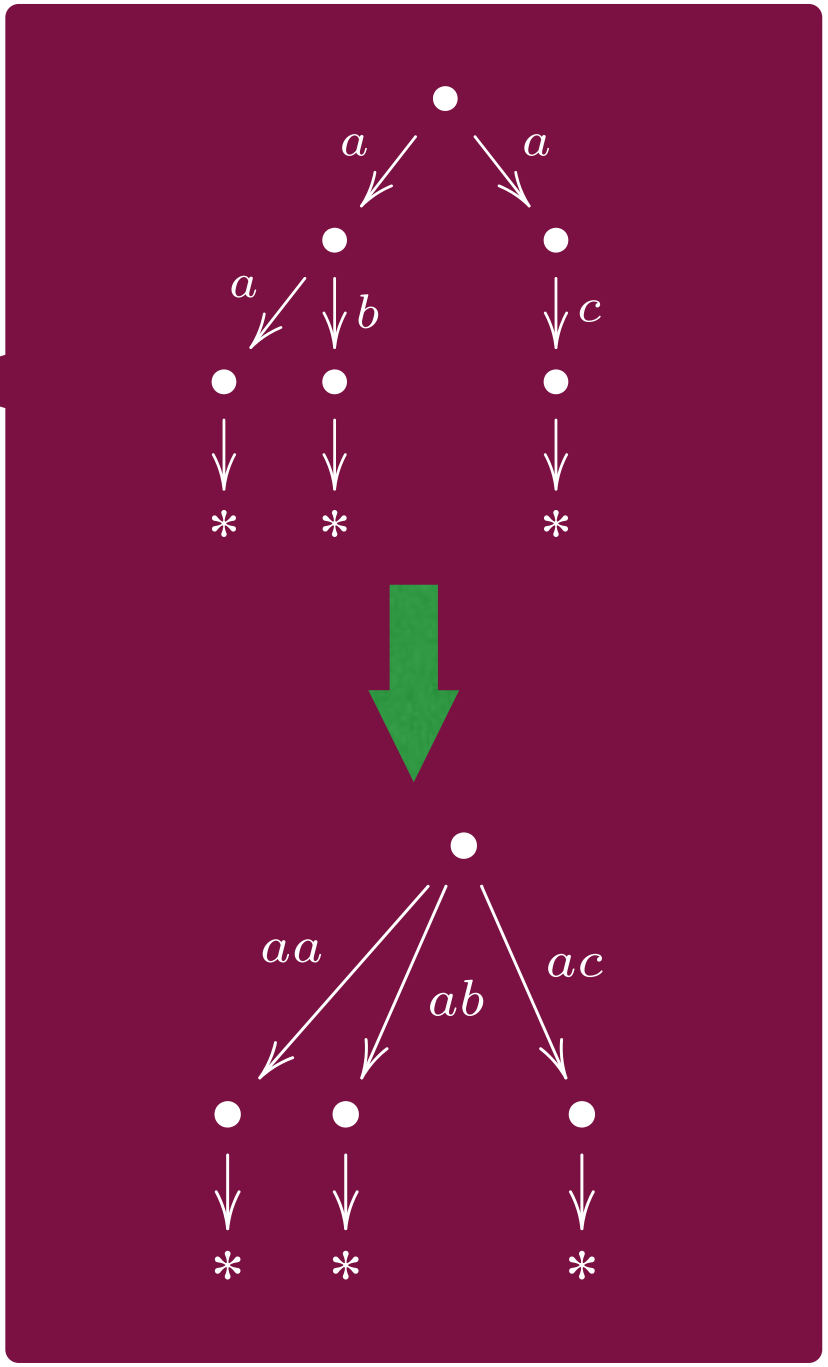


Trace semantics coalgebraically?

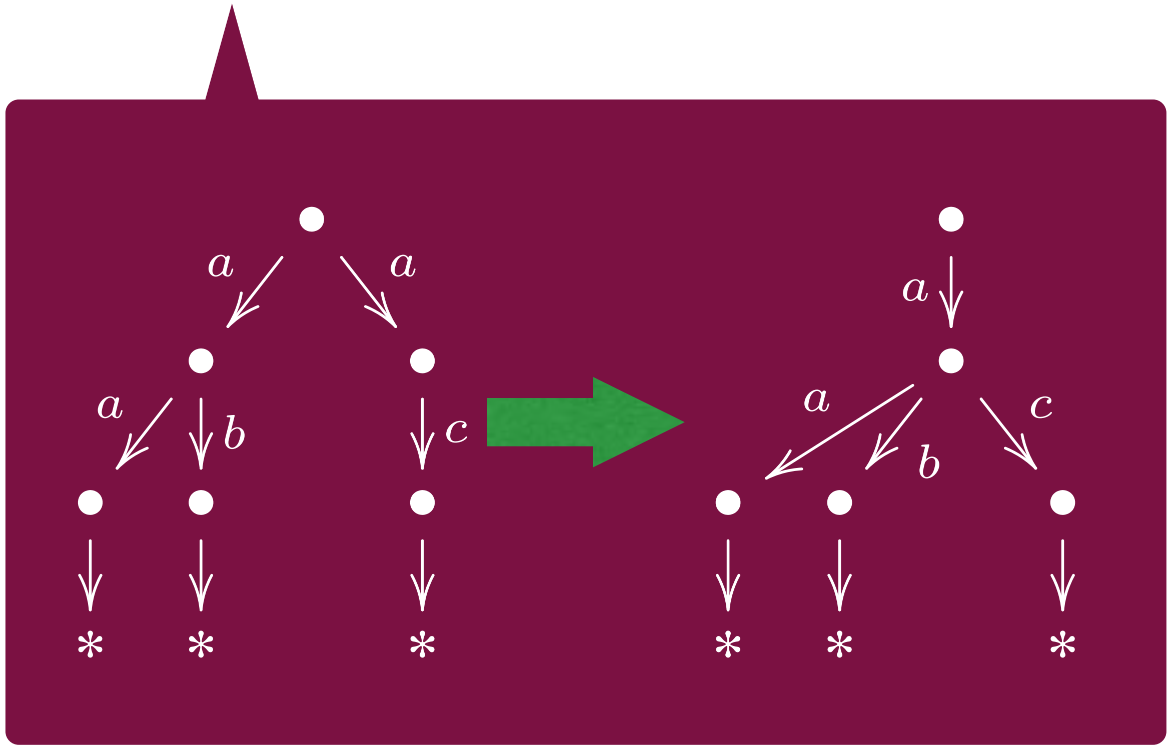
NFA / LTS

Two ideas:

(1) unfold branching + transitions on words



(2) trace = bisimilarity after determinisation

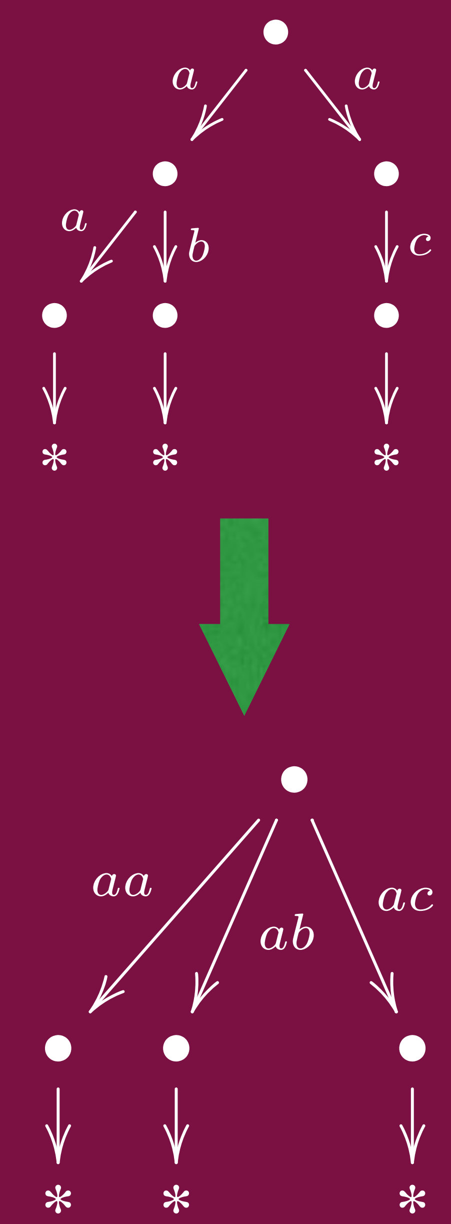


Trace semantics coalgebraically?

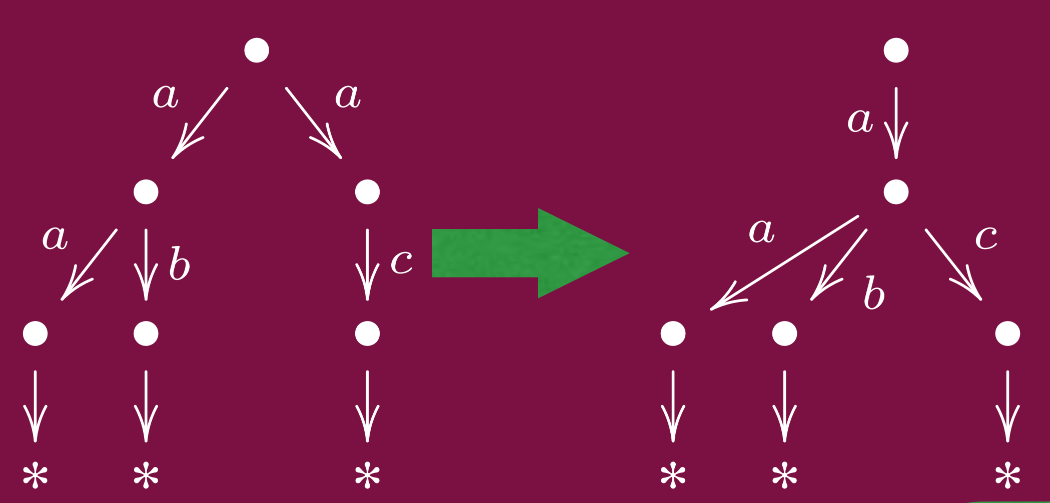
NFA / LTS

Two ideas:

(1) unfold branching + transitions on words



(2) trace = bisimilarity after determinisation



monads !

Trace semantics coalgebraically

we need to move out of $\mathbf{Sets}$

Trace semantics coalgebraically

we need to move out of $\mathbf{Sets}$

Two approaches:

(1) modelling in a Kleisli category

(2) modelling in an Eilenberg-Moore category

Trace semantics coalgebraically

we need to move out of $\mathbf{Sets}$

Two approaches:

(1) modelling in a Kleisli category

(2) modelling in an Eilenberg-Moore category

of a monad T

Trace semantics coalgebraically

we need to move out of $\mathbf{Sets}$

Hasuo, Jacobs, S. LMCS '07

Two approaches:

(1) modelling in a Kleisli category

(2) modelling in an Eilenberg-Moore category

of a monad T

Trace semantics coalgebraically

we need to move out of $\mathbf{Sets}$

Hasuo, Jacobs, S. LMCS ’07

Two approaches:

Silva, Bonchi, Bonsangue, Rutten FSTTCS’10

(1) modelling in a Kleisli category

(2) modelling in an Eilenberg-Moore category

of a monad T

Trace semantics coalgebraically

we need to move out of $\mathbf{Sets}$

Hasuo, Jacobs, S. LMCS '07

Two approaches:

(1) modelling in a Kleisli category

Silva, Bonchi, Bonsangue, Rutten FSTTCS'10

(2) modelling in an Eilenberg-Moore category

of a monad T

we can connect (1) and (2)

Trace semantics coalgebraically

we need to move out of $\mathbf{Sets}$

Hasuo, Jacobs, S. LMCS '07

Two approaches:

(1) modelling in a Kleisli category

Silva, Bonchi, Bonsangue, Rutten FSTTCS'10

(2) modelling in an Eilenberg-Moore category

of a monad T

Jacobs, Silva, S. JCSS'15

we can connect (1) and (2)

Trace semantics coalgebraically

we need to move out of $\mathbf{Sets}$

Two approaches:

(1) modelling in a Kleisli category

(2) modelling in an Eilenberg-Moore category

of a monad T

Trace semantics coalgebraically

we need to move out of $\mathbf{Sets}$

Two approaches:

(1) modelling in a Kleisli category

(2) modelling in an Eilenberg-Moore category

of a monad T

$\mu\colon TT \Rightarrow T$

$\eta\colon Id \Rightarrow T$

Trace semantics coalgebraically

we need to move out of **Sets**

Two approaches:

(1) modelling in a Kleisli category

(2) modelling in an Eilenberg-Moore category

of a monad T

$\mu\colon TT \Rightarrow T$

$\eta\colon Id \Rightarrow T$

$\mathcal{P}$ and $\mathcal{D}$ are monads

Trace semantics coalgebraically

we need to move out of **Sets**

Two approaches:

(1) modelling in a Kleisli category

works for TF-coalgebras

(2) modelling in an Eilenberg-Moore category

of a monad T

$\mu\colon TT \Rightarrow T$

$\eta\colon Id \Rightarrow T$

$\mathcal{P}$ and $\mathcal{D}$ are monads

Trace semantics coalgebraically

we need to move out of $\mathbf{Sets}$

Two approaches:

(1) modelling in a Kleisli category

works for *TF*-coalgebras

(2) modelling in an Eilenberg-Moore category

NFA, Generative PTS,..

of a monad *T*

$\mu \colon TT \Rightarrow T$

$\eta \colon Id \Rightarrow T$

$\mathcal{P}$ and $\mathcal{D}$ are monads

Trace semantics coalgebraically

we need to move out of **Sets**

Two approaches:

(1) modelling in a Kleisli category

works for *TF*-coalgebras

(2) modelling in an Eilenberg-Moore category

of a monad *T*

$\mu \colon TT \Rightarrow T$

$\eta \colon Id \Rightarrow T$

$\mathcal{P}$ and $\mathcal{D}$ are monads

NFA, Generative PTS,..

initial algebra = final coalgebra

Trace semantics coalgebraically

we need to move out of $\mathbf{Sets}$

Two approaches:

(1) modelling in a Kleisli category

works for *TF*-coalgebras

(2) modelling in an Eilenberg-Moore category

works for *FT*-coalgebras

NFA, Generative PTS,..

of a monad *T*

$\mu\colon TT \Rightarrow T$

$\eta\colon Id \Rightarrow T$

$\mathcal{P}$ and $\mathcal{D}$ are monads

initial algebra = final coalgebra

Trace semantics coalgebraically

we need to move out of $\mathbf{Sets}$

Two approaches:

(1) modelling in a Kleisli category

works for *TF*-coalgebras

(2) modelling in an Eilenberg-Moore category

works for *FT*-coalgebras

of a monad *T*

$\mu\colon TT \Rightarrow T$

$\eta\colon Id \Rightarrow T$

$\mathcal{P}$ and $\mathcal{D}$ are monads

NFA, Generative PTS,..

initial algebra = final coalgebra

NFA, Reactive PTS,..

Trace semantics coalgebraically

we need to move out of **Sets**

Two approaches:

(1) modelling in a Kleisli category

(2) modelling in an Eilenberg-Moore category

works for *TF*-coalgebras

works for *FT*-coalgebras

of a monad *T*

$\mu \colon TT \Rightarrow T$

$\eta \colon Id \Rightarrow T$

$\mathcal{P}$ and $\mathcal{D}$ are monads

NFA, Generative PTS,..

initial algebra = final coalgebra

NFA, Reactive PTS,..

via generalised determinisation

Trace semantics coalgebraically

we need to move out of $\mathbf{Sets}$

Two approaches:

(1) modelling in a Kleisli category

works for *TF*-coalgebras

(2) modelling in an Eilenberg-Moore category

works for *FT*-coalgebras

of a monad *T*

$\mu \colon TT \Rightarrow T$

$\eta \colon Id \Rightarrow T$

$\mathcal{P}$ and $\mathcal{D}$ are monads

NFA, Generative PTS,..

initial algebra = final coalgebra

NFA, Reactive PTS,..

via generalised determinisation

generalised² determinization connects (1) and (2)

Trace semantics coalgebraically

we need to move out of **Sets**

Trace semantics coalgebraically

we need to move out of $\mathbf{Sets}$

Two approaches:

(1) TF-coalgebra on Sets becomes an $\underline{F}$-coalgebra on $\mathcal{K}l(T)$

(2) FT-coalgebra on Sets becomes an $\underline{F}$-coalgebra on $\mathcal{EM}(T)$

Trace semantics coalgebraically

we need to move out of **Sets**

Two approaches:

(1) TF-coalgebra on Sets becomes an $\underline{F}$-coalgebra on $\mathcal{K}\ell(T)$

(2) FT-coalgebra on Sets becomes an $\underline{F}$-coalgebra on $\mathcal{EM}(T)$ algebras

Trace semantics coalgebraically

we need to move out of **Sets**

Two approaches:

(1) TF-coalgebra on Sets becomes an $\underline{F}$-coalgebra on $\mathcal{K}\ell(T)$

(2) FT-coalgebra on Sets becomes an $\underline{F}$-coalgebra on $\mathcal{EM}(T)$

algebras

$$\begin{array}{c} \mathcal{D}A \\ \downarrow \alpha \\ A \end{array}$$

Trace semantics coalgebraically

we need to move out of **Sets**

Two approaches:

(1) TF-coalgebra on Sets becomes an $\underline{F}$-coalgebra on $\mathcal{K}\ell(T)$

free ones

(2) FT-coalgebra on Sets becomes an $\underline{F}$-coalgebra on $\mathcal{EM}(T)$

algebras

$$\begin{array}{c} \mathcal{D}A \\ \downarrow \alpha \\ A \end{array}$$

Trace semantics coalgebraically

we need to move out of **Sets**

Two approaches:

(1) TF-coalgebra on Sets becomes an F-coalgebra on $\mathcal{K}\ell(T)$

free ones

(2) FT-coalgebra on Sets becomes an F-coalgebra on $\mathcal{EM}(T)$

algebras

$\mathcal{D}A \xrightarrow{\alpha} A$

$\mathcal{D}\mathcal{D}S \xrightarrow{\mu} \mathcal{D}S$

Trace semantics coalgebraically

we need to move out of **Sets**

Two approaches:

(1) TF-coalgebra on Sets becomes an <u>F</u>-coalgebra on $\mathcal{K}\ell(T)$

free ones

(2) FT-coalgebra on Sets becomes an <u>F</u>-coalgebra on $\mathcal{EM}(T)$

algebras

suitable lifting

$$\begin{array}{c} \mathcal{D}A \\ \downarrow{\alpha} \\ A \end{array} \qquad \begin{array}{c} \mathcal{D}\mathcal{D}S \\ \downarrow{\mu} \\ \mathcal{D}S \end{array}$$

Trace semantics coalgebraically

we need to move out of **Sets**

Two approaches:

(1) TF-coalgebra on Sets becomes an F-coalgebra on $\mathcal{Kl}$(T)

free ones

(2) FT-coalgebra on Sets becomes an F-coalgebra on $\mathcal{EM}$(T)

algebras

$$\begin{array}{c} \mathcal{D}A \\ \downarrow \alpha \\ A \end{array} \qquad \begin{array}{c} \mathcal{D}\mathcal{D}S \\ \downarrow \mu \\ \mathcal{D}S \end{array}$$

suitable lifting

Needed: distributive laws!

(1) $FT \Rightarrow TF$

(2) $TF \Rightarrow FT$

Trace semantics coalgebraically

we need to move out of **Sets**

Two approaches:

(1) TF-coalgebra on Sets becomes an F-coalgebra on $\mathcal{K}\ell$(T)

free ones

(2) FT-coalgebra on Sets becomes an F-coalgebra on $\mathcal{EM}$(T)

algebras

$\mathcal{D}A \xrightarrow{\alpha} A$

$\mathcal{D}\mathcal{D}S \xrightarrow{\mu} \mathcal{D}S$

suitable lifting

Needed: distributive laws!

(1) $FT \Rightarrow TF$

(2) $TF \Rightarrow FT$

trace equivalence is behaviour equivalence

Trace semantics coalgebraically

we need to move out of **Sets**

Two approaches:

(1) TF-coalgebra on Sets becomes an $\underline{F}$-coalgebra on $\mathcal{Kl}$(T)

(2) FT-coalgebra on Sets becomes an $\underline{F}$-coalgebra on $\mathcal{EM}$(T)

Needed: distributive laws!

(1) $FT \Rightarrow TF$

(2) $TF \Rightarrow FT$

Trace semantics coalgebraically

we need to move out of **Sets**

Two approaches:

(1) TF-coalgebra on Sets becomes an F-coalgebra on $\mathcal{K}\ell$(T)

(2) FT-coalgebra on Sets becomes an F-coalgebra on $\mathcal{EM}$(T)

Needed: distributive laws!

(1) $FT \Rightarrow TF$

(2) $TF \Rightarrow FT$

$$X \to TFX \to TFTFX \to TTFFX \to TFFX \cdots$$

Trace semantics coalgebraically

we need to move out of **Sets**

Two approaches:

(1) TF-coalgebra on Sets becomes an F-coalgebra on $\mathcal{Kl}$(T)

(2) FT-coalgebra on Sets becomes an F-coalgebra on $\mathcal{EM}$(T)

Needed: distributive laws!

(1) $FT \Rightarrow TF$

(2) $TF \Rightarrow FT$

distributive law

$$X \to TFX \to TFTFX \to TTFFX \to TFFX \cdots$$

Trace semantics coalgebraically

we need to move out of **Sets**

Two approaches:

(1) TF-coalgebra on Sets becomes an $\underline{F}$-coalgebra on $\mathcal{K}\ell(T)$

(2) FT-coalgebra on Sets becomes an $\underline{F}$-coalgebra on $\mathcal{EM}(T)$

Needed: distributive laws!

(1) $FT \Rightarrow TF$

(2) $TF \Rightarrow FT$

distributive law

monad multiplication

$$X \to TFX \to TFTFX \to TTFFX \to TFFX \cdots$$

Trace semantics coalgebraically

we need to move out of $\mathbf{Sets}$

Two approaches:

(1) TF-coalgebra on Sets becomes an $\underline{F}$-coalgebra on $\mathcal{K}\ell$(T)

(2) FT-coalgebra on Sets becomes an $\underline{F}$-coalgebra on $\mathcal{EM}$(T)

Needed: distributive laws!

(1) $FT \Rightarrow TF$

(2) $TF \Rightarrow FT$

Trace semantics coalgebraically

we need to move out of $\mathbf{Sets}$

Two approaches:

(1) TF-coalgebra on Sets becomes an $\underline{F}$-coalgebra on $\mathcal{K}\ell$(T)

(2) FT-coalgebra on Sets becomes an $\underline{F}$-coalgebra on $\mathcal{EM}$(T)

Needed: distributive laws!

(1) $FT \Rightarrow TF$

(2) $TF \Rightarrow FT$

$$TX \to TFTX \to FTTX \to FTX$$

Trace semantics coalgebraically

we need to move out of **Sets**

Two approaches:

(1) TF-coalgebra on Sets becomes an F̲-coalgebra on $\mathcal{Kl}$(T)

(2) FT-coalgebra on Sets becomes an F̲-coalgebra on $\mathcal{EM}$(T)

Needed: distributive laws!

(1) $FT \Rightarrow TF$

(2) $TF \Rightarrow FT$

distributive law

$$TX \to TFTX \to FTTX \to FTX$$

Trace semantics coalgebraically

we need to move out of **Sets**

Two approaches:

(1) TF-coalgebra on Sets becomes an F-coalgebra on $\mathcal{Kl}$(T)

(2) FT-coalgebra on Sets becomes an F-coalgebra on $\mathcal{EM}$(T)

Needed: distributive laws!

(1) $FT \Rightarrow TF$

(2) $TF \Rightarrow FT$

distributive law

monad multiplication

$$TX \to TFTX \to FTTX \to FTX$$

Trace semantics coalgebraically

we need to move out of **Sets**

Two approaches:

(1) TF-coalgebra on Sets becomes an F-coalgebra on $\mathcal{K}\ell(T)$

(2) FT-coalgebra on Sets becomes an F-coalgebra on $\mathcal{EM}(T)$

must be order enriched

Needed: distributive laws!

(1) $FT \Rightarrow TF$

(2) $TF \Rightarrow FT$

distributive law

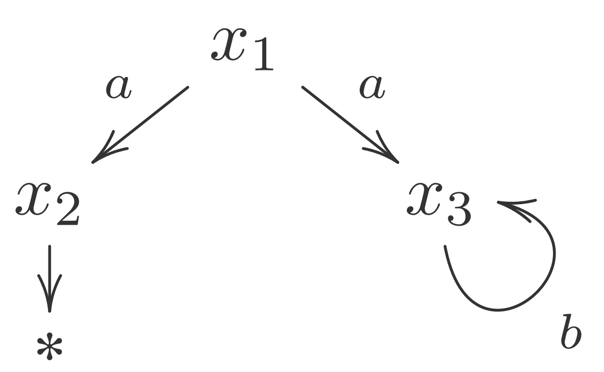
monad multiplication

$$TX \to TFTX \to FTTX \to FTX$$

(1) Traces in Kleisli

NFA

$\mathcal{P}(1 + A \times (-))$



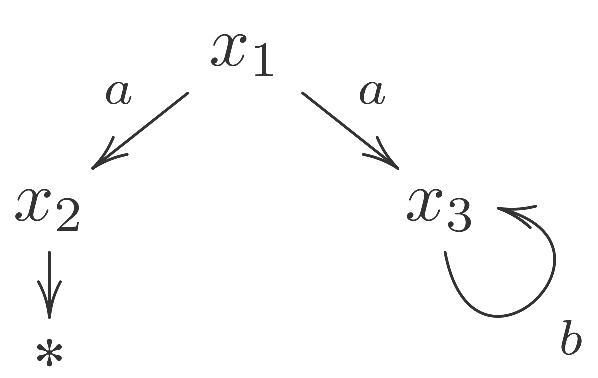
$$\mathrm{tr}(x_1) = \{ab, ac\}$$

$$\mathrm{tr}\colon X \to \mathcal{P}(A^*)$$

(1) Traces in Kleisli

NFA

$\mathcal{P}(1 + A \times (-))$



$$\mathrm{tr}(x_1) = \{ab, ac\}$$

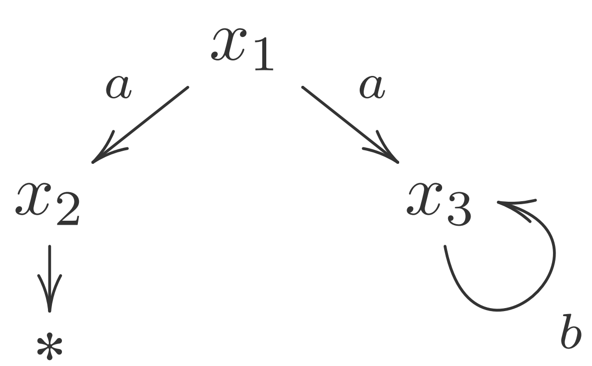
$$\mathrm{tr}\colon X \to \mathcal{P}(A^*)$$

trace equivalence is behaviour equivalence in $\mathcal{K}\ell(\mathcal{P})$

(1) Traces in Kleisli

NFA

$\mathcal{P}(1 + A \times (-))$



$$\mathrm{tr}(x_1) = \{ab, ac\}$$

$$\mathrm{tr}\colon X \to \mathcal{P}(A^*)$$

arrow in $\mathcal{K}\ell(\mathcal{P})$

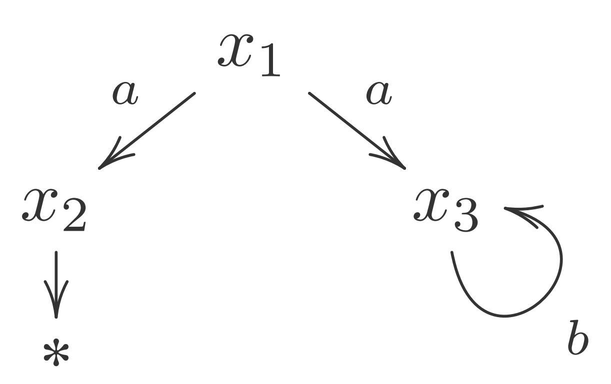
trace equivalence is behaviour equivalence in $\mathcal{K}\ell(\mathcal{P})$

(1) Traces in Kleisli

NFA

$\mathcal{P}(1 + A \times (-))$

lifts to $\mathcal{K}l(\mathcal{P})$
via a distributive law



$$\mathrm{tr}(x_1) = \{ab, ac\}$$

$$\mathrm{tr}\colon X \to \mathcal{P}(A^*)$$

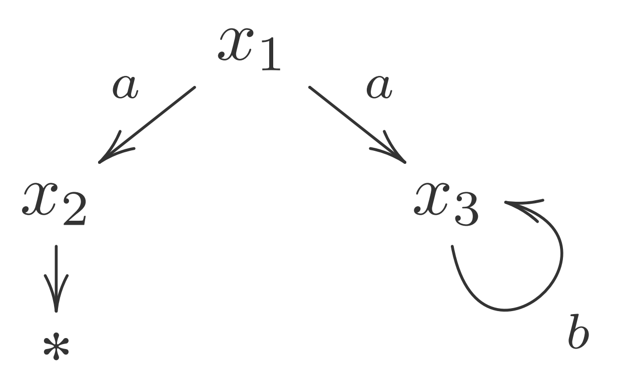
arrow in $\mathcal{K}l(\mathcal{P})$

trace equivalence is behaviour equivalence in $\mathcal{K}l(\mathcal{P})$

(1) Traces in Kleisli

NFA

$\mathcal{P}$ (1 + A x (-))



lifts to $\mathcal{K}\ell(\mathcal{P})$
via a distributive law

$1 + A \times \mathcal{P} \Rightarrow \mathcal{P}(1 + A \times -)$

$\mathrm{tr}(x_1) = \{ab, ac\}$

$\mathrm{tr}\colon X \to \mathcal{P}(A^*)$

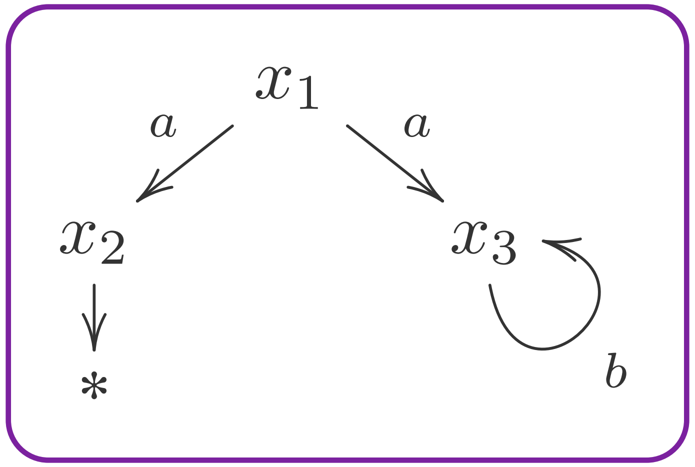
arrow in $\mathcal{K}\ell(\mathcal{P})$

trace equivalence is behaviour equivalence in $\mathcal{K}\ell(\mathcal{P})$

(1) Traces in Kleisli

NFA

$\mathcal{P}(1 + A \times (-))$



lifts to $\mathcal{K}\ell(\mathcal{P})$ via a distributive law

$1 + A \times \mathcal{P} \Rightarrow \mathcal{P}(1 + A \times -)$

$\mathrm{tr}(x_1) = \{ab, ac\}$

$\mathrm{tr}\colon X \to \mathcal{P}(A^*)$

arrow in $\mathcal{K}\ell(\mathcal{P})$

trace equivalence is behaviour equivalence in $\mathcal{K}\ell(\mathcal{P})$

$X \to \mathcal{P}(1 + A \times X) \to \mathcal{P}(1 + A \times \mathcal{P}(1 + A \times X)) \to \mathcal{P}^2(1 + A \times (1 + A \times X)) \to \mathcal{P}(1 + A \times X + A^2 \times X) \cdots$

(1) Traces in Kleisli

$\mathcal{D}$ for $\mathcal{D}_{\leq 1}$

Generative PTS

$\mathcal{D}$ (1 + A x (-))

$$
\begin{tikzcd}
 & x_1 \arrow[dl, "{a,\frac{1}{3}}"'] \arrow[dr, "{a,\frac{1}{4}}"] & \\
x_2 \arrow[d, "{b,\frac{1}{2}}"'] & & x_3 \arrow[d, "{c,1}"] \\
x_4 \arrow[d, "1"'] & & x_5 \arrow[d, "\frac{1}{2}"] \\
* & & *
\end{tikzcd}
$$

(1) Traces in Kleisli

$\mathcal{D}$ for $\mathcal{D}_{\leq 1}$

Generative PTS

$\mathcal{D}$ (1 + A x (-))

$$\begin{tikzcd}
& x_1 \arrow[dl, "{a,\frac{1}{3}}"'] \arrow[dr, "{a,\frac{1}{4}}"] & \\
x_2 \arrow[d, "{b,\frac{1}{2}}"'] & & x_3 \arrow[d, "{c,1}"] \\
x_4 \arrow[d, "1"'] & & x_5 \arrow[d, "\frac{1}{2}"] \\
* & & *
\end{tikzcd}$$

$$\mathrm{tr}(x_1)(ab) = \frac{1}{6} \quad \mathrm{tr}(x_1)(ac) = \frac{1}{8}$$

(1) Traces in Kleisli

$\mathcal{D}$ for $\mathcal{D}_{\leq 1}$

Generative PTS

$\mathcal{D}$ (1 + A x (-))

$$\begin{tikzcd}
 & x_1 \arrow[dl, "{a, \frac{1}{3}}"'] \arrow[dr, "{a, \frac{1}{4}}"] & \\
x_2 \arrow[d, "{b, \frac{1}{2}}"'] & & x_3 \arrow[d, "{c, 1}"] \\
x_4 \arrow[d, "1"'] & & x_5 \arrow[d, "\frac{1}{2}"] \\
* & & *
\end{tikzcd}$$

$$\mathrm{tr}(x_1)(ab) = \frac{1}{6} \qquad \mathrm{tr}(x_1)(ac) = \frac{1}{8}$$

$$\mathrm{tr}\colon X \to \mathcal{D}(A^*)$$

(1) Traces in Kleisli

$\mathcal{D}$ for $\mathcal{D}_{\leq 1}$

Generative PTS

$\mathcal{D}$ (1 + A x (-))

$$\begin{tikzcd}
 & x_1 \arrow[dl, "{a, \frac{1}{3}}"'] \arrow[dr, "{a, \frac{1}{4}}"] & \\
x_2 \arrow[d, "{b, \frac{1}{2}}"'] & & x_3 \arrow[d, "{c, 1}"] \\
x_4 \arrow[d, "1"'] & & x_5 \arrow[d, "\frac{1}{2}"] \\
* & & *
\end{tikzcd}$$

$$\mathrm{tr}(x_1)(ab) = \frac{1}{6} \qquad \mathrm{tr}(x_1)(ac) = \frac{1}{8}$$

$$\mathrm{tr}\colon X \to \mathcal{D}(A^*)$$

trace equivalence is behaviour equivalence in $\mathcal{K}l(\mathcal{D})$

(1) Traces in Kleisli

$\mathcal{D}$ for $\mathcal{D}_{\leq 1}$

Generative PTS

$\mathcal{D}$ (1 + A x (-))

$$\begin{tikzcd}[column sep=small] & x_1 \arrow[dl, "{a, \frac{1}{3}}"'] \arrow[dr, "{a, \frac{1}{4}}"] & \\ x_2 \arrow[d, "{b, \frac{1}{2}}"'] & & x_3 \arrow[d, "{c, 1}"] \\ x_4 \arrow[d, "1"'] & & x_5 \arrow[d, "\frac{1}{2}"] \\ * & & * \end{tikzcd}$$

$$\mathrm{tr}(x_1)(ab) = \frac{1}{6} \qquad \mathrm{tr}(x_1)(ac) = \frac{1}{8}$$

$$\mathrm{tr}\colon X \to \mathcal{D}(A^*)$$

arrow in $\mathcal{K}l(\mathcal{D})$

trace equivalence is behaviour equivalence in $\mathcal{K}l(\mathcal{D})$

(1) Traces in Kleisli

$\mathcal{D}$ for $\mathcal{D}_{\leq 1}$

Generative PTS

$\mathcal{D}$ (1 + A x (-))

lifts to $\mathcal{Kl}(\mathcal{D})$
via a distributive law

$$\begin{tikzcd}[column sep=small]
& x_1 \arrow[dl, "{a, \frac{1}{3}}"'] \arrow[dr, "{a, \frac{1}{4}}"] & \\
x_2 \arrow[d, "{b, \frac{1}{2}}"'] & & x_3 \arrow[d, "{c, 1}"] \\
x_4 \arrow[d, "1"'] & & x_5 \arrow[d, "\frac{1}{2}"] \\
* & & *
\end{tikzcd}$$

$$\mathrm{tr}(x_1)(ab) = \frac{1}{6} \qquad \mathrm{tr}(x_1)(ac) = \frac{1}{8}$$

$$\mathrm{tr}\colon X \to \mathcal{D}(A^*)$$

arrow in $\mathcal{Kl}(\mathcal{D})$

trace equivalence is behaviour equivalence in $\mathcal{Kl}(\mathcal{D})$

(1) Traces in Kleisli

$\mathcal{D}$ for $\mathcal{D}_{\leq 1}$

Generative PTS

$\mathcal{D}$ (1 + A x (-))

$$\begin{tikzcd}
& x_1 \arrow[dl, "{a,\frac{1}{3}}"'] \arrow[dr, "{a,\frac{1}{4}}"] & \\
x_2 \arrow[d, "{b,\frac{1}{2}}"'] & & x_3 \arrow[d, "{c,1}"] \\
x_4 \arrow[d, "1"'] & & x_5 \arrow[d, "\frac{1}{2}"] \\
* & & *
\end{tikzcd}$$

lifts to $\mathcal{K}\ell(\mathcal{D})$
via a distributive law

$$1 + A \times \mathcal{D} \Rightarrow \mathcal{D}(1 + A \times -)$$

$$\mathrm{tr}(x_1)(ab) = \frac{1}{6} \qquad \mathrm{tr}(x_1)(ac) = \frac{1}{8}$$

$$\mathrm{tr}\colon X \to \mathcal{D}(A^*)$$

arrow in $\mathcal{K}\ell(\mathcal{D})$

trace equivalence is behaviour equivalence in $\mathcal{K}\ell(\mathcal{D})$

(1) Traces in Kleisli

$\mathcal{D}$ for $\mathcal{D}_{\leq 1}$

$1 + A \times \mathcal{D} \Rightarrow \mathcal{D}(1 + A \times -)$

Generative PTS

$\mathcal{D}$ (1 + A x (-))

lifts to $\mathcal{K}\ell(\mathcal{D})$ via a distributive law

$$\begin{tikzcd}
 & x_1 \arrow[dl, "a, \frac{1}{3}"'] \arrow[dr, "a, \frac{1}{4}"] & \\
x_2 \arrow[d, "b, \frac{1}{2}"'] & & x_3 \arrow[d, "c, 1"] \\
x_4 \arrow[d, "1"'] & & x_5 \arrow[d, "\frac{1}{2}"] \\
* & & *
\end{tikzcd}$$

$$\mathrm{tr}(x_1)(ab) = \frac{1}{6} \qquad \mathrm{tr}(x_1)(ac) = \frac{1}{8}$$

$\mathrm{tr}\colon X \to \mathcal{D}(A^*)$

arrow in $\mathcal{K}\ell(\mathcal{D})$

trace equivalence is behaviour equivalence in $\mathcal{K}\ell(\mathcal{D})$

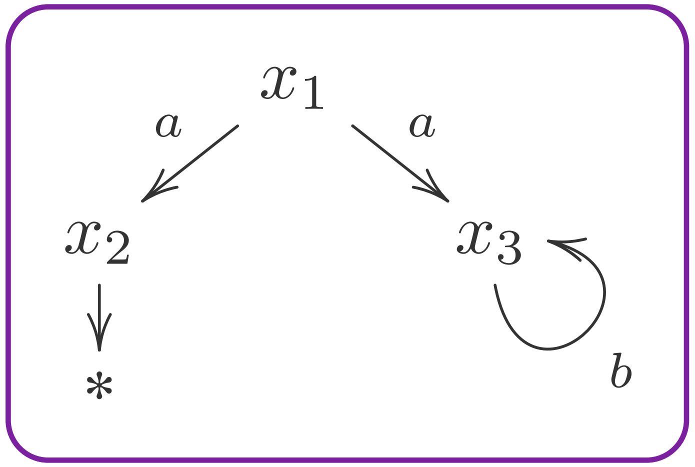
$$X \to \mathcal{D}(1 + A \times X) \to \mathcal{D}(1 + A \times \mathcal{D}(1 + A \times X)) \to \mathcal{D}^2(1 + A \times (1 + A \times X)) \to \mathcal{D}(1 + A \times X + A^2 \times X) \cdots$$

(2) Traces via determinisation

(2) Traces via determinisation

NFA

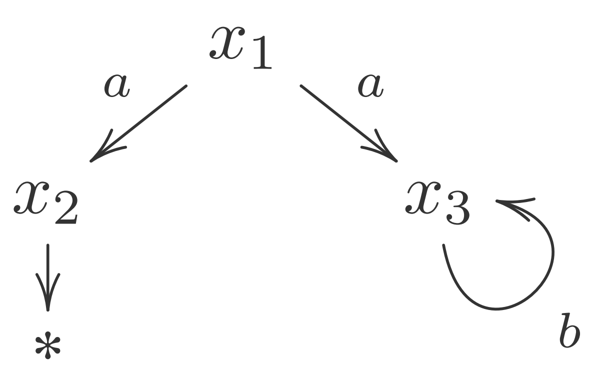
$2 \times \mathcal{P}^A$



(2) Traces via determinisation

NFA

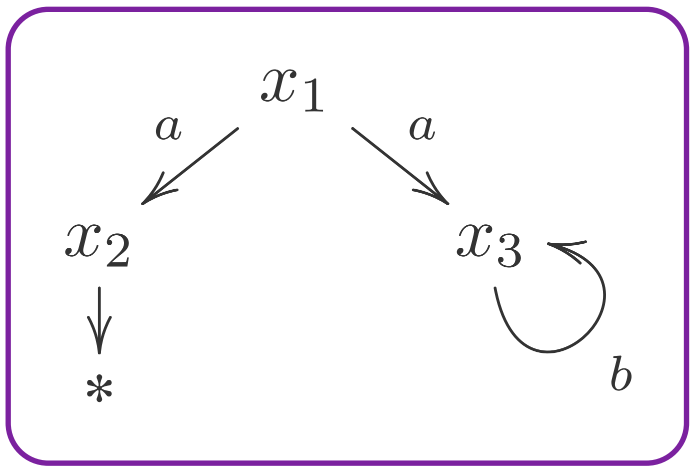
$2 \times \mathcal{P}^A$



(2) Traces via determinisation

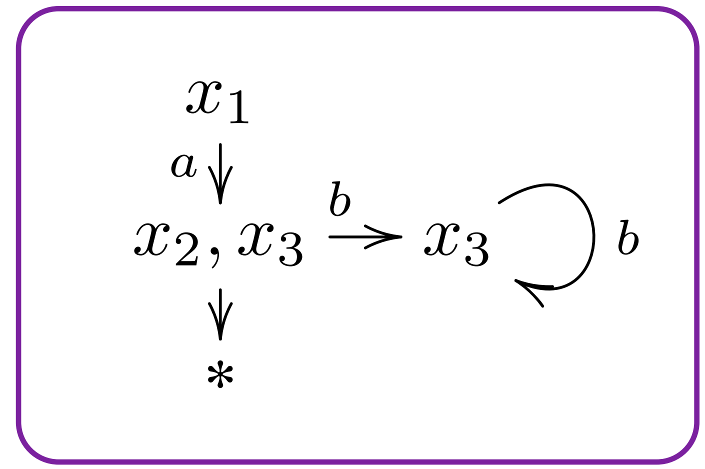
NFA

$2 \times \mathcal{P}^A$



DFA

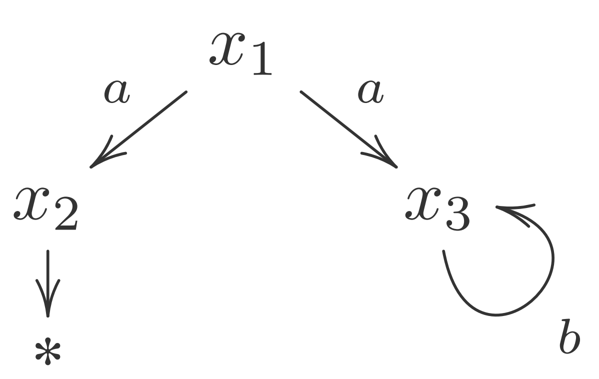
$2 \times (-)^A$ states $\mathcal{P}(-)$



(2) Traces via determinisation

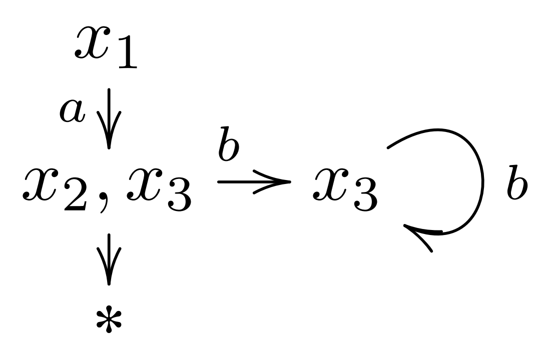
NFA

$2 \times \mathcal{P}^A$



DFA

$2 \times (-)^A$ states $\mathcal{P}(-)$

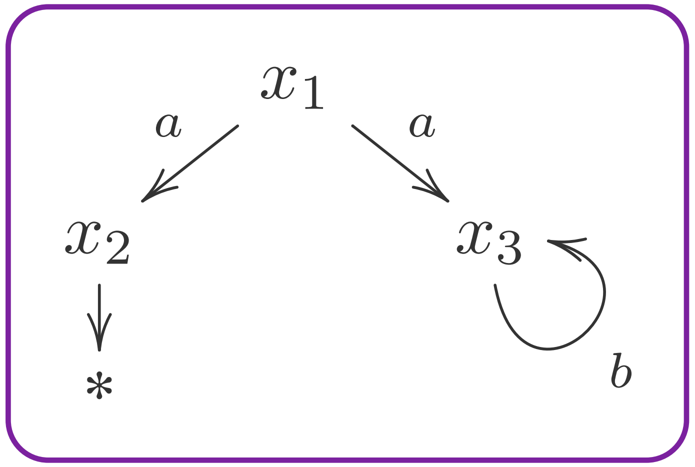


trace = bisimilarity after determinisation

(2) Traces via determinisation

NFA

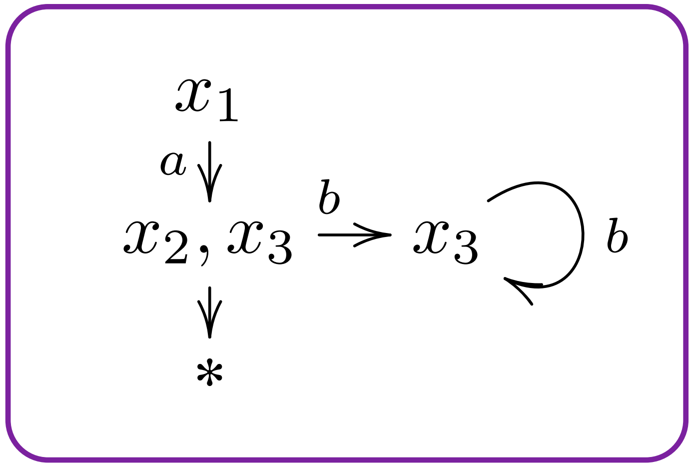
$2 \times \mathcal{P}^A$



DFA

$2 \times (-)^A$

states $\mathcal{P}(-)$



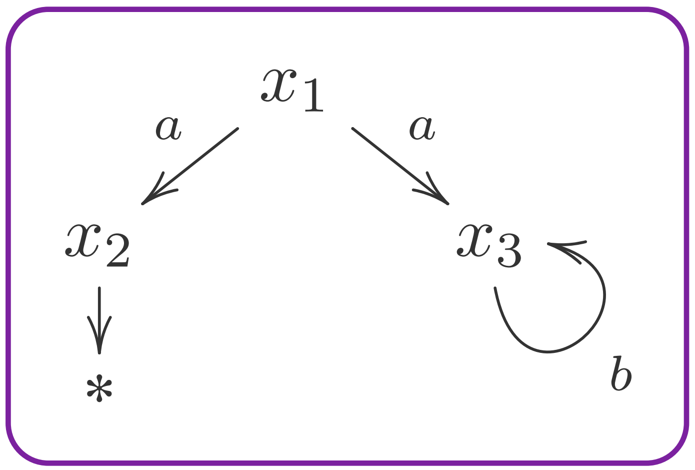
trace = bisimilarity after determinisation

Happens in $\mathcal{EM}(\mathcal{P})$

(2) Traces via determinisation

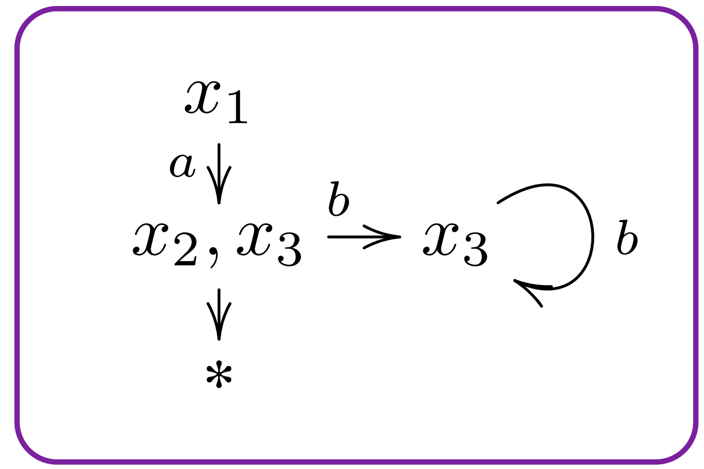
NFA

$2 \times \mathcal{P}^A$



DFA

$2 \times (\text{-})^A$ states $\mathcal{P}(\text{-})$



trace = bisimilarity after determinisation

Happens in $\mathcal{EM}(\mathcal{P})$

join semilattices

(2) Traces via determinisation

(2) Traces via determinisation

Generative PTS

$\mathcal{D}\,(1 + A \times (-))$

$$
\begin{tikzcd}
 & x_1 \arrow[dl, "{a, \frac{1}{2}}"'] \arrow[dr, "{a, \frac{1}{2}}"] & \\
x_2 \arrow[d, "{b, 1}"'] & & x_3 \arrow[d, "{c, 1}"] \\
x_4 \arrow[d, "1"'] & & x_5 \arrow[d, "1"] \\
* & & *
\end{tikzcd}
$$

(2) Traces via determinisation

Generative PTS

$\mathcal{D}(1 + A \times (-))$

$$\begin{tikzcd} & x_1 \arrow[dl, "{a,\frac{1}{2}}"'] \arrow[dr, "{a,\frac{1}{2}}"] & \\ x_2 \arrow[d, "{b,1}"'] & & x_3 \arrow[d, "{c,1}"] \\ x_4 \arrow[d, "1"'] & & x_5 \arrow[d, "1"] \\ * & & * \end{tikzcd}$$

(2) Traces via determinisation

Generative PTS

$\mathcal{D}(1 + A \times (-))$

$$\begin{tikzcd}
& x_1 \arrow[dl, "{a,\frac{1}{2}}"'] \arrow[dr, "{a,\frac{1}{2}}"] & \\
x_2 \arrow[d, "{b,1}"'] & & x_3 \arrow[d, "{c,1}"] \\
x_4 \arrow[d, "1"'] & & x_5 \arrow[d, "1"] \\
* & & *
\end{tikzcd}$$

DFA

$[0,1] \times (-)^A$ states $\mathcal{D}(-)$

$$\begin{tikzcd}
& x_1 \arrow[d, "a"'] & \\
& \frac{1}{2}x_2 + \frac{1}{2}x_3 \arrow[dl, "b"'] \arrow[dr, "c"] & \\
\frac{1}{2}x_4 \arrow[d, "\frac{1}{2}"'] & & \frac{1}{2}x_5 \arrow[d, "\frac{1}{2}"] \\
* & & *
\end{tikzcd}$$

(2) Traces via determinisation

Generative PTS

$\mathcal{D}\,(1 + A \times (-))$

$$
\begin{tikzcd}
& x_1 \arrow[dl, "{a,\frac{1}{2}}"'] \arrow[dr, "{a,\frac{1}{2}}"] & \\
x_2 \arrow[d, "{b,1}"'] & & x_3 \arrow[d, "{c,1}"] \\
x_4 \arrow[d, "1"'] & & x_5 \arrow[d, "1"] \\
* & & *
\end{tikzcd}
$$

DFA

$[0,1] \times (-)^A$ states $\mathcal{D}(-)$

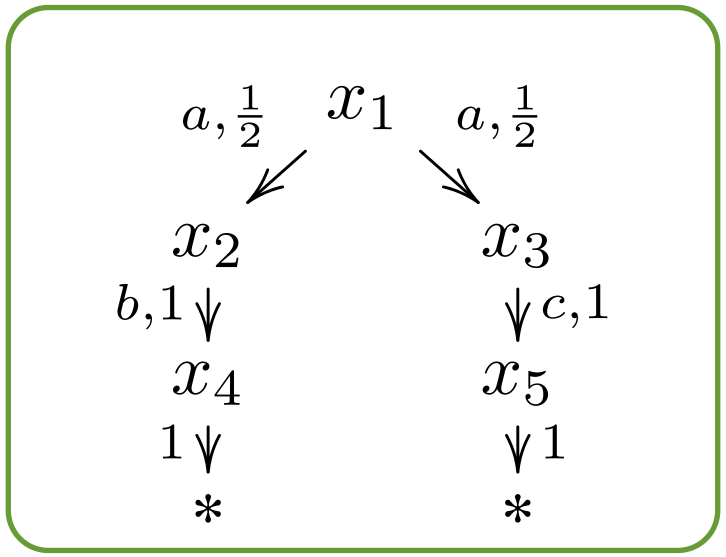
$$
\begin{tikzcd}
& x_1 \arrow[d, "a"'] & \\
& \frac{1}{2}x_2 + \frac{1}{2}x_3 \arrow[dl, "b"'] \arrow[dr, "c"] & \\
\frac{1}{2}x_4 \arrow[d, "\frac{1}{2}"'] & & \frac{1}{2}x_5 \arrow[d, "\frac{1}{2}"] \\
* & & *
\end{tikzcd}
$$

trace = bisimilarity after determinisation

(2) Traces via determinisation

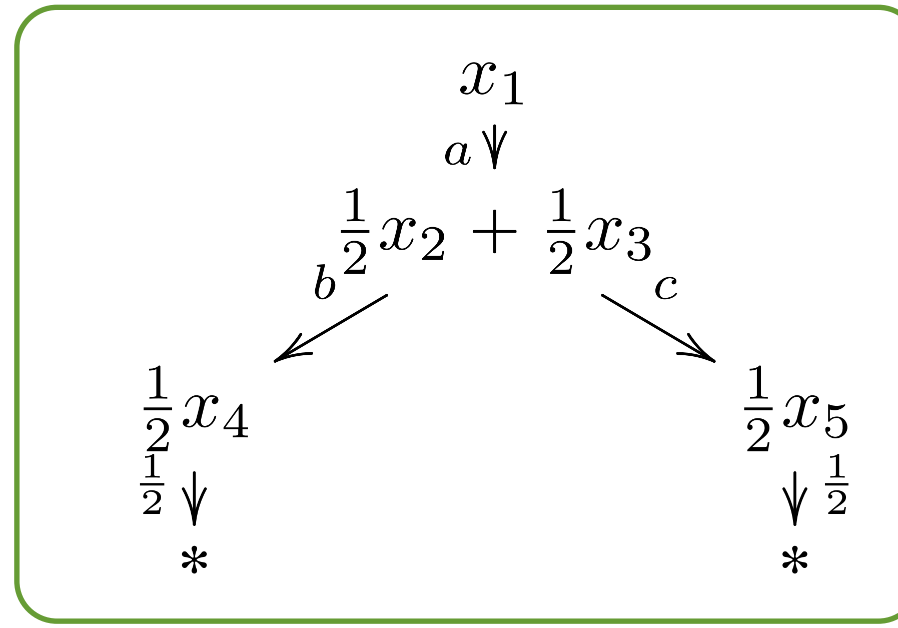
Generative PTS

$\mathcal{D}(1 + A \times (-))$



DFA

$[0,1] \times (-)^A$ states $\mathcal{D}(-)$



trace = bisimilarity after determinisation

Happens in $\mathcal{EM}(\mathcal{D})$

(2) Traces via determinisation

Generative PTS

$\mathcal{D}(1 + A \times (-))$

$$\begin{tikzcd}[column sep=small]
 & x_1 \arrow[dl, "{a,\frac{1}{2}}"'] \arrow[dr, "{a,\frac{1}{2}}"] & \\
x_2 \arrow[d, "{b,1}"'] & & x_3 \arrow[d, "{c,1}"] \\
x_4 \arrow[d, "1"'] & & x_5 \arrow[d, "1"] \\
* & & *
\end{tikzcd}$$

DFA

$[0,1] \times (-)^A$ **states $\mathcal{D}(-)$**

$$\begin{tikzcd}[column sep=small]
 & x_1 \arrow[d, "a"'] & \\
 & \frac{1}{2}x_2 + \frac{1}{2}x_3 \arrow[dl, "b"'] \arrow[dr, "c"] & \\
\frac{1}{2}x_4 \arrow[d, "\frac{1}{2}"'] & & \frac{1}{2}x_5 \arrow[d, "\frac{1}{2}"] \\
* & & *
\end{tikzcd}$$

trace = bisimilarity after determinisation

Happens in $\mathcal{EM}(\mathcal{D})$

(positive) convex algebras