Bisimilarity and trace via coinduction

Ana Sokolova

Computational Systems group, University of Salzburg

Joint work with: Ichiro Hasuo KU, JP & RUN, NL and Bart Jacobs RUN, NL

Outline

- introduction - formal methods, models and semantics
- from LTS to coalgebras
- Bisimilarity can’t be traced, BUT
  - * bisimilarity via coinduction in $\mathbf{Sets}$
  - * trace semantics also via coinduction...

Formal methods

are mathematically based techniques for

- specification
- development
- verification

of software and hardware systems

Formal methods

In general:

- models - transition systems, automata, terms,...
  with a clear semantics
- analysis - model checking
  process algebra
  theorem proving...

Formal methods

Here:

- models - transition systems, coalgebras
- analysis - via behavior semantics

Formal methods

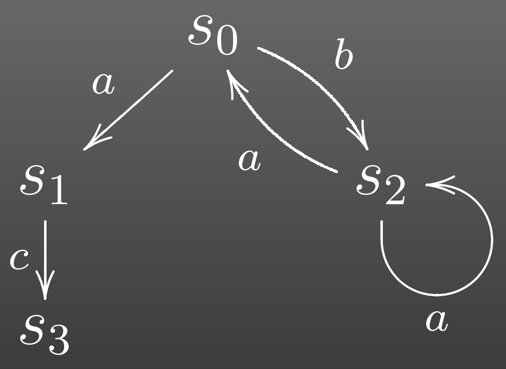
Here:

- models - transition systems, coalgebras
- analysis - via behavior semantics

Aim: One framework for many models and semantics !

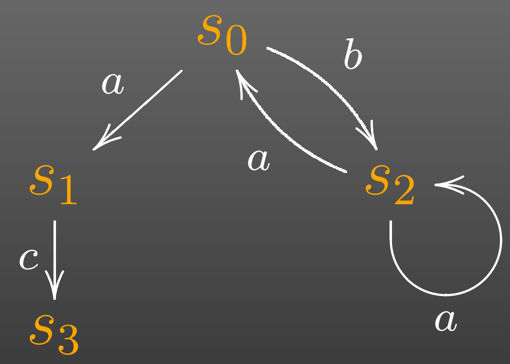
Standard model - LTS

labelled transition systems Σ - labels



Standard model - LTS

labelled transition systems Σ - labels



states $+$ transitions $\alpha : S \to \mathcal{P}(\Sigma \times S)$

$$\alpha(s_0) = \{\langle a, s_1\rangle, \langle b, s_2\rangle\},\ \alpha(s_1) = \{\langle c, s_3\rangle\},\ \ldots$$

Behavior semantics

are used for verification

- behavior equivalence ($\equiv$) identifies states with same behavior
- behavior preorder ($\sqsubseteq$) orders states according to behavior

Behavior semantics

are used for verification

- behavior equivalence ($\equiv$) identifies states with same behavior
- behavior preorder ($\sqsubseteq$) orders states according to behavior

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

Behavior semantics

verification amounts to:

- given
  - * Sys - model of the system, LTS
  - * Spec - model of the specification, LTS

Behavior semantics

verification amounts to:

- given
  - Sys - model of the system, LTS
  - Spec - model of the specification, LTS
- verify if

  Sys $\equiv$ Spec or Sys $\sqsubseteq$ Spec

Bisimulation - LTS

R - equivalence on states, is a bisimulation if

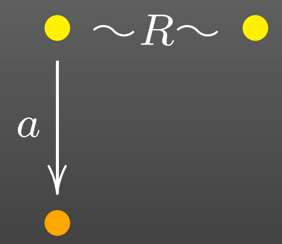
Bisimulation - LTS

R - equivalence on states, is a bisimulation if

$$\bullet \sim R \sim \bullet$$

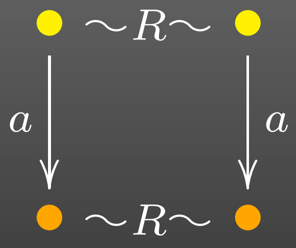
Bisimulation - LTS

R - equivalence on states, is a bisimulation if



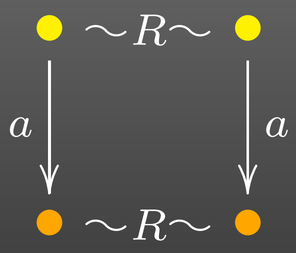
Bisimulation - LTS

R - equivalence on states, is a bisimulation if



Bisimulation - LTS

R - equivalence on states, is a bisimulation if

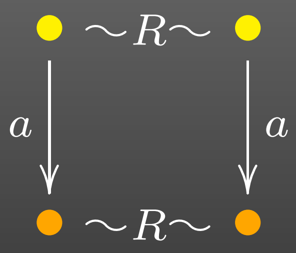


Transfer condition: $\langle s,t\rangle \in R \quad \Longrightarrow$

$$s \xrightarrow{a} s' \Rightarrow (\exists t')\ t \xrightarrow{a} t',\ \langle s',t'\rangle \in R,$$
$$t \xrightarrow{a} t' \Rightarrow (\exists s')\ s \xrightarrow{a} s',\ \langle s',t'\rangle \in R$$

Bisimulation - LTS

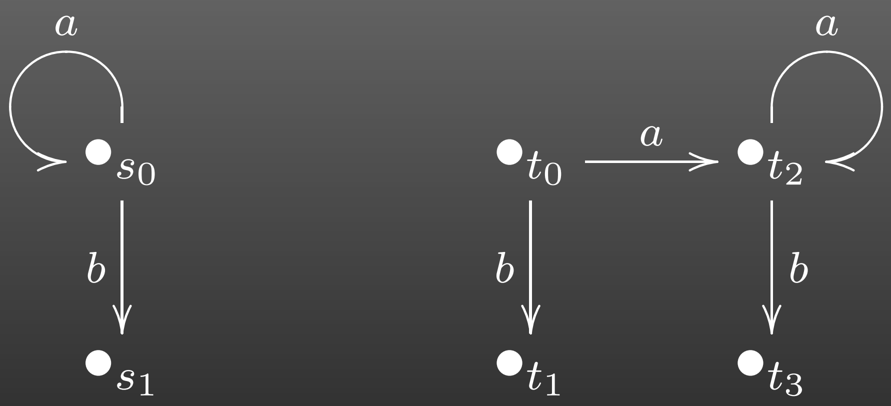
R - equivalence on states, is a bisimulation if



... two states are bisimilar if they are related by some bisimulation

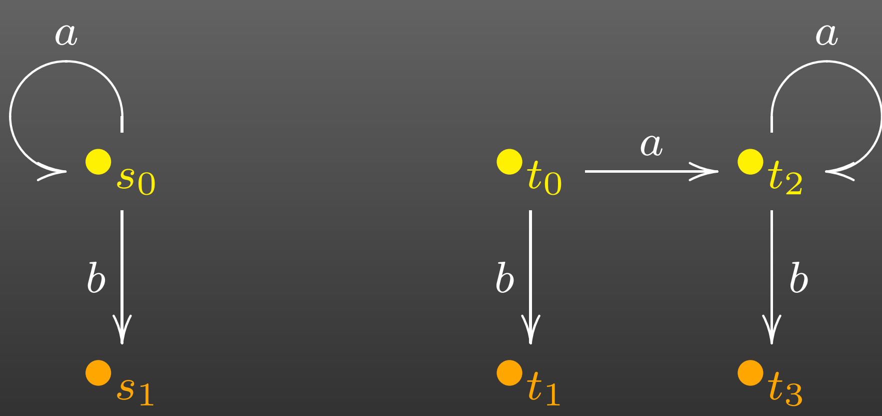
Bisimulation - LTS

Example: Consider the LTS



Bisimulation - LTS

Example: Consider the LTS



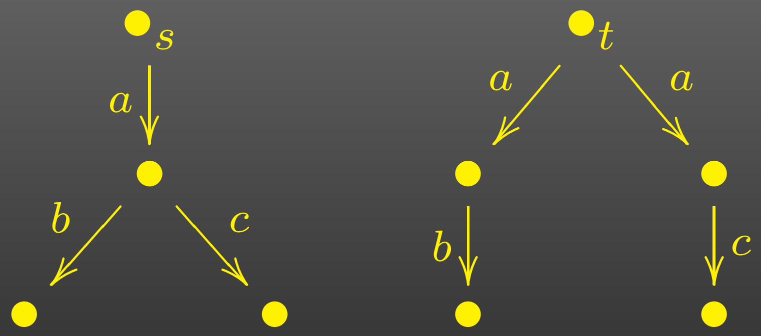
the coloring is a bisimulation, so s_0 and t_0 are bisimilar

LT/BT spectrum

Bisimilarity is not the only semantics...

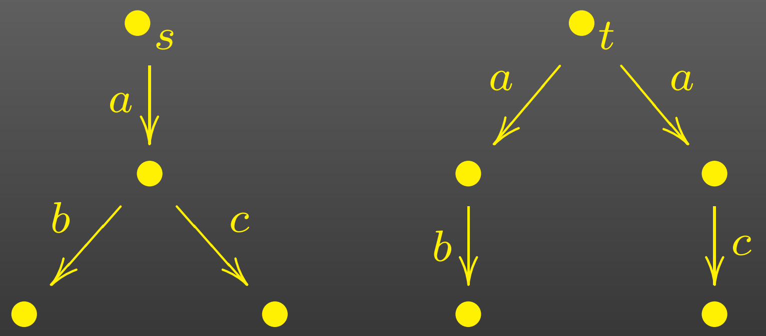
LT/BT spectrum

Are these non-deterministic systems equal ?



LT/BT spectrum

Are these non-deterministic systems equal ?

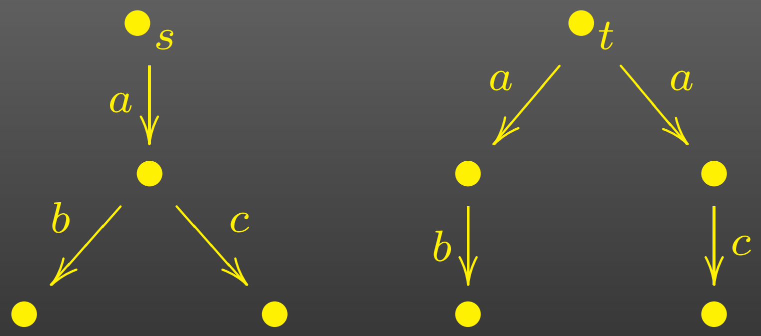


s and t are:

- different wrt. bisimilarity

LT/BT spectrum

Are these non-deterministic systems equal ?



s and t are:

- different wrt. bisimilarity, but
- equivalent wrt. trace semantics
  $\mathrm{tr}(s) = \mathrm{tr}(t) = \{ab, ac\}$

Traces - LTS

For LTS with explicit termination (NA)

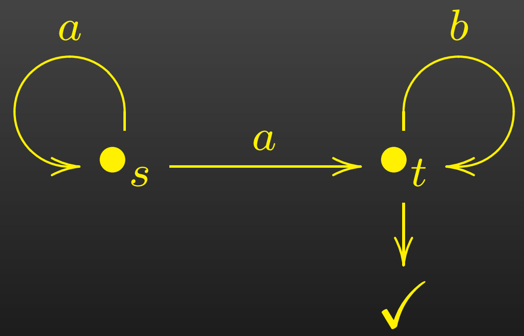
trace = the set of all possible linear behaviors

Traces - LTS

For LTS with explicit termination (NA)

trace = the set of all possible linear behaviors

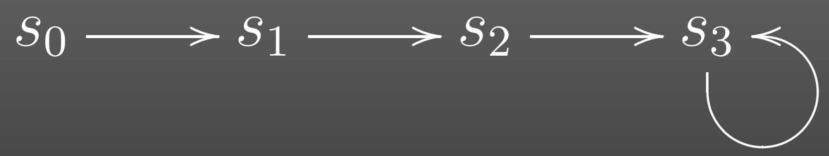
Example:



$$\mathrm{tr}(t) \;=\; b^*, \qquad \mathrm{tr}(s) \;=\; a^+ \cdot \mathrm{tr}(t) \;=\; a^+ \cdot b^*$$

Other models

deterministic systems



Other models

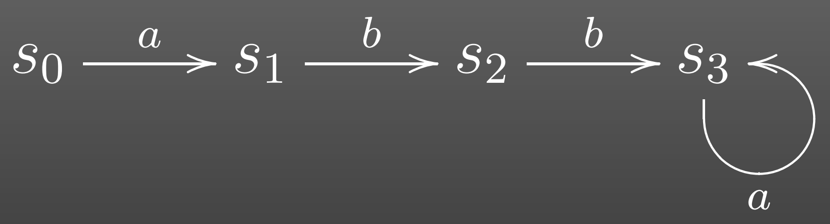
deterministic systems

states $+$ transitions $\alpha : S \to S$

$$\alpha(s_0) = s_1,\ \alpha(s_1) = s_2,\ \ldots$$

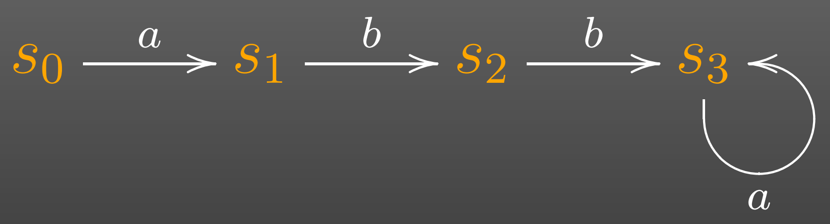
Other models

labelled deterministic systems Σ - labels



Other models

labelled deterministic systems Σ - labels

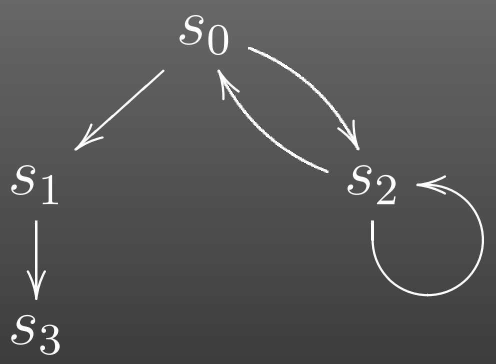


states $+$ transitions $\alpha : S \to \Sigma \times S$

$$\alpha(s_0) = \langle a, s_1 \rangle,\ \alpha(s_1) = \langle b, s_2 \rangle,\ \ldots$$

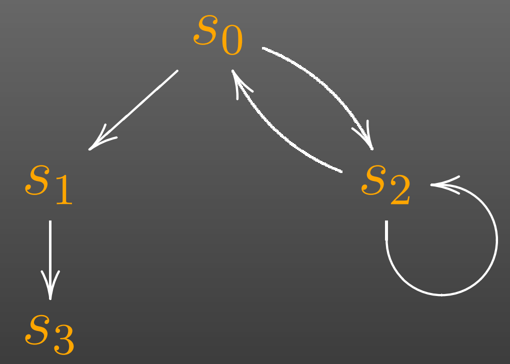
Other models

transition systems



Other models

transition systems

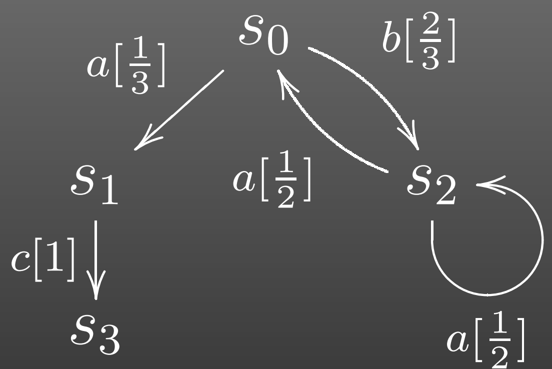


states $+$ transitions $\alpha : S \to \mathcal{P}(S)$

$$\alpha(s_0) = \{s_1, s_2\},\ \alpha(s_1) = \{s_3\},\ \ldots$$

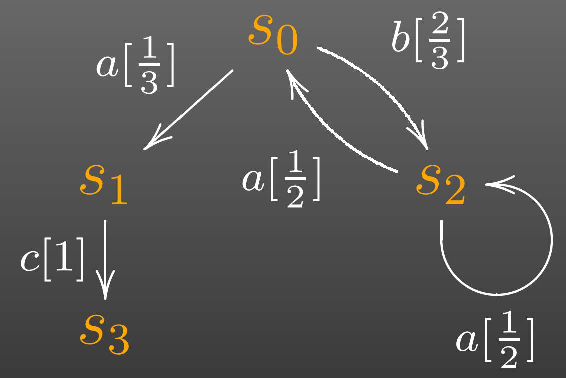
Other models

generative probabilistic systems Σ - labels



Other models

generative probabilistic systems Σ - labels



states $+$ transitions $\alpha : S \to \mathcal{D}(\Sigma \times S) + 1$

$$\alpha(s_0) = \left(\langle a, s_1\rangle \mapsto \tfrac{1}{3}, \langle b, s_2\rangle \mapsto \tfrac{2}{3}\right),$$
$$\alpha(s_1) = (\langle c, s_3\rangle \mapsto 1), \ \ldots$$

Bisimulation - generative

R - equivalence on states, is a bisimulation if

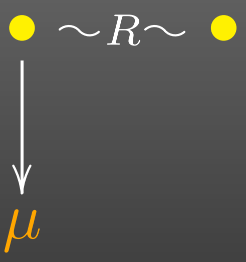
Bisimulation - generative

R - equivalence on states, is a bisimulation if

$$\bullet \sim R \sim \bullet$$

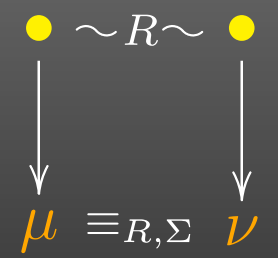
Bisimulation - generative

R - equivalence on states, is a bisimulation if



Bisimulation - generative

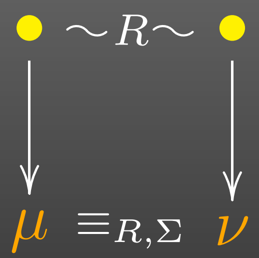
R - equivalence on states, is a bisimulation if



$\equiv_{R,\Sigma}$ relates distributions that assign the same probability to each label and each R-class

Bisimulation - generative

R - equivalence on states, is a bisimulation if

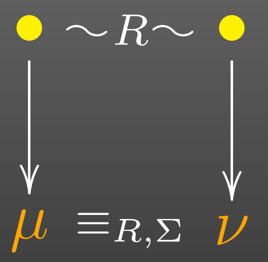


Transfer condition: $\langle s, t \rangle \in R \quad \Longrightarrow$

$$s \to \mu \Rightarrow (\exists \nu)\ t \to \nu,\ \ \mu \equiv_{R,\Sigma} \nu$$

Bisimulation - generative

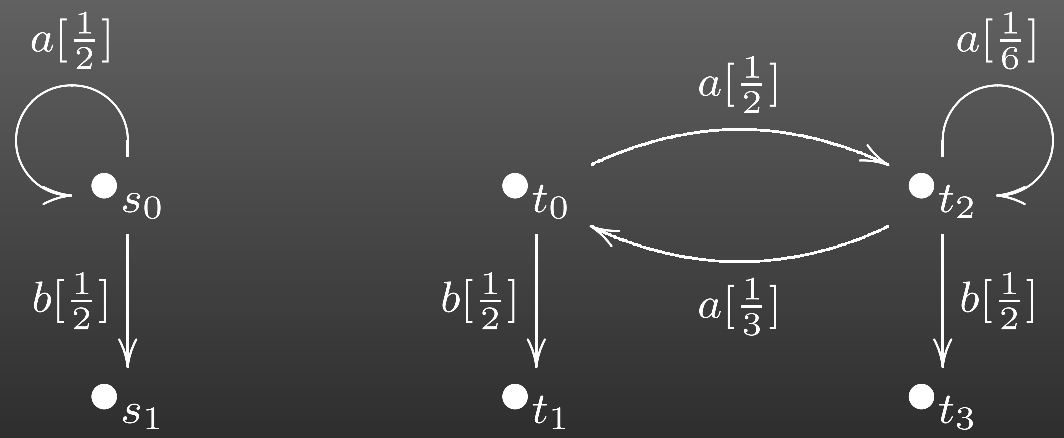
R - equivalence on states, is a bisimulation if



... two states are bisimilar if they are related by some bisimulation

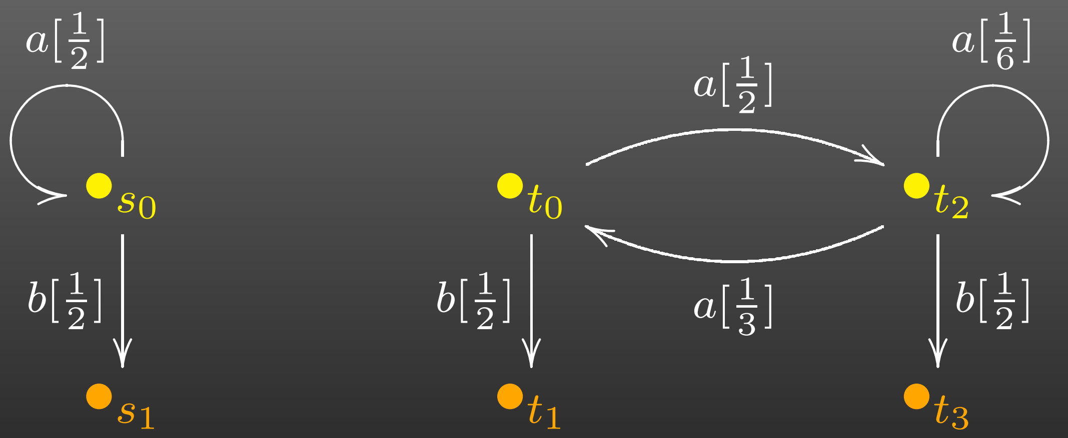
Bisimulation - generative

Consider the generative systems



Bisimulation - generative

Example: Consider the generative systems



the coloring is a bisimulation, so s_0 and t_0 are bisimilar

Traces - generative

For generative probabilistic systems with ex. termination

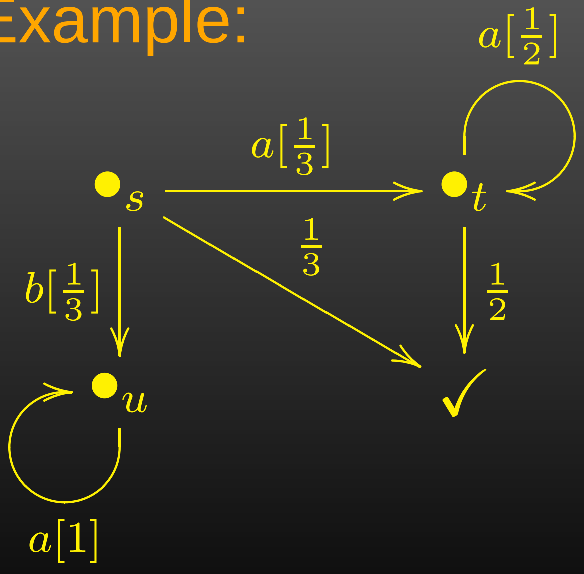
trace = sub-probability distribution over possible linear behaviors

Traces - generative

For generative probabilistic systems with ex. termination

trace = sub-probability distribution over possible linear behaviors

Example:



$$\mathrm{tr}(s): \quad \langle\rangle \mapsto \frac{1}{3}$$

$$a \mapsto \frac{1}{3} \cdot \frac{1}{2}$$

$$a^2 \mapsto \frac{1}{3} \cdot \frac{1}{2} \cdot \frac{1}{2}$$

$$\ldots$$

Coalgebras

are an elegant generalization of transition systems with

states + transitions

Coalgebras

are an elegant generalization of transition systems with

states + transitions

as pairs

$$\langle S, \alpha : S \to \mathcal{F}S \rangle, \text{ for } \mathcal{F} \text{ a functor}$$

Coalgebras

are an elegant generalization of transition systems with

states + transitions

as pairs

$$\langle S, \alpha : S \to \mathcal{F}S \rangle, \text{ for } \mathcal{F} \text{ a functor}$$

- rich mathematical structure
- a uniform way for treating transition systems
- general notions and results, generic notion of bisimulation

Coalgebraic bisimulation

A bisimulation on

$$\langle S, \alpha : S \to \mathcal{F}S \rangle$$

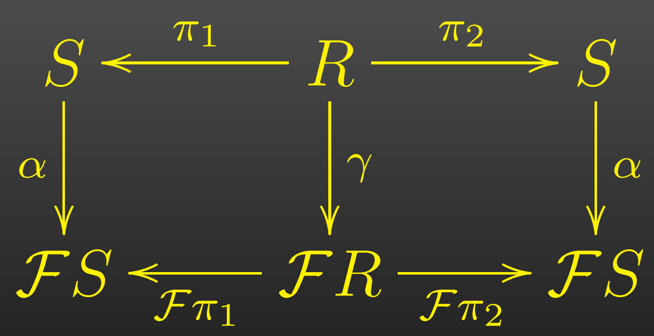
is $R \subseteq S \times S$ such that

Coalgebraic bisimulation

A bisimulation on

$$\langle S, \alpha : S \to \mathcal{F}S \rangle$$

is $R \subseteq S \times S$ such that γ exists:



Coalgebraic bisimulation

A bisimulation on

$$\langle S, \alpha : S \to \mathcal{F}S \rangle$$

is $R \subseteq S \times S$ such that

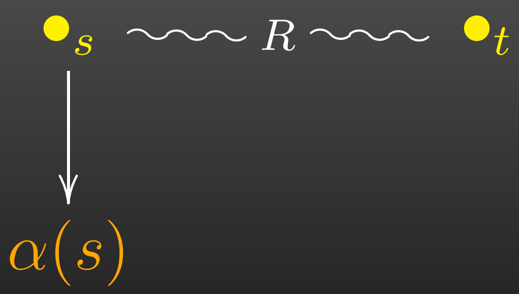
$$\bullet_s \sim\sim\sim R \sim\sim\sim \bullet_t$$

Coalgebraic bisimulation

A bisimulation on

$$\langle S, \alpha : S \to \mathcal{F}S \rangle$$

is $R \subseteq S \times S$ such that

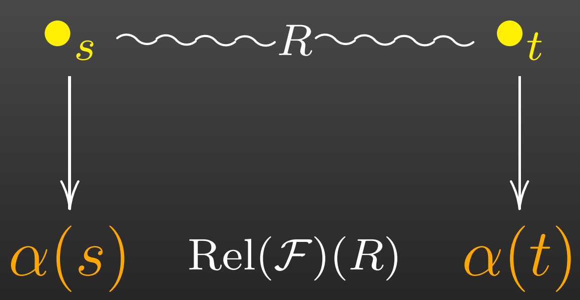


Coalgebraic bisimulation

A bisimulation on

$$\langle S, \alpha : S \to \mathcal{F}S \rangle$$

is $R \subseteq S \times S$ such that

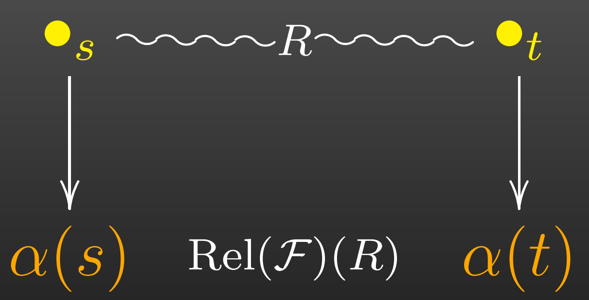


Coalgebraic bisimulation

A bisimulation on

$$\langle S, \alpha : S \to \mathcal{F}S \rangle$$

is $R \subseteq S \times S$ such that



Transfer condition: $\langle s, t \rangle \in R \implies$

$$\langle \alpha(s), \beta(t) \rangle \in \mathrm{Rel}(\mathcal{F})(R)$$

Coalgebraic bisimulation

A bisimulation on

$$\langle S, \alpha : S \to \mathcal{F}S \rangle$$

is $R \subseteq S \times S$ such that

$$\begin{array}{ccc} \bullet_s & \rightsquigarrow R \rightsquigarrow & \bullet_t \\ \downarrow & & \downarrow \\ \alpha(s) & \mathrm{Rel}(\mathcal{F})(R) & \alpha(t) \end{array}$$

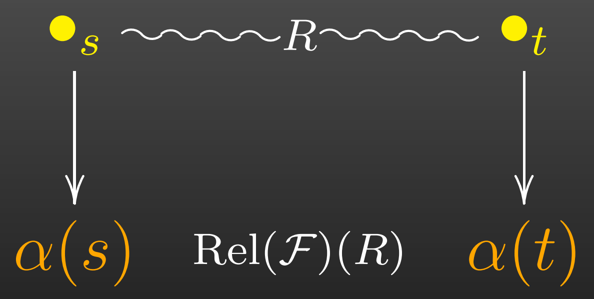
... two states are bisimilar if they are related by some bisimulation

Coalgebraic bisimulation

A bisimulation on

$$\langle S, \alpha : S \to \mathcal{F}S \rangle$$

is $R \subseteq S \times S$ such that



Theorem: Coalgebraic and concrete bisimilarity coincide !

Trace of a coalgebra ?

Trace of a coalgebra ?

- Power&Turi ’99 - $\mathcal{P}(1 + \Sigma \times \_)$
- Jacobs ’04 - $\mathcal{P}\mathcal{F}$
- Hasuo&Jacobs CALCO ’05, CALCO Jnr ’05 - $\mathcal{P}\mathcal{F}$, $\mathcal{D}\mathcal{F}$
- Hasuo&Jacobs&Sokolova CMCS’06, LMCS 3(4:11)’07
  Generic Trace Semantics via Coinduction
  $\mathcal{T}\mathcal{F}$, order-enriched setting

Trace of a coalgebra ?

- Power&Turi '99 - $\mathcal{P}(1 + \Sigma \times \_)$
- Jacobs '04 - $\mathcal{P}\mathcal{F}$
- Hasuo&Jacobs CALCO '05, CALCO Jnr '05 - $\mathcal{P}\mathcal{F}$, $\mathcal{D}\mathcal{F}$
- Hasuo&Jacobs&Sokolova CMCS'06, LMCS 3(4:11)'07
  Generic Trace Semantics via Coinduction
  $\mathcal{T}\mathcal{F}$, order-enriched setting

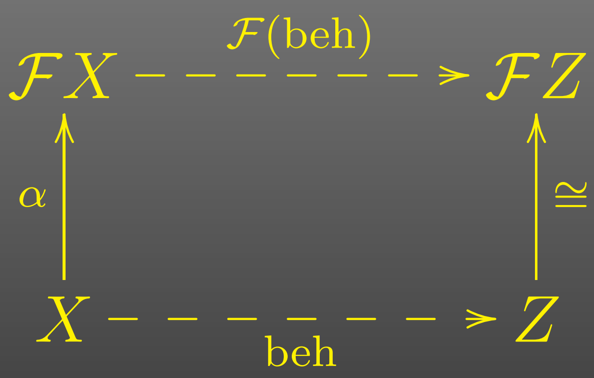
main idea: coinduction in a Kleisli category

Coinduction

$$
\begin{array}{ccc}
\mathcal{F}X & \overset{\mathcal{F}(\mathrm{beh})}{\dashrightarrow} & \mathcal{F}Z \\
\uparrow{\scriptstyle \alpha} & & \uparrow{\scriptstyle \cong} \\
X & \underset{\mathrm{beh}}{\dashrightarrow} & Z
\end{array}
$$

system final coalgebra

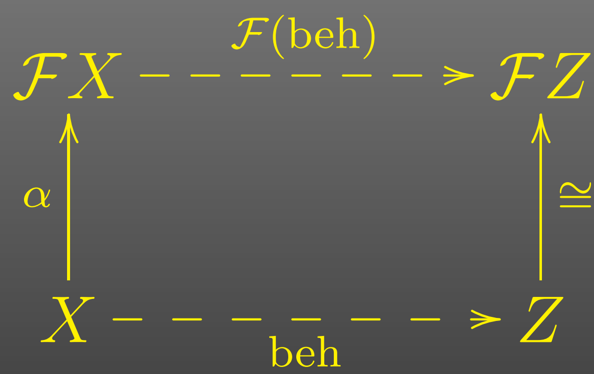
Coinduction



system final coalgebra

- finality = $\exists!$(morphism for any $\mathcal{F}$- coalgebra)
- beh gives the behavior of the system
- this yields final coalgebra semantics

Coinduction



system final coalgebra

- f.c.s. in **Sets** = bisimilarity
- f.c.s. in a Kleisli category = trace semantics

Types of systems

For trace semantics systems are suitably modelled as coalgebras in $\mathbf{Sets}$

$$X \xrightarrow{c} \textcircled{\mathcal{T}}\textcircled{\mathcal{F}} X$$

Types of systems

For trace semantics systems are suitably modelled as coalgebras in $\mathbf{Sets}$

$$X \xrightarrow{c} \mathcal{T}\mathcal{F}X$$

monad - branching type

Types of systems

For trace semantics systems are suitably modelled as coalgebras in $\mathbf{Sets}$

$$X \xrightarrow{c} \mathcal{T}\mathcal{F}X$$

monad - branching type

functor - linear i/o type

Types of systems

For trace semantics systems are suitably modelled as coalgebras in $\mathbf{Sets}$

$$X \xrightarrow{c} \mathcal{T}\,\mathcal{F}X$$

monad - branching type

functor - linear i/o type

needed: distributive law $\mathcal{F}\mathcal{T} \Rightarrow \mathcal{T}\mathcal{F}$

Distributive law

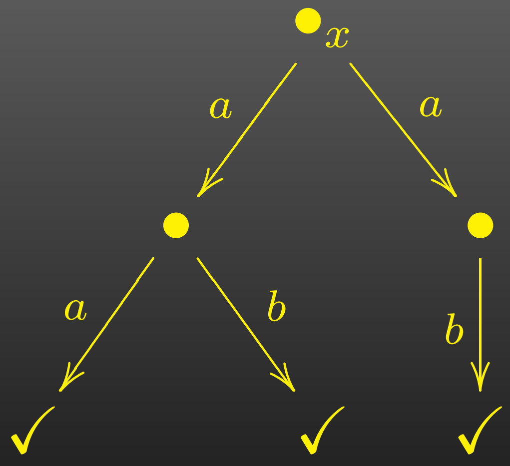
is needed since branching is irrelevant:

LTS with ✓ - $\mathcal{PF} = \mathcal{P}(1 + \Sigma \times \_)$

Distributive law

is needed since branching is irrelevant:

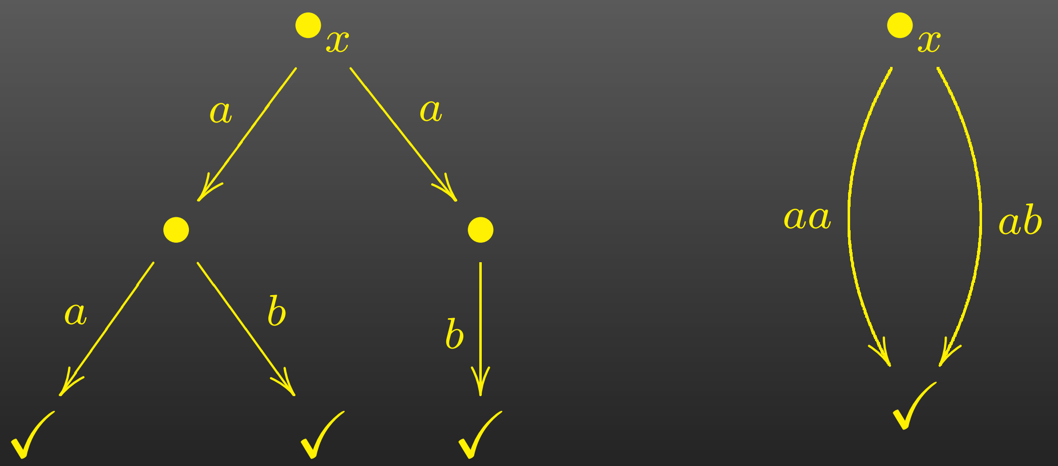
LTS with ✓- $\mathcal{PF} = \mathcal{P}(1 + \Sigma \times \_)$



Distributive law

is needed since branching is irrelevant:

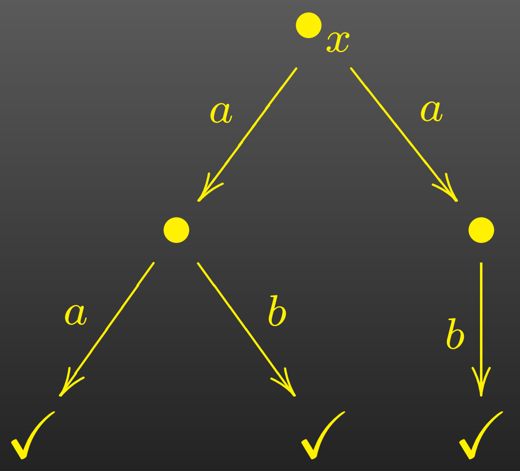
LTS with ✓- $\mathcal{PF} = \mathcal{P}(1 + \Sigma \times \_)$

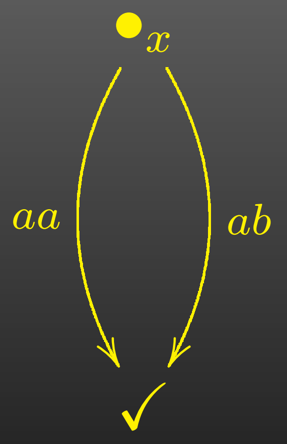


Distributive law

is needed since branching is irrelevant:

LTS with ✓- $\mathcal{P}\mathcal{F} = \mathcal{P}(1 + \Sigma \times \_)$



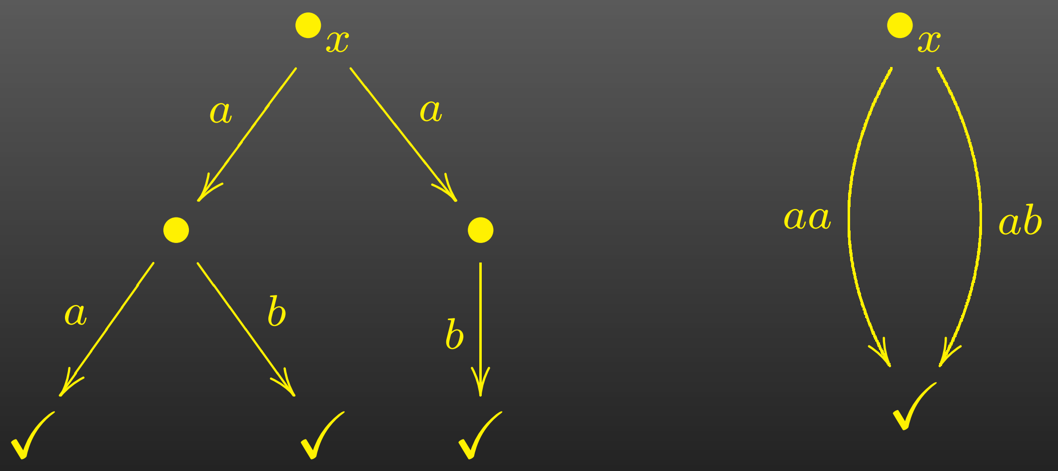


$X \xrightarrow{c} \mathcal{P}\mathcal{F}X$

Distributive law

is needed since branching is irrelevant:

LTS with ✓- $\mathcal{PF} = \mathcal{P}(1 + \Sigma \times \_)$

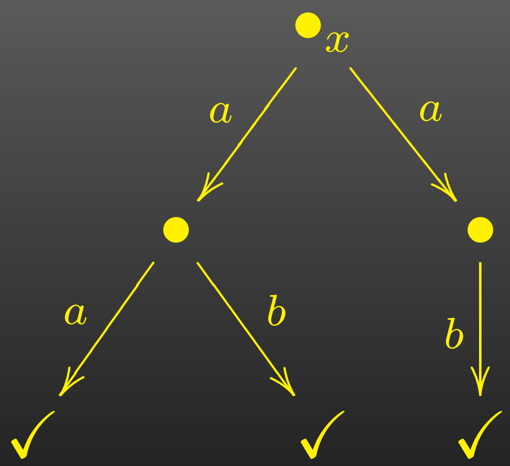


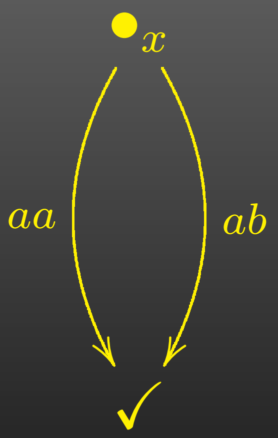
$X \xrightarrow{c} \mathcal{PF}X$

Distributive law

is needed since branching is irrelevant:

LTS with ✓- $\mathcal{PF} = \mathcal{P}(1 + \Sigma \times \_)$



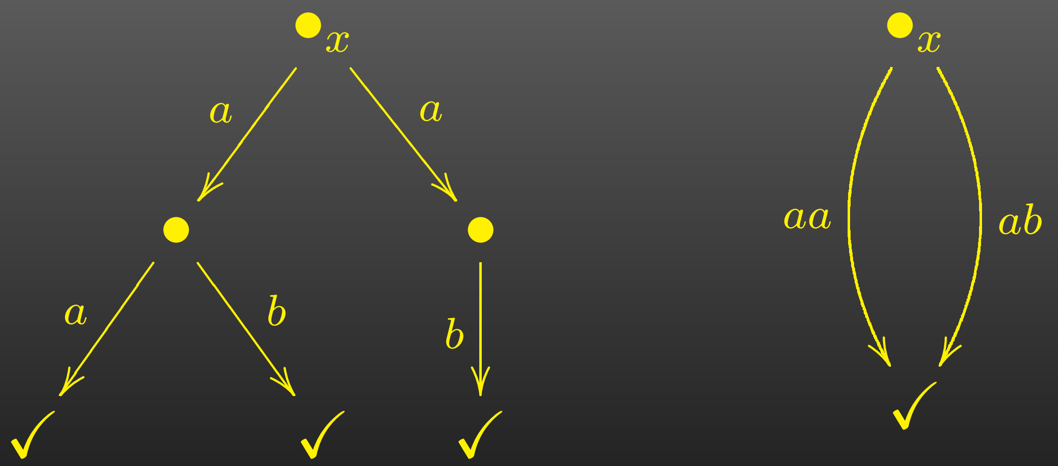


$X \xrightarrow{c} \mathcal{PF}X \xrightarrow{\mathcal{PF}c} \mathcal{PFPF}X$

Distributive law

is needed since branching is irrelevant:

LTS with ✓- $\mathcal{PF} = \mathcal{P}(1 + \Sigma \times \_)$

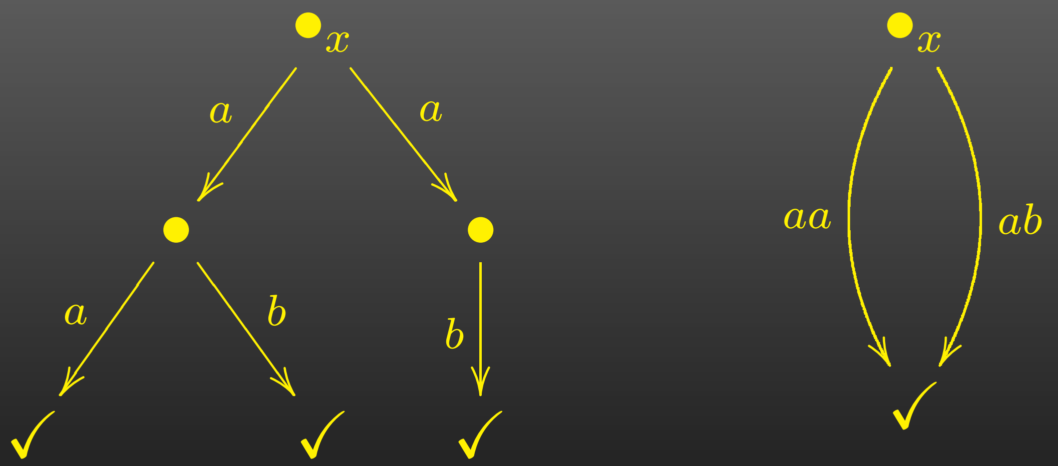


$$X \stackrel{c}{\to} \mathcal{PF}X \stackrel{\mathcal{PF}c}{\to} \mathcal{PFPF}X$$

Distributive law

is needed since branching is irrelevant:

LTS with ✓ - $\mathcal{PF} = \mathcal{P}(1 + \Sigma \times \_)$

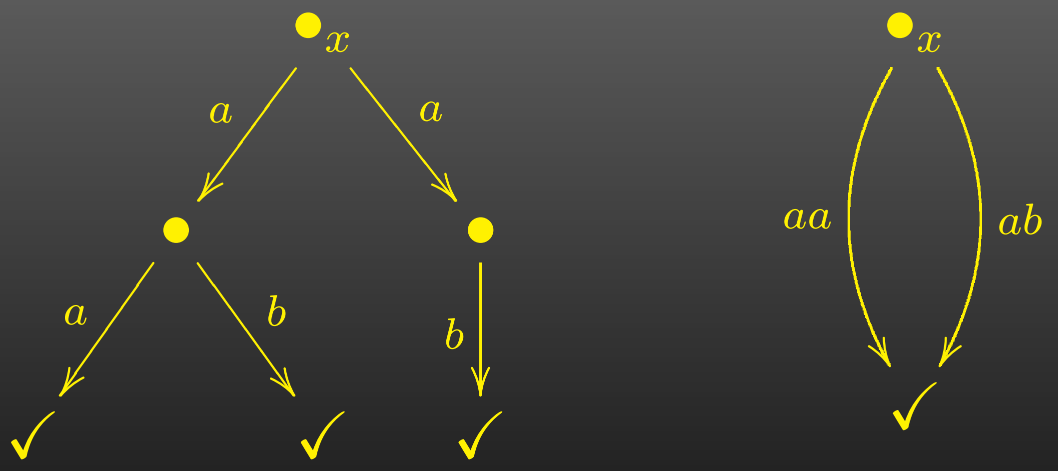


$$X \xrightarrow{c} \mathcal{PF}X \xrightarrow{\mathcal{PF}c} \mathcal{PFPF}X \xrightarrow{\text{d.l.}} \mathcal{PPFF}X$$

Distributive law

is needed since branching is irrelevant:

LTS with ✓ - $\mathcal{PF} = \mathcal{P}(1 + \Sigma \times \_)$

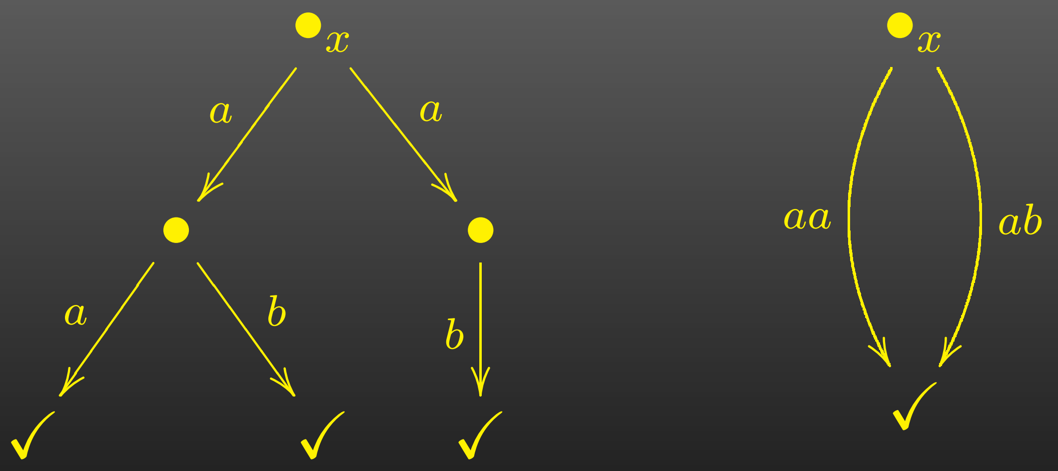


$$X \xrightarrow{c} \mathcal{PF}X \xrightarrow{\mathcal{PF}c} \mathcal{PFPF}X \xrightarrow{\text{d.l.}} \mathcal{PPFF}X$$

Distributive law

is needed since branching is irrelevant:

LTS with ✓ - $\mathcal{PF} = \mathcal{P}(1 + \Sigma \times \_)$

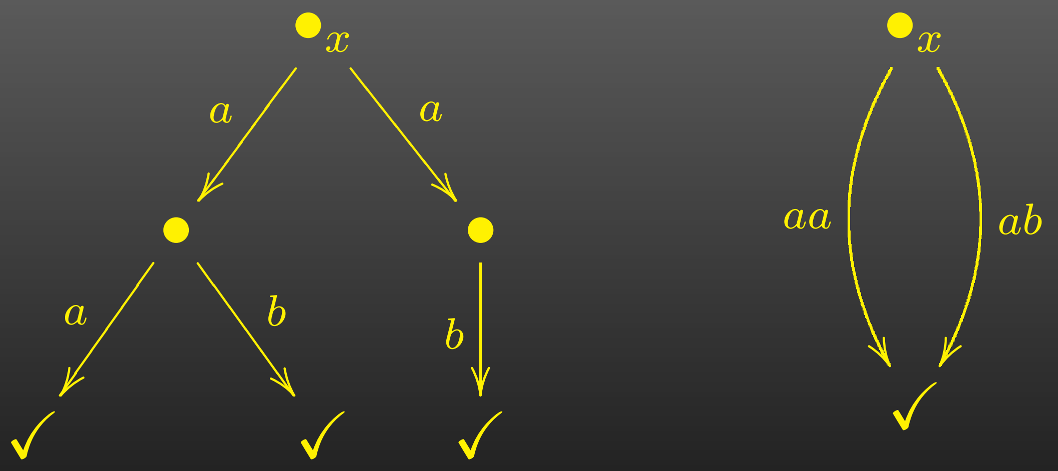


$$X \xrightarrow{c} \mathcal{PF}X \xrightarrow{\mathcal{PF}c} \mathcal{PFPF}X \xrightarrow{\text{d.l.}} \mathcal{PPFF}X \xrightarrow{\text{m.m.}} \mathcal{PFF}X$$

Distributive law

is needed since branching is irrelevant:

LTS with ✓- $\mathcal{PF} = \mathcal{P}(1 + \Sigma \times \_)$



$$X \xrightarrow{c} \mathcal{PF}X \xrightarrow{\mathcal{PF}c} \mathcal{PFPF}X \xrightarrow{\text{d.l.}} \mathcal{PPFF}X \xrightarrow{\text{m.m.}} \mathcal{PFF}X$$

Distributive law

is needed for $X \xrightarrow{c} \mathcal{T}\mathcal{F}X$ to be a coalgebra in the Kleisli category $\mathcal{K}\ell(\mathcal{T})$..

Distributive law

is needed for $X \stackrel{c}{\to} \mathcal{T}\mathcal{F}X$ to be a coalgebra in the Kleisli category $\mathcal{K}\ell(\mathcal{T})$..

- objects - sets
- arrows - $X \stackrel{f}{\to} Y$ are functions $f : X \to \mathcal{T}Y$

Distributive law

is needed for $X \xrightarrow{c} \mathcal{T}\mathcal{F}X$ to be a coalgebra in the Kleisli category $\mathcal{K}\ell(\mathcal{T})$..

$\mathcal{F}\mathcal{T} \Rightarrow \mathcal{T}\mathcal{F}$: $\mathcal{F}$ lifts to $\mathcal{F}_{\mathcal{K}\ell(\mathcal{T})}$ on $\mathcal{K}\ell(\mathcal{T})$.

Distributive law

is needed for $X \xrightarrow{c} \mathcal{T}\mathcal{F}X$ to be a coalgebra in the Kleisli category $\mathcal{K}\ell(\mathcal{T})$..

$\mathcal{F}\mathcal{T} \Rightarrow \mathcal{T}\mathcal{F}:$ $\mathcal{F}$ lifts to $\mathcal{F}_{\mathcal{K}\ell(\mathcal{T})}$ on $\mathcal{K}\ell(\mathcal{T})$.

Hence: coalgebra $X \xrightarrow{c} \mathcal{F}_{\mathcal{K}\ell(\mathcal{T})}X$ in $\mathcal{K}\ell(\mathcal{T})$!!!

Distributive law

is needed for $X \stackrel{c}{\to} \mathcal{T}\mathcal{F}X$ to be a coalgebra in the Kleisli category $\mathcal{K}\ell(\mathcal{T})$..

$\mathcal{F}\mathcal{T} \Rightarrow \mathcal{T}\mathcal{F}$: $\mathcal{F}$ lifts to $\mathcal{F}_{\mathcal{K}\ell(\mathcal{T})}$ on $\mathcal{K}\ell(\mathcal{T})$.

Hence: coalgebra $X \stackrel{c}{\to} \mathcal{F}_{\mathcal{K}\ell(\mathcal{T})}X$ in $\mathcal{K}\ell(\mathcal{T})$!!!

in $\mathcal{K}\ell(\mathcal{T})$: $X \stackrel{c}{\to} \mathcal{F}_{\mathcal{K}\ell(\mathcal{T})}X$

Distributive law

is needed for $X \xrightarrow{c} \mathcal{T}\mathcal{F}X$ to be a coalgebra in the Kleisli category $\mathcal{K}\ell(\mathcal{T})$..

$\mathcal{F}\mathcal{T} \Rightarrow \mathcal{T}\mathcal{F}$: $\mathcal{F}$ lifts to $\mathcal{F}_{\mathcal{K}\ell(\mathcal{T})}$ on $\mathcal{K}\ell(\mathcal{T})$.

Hence: coalgebra $X \xrightarrow{c} \mathcal{F}_{\mathcal{K}\ell(\mathcal{T})}X$ in $\mathcal{K}\ell(\mathcal{T})$!!!

in $\mathcal{K}\ell(\mathcal{T})$: $X \xrightarrow{c} \mathcal{F}_{\mathcal{K}\ell(\mathcal{T})}X$

Distributive law

is needed for $X \stackrel{c}{\to} \mathcal{T}\mathcal{F}X$ to be a coalgebra in the Kleisli category $\mathcal{K}\ell(\mathcal{T})$..

$\mathcal{F}\mathcal{T} \Rightarrow \mathcal{T}\mathcal{F}$: $\mathcal{F}$ lifts to $\mathcal{F}_{\mathcal{K}\ell(\mathcal{T})}$ on $\mathcal{K}\ell(\mathcal{T})$.

Hence: coalgebra $X \stackrel{c}{\to} \mathcal{F}_{\mathcal{K}\ell(\mathcal{T})}X$ in $\mathcal{K}\ell(\mathcal{T})$!!!

in $\mathcal{K}\ell(\mathcal{T})$: $X \stackrel{c}{\to} \mathcal{F}_{\mathcal{K}\ell(\mathcal{T})}X \stackrel{\mathcal{F}_{\mathcal{K}\ell(\mathcal{T})}c}{\to} \mathcal{F}_{\mathcal{K}\ell(\mathcal{T})}\mathcal{F}_{\mathcal{K}\ell(\mathcal{T})}X$

Distributive law

is needed for $X \stackrel{c}{\to} \mathcal{T}\mathcal{F}X$ to be a coalgebra in the Kleisli category $\mathcal{K}\ell(\mathcal{T})$..

$\mathcal{F}\mathcal{T} \Rightarrow \mathcal{T}\mathcal{F}$: $\mathcal{F}$ lifts to $\mathcal{F}_{\mathcal{K}\ell(\mathcal{T})}$ on $\mathcal{K}\ell(\mathcal{T})$.

Hence: coalgebra $X \stackrel{c}{\to} \mathcal{F}_{\mathcal{K}\ell(\mathcal{T})}X$ in $\mathcal{K}\ell(\mathcal{T})$!!!

in $\mathcal{K}\ell(\mathcal{T})$: $X \stackrel{c}{\to} \mathcal{F}_{\mathcal{K}\ell(\mathcal{T})}X \stackrel{\mathcal{F}_{\mathcal{K}\ell(\mathcal{T})}c}{\to} \mathcal{F}_{\mathcal{K}\ell(\mathcal{T})}\mathcal{F}_{\mathcal{K}\ell(\mathcal{T})}X \to \cdots$

Main Theorem

If ♣, then

$$\begin{array}{ccc} \mathcal{F}_{\mathcal{K}\ell(\mathcal{T})}A & \qquad & \mathcal{F}_{\mathcal{K}\ell(\mathcal{T})}A \\ {\scriptstyle \eta_A \circ \alpha}\big\downarrow {\scriptstyle \cong} & & {\scriptstyle \eta_{\mathcal{F}A} \circ \alpha^{-1}}\big\uparrow {\scriptstyle \cong} \\ A & & A \end{array}$$

is initial   is final   in $\mathcal{K}\ell(\mathcal{T})$

Main Theorem

If ♣, then

$$\begin{array}{c} \mathcal{F}_{\mathcal{K}\ell(\mathcal{T})}A \\ {\scriptstyle \eta_A \circ \alpha}\big\downarrow {\scriptstyle \cong} \\ A \end{array} \qquad\qquad \begin{array}{c} \mathcal{F}_{\mathcal{K}\ell(\mathcal{T})}A \\ {\scriptstyle \eta_{\mathcal{F}A} \circ \alpha^{-1}}\big\uparrow {\scriptstyle \cong} \\ A \end{array}$$

is initial   is final   in $\mathcal{K}\ell(\mathcal{T})$

[$\alpha : \mathcal{F}A \xrightarrow{\cong} A$ denotes the initial $\mathcal{F}$-algebra in $\mathbf{Sets}$]

Main Theorem

If ♣, then

$$\begin{array}{c} \mathcal{F}_{\mathcal{K}\ell(\mathcal{T})}A \\ {\scriptstyle \eta_A \circ \alpha}\downarrow{\scriptstyle \cong} \\ A \end{array} \qquad\qquad \begin{array}{c} \mathcal{F}_{\mathcal{K}\ell(\mathcal{T})}A \\ {\scriptstyle \eta_{\mathcal{F}A} \circ \alpha^{-1}}\uparrow{\scriptstyle \cong} \\ A \end{array}$$

is initial   is final   in $\mathcal{K}\ell(\mathcal{T})$

[$\alpha : \mathcal{F}A \xrightarrow{\cong} A$ denotes the initial $\mathcal{F}$-algebra in $\mathbf{Sets}$]

proof: via limit-colimit coincidence Smyth&Plotkin '82

The assumptions ♣:

- A monad $\mathcal{T}$ s.t. $\mathcal{K}\ell(\mathcal{T})$ is $\mathbf{DCpo}_\perp$-enriched
  left-strict composition

The assumptions ♣:

- A monad $\mathcal{T}$ s.t. $\mathcal{K}\ell(\mathcal{T})$ is $\mathbf{DCpo}_\bot$-enriched left-strict composition
- A functor $\mathcal{F}$ that preserves ω-colimits

The assumptions ♣:

- A monad $\mathcal{T}$ s.t. $\mathcal{K}\ell(\mathcal{T})$ is $\mathbf{DCpo}_\perp$-enriched
  left-strict composition
- A functor $\mathcal{F}$ that preserves ω-colimits
- A distributive law $\mathcal{F}\mathcal{T} \Rightarrow \mathcal{T}\mathcal{F}$: lifting $\mathcal{F}_{\mathcal{K}\ell(\mathcal{T})}$

The assumptions ♣:

- A monad $\mathcal{T}$ s.t. $\mathcal{K}\ell(\mathcal{T})$ is $\mathbf{DCpo}_\perp$-enriched left-strict composition
- A functor $\mathcal{F}$ that preserves ω-colimits
- A distributive law $\mathcal{F}\mathcal{T} \Rightarrow \mathcal{T}\mathcal{F}$: lifting $\mathcal{F}_{\mathcal{K}\ell(\mathcal{T})}$
- $\mathcal{F}_{\mathcal{K}\ell(\mathcal{T})}$ should be locally monotone

Proof sketch

In Sets

$$0 \xrightarrow{\mathsf{i}} \mathcal{F}0 \xrightarrow{\mathcal{F}\,\mathsf{i}} \ldots \mathcal{F}^n 0 \xrightarrow{\mathcal{F}^n\,\mathsf{i}} \mathcal{F}^{n+1}0 \xrightarrow{\mathcal{F}^{n+1}\,\mathsf{i}} \cdots$$

Proof sketch

In Sets

$$\cdots \xrightarrow{\mathcal{F}^{n-1}\,\text{i}} \mathcal{F}^{n}0 \xrightarrow{\mathcal{F}^{n}\,\text{i}} \mathcal{F}^{n+1}0 \xrightarrow{\mathcal{F}^{n+1}\,\text{i}} \cdots$$

Proof sketch

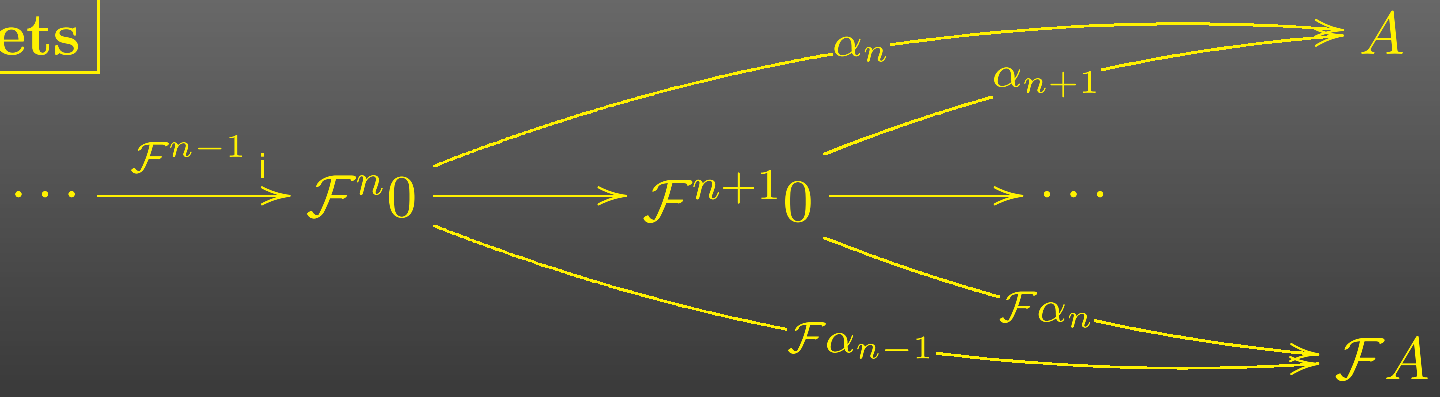
In Sets

$$\cdots \xrightarrow{\mathcal{F}^{n-1}\,¡} \mathcal{F}^n 0 \longrightarrow \mathcal{F}^{n+1} 0 \longrightarrow \cdots$$

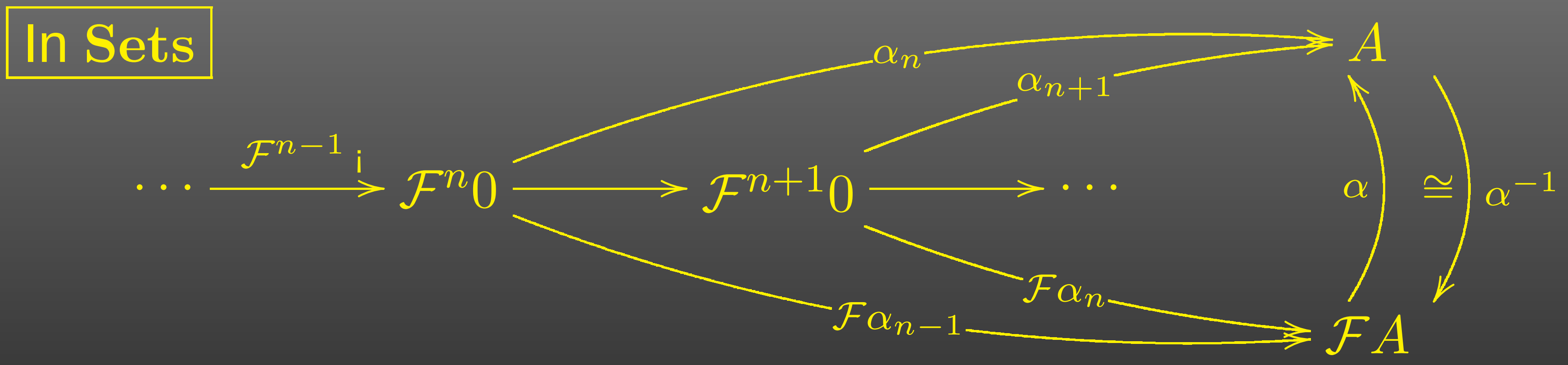
$$\alpha_n : \mathcal{F}^n 0 \to A, \qquad \alpha_{n+1} : \mathcal{F}^{n+1} 0 \to A$$

Proof sketch

In Sets

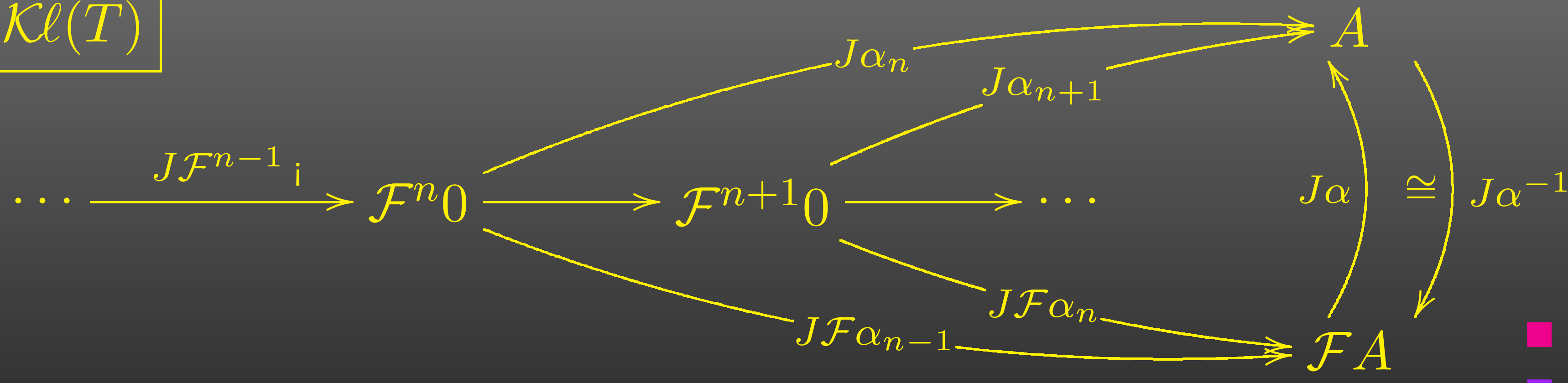


Proof sketch

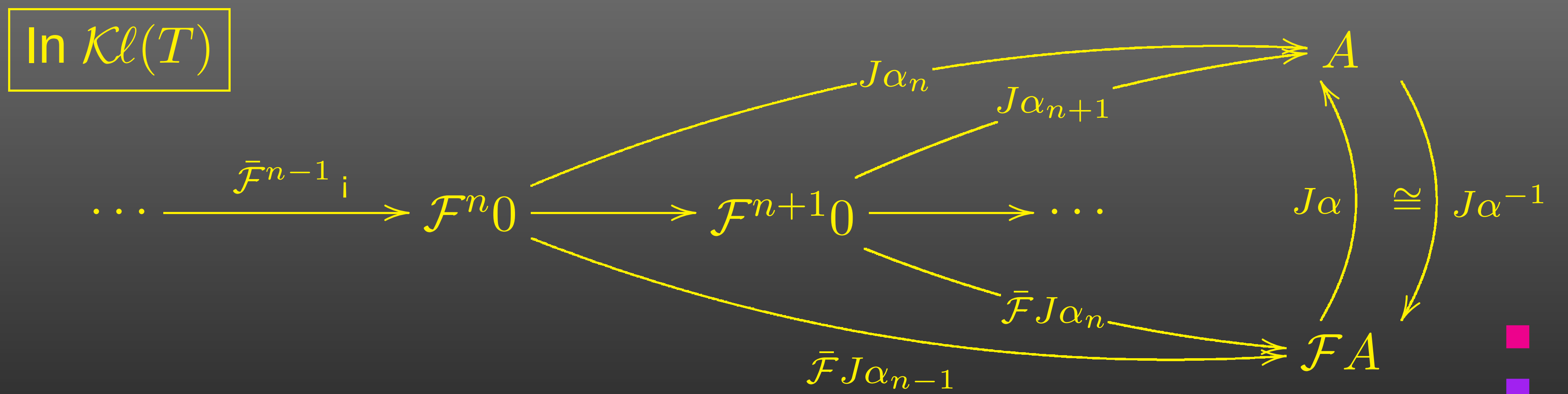


Proof sketch

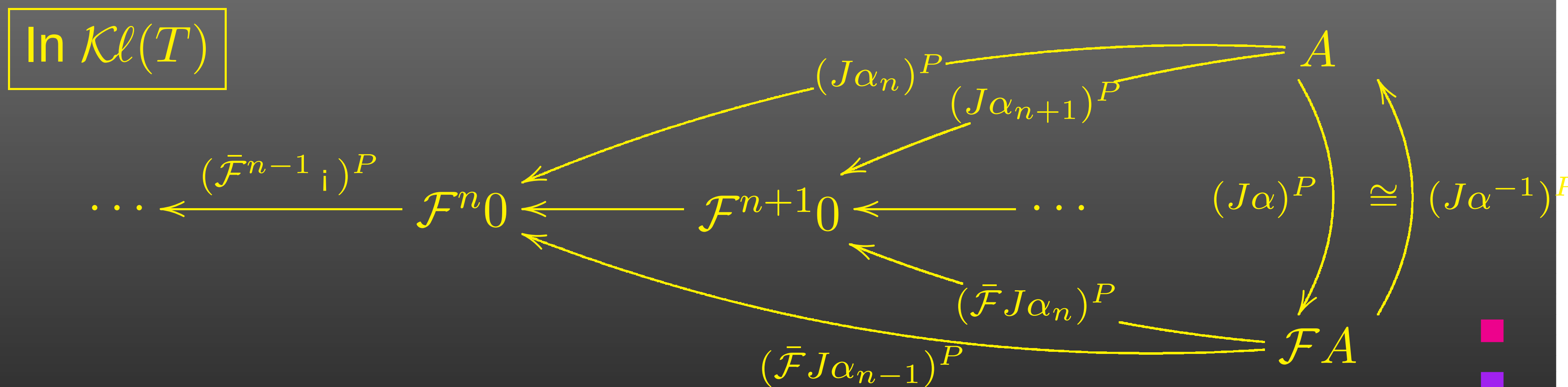
In $\mathcal{K}\ell(T)$



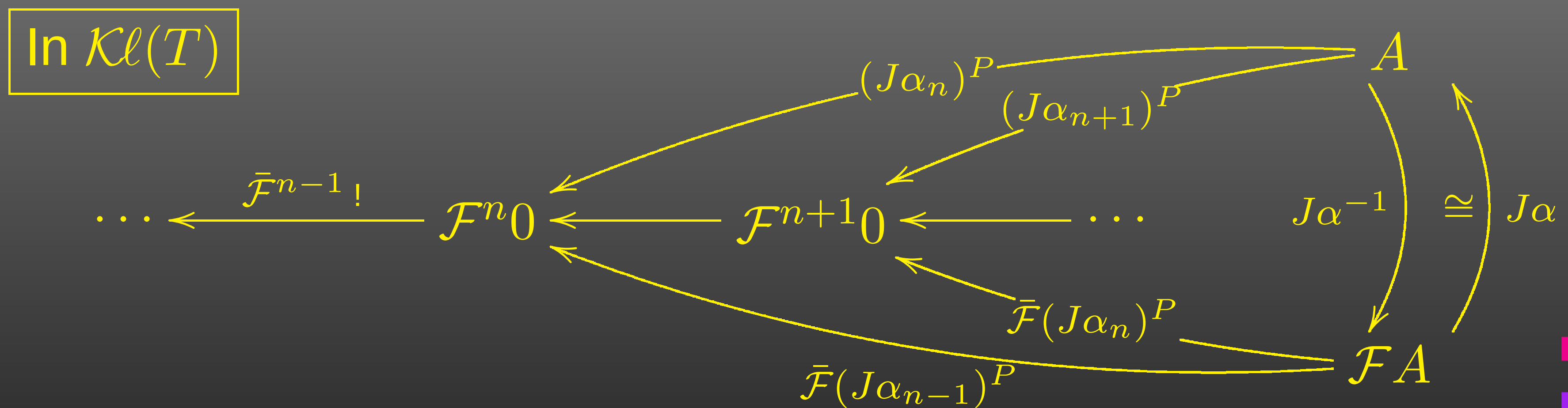
Proof sketch



Proof sketch



Proof sketch



Corollary (♣)

For $X \xrightarrow{c} \mathcal{F}_{\mathcal{K}\ell(\mathcal{T})}X$ in $\mathcal{K}\ell(\mathcal{T})$

Corollary (♣)

For $X \xrightarrow{c} \mathcal{F}_{\mathcal{K}\ell(\mathcal{T})} X$ in $\mathcal{K}\ell(\mathcal{T})$ ... $X \xrightarrow{c} \mathcal{T}\mathcal{F}X$ in $\mathbf{Sets}$

Corollary (♣)

For $X \xrightarrow{c} \mathcal{F}_{\mathcal{K}\ell(\mathcal{T})}X$ in $\mathcal{K}\ell(\mathcal{T})$ ... $X \xrightarrow{c} \mathcal{T}\mathcal{F}X$ in $\mathbf{Sets}$

$\exists!$ finite trace map $\mathrm{tr}_c : X \to \mathcal{T}A$ in $\mathbf{Sets}$:

Corollary (♣)

For $X \stackrel{c}{\to} \mathcal{F}_{\mathcal{K}\ell(\mathcal{T})}X$ in $\mathcal{K}\ell(\mathcal{T})$ ... $X \stackrel{c}{\to} \mathcal{T}\mathcal{F}X$ in $\mathbf{Sets}$

$\exists!$ finite trace map $\mathrm{tr}_c : X \to \mathcal{T}A$ in $\mathbf{Sets}$:

in $\mathcal{K}\ell(\mathcal{T})$

$$\begin{array}{ccc}
\mathcal{F}_{\mathcal{K}\ell(\mathcal{T})}X & \xrightarrow{\mathcal{F}_{\mathcal{K}\ell(\mathcal{T})}(\mathrm{tr}_c)} & \mathcal{F}_{\mathcal{K}\ell(\mathcal{T})}A \\
{\scriptstyle c}\uparrow & & \uparrow{\scriptstyle \cong} \\
X & \xrightarrow[\mathrm{tr}_c]{} & A
\end{array}$$

It works for...

- branching types:
  - * lift monad $1 + \_$

    systems with non-termination, exception
  - * powerset monad $\mathcal{P}$

    non-deterministic systems
  - * subdistribution monad $\mathcal{D}$

    probabilistic systems

It works for...

- branching types:
  - * lift monad $1 + \_$

    systems with non-termination, exception
  - * powerset monad $\mathcal{P}$

    non-deterministic systems
  - * subdistribution monad $\mathcal{D}$

    probabilistic systems

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

It works for...

- branching types:
  - * lift monad $1 + \_$
    systems with non-termination, exception
  - * powerset monad $\mathcal{P}$
    non-deterministic systems
  - * subdistribution monad $\mathcal{D}$
    probabilistic systems

all with pointwise order !

together with...

- linear I/O types:

together with...

- linear I/O types: shapely functors

together with...

- linear I/O types: shapely functors

$$\mathcal{F} = \textit{id} \mid \Sigma \mid F \times F \mid \coprod_i F_i$$

together with...

- linear I/O types: shapely functors

$$\mathcal{F} = \textit{id} \mid \Sigma \mid F \times F \mid \coprod_i F_i$$

  - * modular distributive law between commutative monads and shapely functors
  - * our monads are commutative

Hence, it works...

- for LTS with explicit termination

$$\mathcal{P}(1 + \Sigma \times \_\,)$$

Hence, it works...

- for LTS with explicit termination

$$\mathcal{P}(1 + \Sigma \times \_\,)$$

- for generative systems with explicit termination

$$\mathcal{D}(1 + \Sigma \times \_\,)$$

Hence, it works...

- for LTS with explicit termination

$$\mathcal{P}(1 + \Sigma \times \_\ )$$

- for generative systems with explicit termination

$$\mathcal{D}(1 + \Sigma \times \_\ )$$

Note: Initial $1 + \Sigma \times \_$ - algebra is

$$\Sigma^* \xrightarrow[\cong]{[\text{nil},\text{cons}]} 1 + \Sigma \times \Sigma^*$$

Finite traces - LTS with $\checkmark$

the finality diagram in $\mathcal{K}\ell(\mathcal{P})$

$$
\begin{array}{ccc}
\mathcal{F}_{\mathcal{K}\ell(\mathcal{P})}X & \xrightarrow{\mathcal{F}_{\mathcal{K}\ell(\mathcal{P})}(\mathrm{tr}_c)} & \mathcal{F}_{\mathcal{K}\ell(\mathcal{P})}\Sigma^* \\
c \uparrow & & \uparrow \cong \\
X & \xrightarrow[\mathrm{tr}_c]{} & \Sigma^*
\end{array}
$$

Finite traces - LTS with ✓

the finality diagram in $\mathcal{K}\ell(\mathcal{P})$

$$\begin{array}{ccc}
1+\Sigma\times X & \xrightarrow{(1+\Sigma\times_)_{\mathcal{K}\ell(\mathcal{P})}(\mathrm{tr}_c)} & 1+\Sigma\times\Sigma^* \\
\uparrow c & & \uparrow \cong \\
X & \xrightarrow[\mathrm{tr}_c]{} & \Sigma^*
\end{array}$$

Finite traces - LTS with ✓

the finality diagram in $\mathcal{K}\ell(\mathcal{P})$

$$\begin{array}{ccc}
1+\Sigma\times X & \xrightarrow{(1+\Sigma\times_)_{\mathcal{K}\ell(\mathcal{P})}(\mathrm{tr}_c)} & 1+\Sigma\times\Sigma^* \\
\uparrow c & & \uparrow \cong \\
X & \xrightarrow[\mathrm{tr}_c]{} & \Sigma^*
\end{array}$$

amounts to

- $\langle\rangle \in \mathrm{tr}_c(x) \iff \checkmark \in c(x)$
- $a \cdot w \in \mathrm{tr}_c(x) \iff (\exists x')\langle a, x'\rangle \in c(x),\ w \in \mathrm{tr}_c(x')$

Finite traces - generative ✓

the finality diagram in $\mathcal{K}\ell(\mathcal{D})$

$$
\begin{array}{ccc}
\mathcal{F}_{\mathcal{K}\ell(\mathcal{D})}X & \xrightarrow{\mathcal{F}_{\mathcal{K}\ell(\mathcal{D})}(\mathrm{tr}_c)} & \mathcal{F}_{\mathcal{K}\ell(\mathcal{D})}\Sigma^* \\
c \uparrow & & \uparrow \cong \\
X & \xrightarrow[\mathrm{tr}_c]{} & \Sigma^*
\end{array}
$$

Finite traces - generative ✓

the finality diagram in $\mathcal{K}\ell(\mathcal{D})$

$$\begin{array}{ccc}
1+\Sigma\times X & \xrightarrow{(1+\Sigma\times _)_{\mathcal{K}\ell(\mathcal{D})}(\mathrm{tr}_c)} & 1+\Sigma\times\Sigma^* \\
\uparrow c & & \uparrow \cong \\
X & \xrightarrow[\mathrm{tr}_c]{} & \Sigma^*
\end{array}$$

Finite traces - generative ✓

the finality diagram in $\mathcal{K}\ell(\mathcal{D})$

$$\begin{array}{ccc}
1+\Sigma\times X & \xrightarrow{(1+\Sigma\times_)_{\mathcal{K}\ell(\mathcal{D})}(\mathrm{tr}_c)} & 1+\Sigma\times\Sigma^* \\
\uparrow c & & \uparrow \cong \\
X & \xrightarrow[\mathrm{tr}_c]{} & \Sigma^*
\end{array}$$

amounts to $\mathrm{tr}_c(x)$:

- $\langle\rangle \mapsto c(x)(\checkmark)$
- $a\cdot w \mapsto \sum_{y\in X} c(x)(a,y)\cdot c(y)(w)$

Conclusions

- Systems as coalgebras
- Behaviour via coinduction

Conclusions

- Systems as coalgebras
- Behaviour via coinduction
  - * bisimilarity: coinduction in $\mathbf{Sets}$
  - * trace semantics: coinduction

    in $\mathcal{K}\ell(\mathcal{T})$

$$
\begin{array}{ccc}
\mathcal{F}_{\mathcal{K}\ell(\mathcal{T})} X & \xrightarrow{\;\mathcal{F}_{\mathcal{K}\ell(\mathcal{T})}(\mathrm{tr}_c)\;} & \mathcal{F}_{\mathcal{K}\ell(\mathcal{T})} A \\
{\scriptstyle c}\uparrow & & \uparrow{\scriptstyle \cong} \\
X & \xrightarrow[\mathrm{tr}_c]{} & A
\end{array}
$$

Conclusions

- Systems as coalgebras
- Behaviour via coinduction
  - * bisimilarity: coinduction in $\mathbf{Sets}$
  - * trace semantics: coinduction

    in $\mathcal{K}\ell(\mathcal{T})$

$$\begin{array}{ccc} \mathcal{F}_{\mathcal{K}\ell(\mathcal{T})}X & \overset{\mathcal{F}_{\mathcal{K}\ell(\mathcal{T})}(\mathrm{tr}_c)}{\dashrightarrow} & \mathcal{F}_{\mathcal{K}\ell(\mathcal{T})}A \\ c \uparrow & & \uparrow \cong \\ X & \underset{\mathrm{tr}_c}{\dashrightarrow} & A \end{array}$$

- Main technical result: initial algebra = final coalgebra

Future work

- Combined monads:

Future work

- Combined monads:
  * non-determinism + probability
    [Vardi ’85, Segala & Lynch ’95]
    monad/order structure yet to be found
    [Varacca & Winskel ’05]
    initial work by Jacobs’08

Future work

- Combined monads:
  * non-determinism + probability
    [Vardi '85, Segala & Lynch '95]
    monad/order structure yet to be found
    [Varacca & Winskel '05]
    initial work by Jacobs'08
  * $\mathcal{PP}$ [Kupke & Venema '05]

Future work

- Combined monads:
  * non-determinism + probability
    [Vardi '85, Segala & Lynch '95]
    monad/order structure yet to be found
    [Varacca & Winskel '05]
    initial work by Jacobs'08
  * $\mathcal{P}\mathcal{P}$ [Kupke & Venema '05]
- Between bisimilarity and trace in the spectrum

Future work

- Combined monads:
  * non-determinism + probability
    [Vardi ’85, Segala & Lynch ’95]
    monad/order structure yet to be found
    [Varacca & Winskel ’05]
    initial work by Jacobs’08
  * $\mathcal{P}\mathcal{P}$ [Kupke & Venema ’05]
- Between bisimilarity and trace in the spectrum