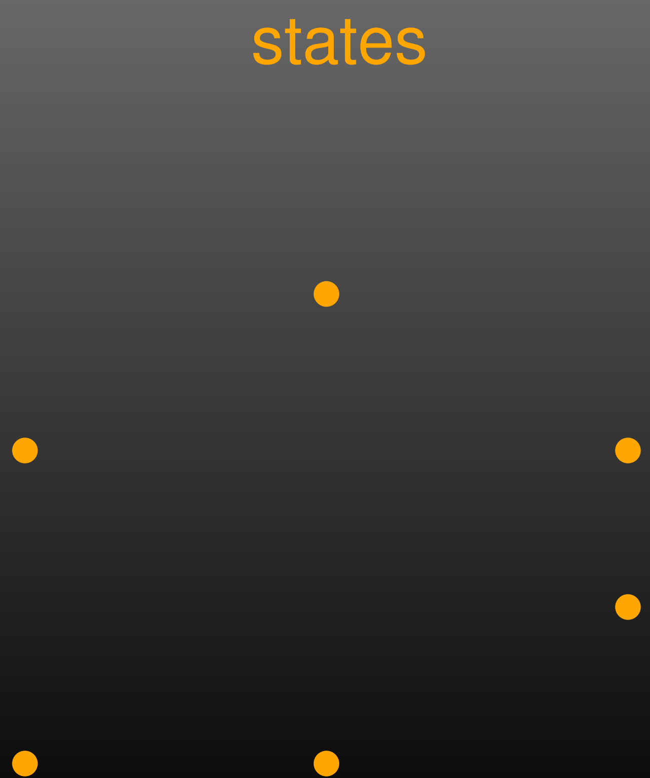
Generic Trace Theory

Ichiro Hasuo, Bart Jacobs and Ana Sokolova

SOS group - Radboud University Nijmegen

Talk about...

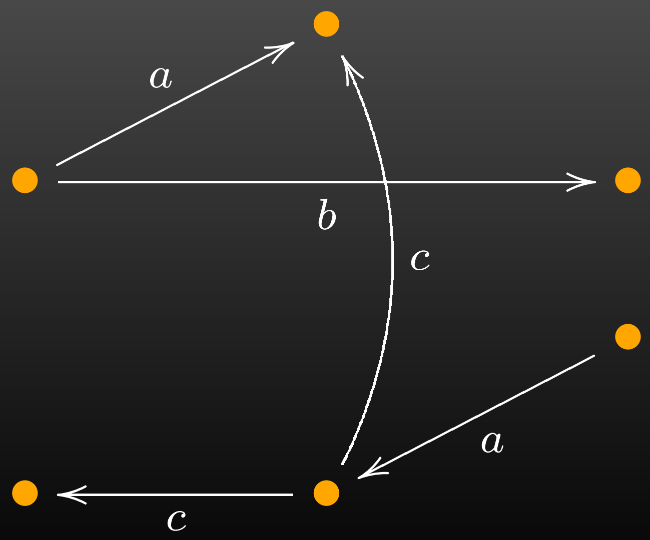
- systems as coalgebras



Talk about...

- systems as coalgebras

states + transitions



Talk about...

- systems as coalgebras

  states + transitions

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

Talk about...

- systems as coalgebras

  states + transitions

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

- semantic relations represent behaviour

Talk about...

- systems as coalgebras

  states + transitions

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

- semantic relations represent behaviour

  LT/BT spectrum

Talk about...

- systems as coalgebras

  states + transitions

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

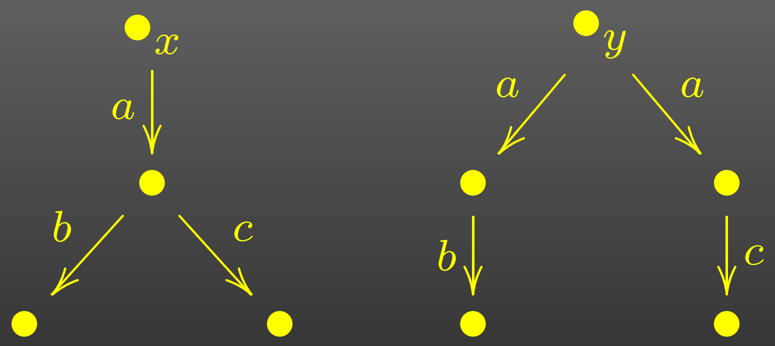
- semantic relations represent behaviour

  LT/BT spectrum

  ... linear-time behaviour via trace semantics

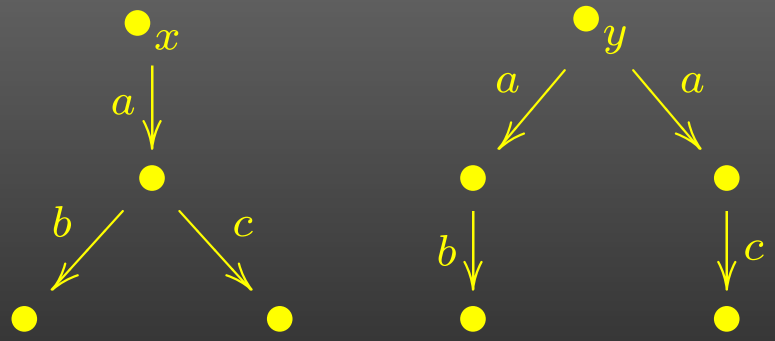
LT/BT spectrum

Are these non-deterministic systems equal ?



LT/BT spectrum

Are these non-deterministic systems equal ?

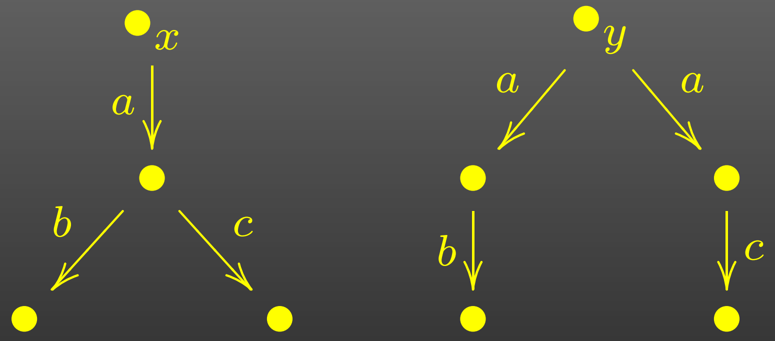


x and y are:

- different wrt. bisimilarity

LT/BT spectrum

Are these non-deterministic systems equal ?



x and y are:

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

Traces - LTS

For LTS with explicit termination (NA)

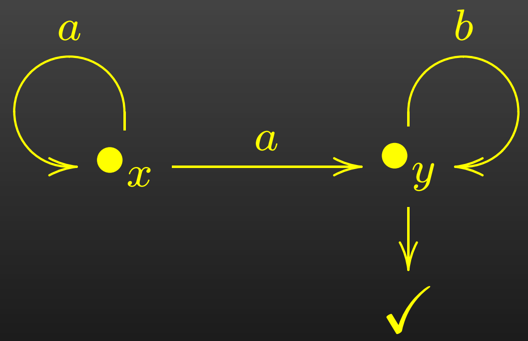
trace = the set of all possible linear behaviors

Traces - LTS

For LTS with explicit termination (NA)

trace = the set of all possible linear behaviors

Example:



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

Traces - generative

For generative probabilistic systems with ex. termination

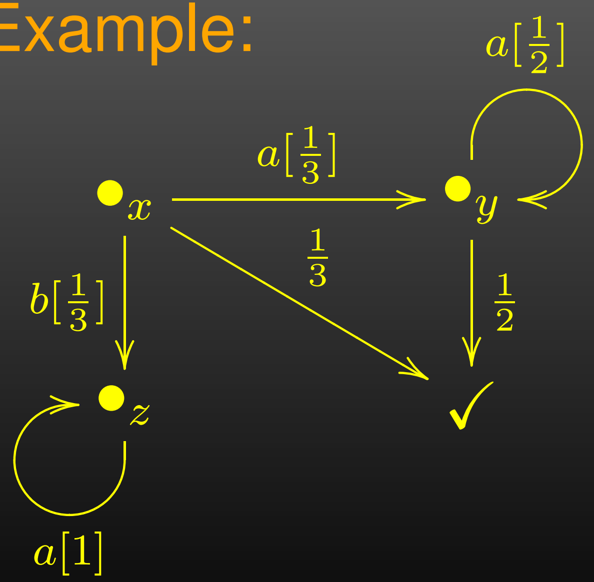
trace = sub-probability distribution over possible linear behaviors

Traces - generative

For generative probabilistic systems with ex. termination

trace = sub-probability distribution over possible linear behaviors

Example:



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

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

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

$$\cdots$$

Trace of a coalgebra ?

Trace of a coalgebra ?

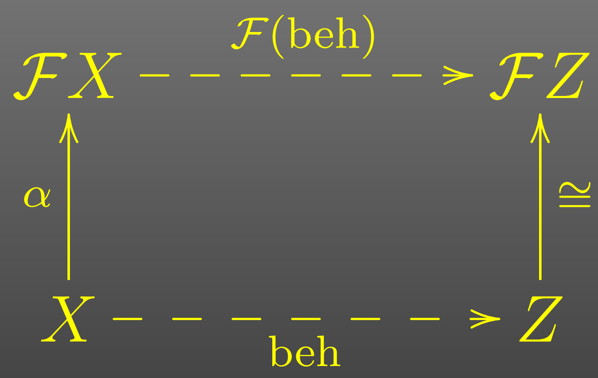
- Power&Turi '99 - $\mathcal{P}(1 + \Sigma \times \_)$
- Jacobs '04 - $\mathcal{PF}$
- Hasuo&Jacobs CALCO '05 - $\mathcal{PF}$, shapely $\mathcal{F}$
- Hasuo&Jacobs CALCO Jnr '05 - $\mathcal{DF}$, shapely $\mathcal{F}$
- Generic Trace Theory - $\mathcal{TF}$, order-enriched setting

Trace of a coalgebra ?

- Power&Turi '99 - $\mathcal{P}(1 + \Sigma \times \_)$
- Jacobs '04 - $\mathcal{P}\mathcal{F}$
- Hasuo&Jacobs CALCO '05 - $\mathcal{P}\mathcal{F}$, shapely $\mathcal{F}$
- Hasuo&Jacobs CALCO Jnr '05 - $\mathcal{D}\mathcal{F}$, shapely $\mathcal{F}$
- Generic Trace Theory - $\mathcal{T}\mathcal{F}$, order-enriched setting

main idea: coinduction in a Kleisli category

Coinduction



system final coalgebra

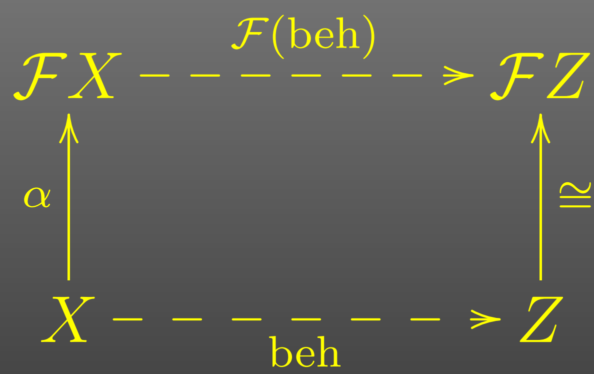
Coinduction

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

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 $\mathbf{Sets}$ = bisimilarity
- f.c.s. in a Kleisli category = trace semantics

Types of systems

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

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

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 $\mathbf{Sets}$

$$X \stackrel{c}{\to} \mathcal{T}\,\mathcal{F}X$$

monad - branching type

functor - linear i/o type

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

Distributive law

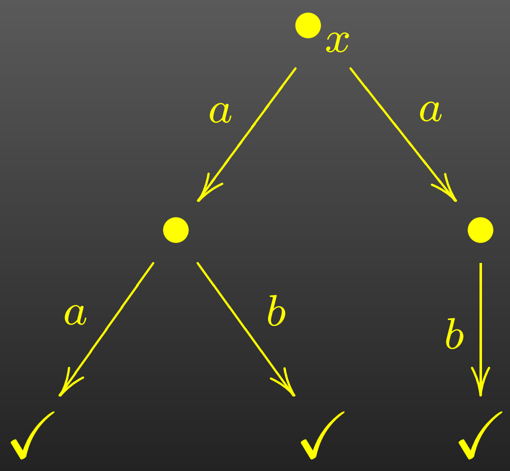
is needed since branching is irrelevant:

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

Distributive law

is needed since branching is irrelevant:

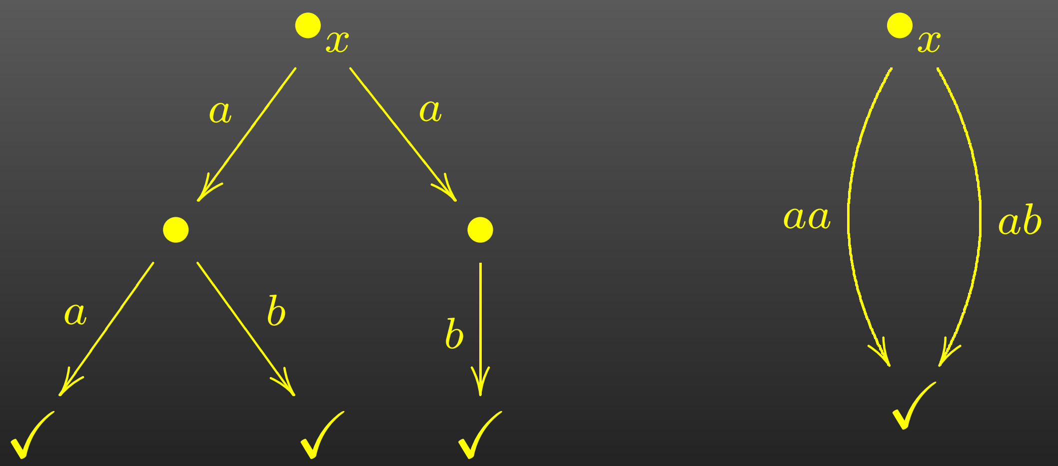
LTS with ✓ - $\mathcal{PF} = \mathcal{P}(1 + \Sigma \times \_)$



Distributive law

is needed since branching is irrelevant:

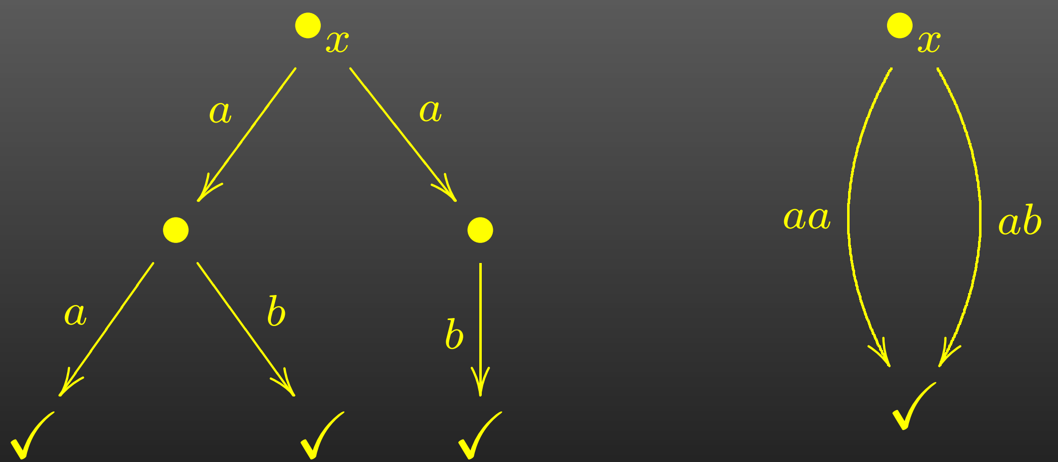
LTS with ✓- $\mathcal{PF} = \mathcal{P}(1 + \Sigma \times \_)$



Distributive law

is needed since branching is irrelevant:

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

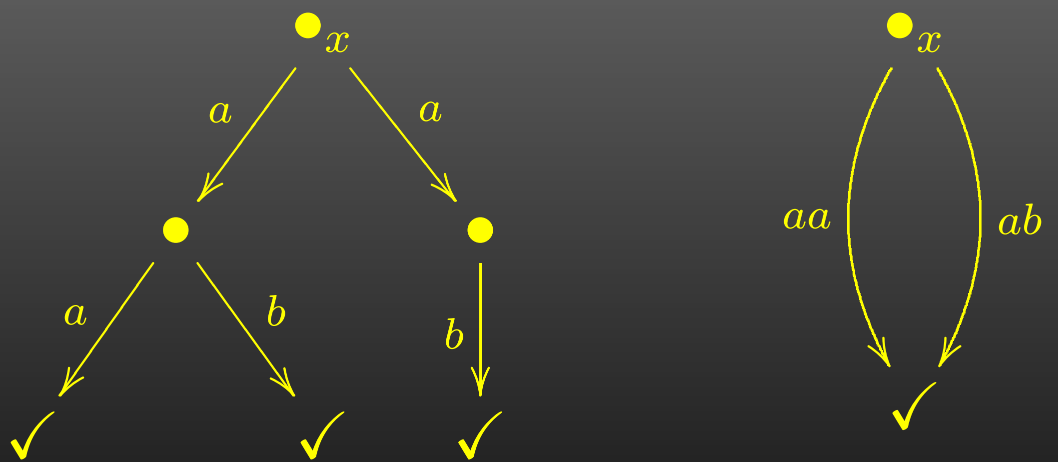


$X \xrightarrow{c} \mathcal{PF}X$

Distributive law

is needed since branching is irrelevant:

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

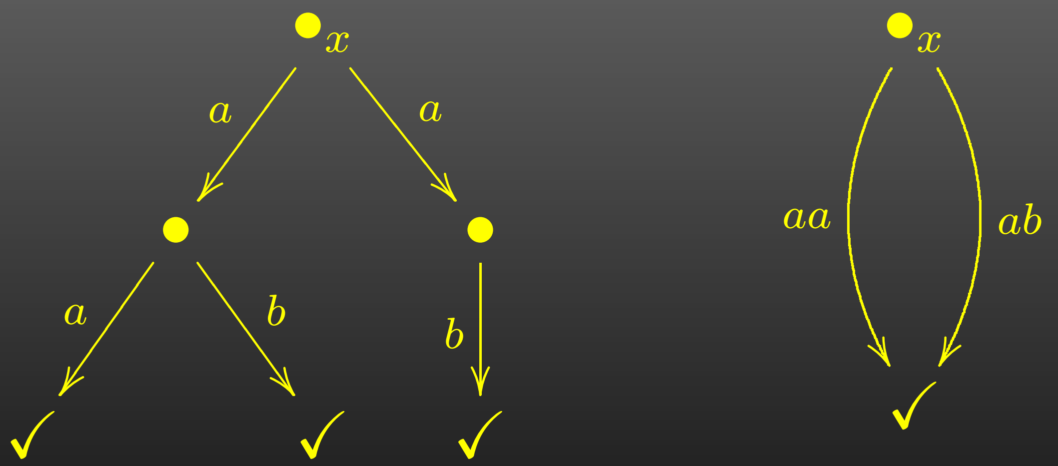


$X \xrightarrow{c} \mathcal{PF}X$

Distributive law

is needed since branching is irrelevant:

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

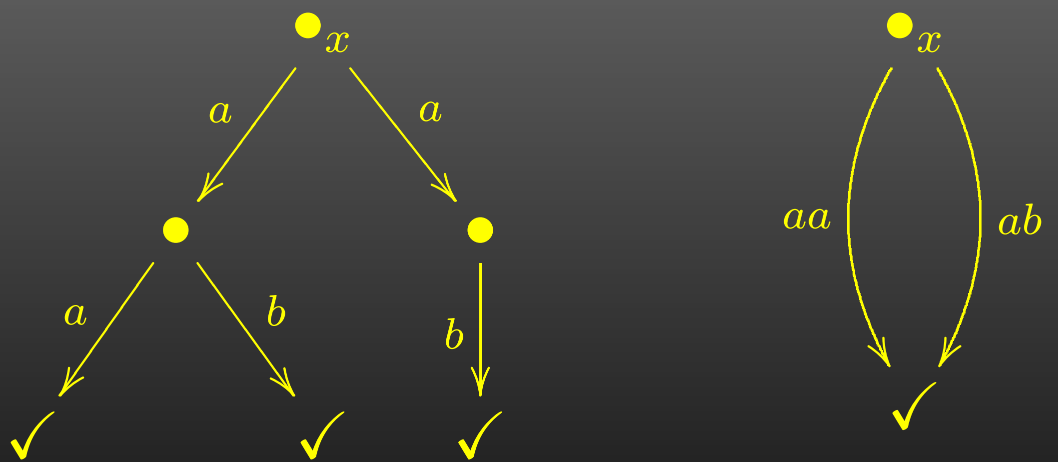


$$X \xrightarrow{c} \mathcal{PF}X \xrightarrow{\mathcal{PF}c} \mathcal{PFPF}X$$

Distributive law

is needed since branching is irrelevant:

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

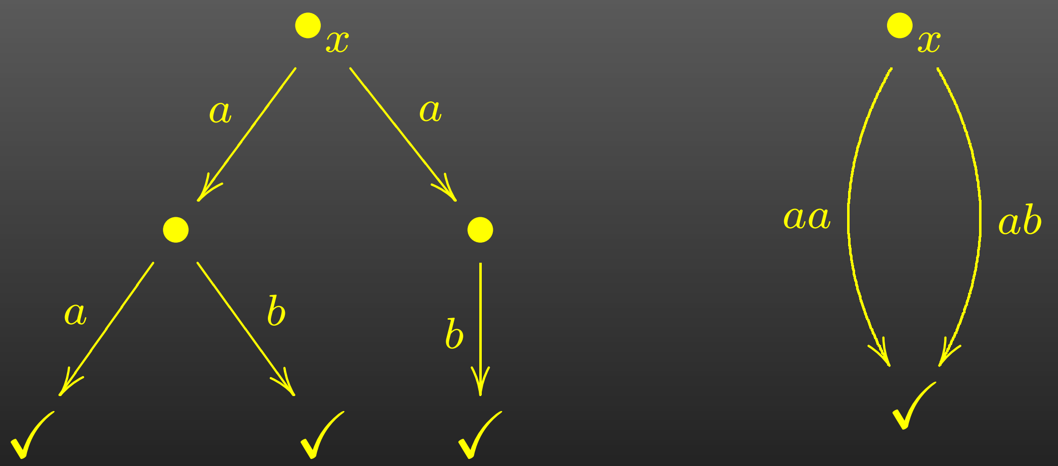


$X \xrightarrow{c} \mathcal{PF}X \xrightarrow{\mathcal{PF}c} \mathcal{PFPF}X$

Distributive law

is needed since branching is irrelevant:

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

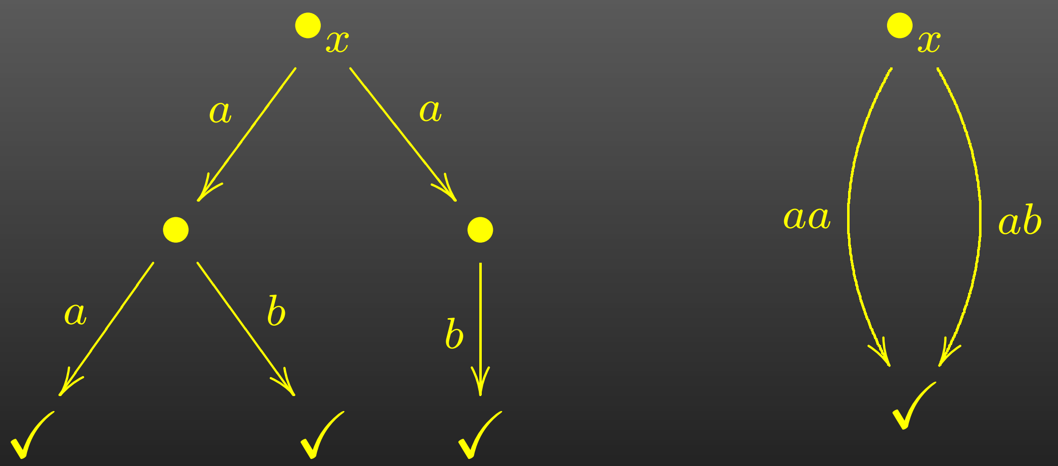


$$X \xrightarrow{c} \mathcal{PF}X \xrightarrow{\mathcal{PF}c} \mathcal{PFPF}X \xrightarrow{\text{d.l.}} \mathcal{PPFF}X$$

Distributive law

is needed since branching is irrelevant:

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

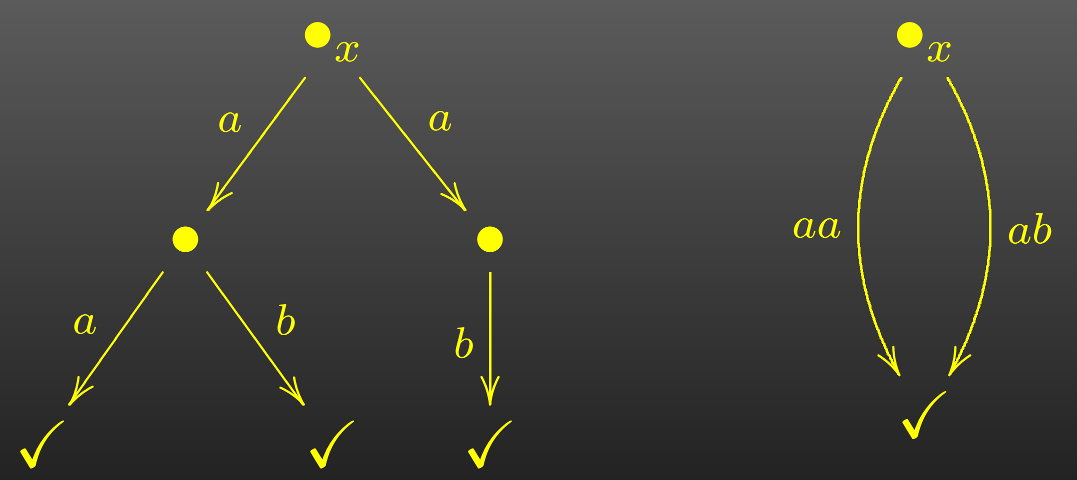


$$X \stackrel{c}{\to} \mathcal{P}\mathcal{F}X \stackrel{\mathcal{P}\mathcal{F}c}{\to} \mathcal{P}\mathcal{F}\mathcal{P}\mathcal{F}X \stackrel{\text{d.l.}}{\to} \mathcal{P}\mathcal{P}\mathcal{F}\mathcal{F}X$$

Distributive law

is needed since branching is irrelevant:

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

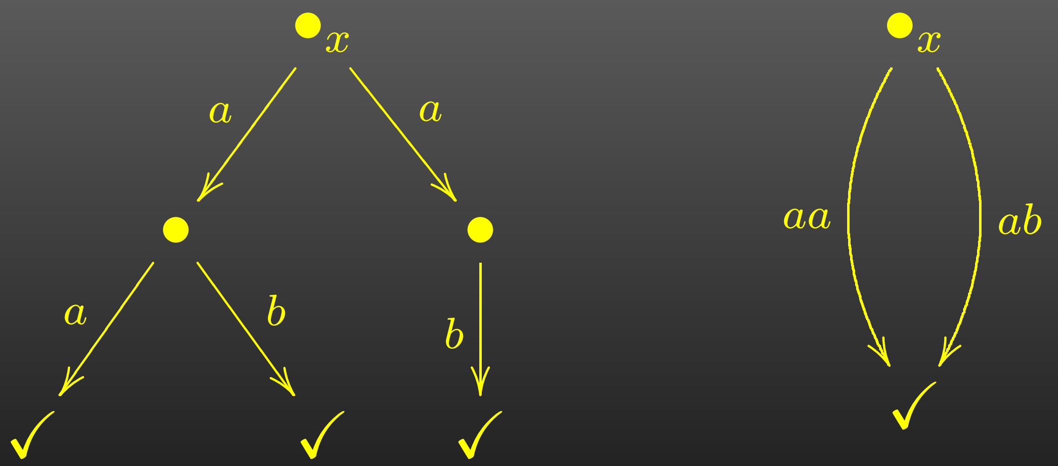


$$X \stackrel{c}{\to} \mathcal{P}\mathcal{F}X \stackrel{\mathcal{P}\mathcal{F}c}{\to} \mathcal{P}\mathcal{F}\mathcal{P}\mathcal{F}X \stackrel{\text{d.l.}}{\to} \mathcal{P}\mathcal{P}\mathcal{F}\mathcal{F}X \stackrel{\text{m.m.}}{\to} \mathcal{P}\mathcal{F}\mathcal{F}X$$

Distributive law

is needed since branching is irrelevant:

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



$$X \xrightarrow{c} \mathcal{P}\mathcal{F}X \xrightarrow{\mathcal{P}\mathcal{F}c} \mathcal{P}\mathcal{F}\mathcal{P}\mathcal{F}X \xrightarrow{\text{d.l.}} \mathcal{P}\mathcal{P}\mathcal{F}\mathcal{F}X \xrightarrow{\text{m.m.}} \mathcal{P}\mathcal{F}\mathcal{F}X$$

Distributive law

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

Distributive law

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

- objects - sets
- arrows - $X \xrightarrow{f} Y$ 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 $X \stackrel{c}{\to} \mathcal{T}\mathcal{F}X$ 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 $\mathcal{K}\ell(\mathcal{T})$!!!

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

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

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

Main Theorem

If ♣, then

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

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

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

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

[$\alpha : \mathcal{F}A \stackrel{\cong}{\to} A$ 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 $\mathbf{DCpo}_{\perp}$-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

$$0 \xrightarrow{\text{¡}} \mathcal{F}0 \xrightarrow{\mathcal{F}\,\text{¡}} \ldots \mathcal{F}^n 0 \xrightarrow{\mathcal{F}^n\,\text{¡}} \mathcal{F}^{n+1}0 \xrightarrow{\mathcal{F}^{n+1}\,\text{¡}} \cdots$$

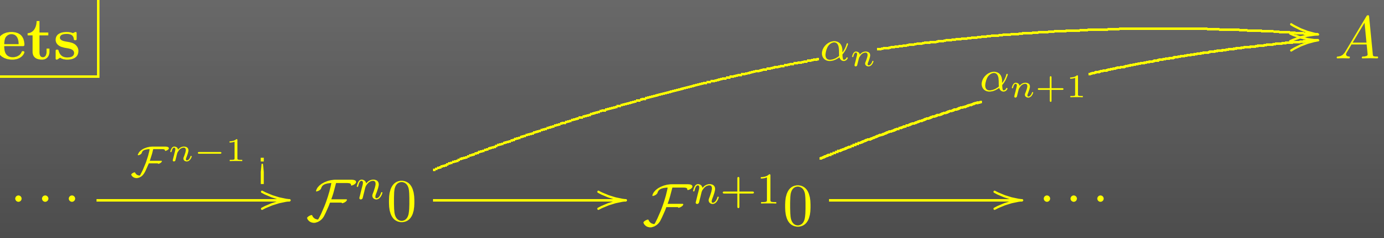
Proof sketch

In Sets

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

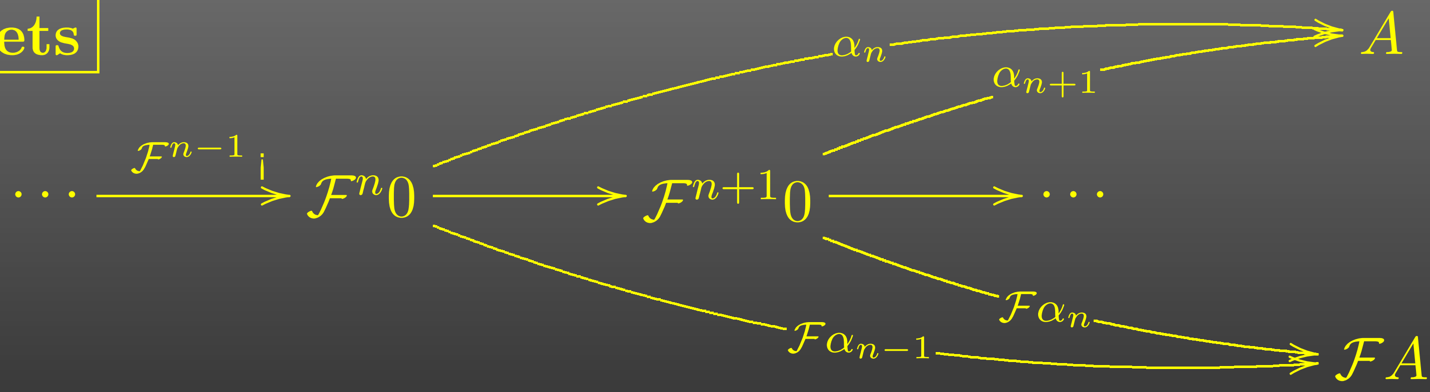
Proof sketch

In Sets



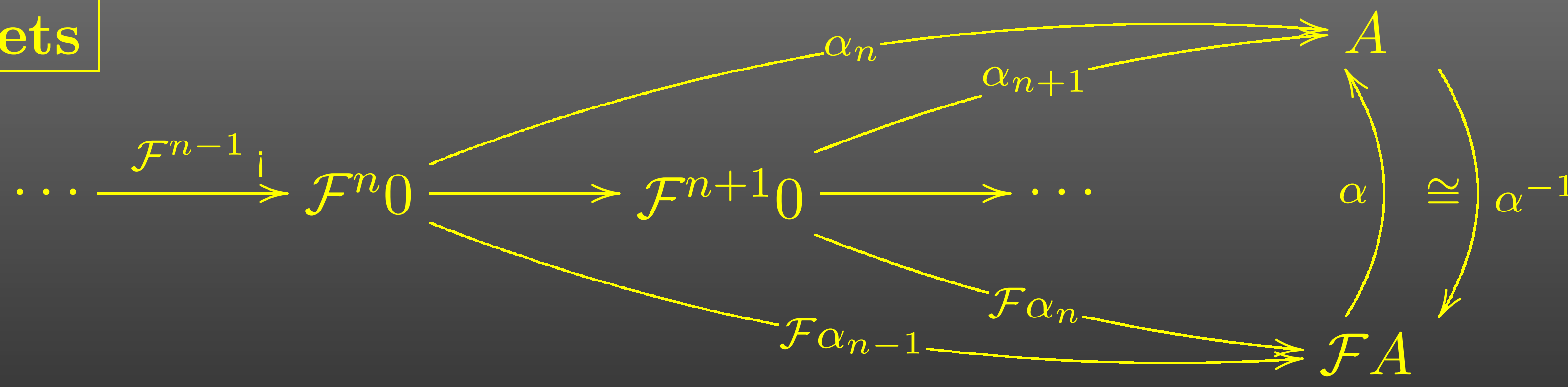
Proof sketch

In Sets



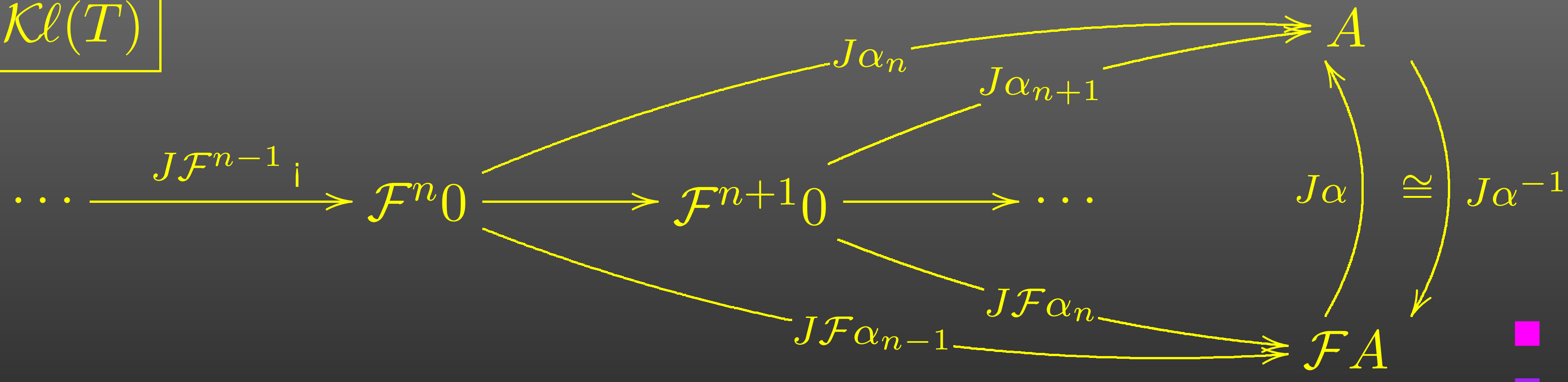
Proof sketch

In Sets



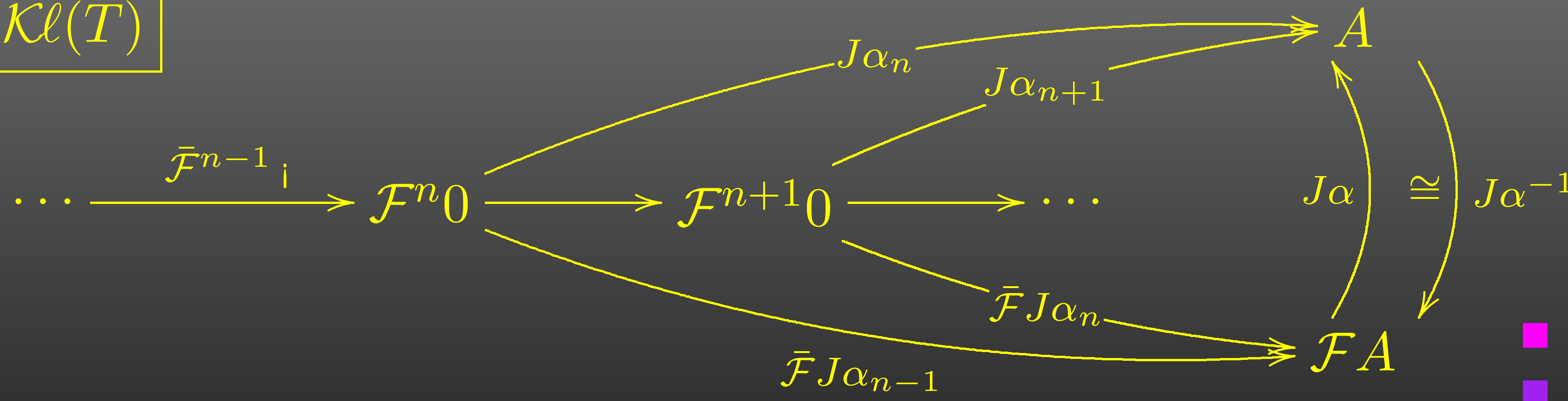
Proof sketch

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

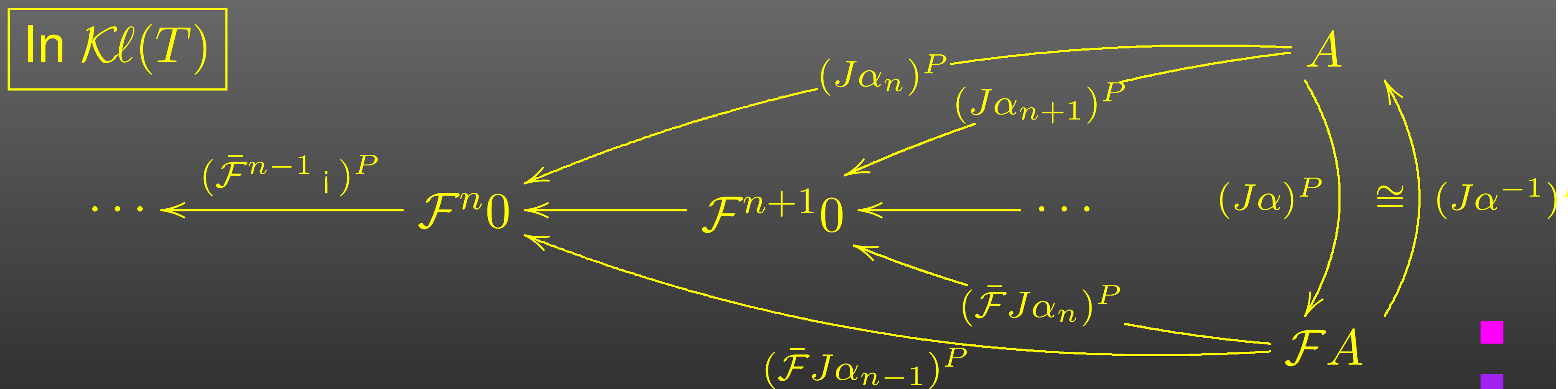


Proof sketch

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

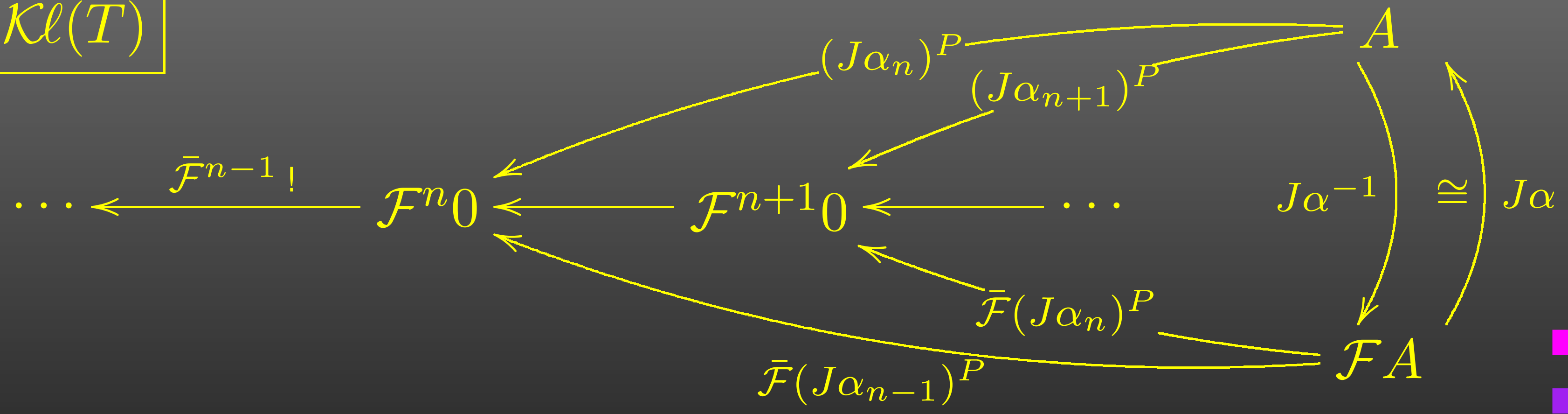


Proof sketch



Proof sketch

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



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

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 $\mathbf{Sets}$:

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

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

together with...

- linear I/O types: shapely functors

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

  * 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

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

Finite traces - LTS with $\checkmark$

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

$$\begin{array}{ccc}
\mathcal{F}_{\mathcal{K}\ell(\mathcal{P})} X & \overset{\mathcal{F}_{\mathcal{K}\ell(\mathcal{P})}(\mathrm{tr}_c)}{- - - - - - \to} & \mathcal{F}_{\mathcal{K}\ell(\mathcal{P})} \Sigma^* \\
c \uparrow & & \uparrow \cong \\
X & \underset{\mathrm{tr}_c}{- - - - - - - - - \to} & \Sigma^*
\end{array}$$

Finite traces - LTS with $\checkmark$

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

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

Finite traces - generative ✓

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

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

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

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

Parallel composition

For $u, v \in \mathcal{P}(\Sigma^*)$ the (shuffle) parallel composition $u \parallel v$:

$$\langle\rangle \in u \parallel v \quad \stackrel{\text{def}}{\Longleftrightarrow} \quad \langle\rangle \in u \;\text{ and }\; \langle\rangle \in v$$

$$a \cdot w \in u \parallel v \quad \stackrel{\text{def}}{\Longleftrightarrow} \quad w \in \partial_a u \parallel v \;\text{ or }\; w \in u \parallel \partial_a v$$

for $\partial_a u = \{w \in \Sigma^* \mid a \cdot w \in u\}$
can be defined by coinduction

Parallel composition

For $u, v \in \mathcal{P}(\Sigma^*)$ the (shuffle) parallel composition $u \parallel v$:

$$\langle\rangle \in u \parallel v \quad \stackrel{\text{def}}{\Longleftrightarrow} \quad \langle\rangle \in u \ \text{ and } \ \langle\rangle \in v$$

$$a \cdot w \in u \parallel v \quad \stackrel{\text{def}}{\Longleftrightarrow} \quad w \in \partial_a u \parallel v \ \text{ or } \ w \in u \parallel \partial_a v$$

for $\partial_a u = \{w \in \Sigma^* \mid a \cdot w \in u\}$

can be defined by coinduction

Also: Equations

$$u \parallel v = v \parallel u,\ (u \parallel v) \parallel w = u \parallel (v \parallel w), \ldots$$

can be proved by coinduction

Conclusions

- Systems as coalgebras
- Behaviour via coinduction

Conclusions

- Systems as coalgebras
- Behaviour via coinduction
  - * generic trace semantics: coinduction

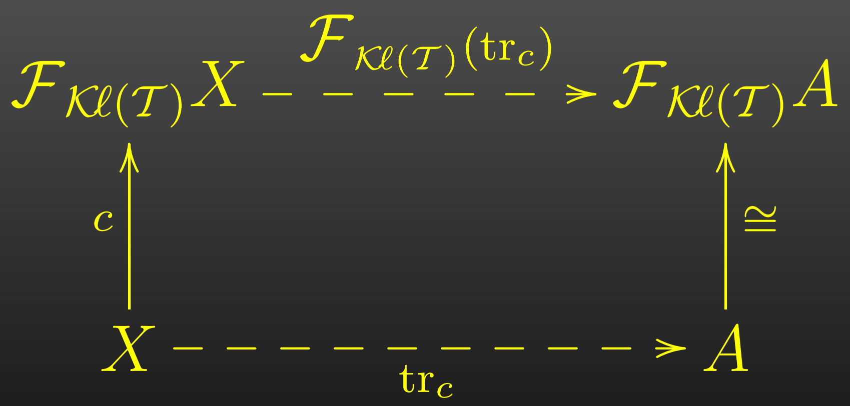
in $\mathcal{K}\ell(\mathcal{T})$

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

Conclusions

- Systems as coalgebras
- Behaviour via coinduction
  - * generic trace semantics: coinduction

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



- Main technical result: initial algebra = final coalgebra

  in an order enriched setting

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]

Future work

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