Foundations of Total Functional Data-Flow Programming, Coinductively

(Cross-Complement from MSFP 2014)

Baltasar Trancón y Widemann[1,2] Markus Lepper[2]

[1] Ilmenau University of Technology

[2] <semantics/> GmbH

12th CMCS Workshop
2014-04-05/06

Agenda

Agenda

Exercise #1

Functor $\mathcal{S}_A = A \times (-)$

Coalgebras Deterministic total automata, output A (only)

Final Streams A^ω with deconstruction

$$\text{out}_{\mathcal{S}_A} = \text{cons}_A^{-1} = \langle \text{head}_A, \text{tail}_A \rangle$$

Coiteration

$$\big(X, \varphi = \langle h, t \rangle\big) \longmapsto [\![\varphi]\!]_{\mathcal{S}_A}(x)_n = h\big(t^n(x)\big)$$

$$h : X \to A$$

$$t : X \to X$$

Exercise #2

Functor $\mathcal{T}_{AB} = (B \times -)^A$

Coalgebras Total Mealy machines, input/output A/B

Final Causal stream functions $A^\omega \xrightarrow{!} B^\omega$

$$\text{out}_{\mathcal{T}_{AB}} = \lambda f. \left\langle \text{head}_{AB}(f), \text{tail}_{AB}(f) \right\rangle$$

Coiteration $(X, \varphi) \longmapsto [\![\varphi]\!]_{\mathcal{T}_{AB}}(x)(a)_n = \ldots$

Aside: Causality

$$a|_n = a'|_n \implies f(a)|_n = f(a')|_n$$

$$\text{head}_{AB}(f) : A \to B \qquad \text{tail}_{AB}(f) : A \to (A^\omega \xrightarrow{!} B^\omega)$$

$$\text{head}_{AB}(f)(a_0) = \text{head}_B\left(f\left(\text{cons}_A(a_0, \ldots)\right)\right)$$

$$\text{tail}_{AB}(f)(a_0)(a') = \text{tail}_B\left(f\left(\text{cons}_A(a_0, a')\right)\right)$$

Agenda

Declarative Online Stream Programming

Characteristics discrete (real) time, clocked synchronous, modular, data-centric

Applications very broad

- embedded systems
- numerical simulations
- scientific models
- digitial music & event arts

Paradigms deeply divided

Professionals Data-flow graph (DFG) visual systems Simulink, system dynamics, SuperCollider, ...

Academics Functional reactive programming (FRP)

Long-Term Goal: The Best of Both Worlds

FRP type discipline, symbolic computation, pattern matching

DFG aesthetic appeal, complex routing, ***standard math***

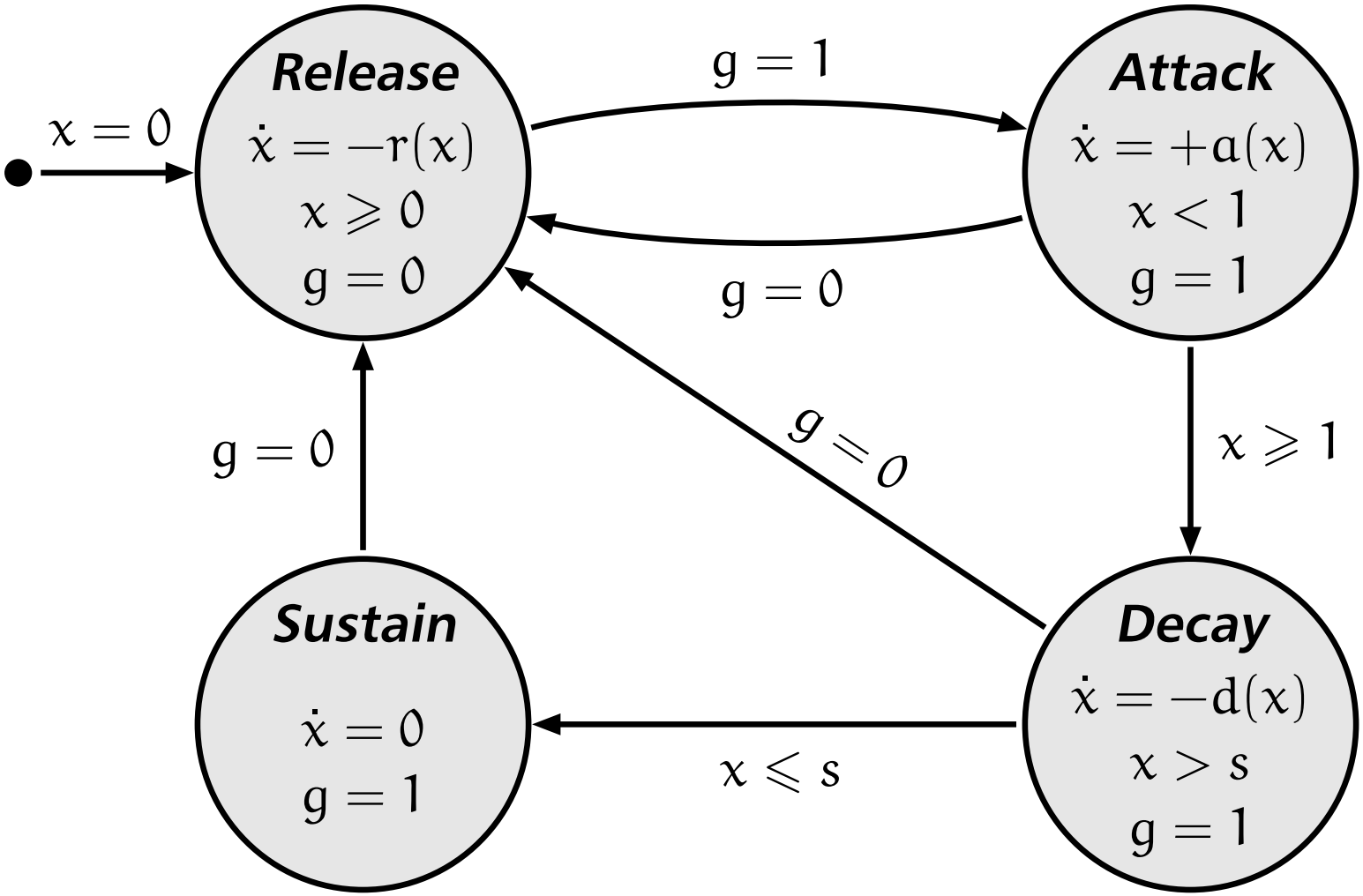
Excuse the pun, but. . .

Domain experts must not be required to study domain theory!

Unfortunately non-standard math lurking

Fortunately just coinduction

Example (1) – ADSR Hybrid Automaton



Example (1) – ADSR Functional Program

$$\textbf{data}\ \mathit{State} = \mathit{Attack} \mid \mathit{Decay} \mid \mathit{Sustain} \mid \mathit{Release}$$

$$\begin{aligned}
&\mathit{adsr} :: (\mathit{Num}\ \tau, \mathit{Ord}\ \tau) \Rightarrow (\tau \to \tau) \to (\tau \to \tau) \to \tau \to (\tau \to \tau) \to \\
&\qquad\qquad [\mathit{Bool}] \to [\tau] \\
&\mathit{adsr}\ \mathit{a_rate}\ \mathit{d_rate}\ \mathit{s_level}\ \mathit{r_rate} = \mathit{loop}\ \mathit{Release}\ 0 \\
&\quad \textbf{where}\ \mathit{loop}\ \mathit{state}\ \mathit{out}\ (\mathit{gate} : \mathit{gates}) = \mathit{out}' : \mathit{loop}\ \mathit{state}'\ \mathit{out}'\ \mathit{gates} \\
&\qquad\qquad \textbf{where}\ \mathit{out}' = \textbf{case}\ \mathit{state}\ \textbf{of} \\
&\qquad\qquad\qquad \begin{array}{lll}
\mathit{Attack} & \to \mathit{min}\ 1 & \$\ \mathit{out} + \mathit{a_rate}\ \mathit{out} \\
\mathit{Decay} & \to \mathit{max}\ \mathit{s_level} & \$\ \mathit{out} - \mathit{d_rate}\ \mathit{out} \\
\mathit{Sustain} & \to & \mathit{out} \\
\mathit{Release} & \to \mathit{max}\ 0 & \$\ \mathit{out} - \mathit{r_rate}\ \mathit{out}
\end{array} \\
&\qquad\qquad\quad \mathit{state}' = \textbf{case}\ \mathit{state}\ \textbf{of} \\
&\qquad\qquad\qquad \begin{array}{lll}
\mathit{Release} \mid & \mathit{gate} & \to \mathit{Attack} \\
\mathit{Attack} \mid & \mathit{gate} \wedge \mathit{out}' \geqslant 1 & \to \mathit{Decay} \\
\mathit{Decay} \mid & \mathit{gate} \wedge \mathit{out}' \leqslant \mathit{s_level} & \to \mathit{Sustain} \\
_ \mid & \neg\ \mathit{gate} & \to \mathit{Release} \\
_ & & \to \mathit{state}
\end{array}
\end{aligned}$$

Agenda

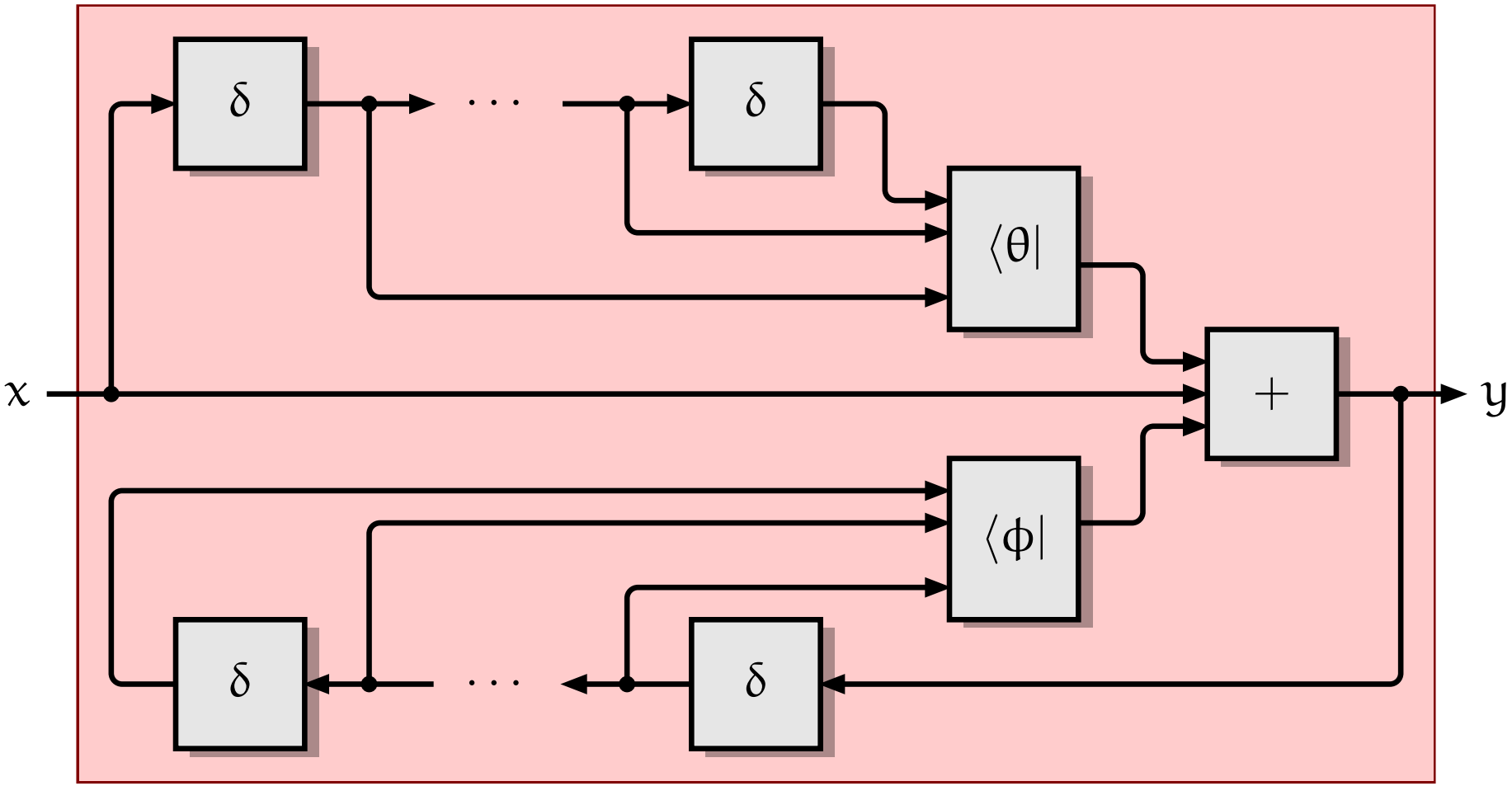
How State Comes In

Empirical Programming Lore

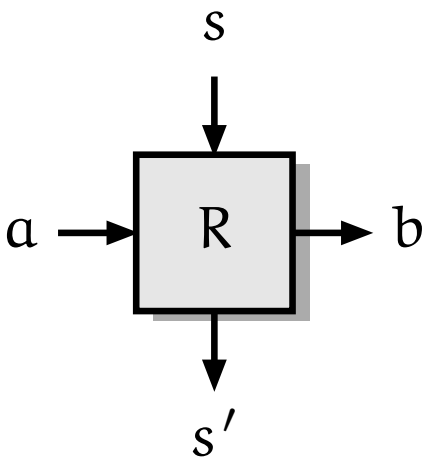
1. Computations are conceived as if per element
2. Most operations are stateless (repeated independently)
3. Interesting behavior arises from delay
4. State arises from delayed feedback

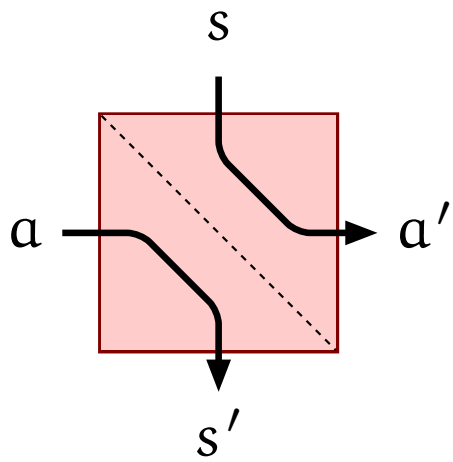
- Introduce state without breaking 1
- Exploit 2 – 4 as convenient

Example (2) – Auto-Regressive Moving Average



The Z Approach to Stateful Computation





$$R \subseteq S \times A \times B \times S$$

$$R : S \times A \to M(B \times S)$$

$$s \,/\, a \overset{R}{\longmapsto} b \,/\, s'$$

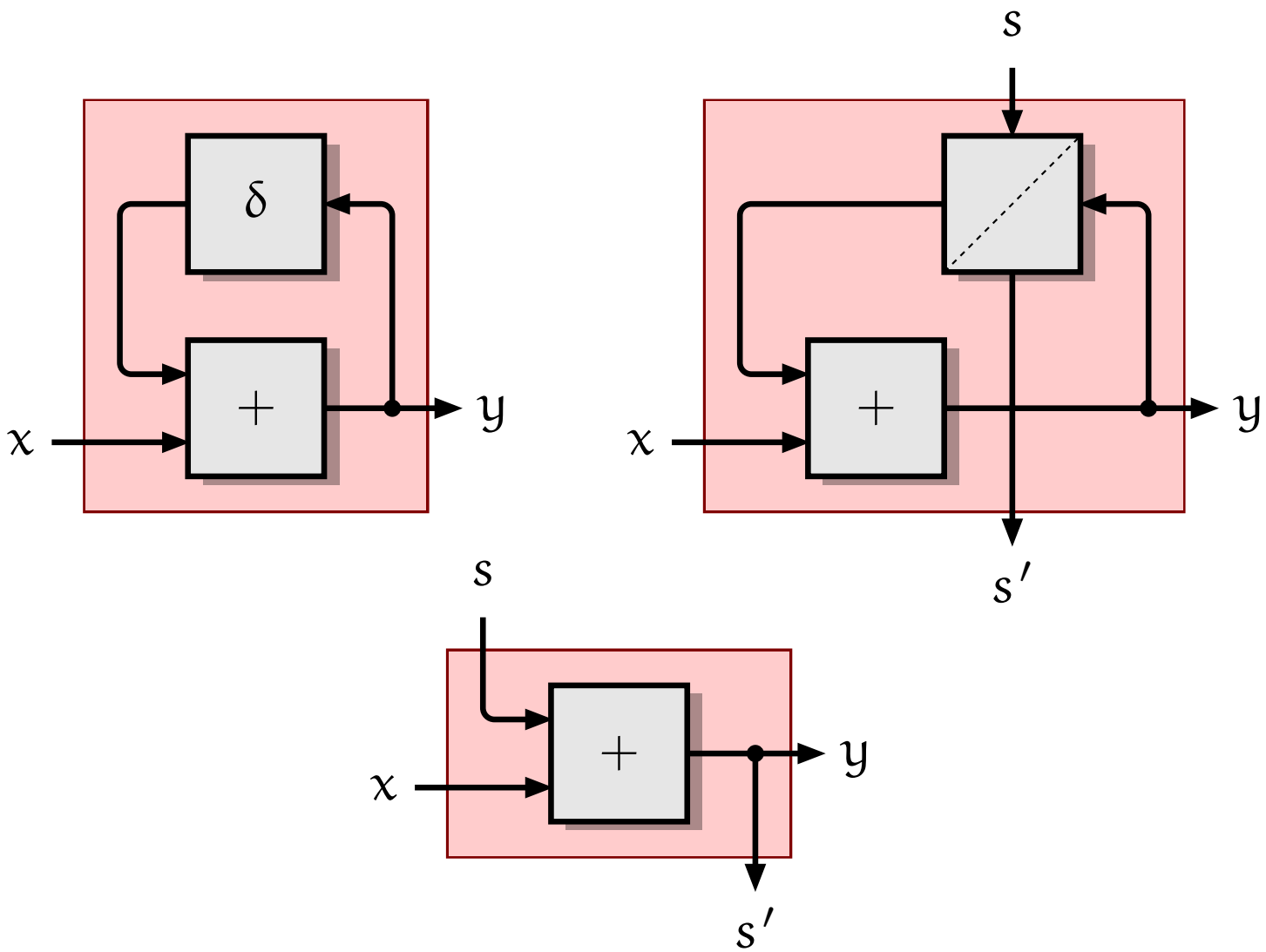
$$\delta \subseteq A \times A \times A \times A$$

$$\delta : A \times A \to M(A \times A)$$

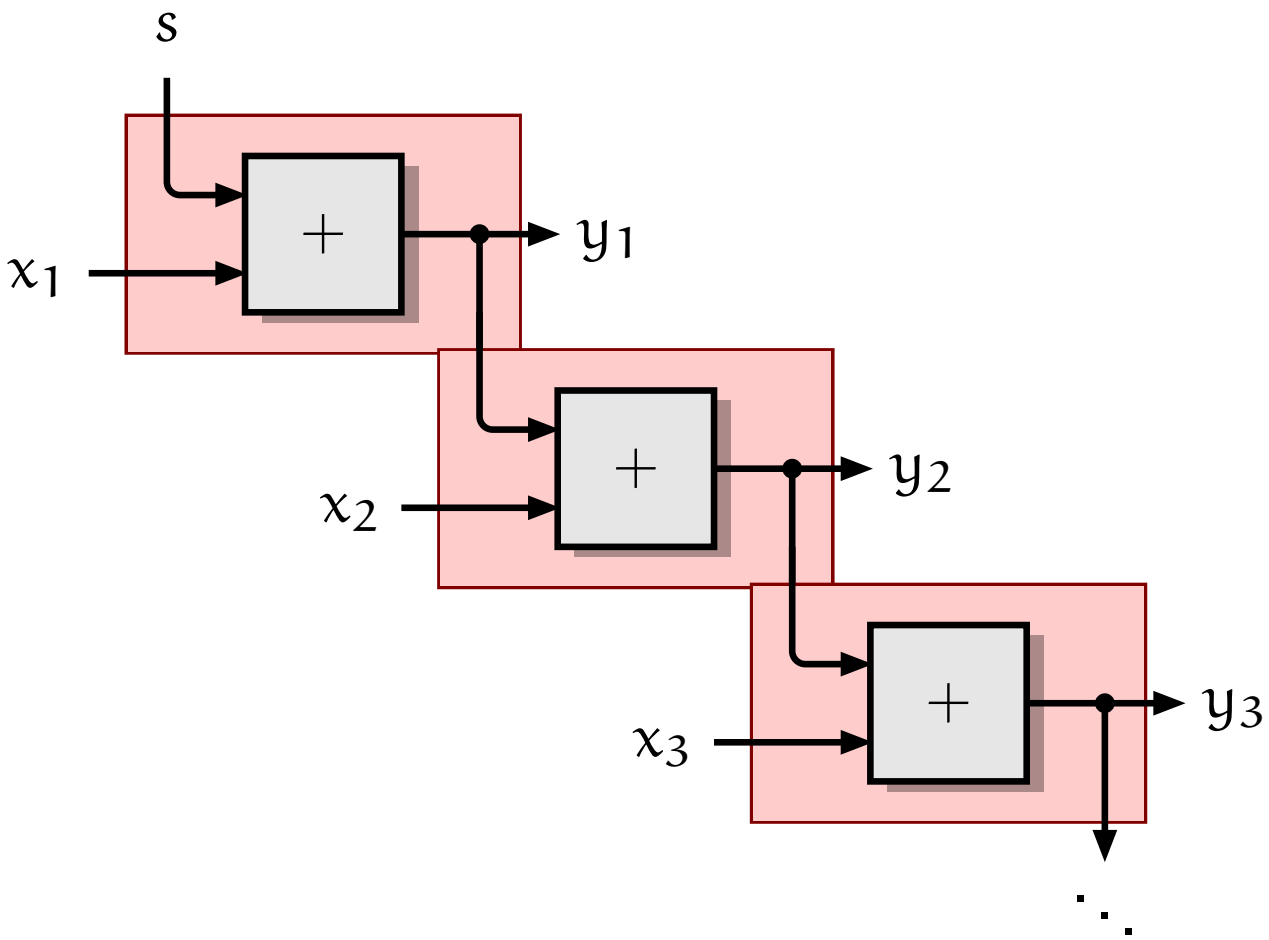
$$s \,/\, a \overset{\delta}{\longmapsto} a' \,/\, s'$$

$$s = a' \wedge a = s'$$

State Inference & Delay Elimination



State Inference & Delay Elimination



Simple Coinduction

$$M = \mathrm{Id}$$

$$\mathrm{curry}\, R : S \times A \to (B \times S)^A$$

$$[\![\mathrm{curry}\, R]\!]_{\mathcal{T}_{AB}} : S \to (A^\omega \xrightarrow{!} B^\omega)$$

- Initial state from delay node
  - Annotation (`fby`)
  - Type-based default

Beyond Stream Functions: Stream Relations

Features Beyond Basic Data Flow

Control by pattern matching
sometimes no value (locally)

Specification loosely with nondeterministic choice
sometimes more than one value

Monadic Coinduction (1)
(Pardo 1998)

1. Lift final coalgebra diagram to Kleisli category

$$\begin{array}{ccc} S & \xrightarrow{\quad f \quad} & M(A^\omega \xrightarrow{!} B^\omega) \\ {\scriptstyle\varphi}\downarrow & & \downarrow{\scriptstyle M\,\mathrm{out}_{\mathcal{T}_{AB}}} \\ M\mathcal{T}_{AB}S & \xrightarrow[(\widehat{\mathcal{T}}_{AB} f)^\star]{} & M\mathcal{T}_{AB}(A^\omega \xrightarrow{!} B^\omega) \end{array}$$

   - Kleisli extension *; distributive law; functor lifting $\widehat{\mathcal{T}}_{AB}$
   - Solutions ***not*** guaranteed/unique

2. Tarski to the rescue

$$f = M\,\mathsf{out}^{-1} \circ \mu \circ M\widehat{\mathcal{T}}_{AB} f \circ \varphi$$

   - $M = \mathcal{P} \implies$ monotonic on complete lattice
   - Greatest fixed point

Monadic Coinduction (2)

$$M = \mathcal{P} \quad \text{etc.}$$

$$\lambda_{B\times S} \circ \mathsf{curry}\, R : S \times A \to \left(M\left((B \times S)\right)^A\right)$$

$$\lfloor \lambda_{B\times S} \circ \mathsf{curry}\, R \rfloor^M_{\mathcal{T}_{AB}} : S \to M(A^\omega \xrightarrow{!} B^\omega)$$

Coinduction?

$$\lfloor \lambda_{B\times S} \circ \mathsf{curry}\, R \rfloor^M_{\mathcal{T}_{AB}}(s) = \ldots$$

Agenda

Conclusion

- Semantics for causal (online) stream functions
- Off-the-shelf semantic components
  - Z-style relational stateful computation
  - coalgebraic streams
  - monadic coinduction
- Coinduction essential to comprehensible semantics
- Deterministic programs: implementation straightforward ($\leqslant 2$ kLoC; $\leqslant 1$ person-week)