Stream Functions from a Coalgebraic Perspective

Helle Hvid Hansen

TU/e and CWI

Streams Seminar, RU Nijmegen, 18 January 2010

From Streams to Stream Functions

In previous lecture:

- Systems with output: $X \longrightarrow O \times X$
- System behaviour: stream.
- Stream definition principle: coinduction.

In this lecture:

- Systems with input and output: $X \longrightarrow (B \times X)^A$
- System behaviour: stream function.
- Algebraic specification and coalgebraic synthesis of certain bitstream functions.

Overview

- Mealy machines.
- Bitstream arithmetic.
- Specifying bitstream functions in infinitary binary arithmetic.
- Synthesis of rational bitstream functions.
- Conclusion and related work.

Mealy Machines

- A Mealy machine with input in A and output in B:

$$m\colon X \to (B \times X)^A$$

- Notation: $m(x)(a) = \langle b, y \rangle$ iff $x \xrightarrow{a|b} y$

Mealy Machines

- A Mealy machine with input in A and output in B:

$$m\colon X \to (B \times X)^A$$

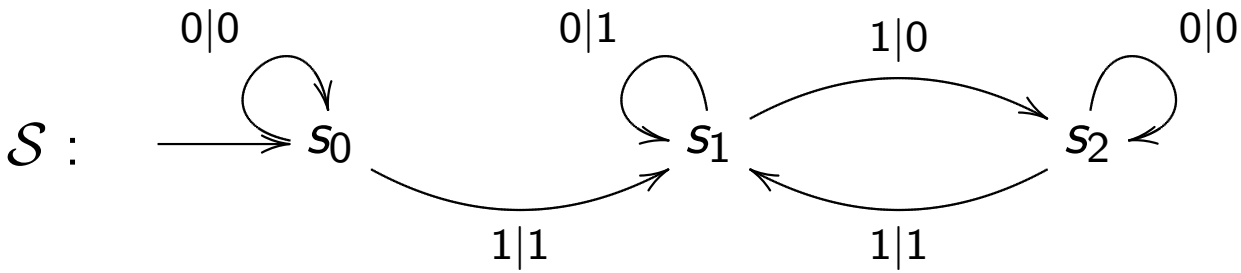
- Notation: $m(x)(a) = \langle b, y \rangle$ iff $x \xrightarrow{a|b} y$
- A map $f : X \to Y$ is a Mealy homomorphism if:

$$\begin{array}{ccc} X & \xrightarrow{\quad f \quad} & Y \\ {\scriptstyle m}\downarrow & & \downarrow{\scriptstyle n} \\ (B \times X)^A & \xrightarrow{(B \times f)^A} & (B \times Y)^A \end{array}$$

i.e. $\quad x \xrightarrow{a|b} y \quad \Rightarrow \quad f(x) \xrightarrow{a|b} f(y) \quad$ for all $x \in X, a \in A$.

Example: Single bit binary counter

(Input and output $A = B = 2$)



- after even number of 1's, output 0.
- after odd number of 1's, output 1.

Stream Function Behaviour

- The (input-output) behaviour of a state x_0:
  $beh(x_0)\colon (a_0, a_1, a_2, \dots) \mapsto (b_0, b_1, b_2, \dots)$ where

$$x_0 \xrightarrow{a_0|b_0} x_1 \xrightarrow{a_1|b_1} \cdots \xrightarrow{a_k|b_k} x_{k+1} \xrightarrow{a_{k+1}|b_{k+1}} \cdots$$

  I.e. $beh(x_0)\colon A^\omega \longrightarrow B^\omega$ is stream function.

Stream Function Behaviour

- The (input-output) behaviour of a state x_0:
  $beh(x_0)\colon (a_0, a_1, a_2, \dots) \mapsto (b_0, b_1, b_2, \dots)$ where

$$x_0 \xrightarrow{a_0|b_0} x_1 \xrightarrow{a_1|b_1} \cdots \xrightarrow{a_k|b_k} x_{k+1} \xrightarrow{a_{k+1}|b_{k+1}} \cdots$$

  I.e. $beh(x_0)\colon A^\omega \to B^\omega$ is stream function.

- Single bit binary counter: $beh(s_0) = f$ where:

$$f(\alpha)(k) = \begin{cases} 0 & \text{if } \alpha(0), \dots, \alpha(k) \text{ contains even number of 1's} \\ 1 & \text{otherwise} \end{cases}$$

- Note: k-th output is determined by first k inputs.
  In general: $beh(x)$ is causal.

Causal Stream Functions

- Def. $f : A^\omega \to B^\omega$ is causal for all $\alpha, \sigma \in A^\omega$,

$$(\forall i \leq k : \alpha(i) = \sigma(i)) \quad \Rightarrow \quad f(\alpha)(k) = f(\sigma)(k).$$

- Let $\Gamma = \{ f \mid f : A^\omega \to B^\omega \text{ and } f \text{ is causal } \}$.

Causal Stream Functions

- Def. $f : A^\omega \to B^\omega$ is causal for all $\alpha, \sigma \in A^\omega$,

$$(\forall i \leq k : \alpha(i) = \sigma(i)) \quad \Rightarrow \quad f(\alpha)(k) = f(\sigma)(k).$$

- Let $\Gamma = \{ f \mid f : A^\omega \to B^\omega \text{ and } f \text{ is causal } \}$.
- Next: we define Mealy structure $\gamma : \Gamma \to (B \times \Gamma)^A$.
- I.e., for $f \in \Gamma$ and input $a \in 2$ define output $f[a] \in 2$ and next state $f_a \in \Gamma$.

$$f \xrightarrow{a|f[a]} f_a$$

Mealy Machine of Causal Stream Functions

Given $f \in \Gamma$ and input $a \in A$,

Mealy Machine of Causal Stream Functions

Given $f \in \Gamma$ and input $a \in A$,

$$A^\omega \xrightarrow{f(a:-)} B^\omega$$

$$\sigma \longmapsto f(a:\sigma)$$

Mealy Machine of Causal Stream Functions

Given $f \in \Gamma$ and input $a \in A$,

$$A^\omega \xrightarrow{f(a:-)} B^\omega \xrightarrow{\langle hd, tl \rangle} B \times B^\omega$$

$$\sigma \longmapsto f(a:\sigma) \longmapsto \langle hd(f(a:\sigma)), tl(f(a:\sigma)) \rangle$$

Mealy Machine of Causal Stream Functions

Given $f \in \Gamma$ and input $a \in A$,

$$A^\omega \xrightarrow{f(a:-)} B^\omega \xrightarrow{\langle hd, tl \rangle} B \times B^\omega$$

$$\sigma \longmapsto f(a:\sigma) \longmapsto \langle hd(f(a:\sigma)), tl(f(a:\sigma)) \rangle$$

$$\begin{array}{lcl} hd \circ (f(a:-)) & : & A^\omega \to B \\ tl \circ (f(a:-)) & : & A^\omega \to B^\omega \end{array}$$

Mealy Machine of Causal Stream Functions

Given $f \in \Gamma$ and input $a \in A$,

$$A^\omega \xrightarrow{f(a:-)} B^\omega \xrightarrow{\langle hd, tl \rangle} B \times B^\omega$$

$$\sigma \longmapsto f(a:\sigma) \longmapsto \langle hd(f(a:\sigma)), tl(f(a:\sigma)) \rangle$$

			since f causal:
$hd \circ (f(a:-))$	:	$A^\omega \to B$	constant, hence in B
$tl \circ (f(a:-))$	:	$A^\omega \to B^\omega$	causal, hence in Γ

Mealy Machine of Causal Stream Functions

Given $f \in \Gamma$ and input $a \in A$,

$$A^\omega \xrightarrow{f(a:-)} B^\omega \xrightarrow{\langle hd, tl \rangle} B \times B^\omega$$

$$\sigma \longmapsto f(a:\sigma) \longmapsto \langle hd(f(a:\sigma)), tl(f(a:\sigma)) \rangle$$

			since f causal:
$hd \circ (f(a:-))$	:	$A^\omega \to B$	constant, hence in B
$tl \circ (f(a:-))$	:	$A^\omega \to B^\omega$	causal, hence in Γ

Define:

$f[a]$	:=	$hd \circ (f(a:-))$	$\in$	B	(initial output)
f_a	:=	$tl \circ (f(a:-))$	$\in$	Γ	(stream function derivative)

Final Mealy Machine

Theorem: The Mealy machine of causal functions $\langle \Gamma, \gamma \rangle$ is a final Mealy machine with the behaviour map as the final map.

That is:

For all Mealy machines $\langle X, m \rangle$, the behavior map $beh \colon X \to \Gamma$ is the unique Mealy homomorphism $beh \colon \langle X, m \rangle \to \langle \Gamma, \gamma \rangle$:

$$\begin{array}{ccc} X & \xrightarrow{!beh} & \Gamma \\ {\scriptstyle \forall m}\downarrow & & \downarrow{\scriptstyle \gamma} \\ (B \times X)^A & \xrightarrow{(B \times beh)^A} & (B \times \Gamma)^A \end{array}$$

Minimal Realisations

Let $\langle X, m \rangle$ be a Mealy machine, and $x \in X$:

- Def. The submachine generated by x in $\langle X, m \rangle$ is the least set $S \subseteq X$ such that $x \in S$ and S is transition closed.
- Def. $\langle x \rangle$ is realisation of $f : A^\omega \to B^\omega$ if $beh(x) = f$.
- Def. $\langle X, m \rangle$ is minimal, if $beh : \langle X, m \rangle \to \langle \Gamma, \gamma \rangle$ is injective.

Minimal Realisations

Let $\langle X, m\rangle$ be a Mealy machine, and $x \in X$:

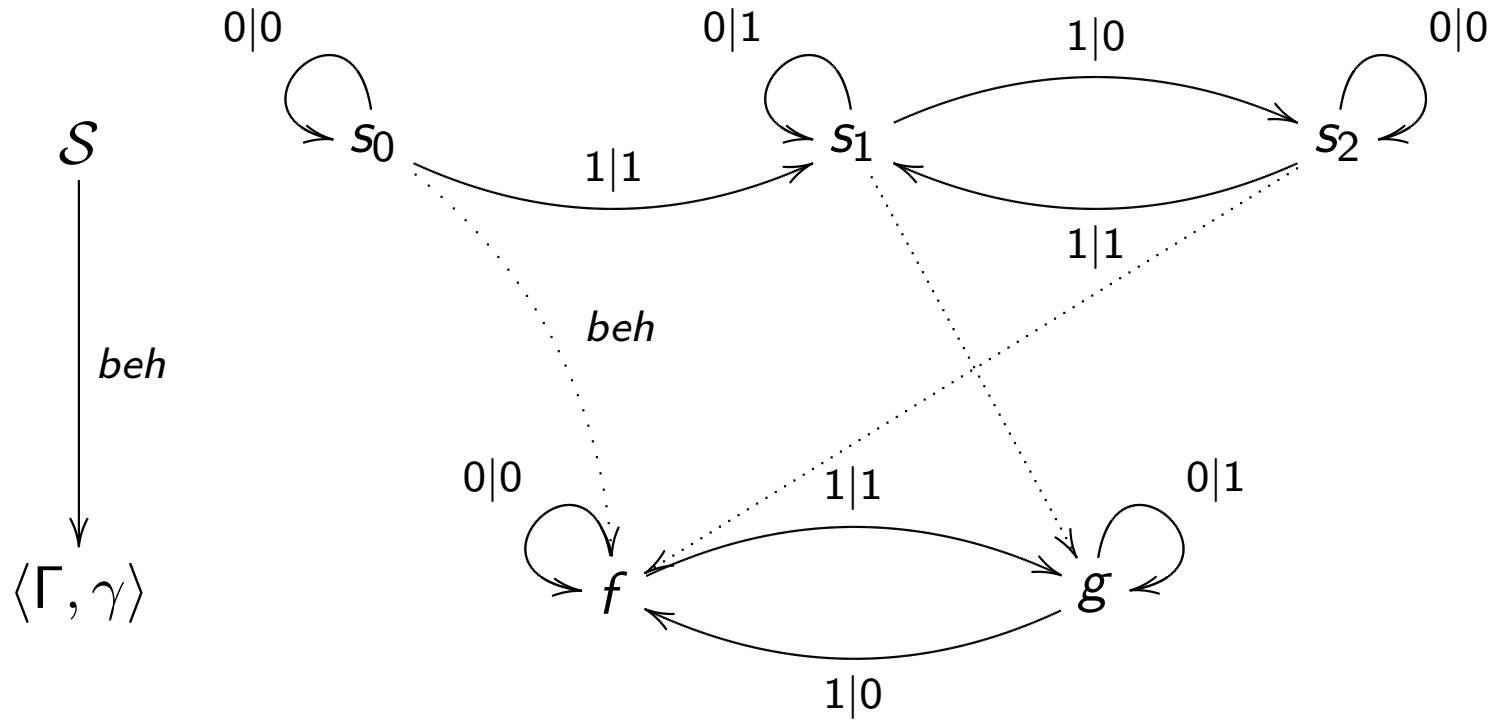
- Def. The submachine generated by x in $\langle X, m\rangle$ is the least set $S \subseteq X$ such that $x \in S$ and S is transition closed.
- Def. $\langle x\rangle$ is realisation of $f : A^\omega \to B^\omega$ if $beh(x) = f$.
- Def. $\langle X, m\rangle$ is minimal, if $beh : \langle X, m\rangle \to \langle \Gamma, \gamma\rangle$ is injective.

By finality:

- $beh(f) = f$ for all $f \in \Gamma$, hence
- For all $f \in \Gamma$, $\langle f\rangle$ is a minimal realisation of f.

Example: Single bit binary counter

$(\langle s_0 \rangle = \mathcal{S})$



- I.e. $f = beh(s_0) = beh(s_2)$ and $g = beh(s_1)$).
- beh maps $\langle s_0 \rangle$ to its minimization $\langle beh(s_0) \rangle \subseteq \Gamma$.

Mealy Machines, Digital Circuits and Bitstreams

Mealy Machines, Digital Circuits and Bitstreams

Specifying digital circuits

- Mealy machines invented as models of sequential circuits (G.H. Mealy, 1955), (state = register contents).
- Mealy machine specifies circuit behaviour.
- Circuit synthesis: from Mealy machine to sequential circuit.

Mealy Machines, Digital Circuits and Bitstreams

Specifying digital circuits

- Mealy machines invented as models of sequential circuits (G.H. Mealy, 1955), (state = register contents).
- Mealy machine specifies circuit behaviour.
- Circuit synthesis: from Mealy machine to sequential circuit.

Specifying binary Mealy machines $m\colon X \longrightarrow (2 \times X)^2$

- Binary Mealy machines realise bitstream functions.
- Bitstream functions specified in infinitary binary arithmetic.
- Mealy synthesis:
  from arithmetic function expression to Mealy machine.

Infinitary Binary Expansions

- Bitstreams are 2-adic numbers (infinitary base-2 expansion): $(a_0, a_1, a_2, a_3, \ldots) \in 2^\omega$ represents

$$\Sigma_{i<\omega} a_i 2^i = a_0 + a_1 2 + a_2 2^2 + a_3 2^3 + \ldots$$

Infinitary Binary Expansions

- Bitstreams are 2-adic numbers (infinitary base-2 expansion): $(a_0, a_1, a_2, a_3, \ldots) \in 2^\omega$ represents

$$\Sigma_{i<\omega} a_i 2^i = a_0 + a_1 2 + a_2 2^2 + a_3 2^3 + \ldots$$

- Examples:

$$\begin{aligned} B(2) &= (0,1,0,0,0,0,\ldots), \\ B(5) &= (1,0,1,0,0,0,\ldots), \end{aligned}$$

Infinitary Binary Expansions

- Bitstreams are 2-adic numbers (infinitary base-2 expansion): $(a_0, a_1, a_2, a_3, \ldots) \in 2^\omega$ represents

$$\Sigma_{i<\omega} a_i 2^i = a_0 + a_1 2 + a_2 2^2 + a_3 2^3 + \ldots$$

- Examples:

$$\begin{aligned} B(2) &= (0,1,0,0,0,0,\ldots), \\ B(5) &= (1,0,1,0,0,0,\ldots), \\ B(-5) &= (1,1,0,1,1,1,\ldots), \end{aligned}$$

Infinitary Binary Expansions

- Bitstreams are 2-adic numbers (infinitary base-2 expansion): $(a_0, a_1, a_2, a_3, \ldots) \in 2^\omega$ represents

$$\Sigma_{i<\omega} a_i 2^i = a_0 + a_1 2 + a_2 2^2 + a_3 2^3 + \ldots$$

- Examples:

$$\begin{array}{lcl} B(2) & = & (0,1,0,0,0,0,\ldots), \\ B(5) & = & (1,0,1,0,0,0,\ldots), \\ B(-5) & = & (1,1,0,1,1,1,\ldots), \\ B(1/5) & = & (1,0,1,1,0,0,1,1,0,0,1,1,0,0,\ldots). \end{array}$$

Bitstream Algebra $\mathcal{A}$

- Notation: $[0] = B(0) = (0, 0, 0, 0, \ldots)$
  $[1] = B(1) = (1, 0, 0, 0, \ldots)$
- Def. The bitstream algebra $\mathcal{A} = \langle 2^\omega, +, -, \times, 1/-, [0], [1] \rangle$ consists of bitstreams with 2-adic operations (details coming).

Bitstream Algebra $\mathcal{A}$

- Notation: $[0] = B(0) = (0, 0, 0, 0, \ldots)$
  $[1] = B(1) = (1, 0, 0, 0, \ldots)$
- Def. The bitstream algebra $\mathcal{A} = \langle 2^\omega, +, -, \times, 1/-, [0], [1] \rangle$ consists of bitstreams with 2-adic operations (details coming).
- $\mathcal{A}$ is integral domain, i.e., commutative ring, no zero-divisors: $x \times y = 0 \;\Rightarrow\; x = 0$ or $y = 0$.
- Binary expansion $B \colon \mathbb{Q}_{odd} \to \mathcal{A}$ is homomorphism of integral domains, where $\mathbb{Q}_{odd} = \{n/(2m+1) \mid n, m \in \mathbb{Z}\}$.

The Bitstream X

- Def. $X = B(2) = (0, 1, 0, 0, 0, \ldots)$.
- X plays role of formal power series variable.

The Bitstream X

- Def. $X = B(2) = (0, 1, 0, 0, 0, \ldots)$.
- X plays role of formal power series variable.
- Multiplying with $X = B(2)$ is shift-right:

$$\begin{aligned} 2 \times (a_0 + a_1 2 + a_2 2^2 + \ldots) &= 0 + a_0 2 + a_1 2^2 + a_2 2^3 + \ldots) \\ X \times (a_0, a_1, a_2, \ldots) &= (0, a_0, a_1, a_2, \ldots) \end{aligned}$$

The Bitstream X

- Def. $X = B(2) = (0, 1, 0, 0, 0, \ldots)$.
- X plays role of formal power series variable.
- Multiplying with $X = B(2)$ is shift-right:

$$\begin{aligned} 2 \times (a_0 + a_1 2 + a_2 2^2 + \ldots) &= 0 + a_0 2 + a_1 2^2 + a_2 2^3 + \ldots) \\ X \times (a_0, a_1, a_2, \ldots) &= (0, a_0, a_1, a_2, \ldots) \end{aligned}$$

- For all $\alpha = (a_0, a_1, a_2, \ldots)$, $\alpha = [\alpha(0)] + (X \times \alpha')$:

$$\begin{array}{rrcl} & [\alpha(0)] & = & (a_0, 0, 0, 0, \ldots) \\ + & X \times \alpha' & = & (0, a_1, a_2, a_3, \ldots) \\ \hline & \alpha & = & (a_0, a_1, a_2, a_3, \ldots) \end{array}$$

Some 2-Adic Bitstream Identities

Lemma: For all $\alpha = (a_0, a_1, a_2, a_3, \ldots) \in 2^\omega$:

$$\begin{aligned} X \times \alpha &= 0 : \alpha = (0, a_0, a_1, a_2, \ldots) \\ [1] + (X \times \alpha) &= 1 : \alpha = (1, a_0, a_1, a_2, \ldots) \\ \alpha &= [\alpha(0)] + (X \times \alpha') \\ \alpha + \alpha &= X \times \alpha \end{aligned}$$

2-Adic Bitstream Operations

Stream differential equations

- Coinductive and inductive definition of 2-adic operations.
- Inductive: e.g. define $\alpha + \beta$ in terms of α and β.
- Coinductive: e.g. define $\alpha + \beta$ by defining head and tail $(\alpha + \beta)(0)$ and $(\alpha + \beta)'$.
- Useful later when defining Mealy structure on expressions.

2-Adic Bitstream Operations

Stream differential equations

- Coinductive and inductive definition of 2-adic operations.
- Inductive: e.g. define $\alpha + \beta$ in terms of α and β.
- Coinductive: e.g. define $\alpha + \beta$ by defining head and tail $(\alpha + \beta)(0)$ and $(\alpha + \beta)'$.
- Useful later when defining Mealy structure on expressions.

Boolean operations:

For $a, b \in 2$

$$\begin{array}{lcll} a \oplus b & = & (a + b) \mod 2 & \text{(Boolean XOR)} \\ a \wedge b & = & \min\{a, b\} & \text{(Boolean AND)} \end{array}$$

2-Adic Bitstream Operations

2-adic sum (carry-bits propagate indefinitely):

- Example:

$$\begin{array}{rl} & 1\quad 1\quad 1\cdots \\ & (1,0,0,0,\ldots) \\ + & (1,1,1,1,\ldots) \\ \hline & (0,0,0,0,\ldots) \end{array}$$

- Stream differential equation:

$$\begin{aligned} (\alpha+\beta)(0) &= \alpha(0)\oplus\beta(0) \\ (\alpha+\beta)' &= (\alpha'+\beta') + \underbrace{[\alpha(0)\wedge\beta(0)]}_{\text{carry}} \end{aligned}$$

2-Adic Bitstream Operations

2-adic product:

- Using $\alpha = [\alpha(0)] + (X \times \alpha')$ we have:

$$\begin{aligned}(\alpha \times \beta) &= ([\alpha(0)] + (X \times \alpha')) \times \beta \\ &= ([\alpha(0)] \times \beta) + ((X \times \alpha') \times \beta)\end{aligned}$$

2-Adic Bitstream Operations

2-adic product:

- Using $\alpha = [\alpha(0)] + (X \times \alpha')$ we have:

$$\begin{aligned}(\alpha \times \beta) &= ([\alpha(0)] + (X \times \alpha')) \times \beta \\ &= ([\alpha(0)] \times \beta) + ((X \times \alpha') \times \beta)\end{aligned}$$

- Using stream.diff.eq. for $+$ together with $([a] \times \beta)' = [a] \times \beta'$ and $(X \times \beta)' = \beta$ ...

2-Adic Bitstream Operations

2-adic product:

- Using $\alpha = [\alpha(0)] + (X \times \alpha')$ we have:

$$\begin{aligned}(\alpha \times \beta) &= ([\alpha(0)] + (X \times \alpha')) \times \beta \\ &= ([\alpha(0)] \times \beta) + ((X \times \alpha') \times \beta)\end{aligned}$$

- Using stream.diff.eq. for $+$ together with $([a] \times \beta)' = [a] \times \beta'$ and $(X \times \beta)' = \beta$ ...
- Stream differential equation:

$$\begin{aligned}(\alpha \times \beta)(0) &= \alpha(0) \wedge \beta(0) \\ (\alpha \times \beta)' &= (\alpha' \times \beta) + ([\alpha(0)] \times \beta')\end{aligned}$$

2-Adic Bitstream Operations

2-adic minus:

- Infinitary two's complement, e.g.

$$\begin{aligned} (1,0,0,0,\ldots) + (1,1,1,1,\ldots) &= [0] \\ \Rightarrow \quad -[1] = -(1,0,0,\ldots) &= (1,1,1,\ldots) \end{aligned}$$

2-Adic Bitstream Operations

2-adic minus:

- Infinitary two's complement, e.g.

$$\begin{aligned} (1,0,0,0,\ldots) + (1,1,1,1,\ldots) &= [0] \\ \Rightarrow \qquad -[1] = -(1,0,0,\ldots) &= (1,1,1,\ldots) \end{aligned}$$

- Additive inverse: $(-\alpha) + \alpha = [0]$, hence

$$\begin{aligned} (-\alpha)(0) \oplus \alpha(0) &= 0, \text{ and} \\ ((-\alpha)' + \alpha') + [(-\alpha)(0) \wedge \alpha(0)] &= [0] \end{aligned}$$

2-Adic Bitstream Operations

2-adic minus:

- Infinitary two's complement, e.g.

$$\begin{aligned} (1,0,0,0,\ldots) + (1,1,1,1,\ldots) &= [0] \\ \Rightarrow \quad -[1] = -(1,0,0,\ldots) &= (1,1,1,\ldots) \end{aligned}$$

- Additive inverse: $(-\alpha) + \alpha = [0]$, hence

$$\begin{aligned} (-\alpha)(0) \oplus \alpha(0) &= 0, \text{ and} \\ ((-\alpha)' + \alpha') + [(-\alpha)(0) \wedge \alpha(0)] &= [0] \end{aligned}$$

- Stream differential equation:

$$\begin{aligned} (-\alpha)(0) &= \alpha(0) \\ (-\alpha)' &= -(\alpha' + [\alpha(0)]) \end{aligned}$$

2-Adic Bitstream Operations

2-adic inverse:

- Multiplicative inverse: $1/\beta$ satisfies:

$$\beta \times (1/\beta) = [1]$$

- Consequently, $1/\beta$ only defined if $\beta(0) = 1$.

2-Adic Bitstream Operations

2-adic inverse:

- Multiplicative inverse: $1/\beta$ satisfies:

$$\beta \times (1/\beta) = [1]$$

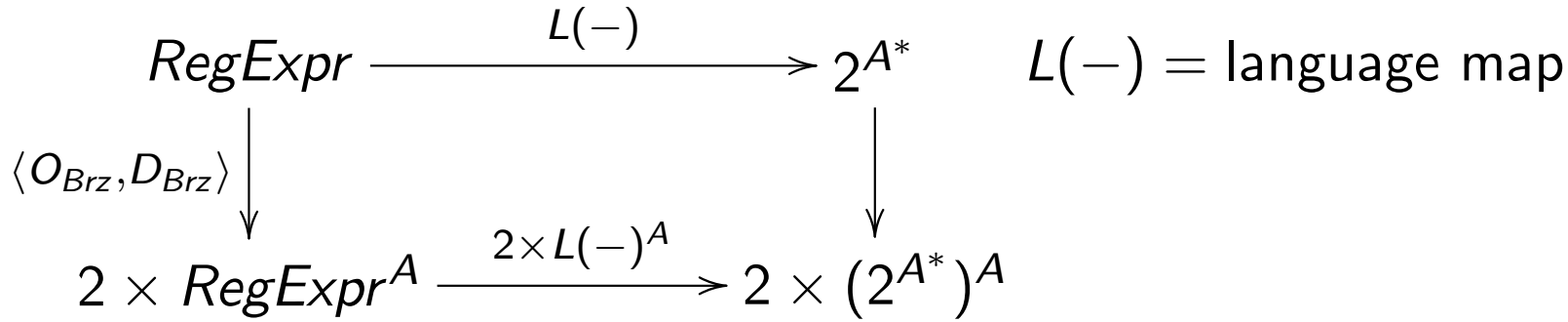
- Consequently, $1/\beta$ only defined if $\beta(0) = 1$.
- Stream differential equation (condition $\beta(0) = 1$):

$$\begin{array}{lcl} (1/\beta)(0) & = & 1 \\ (1/\beta)' & = & -(\beta' \times (1/\beta)) \end{array}$$

- Notation: $\alpha/\beta \;:=\; \frac{\alpha}{\beta} \;:=\; \alpha \times (1/\beta)$

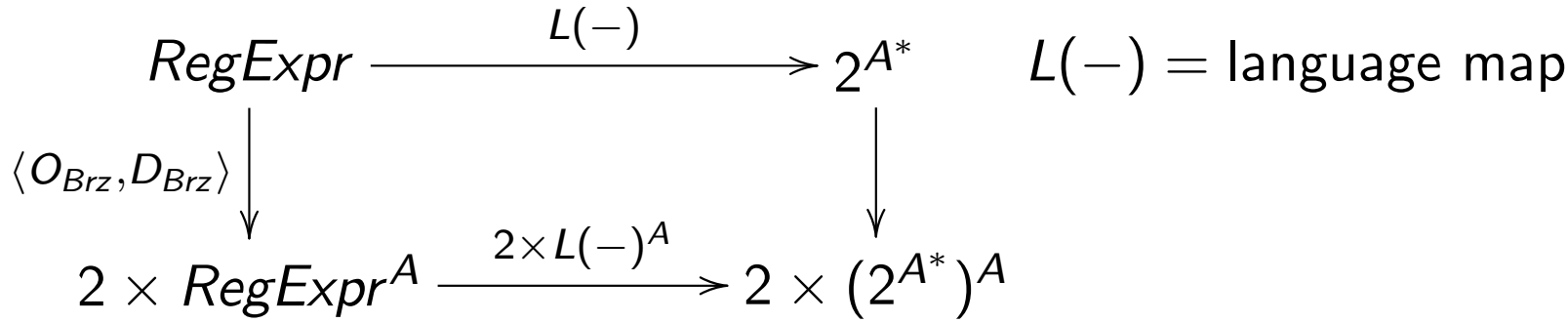
Specifying 2-Adic Bitstream Functions

- Recall Brzozowski, regular expressions, formal languages:

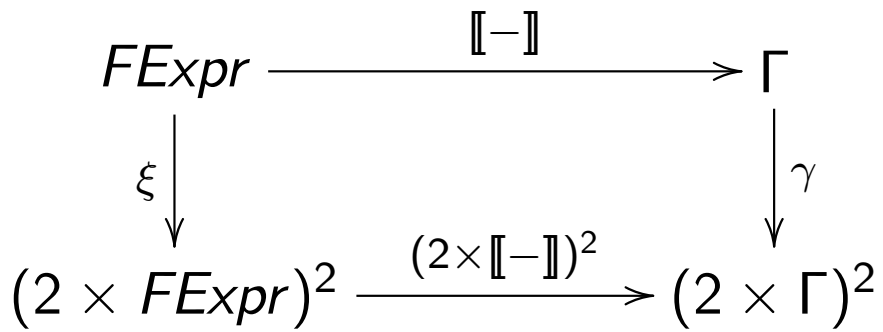


Specifying 2-Adic Bitstream Functions

- Recall Brzozowski, regular expressions, formal languages:



- We define function expressions *FExpr* with
  - algebraic semantics $[\![-]\!]$ (given by $\mathcal{A}$), and
  - coalgebraic semantics ξ (Brz. analogue) such that:



Function Expressions

We specify functions $f : 2^\omega \longrightarrow 2^\omega$ defined in language of $\mathcal{A}$.

- The set *FExpr* of function expressions is generated by grammar:

$$\mathbf{E}, \mathbf{F} ::= \mathbf{0} \mid \mathbf{1} \mid \mathbf{X} \mid \mathbf{s} \mid \mathbf{E} + \mathbf{F} \mid -\mathbf{E} \mid \mathbf{E} \times \mathbf{F} \mid 1/(\mathbf{1} + (\mathbf{X} \times \mathbf{E})$$

  where $\mathbf{s}$ is a bitstream variable.

Function Expressions

We specify functions $f : 2^\omega \to 2^\omega$ defined in language of $\mathcal{A}$.

- The set *FExpr* of function expressions is generated by grammar:

$$\mathbf{E}, \mathbf{F} ::= \mathbf{0} \mid \mathbf{1} \mid \mathbf{X} \mid \mathbf{s} \mid \mathbf{E} + \mathbf{F} \mid -\mathbf{E} \mid \mathbf{E} \times \mathbf{F} \mid 1/(\mathbf{1} + (\mathbf{X} \times \mathbf{E})$$

  where $\mathbf{s}$ is a bitstream variable.

- Example:

$$\mathbf{E} = \frac{-\mathbf{X} + ((\mathbf{1} + \mathbf{X}^\mathbf{2}) \times \mathbf{s})}{\mathbf{1} + (\mathbf{X} \times (\mathbf{X} + \mathbf{X}))}$$

Algebraic Semantics of Function Expressions

- Algebraic semantics $[\![\mathbf{E}]\!]: 2^\omega \longrightarrow 2^\omega$ by interpretation in $\mathcal{A}$:

Algebraic Semantics of Function Expressions

- Algebraic semantics $[\![\mathbf{E}]\!]\colon 2^\omega \longrightarrow 2^\omega$ by interpretation in $\mathcal{A}$:

$$\begin{array}{lcl} [\![\mathbf{0}]\!](\sigma) & = & [0] \\ [\![\mathbf{1}]\!](\sigma) & = & [1] \\ [\![\mathbf{X}]\!](\sigma) & = & X \\ [\![\mathbf{s}]\!](\sigma) & = & \sigma \end{array}$$

Algebraic Semantics of Function Expressions

- Algebraic semantics $[\![\mathbf{E}]\!]\colon 2^\omega \longrightarrow 2^\omega$ by interpretation in $\mathcal{A}$:

$[\![\mathbf{0}]\!](\sigma)$	$=$	$[0]$	$[\![\mathbf{E}+\mathbf{F}]\!](\sigma)$	$=$	$[\![\mathbf{E}]\!](\sigma)+[\![\mathbf{F}]\!](\sigma)$
$[\![\mathbf{1}]\!](\sigma)$	$=$	$[1]$			
$[\![\mathbf{X}]\!](\sigma)$	$=$	X			
$[\![\mathbf{s}]\!](\sigma)$	$=$	σ			

Algebraic Semantics of Function Expressions

- Algebraic semantics $[\![\mathbf{E}]\!]\colon 2^\omega \to 2^\omega$ by interpretation in $\mathcal{A}$:

$[\![\mathbf{0}]\!](\sigma)$	$=$	$[0]$	$[\![\mathbf{E}+\mathbf{F}]\!](\sigma)$	$=$	$[\![\mathbf{E}]\!](\sigma)+[\![\mathbf{F}]\!](\sigma)$
$[\![\mathbf{1}]\!](\sigma)$	$=$	$[1]$	$[\![-\mathbf{E}]\!](\sigma)$	$=$	$-[\![\mathbf{E}]\!](\sigma)$
$[\![\mathbf{X}]\!](\sigma)$	$=$	X	$[\![\mathbf{E}\times\mathbf{F}]\!](\sigma)$	$=$	$[\![\mathbf{E}]\!](\sigma)\times[\![\mathbf{F}]\!](\sigma)$
$[\![\mathbf{s}]\!](\sigma)$	$=$	σ	$[\![1/(\mathbf{1}+\mathbf{X}\times\mathbf{E})]\!]$	$=$	$1/[1]+(X\times[\![\mathbf{E}]\!])$

- Note: inverse is always defined

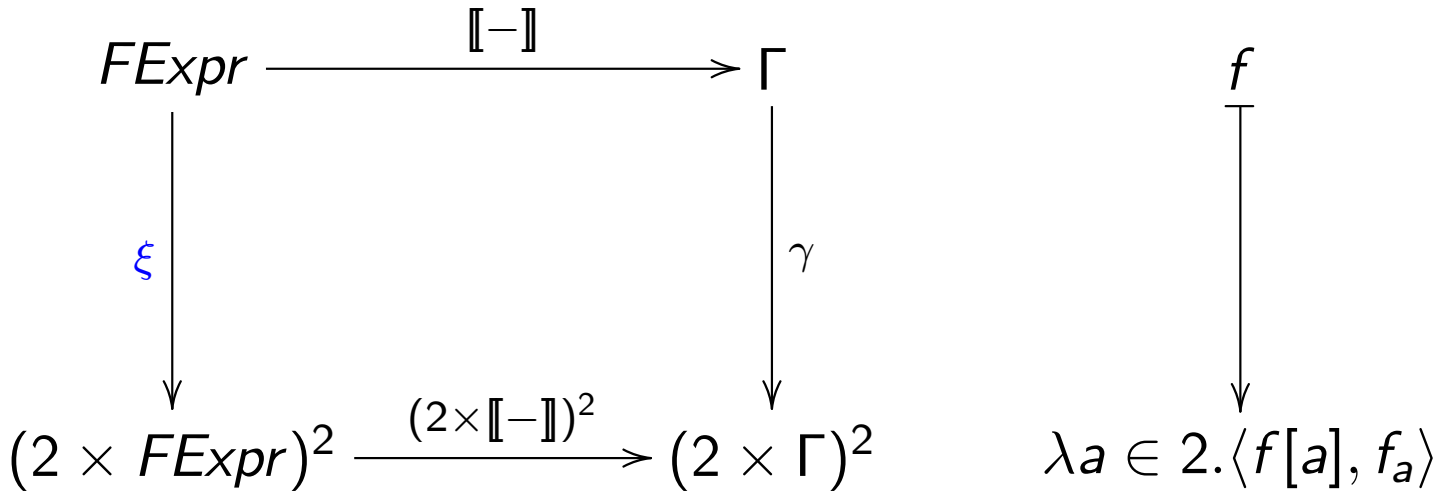
$$\begin{aligned}[\![1/(\mathbf{1}+\mathbf{X}\times\mathbf{E})]\!](\sigma) &= 1/[1]+(X\times[\![\mathbf{E}]\!](\sigma)) \\ &= 1/(1:[\![\mathbf{E}]\!](\sigma))\end{aligned}$$

Mealy Machine of Function Expressions

Want to define Mealy structure $\xi \colon \mathit{FExpr} \to (2 \times \mathit{FExpr})^2$ such that $[\![-]\!]$ is Mealy homomorphism.

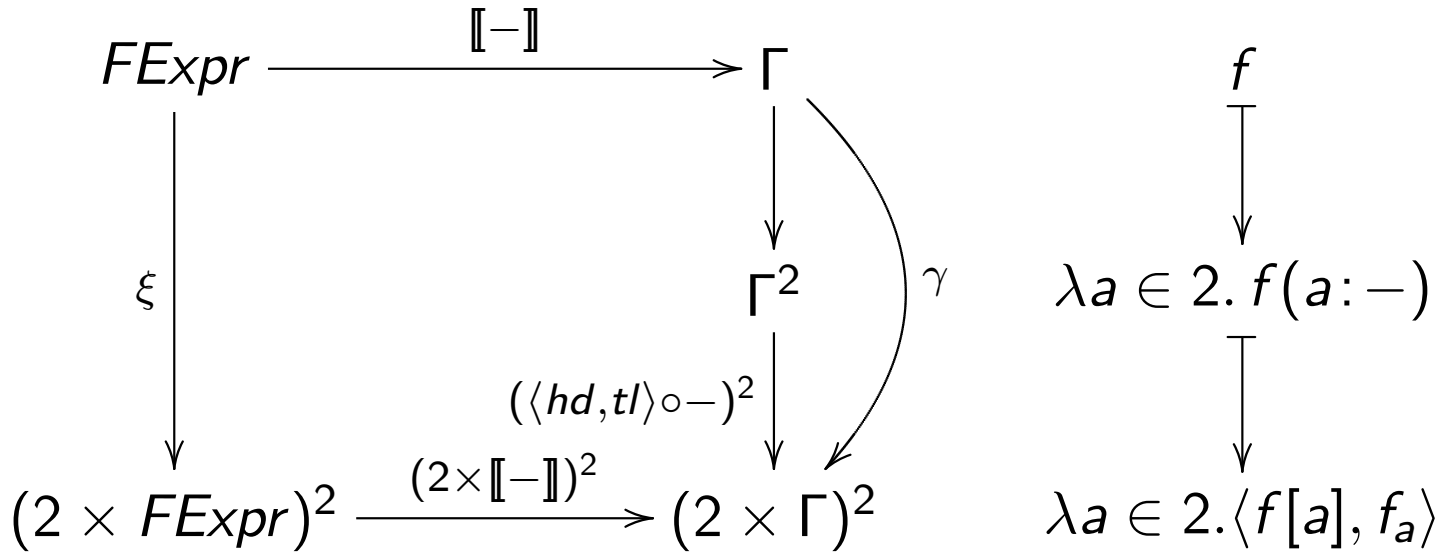
Mealy Machine of Function Expressions

Want to define Mealy structure $\xi \colon \mathit{FExpr} \to (2 \times \mathit{FExpr})^2$ such that $[\![-]\!]$ is Mealy homomorphism.



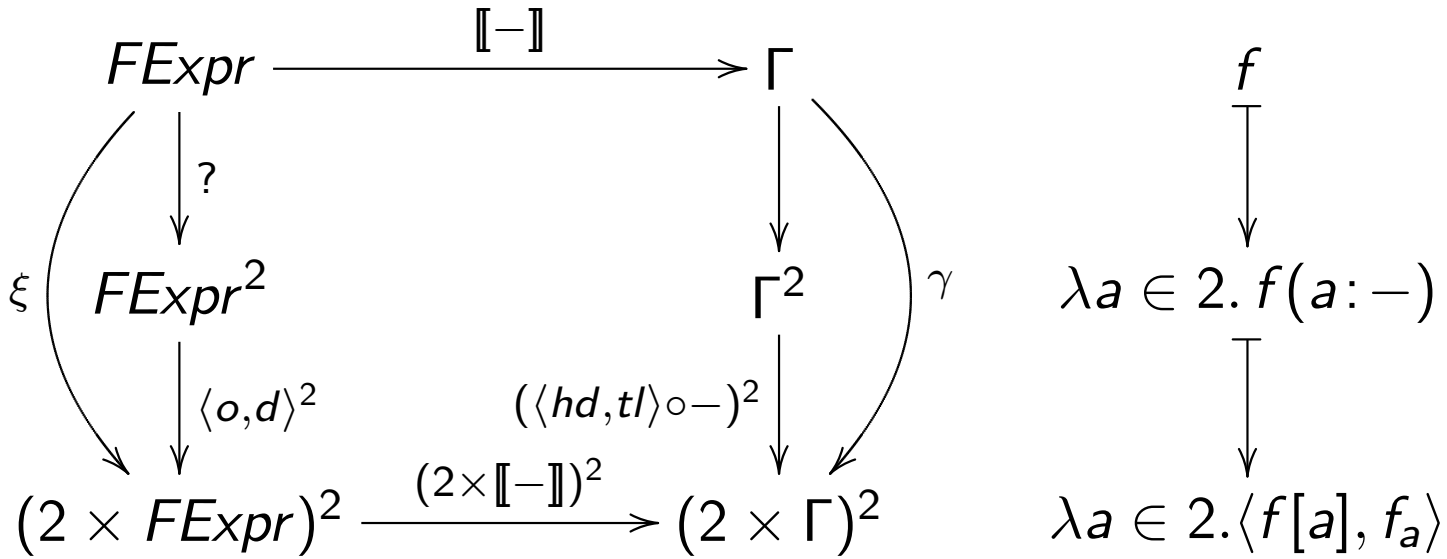
Mealy Machine of Function Expressions

Want to define Mealy structure $\xi\colon \mathit{FExpr} \to (2 \times \mathit{FExpr})^2$ such that $[\![-]\!]$ is Mealy homomorphism.



Mealy Machine of Function Expressions

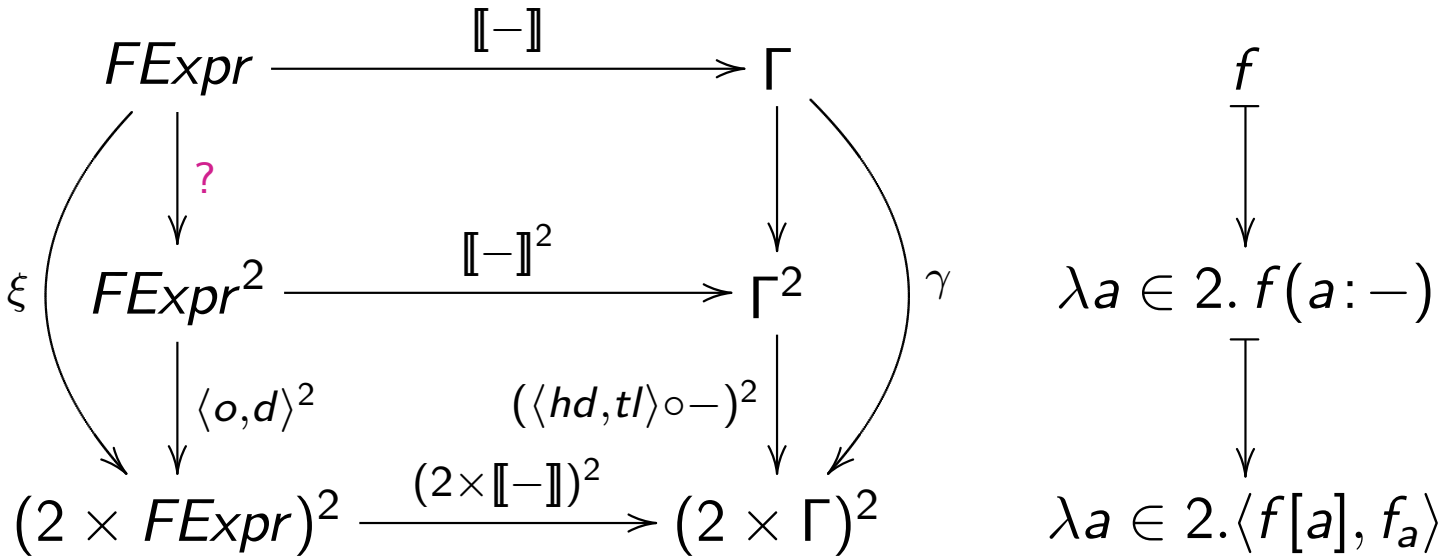
Want to define Mealy structure $\xi\colon FExpr \to (2 \times FExpr)^2$ such that $[\![-]\!]$ is Mealy homomorphism.



- $\langle o, d\rangle$ given by 2-adic stream differential equations.

Mealy Machine of Function Expressions

Want to define Mealy structure $\xi\colon FExpr \to (2 \times FExpr)^2$ such that $[\![-]\!]$ is Mealy homomorphism.



- $\langle o, d\rangle$ given by 2-adic stream differential equations.
- Need syntactic analogue of $f(a:-)$.

Syntactic Analogue of $f(a:-)$

- Given $\mathbf{E}$ and $a \in 2$, we want $\mathbf{E}(a:-) \in \mathit{FExpr}$ such that $[\![\mathbf{E}(a:-)]\!](\sigma) = [\![\mathbf{E}]\!](a:\sigma)$.
- Note: For $\sigma \in 2^\omega$: $\quad 0:\sigma = [\![\mathbf{X} \times \mathbf{s}]\!](\sigma)$
  $\qquad\qquad\qquad\qquad\qquad\qquad\quad 1:\sigma = [\![\mathbf{1} + (\mathbf{X} \times \mathbf{s})]\!](\sigma)$

Syntactic Analogue of $f(a:-)$

- Given $\mathbf{E}$ and $a \in 2$, we want $\mathbf{E}(a:-) \in \mathit{FExpr}$ such that $[\![\mathbf{E}(a:-)]\!](\sigma) = [\![\mathbf{E}]\!](a:\sigma)$.
- Note: For $\sigma \in 2^\omega$: $\quad 0:\sigma = [\![\mathbf{X} \times \mathbf{s}]\!](\sigma)$
  $\qquad\qquad\qquad\qquad\qquad\quad 1:\sigma = [\![\mathbf{1} + (\mathbf{X} \times \mathbf{s})]\!](\sigma)$
- Define: $\quad 0:\mathbf{s} \;:=\; \mathbf{X} \times \mathbf{s}$, and
  $\qquad\quad 1:\mathbf{s} \;:=\; \mathbf{1} + \mathbf{X} \times \mathbf{s}$
  $\qquad\quad \mathbf{E}(a:\mathbf{s}) \;:=\;$ result of substituting $a:\mathbf{s}$ for $\mathbf{s}$ in $\mathbf{E}$
- Lemma: $[\![\mathbf{E}(a:\mathbf{s})]\!](\sigma) = [\![\mathbf{E}]\!](a:\sigma)$ for all $\sigma \in 2^\omega$.

Syntactic Analogue of $f(a:-)$

- Given $\mathbf{E}$ and $a \in 2$, we want $\mathbf{E}(a:-) \in \mathit{FExpr}$ such that $[\![\mathbf{E}(a:-)]\!](\sigma) = [\![\mathbf{E}]\!](a:\sigma)$.
- Note: For $\sigma \in 2^\omega$: $\quad 0:\sigma = [\![\mathbf{X} \times \mathbf{s}]\!](\sigma)$
  $\qquad\qquad\qquad\qquad\qquad\quad 1:\sigma = [\![\mathbf{1} + (\mathbf{X} \times \mathbf{s})]\!](\sigma)$
- Define:
$$\begin{array}{lll} 0:\mathbf{s} & := & \mathbf{X} \times \mathbf{s}, \text{ and} \\ 1:\mathbf{s} & := & \mathbf{1} + \mathbf{X} \times \mathbf{s} \\ \mathbf{E}(a:\mathbf{s}) & := & \text{result of substituting } a:\mathbf{s} \text{ for } \mathbf{s} \text{ in } \mathbf{E} \end{array}$$
- Lemma: $[\![\mathbf{E}(a:\mathbf{s})]\!](\sigma) = [\![\mathbf{E}]\!](a:\sigma)$ for all $\sigma \in 2^\omega$.
- Mealy structure: essentially

$$\xi(\mathbf{E})(a) = \langle (o(\mathbf{E}(a:\mathbf{s}))), d(\mathbf{E}(a:\mathbf{s})) \rangle$$

($\langle o, d \rangle$ given by 2-adic stream differential equations)

Mealy Machine of Function Expressions

Mealy structure $\xi : \mathit{FExpr} \to (2 \times \mathit{FExpr})^2$ defined by:

Mealy Machine of Function Expressions

Mealy structure $\xi\colon \mathit{FExpr} \to (2 \times \mathit{FExpr})^2$ defined by:

initial output (syntax)					
$\mathbf{s}[a]$	$=$	a	$(-\mathbf{E})[a]$	$=$	$\mathbf{E}[a]$
$\mathbf{0}[a]$	$=$	0	$(\mathbf{E}+\mathbf{F})[a]$	$=$	$\mathbf{E}[a] \oplus \mathbf{F}[a]$
$\mathbf{1}[a]$	$=$	1	$(\mathbf{E}\times\mathbf{F})[a]$	$=$	$\mathbf{E}[a] \wedge \mathbf{F}[a]$
$\mathbf{X}[a]$	$=$	0	$(\mathbf{1}/(\mathbf{1}+(\mathbf{X}\times\mathbf{E})))[a]$	$=$	1

Mealy Machine of Function Expressions

Mealy structure $\xi\colon \mathit{FExpr} \to (2 \times \mathit{FExpr})^2$ defined by:

initial output (syntax)					
$\mathbf{s}[a]$	$=$	a	$(-\mathbf{E})[a]$	$=$	$\mathbf{E}[a]$
$\mathbf{0}[a]$	$=$	0	$(\mathbf{E}+\mathbf{F})[a]$	$=$	$\mathbf{E}[a] \oplus \mathbf{F}[a]$
$\mathbf{1}[a]$	$=$	1	$(\mathbf{E}\times\mathbf{F})[a]$	$=$	$\mathbf{E}[a] \wedge \mathbf{F}[a]$
$\mathbf{X}[a]$	$=$	0	$(\mathbf{1}/(\mathbf{1}+(\mathbf{X}\times\mathbf{E})))[a]$	$=$	1

stream function derivative (syntax)					
$\mathbf{s}_a$	$=$	$\mathbf{s}$	$(-\mathbf{E})_a$	$=$	$-(\mathbf{E}_a + \iota(\mathbf{E}[a]))$
$\mathbf{0}_a$	$=$	$\mathbf{0}$	$(\mathbf{E}+\mathbf{F})_a$	$=$	$(\mathbf{E}_a+\mathbf{F}_a) + \iota(\mathbf{E}[a] \wedge \mathbf{F}[a])$
$\mathbf{1}_a$	$=$	$\mathbf{0}$	$(\mathbf{E}\times\mathbf{F})_a$	$=$	$(\mathbf{E}_a \times \mathbf{F}(a{:}\mathbf{s})) + (\iota(\mathbf{E}[a]) \times \mathbf{F}_a)$
$\mathbf{X}_a$	$=$	$\mathbf{1}$	$(\mathbf{1}/(\mathbf{1}+(\mathbf{X}\times\mathbf{E})))_a$	$=$	$-(\mathbf{E}(a{:}\mathbf{s}))/(\mathbf{1}+(\mathbf{X}\times\mathbf{E}(a{:}\mathbf{s})))$

where $\iota(0) = \mathbf{0}$ and $\iota(1) = \mathbf{1}$.

Algebraic Semantics is Final Morphism

Theorem:
For all $\mathbf{E} \in \mathit{FExpr}$ and all $a \in 2$: $\llbracket \mathbf{E} \rrbracket[a] = \mathbf{E}[a]$, and $\llbracket \mathbf{E}_a \rrbracket = \llbracket \mathbf{E} \rrbracket_a$.

I.e., $\llbracket - \rrbracket \colon \langle \mathit{FExpr}, \xi \rangle \to \langle \Gamma, \gamma \rangle$ is a Mealy homomorphism, hence $\llbracket - \rrbracket$ equals behaviour map *beh*

$$\begin{array}{ccc} \mathit{FExpr} & \xrightarrow{\llbracket - \rrbracket} & \Gamma \\ \Big\downarrow{\scriptstyle \xi} & & \Big\downarrow{\scriptstyle \gamma} \\ (2 \times \mathit{FExpr})^2 & \xrightarrow{(2 \times \llbracket - \rrbracket)^2} & (2 \times \Gamma)^2 \end{array}$$

Generating Realisations

Realisation as generated subcoalgebra:

- We can compute $\langle \mathbf{E} \rangle$ by least fixpoint computation.
- Mealy machine $\langle \mathbf{E} \rangle$ is a realisation of $[\![\mathbf{E}]\!]$.
- In general: minimal realisation $\langle [\![\mathbf{E}]\!] \rangle$ is infinite.
  (examples: $\mathbf{s} \times \mathbf{s}$ and $\mathbf{1}/(\mathbf{1} + (\mathbf{X} \times \mathbf{s}))$)
- In general: $\langle \mathbf{E} \rangle$ is not minimal (and not finite).

Generating Realisations

Realisation as generated subcoalgebra:

- We can compute $\langle \mathbf{E} \rangle$ by least fixpoint computation.
- Mealy machine $\langle \mathbf{E} \rangle$ is a realisation of $[\![\mathbf{E}]\!]$.
- In general: minimal realisation $\langle [\![\mathbf{E}]\!] \rangle$ is infinite.
  (examples: $\mathbf{s} \times \mathbf{s}$ and $\mathbf{1}/(\mathbf{1} + (\mathbf{X} \times \mathbf{s}))$)
- In general: $\langle \mathbf{E} \rangle$ is not minimal (and not finite).

Ensuring finiteness and termination:

Restrict to rational function expressions.

Rational Function Expressions

- Def. $\mathbf{E} \in \mathit{FExpr}$ is rational if

$$\mathbf{E} = \frac{\mathbf{D} + (\mathbf{F} \times \mathbf{s})}{\mathbf{1} + (\mathbf{X} \times \mathbf{G})}$$

  where $\mathbf{D}, \mathbf{F}, \mathbf{G} \in \mathit{FExpr}$ do not contain inverse / and $\mathbf{s}$.

- Example:

$$\mathbf{E} = \frac{(-\mathbf{X}) + ((1 + \mathbf{X}^2) \times \mathbf{s})}{1 + (\mathbf{X} \times (\mathbf{X} + \mathbf{X}))} \qquad \left(\frac{-2 + (5 \times \mathbf{s})}{9} \right)$$

Rational Function Expressions

- Def. $\mathbf{E} \in \mathit{FExpr}$ is rational if

$$\mathbf{E} = \frac{\mathbf{D} + (\mathbf{F} \times \mathbf{s})}{\mathbf{1} + (\mathbf{X} \times \mathbf{G})}$$

  where $\mathbf{D}, \mathbf{F}, \mathbf{G} \in \mathit{FExpr}$ do not contain inverse $/$ and $\mathbf{s}$.

- Example:

$$\mathbf{E} = \frac{(-\mathbf{X}) + ((1 + \mathbf{X}^2) \times \mathbf{s})}{1 + (\mathbf{X} \times (\mathbf{X} + \mathbf{X}))} \qquad \left(\frac{-2 + (5 \times \mathbf{s})}{9}\right)$$

Lemma: Let $\mathbf{E}$ be rational, and $f = [\![\mathbf{E}]\!]$

- For all $g \in \langle f \rangle$, g is rational.
- $\langle f \rangle$ is finite.

Synthesis from Rational Function Expressions

Ensure minimality:

- Compute $\langle \mathbf{E} \rangle$ modulo equivalence $\equiv$ where $\mathbf{E} \equiv \mathbf{F}$ if $[\![\mathbf{E}]\!] = [\![\mathbf{F}]\!]$.
- $\equiv$ is axiomatised by integral domain identities.
- Implemented by reduction to normal forms.

Synthesis from Rational Function Expressions

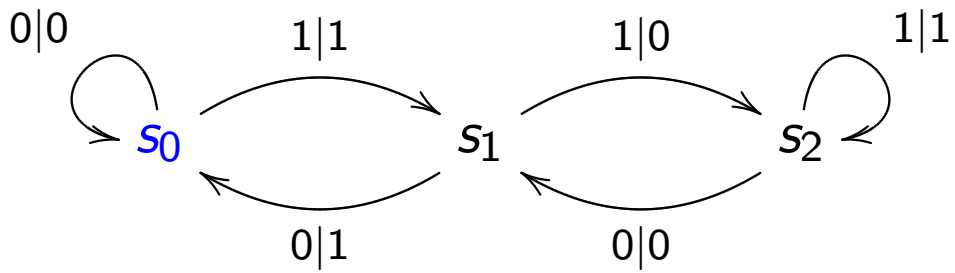
Ensure minimality:

- Compute $\langle \mathbf{E} \rangle$ modulo equivalence $\equiv$ where $\mathbf{E} \equiv \mathbf{F}$ if $[\![\mathbf{E}]\!] = [\![\mathbf{F}]\!]$.
- $\equiv$ is axiomatised by integral domain identities.
- Implemented by reduction to normal forms.

Theorem (Synthesis)
For all rational $\mathbf{E} = \frac{\mathbf{D}+(\mathbf{F}\times\mathbf{s})}{\mathbf{1}+(\mathbf{X}\times\mathbf{G})} \in \mathit{FExpr}$, we can construct a finite, minimal realisation of $[\![\mathbf{E}]\!]$ by computing $\langle \mathbf{E} \rangle / \equiv$.

Example $\mathbf{E} = (\mathbf{1} + \mathbf{X}) \times \mathbf{s}$

$[\![\mathbf{E}]\!](\sigma) = 3 \times \sigma = 3\sigma.$

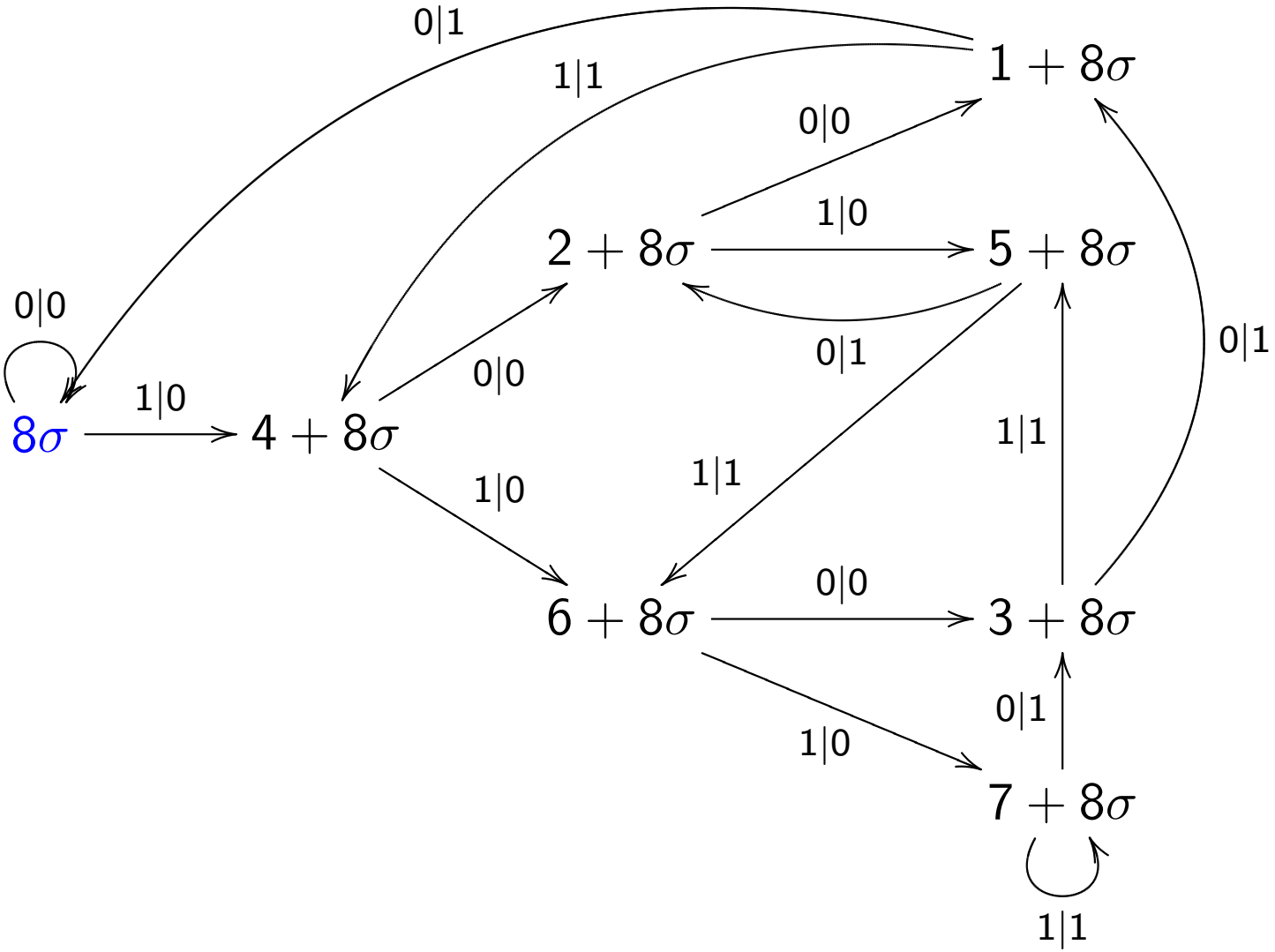


where

$$\begin{array}{lcccc} beh(s_0) & = & [\![\mathbf{E}]\!] & = & 3\sigma \\ beh(s_1) & = & [\![\mathbf{1} + \mathbf{E}]\!] & = & 1 + 3\sigma \\ beh(s_2) & = & [\![\mathbf{X} + \mathbf{E}]\!] & = & 2 + 3\sigma \end{array}$$

Example $\mathbf{E} = \mathbf{X}^3 \times \mathbf{s}$

$[\![\mathbf{E}]\!](\sigma) = 8 \times \sigma = 8\sigma.$



Summary

Specification		Semantics		Realisation
FExpr	$\longrightarrow$	Causal functions	$\longleftrightarrow$	Mealy
Rational *FExpr*	$\longrightarrow$	Fin. state causal func.	$\longleftrightarrow$	Fin. Mealy

Summary

Specification		Semantics		Realisation
FExpr	$\longrightarrow$	Causal functions	$\longleftrightarrow$	Mealy
Rational *FExpr*	$\longrightarrow$	Fin. state causal func.	$\longleftrightarrow$	Fin. Mealy
Regular expr.	$\longleftrightarrow$	Regular languages	$\longleftrightarrow$	DFA

FExpr versus *RegExp*:

- Regular expression $\longrightarrow$ DFA (Brzozowski derivative)
- Rational *FExpr* $\longrightarrow$ Finite Mealy (stream function derivative)

Summary

Specification		Semantics		Realisation
FExpr	$\longrightarrow$	Causal functions	$\longleftrightarrow$	Mealy
Rational *FExpr*	$\longrightarrow$	Fin. state causal func.	$\longleftrightarrow$	Fin. Mealy
Regular expr.	$\longleftrightarrow$	Regular languages	$\longleftrightarrow$	DFA

FExpr versus *RegExp*:

- Regular expression $\longrightarrow$ DFA (Brzozowski derivative)
- Rational *FExpr* $\longrightarrow$ Finite Mealy (stream function derivative)
- Kleene theorem: DFA $\longleftrightarrow$ Regular expressions.
- But: Finite Mealy $\not\longrightarrow$ Rational *FExpr*.

Summary

Specification		Semantics		Realisation
FExpr	$\longrightarrow$	Causal functions	$\longleftrightarrow$	Mealy
Rational *FExpr*	$\longrightarrow$	Fin. state causal func.	$\longleftrightarrow$	Fin. Mealy
Regular expr.	$\longleftrightarrow$	Regular languages	$\longleftrightarrow$	DFA

FExpr versus *RegExp*:

- Regular expression $\longrightarrow$ DFA (Brzozowski derivative)
- Rational *FExpr* $\longrightarrow$ Finite Mealy (stream function derivative)
- Kleene theorem: DFA $\longleftrightarrow$ Regular expressions.
- But: Finite Mealy $\not\longrightarrow$ Rational *FExpr*.

Synthesis (from specification to circuit):

- Generate subcoalgebra in Mealy machine of expressions.
- Rational *FExpr* $\longrightarrow$ Finite Mealy machine (impl. in Haskell)
  $\longrightarrow$ Seq. circuit (can be fully automated)

Related Work

- Logic synthesis of Mealy machines:
  Specification formula φ (in e.g. LTL). Synthesis as constructive non-emptiness test of Büchi automaton $\mathbb{A}_\varphi$.
- Coalgebraic generalisation of Kleene theorem:
  Kripke polynomial coalgebras, generalised regular expressions.
  (work by Bonsangue, Rutten, Silva)
- Generalised structural operational semantics (GSOS)
  (algebra-coalgebra interaction).