

Bialgebras, Bitstream Arithmetic and Mealy Machines

Helle Hvid Hansen

Technische Universiteit Eindhoven

Dagstuhl seminar: Coalgebraic Logic, 7-9 December 2009

Constructing coalgebras from syntax (coalgebraic synthesis):

- ▶ Construct deterministic automaton from regular expression, cf. [Brzozowski, 1964]
- ▶ Construct Mealy machine from arithmetic expression, cf. [Costa,HHH,Rutten, 2005]
- ▶ Construct F -coalgebra from F -expression (F is KPF+), cf. [Bonchi,Bonsangue,Rutten,Silva, 2009]

Constructing coalgebras from syntax (coalgebraic synthesis):

- ▶ Construct deterministic automaton from regular expression, cf. [Brzozowski, 1964]
- ▶ Construct Mealy machine from arithmetic expression, cf. [Costa,HHH,Rutten, 2005]
- ▶ Construct F -coalgebra from F -expression (F is KPF+), cf. [Bonchi,Bonsangue,Rutten,Silva, 2009]

Common basic idea:

- ▶ Expressions have algebraic and coalgebraic structure.
- ▶ Algebraic and coalgebraic semantics coincide.

Question: Are these constructions bialgebraic?

Regular Expressions and Deterministic Automata

[Jacobs, 2006]: Regular expressions, deterministic automata and formal languages form bialgebras (for a GSOS law λ):

$$\begin{array}{ccc} R^*(Reg) & \xrightarrow{R^*[-]} & R^*(\mathcal{P}(A^*)) & \langle \mathcal{P}(A^*), \alpha, \langle O, D \rangle \rangle \text{ is} \\ \mu \downarrow & & \downarrow \alpha & \text{a final } \lambda\text{-bialgebra.} \\ Reg & \xrightarrow{[-]} & \mathcal{P}(A^*) & \\ \langle O_{Brz}, D_{Brz} \rangle \downarrow & & \downarrow \langle O, D \rangle & \\ 2 \times Reg^A & \xrightarrow{2 \times [-]^A} & 2 \times (\mathcal{P}(A^*))^A & \end{array}$$

where

- ▶ $\langle R^*, \eta, \mu \rangle$ is free monad generated by Reg -signature functor.
- ▶ α is the algebraic semantics of Reg .
- ▶ $\langle O_{Brz}, D_{Brz} \rangle$ is Brzozowski's output and derivative.

Bitstream Function Expressions and Mealy Machines

[Rutten, 2005]: Bitstream functions expressions $FExpr$, binary Mealy machines and final Mealy machine $\langle \Phi, \phi \rangle$:

$$\begin{array}{ccc} T(FExpr) & \xrightarrow{T[-]} & T(\Phi) \\ \mu \downarrow & & \downarrow \alpha \\ FExpr & \xrightarrow{[-]} & \Phi \\ \xi \downarrow & & \downarrow \phi \\ (2 \times FExpr)^2 & \xrightarrow{(2 \times [-])^2} & (2 \times \Phi)^2 \end{array}$$

where

- ▶ $\langle T, \eta, \mu \rangle$ is free monad generated by $FExpr$ -signature functor.
- ▶ α is the algebraic semantics of $FExpr$.
- ▶ $\xi: FExpr \rightarrow (2 \times FExpr)^2$ is “syntactic” Mealy machine.

Bitstream Function Expressions and Mealy Machines

[Rutten, 2005]: Bitstream functions expressions $FExpr$, binary Mealy machines and final Mealy machine $\langle \Phi, \phi \rangle$:

$$\begin{array}{ccc} T(FExpr) & \xrightarrow{T[-]} & T(\Phi) \\ \mu \downarrow & & \downarrow \alpha \\ FExpr & \xrightarrow{[-]} & \Phi \\ \xi \downarrow & & \downarrow \phi \\ (2 \times FExpr)^2 & \xrightarrow{(2 \times [-])^2} & (2 \times \Phi)^2 \end{array}$$

$[-]$ is a Mealy morphism and T -algebra morphism.

where

- ▶ $\langle T, \eta, \mu \rangle$ is free monad generated by $FExpr$ -signature functor.
- ▶ α is the algebraic semantics of $FExpr$.
- ▶ $\xi: FExpr \rightarrow (2 \times FExpr)^2$ is “syntactic” Mealy machine.

Bitstream Function Expressions and Mealy Machines

[Rutten, 2005]: Bitstream functions expressions $FExpr$, binary Mealy machines and final Mealy machine $\langle \Phi, \phi \rangle$:

$$\begin{array}{ccc} T(FExpr) & \xrightarrow{T[-]} & T(\Phi) \\ \mu \downarrow & & \downarrow \alpha \\ FExpr & \xrightarrow{[-]} & \Phi \\ \xi \downarrow & & \downarrow \phi \\ (2 \times FExpr)^2 & \xrightarrow{(2 \times [-])^2} & (2 \times \Phi)^2 \end{array}$$

$[-]$ is a Mealy morphism and T -algebra morphism.

Is there a distributive law?

where

- ▶ $\langle T, \eta, \mu \rangle$ is free monad generated by $FExpr$ -signature functor.
- ▶ α is the algebraic semantics of $FExpr$.
- ▶ $\xi: FExpr \rightarrow (2 \times FExpr)^2$ is “syntactic” Mealy machine.

Overview

Overview

The rest of this talk:

- ▶ Bialgebra basics
- ▶ Mealy machines and 2-adic bitstream expressions.
- ▶ Distributive (GSOS) law for 2-adic bitstreams.
- ▶ Distributive law for Mealy machines and 2-adic bitstream functions: an operation must be added to syntax.

Distributive law

- ▶ Given monad $\langle T, \eta, \mu \rangle$ (syntax) and
- ▶ functor G (behaviour type).
- ▶ A distributive law of $\langle T, \eta, \mu \rangle$ over G is a natural transformation $\lambda: TG \Rightarrow GT$ such that

$$\begin{array}{ccc} GX & \xrightarrow{\eta_{GX}} & TGX \\ & \searrow^{G(\eta_X)} & \downarrow \lambda_X \\ & & GTX \end{array} \qquad \begin{array}{ccccc} TTGX & \xrightarrow{T(\lambda_X)} & TGTX & \xrightarrow{\lambda_{TX}} & GTTX \\ \downarrow \mu_{GX} & & & & \downarrow G(\mu_X) \\ TGX & \xrightarrow{\lambda_X} & & & GTX \end{array}$$

- ▶ Let λ be a distrib. law of a monad $\langle T, \eta, \mu \rangle$ over a functor G .
- ▶ A λ -bialgebra $\langle X, \alpha, \beta \rangle$ is a pair of maps

$$TX \xrightarrow{\alpha} X \xrightarrow{\beta} GX$$

such that $TX \xrightarrow{\alpha} X$ is an Eilenberg-Moore algebra, i.e., $\alpha \circ \eta = id$ and $\alpha \circ T(\alpha) = \alpha \circ \mu$, and

$$\begin{array}{ccccc} TX & \xrightarrow{\alpha} & X & \xrightarrow{\beta} & GX \\ T(\beta) \downarrow & & & & \uparrow G\alpha \\ TGX & \xrightarrow{\lambda_X} & & & GTX \end{array}$$

Morphisms of λ -bialgebras

A λ -bialgebra morphism $f: (X, \alpha, \beta) \rightarrow (Y, \gamma, \delta)$ is a function $f: X \rightarrow Y$ such that:

$$\begin{array}{ccccc} TX & \xrightarrow{\alpha} & X & \xrightarrow{\beta} & GX \\ Tf \downarrow & & f \downarrow & & \downarrow Gf \\ TY & \xrightarrow{\gamma} & Y & \xrightarrow{\delta} & GY \end{array}$$

i.e., f is a morphism of T -algebras and of G -coalgebras.

Abstract GSOS laws and Free monads

- ▶ Let $\langle T, \eta, \mu \rangle$ be monad, G functor.
- ▶ An (abstract) GSOS law is a distributive law $\lambda: T(G \times Id) \Rightarrow (G \times Id)T$ such that $\pi_2 \circ \lambda = T(\pi_2)$.
- ▶ If $\langle T_\Sigma, \eta, \mu \rangle$ is the free monad generated by a signature functor Σ , then there is a 1-1 correspondence

$$\frac{\text{GSOS laws } \lambda: T_\Sigma(G \times Id) \Rightarrow (G \times Id)T_\Sigma}{\text{(plain) natural transformations } \rho: \Sigma(G \times Id) \Rightarrow GT_\Sigma}$$

Theorem:

If $\lambda: T(G \times Id) \Rightarrow (G \times Id)T$ is a GSOS law, and $\zeta: Z \rightarrow GZ$ is a final G -coalgebra, then there is an Eilenberg-Moore algebra $\alpha: TZ \rightarrow Z$ such that $\langle Z, \alpha, \zeta \rangle$ is a final λ -bialgebra.

Quick recap...

Notation: For $\sigma \in 2^\omega$,

- ▶ $\sigma = (\sigma(0), \sigma(1), \sigma(2), \dots)$
- ▶ $\sigma(0)$ initial value (head) of σ
- ▶ σ' stream derivative (tail) of σ
- ▶ $a:\sigma = (a, \sigma(0), \sigma(1), \sigma(2), \dots)$ for $a \in 2$.

Mealy Machines

- ▶ (Binary) Mealy machine $m: S \rightarrow (2 \times S)^2$ where $2 = \{0, 1\}$.
- ▶ Final Mealy machine $\phi: \Phi \rightarrow (2 \times \Phi)^2$ where

$$\Phi = \{f: 2^\omega \rightarrow 2^\omega \mid f \text{ is causal}\}.$$

$$\phi(f)(a) = \langle f[a], f_a \rangle$$

where for all $a \in 2$ and $\sigma \in 2^\omega$:

$$f[a] = (f(a:\sigma))(0) \quad (\text{initial output})$$

$$f_a(\sigma) = (f(a:\sigma))' \quad (\text{stream function derivative})$$

Arithmetic Expressions

Expressions:

- ▶ Arithmetic signature $\mathbf{0, 1, X, +, -, \times, 1/-}$: Signature functor $\Sigma X = 1 + 1 + 1 + (X \times X) + X + (X \times X) + X$
- ▶ Bitstream expressions = closed Σ -terms.
- ▶ Function expressions = Σ -terms over variable \mathbf{s} .

Algebraic semantics: 2-adic arithmetic.

- ▶ bitstream $\sigma \in 2^\omega$ represents 2-adic integer $\sum_{i < \omega} \sigma(i) \cdot 2^i$.
- ▶ bitstream operations are 2-adic sum, minus, product, inverse. (Infinitary binary arithmetic.)

Constants: $\mathbf{0} = (0, 0, 0, 0, \dots)$
 $\mathbf{1} = (1, 0, 0, 0, \dots)$
 $\mathbf{X} = (0, 1, 0, 0, 0, \dots)$

Operations (defined by stream differential equations):

Constants: $\mathbf{0} = (0, 0, 0, 0, \dots)$
 $\mathbf{1} = (1, 0, 0, 0, \dots)$
 $\mathbf{X} = (0, 1, 0, 0, 0, \dots)$

Operations (defined by stream differential equations):

sum:	$(\sigma + \tau)(0) = \sigma(0) \oplus \tau(0)$ $(\sigma + \tau)' = (\sigma' + \tau') + [\sigma(0) \wedge \tau(0)]$
minus:	$(-\sigma)(0) = \sigma(0)$ $(-\sigma)' = -(\sigma' + [\sigma(0)])$
product:	$(\sigma \times \tau)(0) = \sigma(0) \wedge \tau(0)$ $(\sigma \times \tau)' = (\sigma' \times \tau) + ([\sigma(0)] \times \tau')$
inverse: ($\sigma(0) = 1$)	$(1/\sigma)(0) = 1$ $(1/\sigma)' = -(\sigma' \times (1/\sigma))$

using bit operations: $\oplus =$ addition modulo 2, $\wedge =$ boolean AND.

We want a totally defined algebraic semantics.

- ▶ Problem: $1/\sigma$ only defined if $\sigma(0) = 1$.
- ▶ So we do not have (total) map $\Sigma 2^\omega \rightarrow 2^\omega$.
- ▶ Solution: replace with “guarded inverse” $1//\sigma$:
 $1//\sigma := 1/(1:\sigma)$ where $1:\sigma = (1, \sigma(0), \sigma(1), \dots)$.
- ▶ Stream differential equations:
 $(1//\sigma)(0) = 1$
 $(1//\sigma)' = (1/(1:\sigma))' = -((1:\sigma)' \times (1/(1:\sigma))) = -(\sigma \times (1//\sigma))$

From now on: Σ is signature functor for $\mathbf{0}, \mathbf{1}, \mathbf{X}, +, -, \times, 1//$.

GSOS law for Bitstream Expressions

Bitstream functor: $BX = 2 \times X$. GSOS law of $\langle T_\Sigma, \eta, \mu \rangle$ over B : natural transform. $\lambda: \Sigma(B \times Id) \Rightarrow BT_\Sigma$ s.t. $\pi_2 \circ \lambda = T_\Sigma(\pi_2)$. We use notation: $\iota(0) = \mathbf{0}$ and $\iota(1) = \mathbf{1}$.

$\Sigma(2 \times X \times X)$	$\xrightarrow{\lambda_X}$	$2 \times T_\Sigma X$
$\mathbf{0}$	$\xrightarrow{\text{zero}}$	$\langle 0, \mathbf{0} \rangle$
$\mathbf{1}$	$\xrightarrow{\text{one}}$	$\langle 1, \mathbf{0} \rangle$
\mathbf{X}	\xrightarrow{X}	$\langle 0, \mathbf{1} \rangle$
$\langle \langle a, s', s \rangle, \langle b, t', t \rangle \rangle$	$\xrightarrow{\text{sum}}$	$\langle a \oplus b, (s' + t') + \iota(a \wedge b) \rangle$
$\langle a, s', s \rangle$	$\xrightarrow{\text{minus}}$	$\langle a, -(s' + \iota(a)) \rangle$
$\langle \langle a, s', s \rangle, \langle b, t', t \rangle \rangle$	$\xrightarrow{\text{product}}$	$\langle a \wedge b, (s' \times t) + \iota(a) \times t' \rangle$
$\langle a, s', s \rangle$	$\xrightarrow{\text{g.inv.}}$	$\langle 1, -(s \times (1 // s)) \rangle$

Bialgebra of Bitstream Expressions and Bitstreams

Diagram of λ -bialgebras (for GSOS-law λ):

$$\begin{array}{ccc} T(BExpr) & \xrightarrow{\tau[-]} & T(2^\omega) \\ \mu \downarrow & & \downarrow \mathcal{A}_{2adic} \\ BExpr & \xrightarrow{[-]} & 2^\omega \\ \beta \downarrow & & \downarrow \langle \text{hd}, \text{tl} \rangle \\ 2 \times BExpr & \xrightarrow{2 \times [-]} & (2 \times 2^\omega) \end{array}$$

- ▶ 2-adic operations well-defined by stream differential equations.
- ▶ The 2-adic algebraic semantics $[-]: BExpr \rightarrow 2^\omega$ is the final $2 \times (-)$ -morphism into $\langle 2^\omega, \text{hd}, \text{tl} \rangle$.
- ▶ $\langle 2^\omega, \mathcal{A}_{2adic}, \langle \text{hd}, \text{tl} \rangle \rangle$ is a final bialgebra.

GSOS law for Function Expressions and Mealy Machines?

- ▶ Same signature Σ : $\mathbf{0}, \mathbf{1}, \mathbf{X}, +, -, \times, \mathbf{1} //$
- ▶ Function expressions $FExpr = T_{\Sigma}(\{\mathbf{s}\})$ (Σ -terms over var. \mathbf{s})

GSOS law for Function Expressions and Mealy Machines?

- ▶ Same signature Σ : $\mathbf{0}, \mathbf{1}, \mathbf{X}, +, -, \times, 1//$
- ▶ Function expressions $FExpr = T_{\Sigma}(\{\mathbf{s}\})$ (Σ -terms over var. \mathbf{s})
- ▶ Algebraic 2-adic semantics:

For $E \in FExpr$, $\llbracket E \rrbracket : 2^{\omega} \rightarrow 2^{\omega}$ defined for all $\sigma \in 2^{\omega}$ by:

$$\begin{aligned}\llbracket \mathbf{0} \rrbracket(\sigma) &= (0, 0, 0, \dots) \\ \llbracket \mathbf{1} \rrbracket(\sigma) &= (1, 0, 0, \dots) \\ \llbracket \mathbf{X} \rrbracket(\sigma) &= (0, 1, 0, \dots) \\ \llbracket \mathbf{s} \rrbracket(\sigma) &= \sigma \\ \llbracket E + F \rrbracket(\sigma) &= \llbracket E \rrbracket(\sigma) + \llbracket F \rrbracket(\sigma) \\ \llbracket -E \rrbracket(\sigma) &= -\llbracket E \rrbracket(\sigma) \\ \llbracket E \times F \rrbracket(\sigma) &= \llbracket E \rrbracket(\sigma) \times \llbracket F \rrbracket(\sigma) \\ \llbracket 1//E \rrbracket(\sigma) &= 1//\llbracket E \rrbracket(\sigma)\end{aligned}$$

(i.e., assign \mathbf{s} to $id_{2^{\omega}}$, take pointwise extension of 2-adic operations to $\{f : 2^{\omega} \rightarrow 2^{\omega}\}$ plus freeness)

Mealy Machine of Function Expressions

- ▶ We define Mealy structure $\xi: FExpr \rightarrow (2 \times FExpr)^2$.
- ▶ Recall, in final Mealy machine (causal functions):

$$f[a] = (f(a:\sigma))(0) \quad (\text{initial output})$$

$$f'_a(\sigma) = (f(a:\sigma))' \quad (\text{stream function derivative})$$

Basic idea: Mimic in syntax.

Mealy Machine of Function Expressions

- ▶ We define Mealy structure $\xi: FExpr \rightarrow (2 \times FExpr)^2$.

- ▶ Recall, in final Mealy machine (causal functions):

$$f[a] = (f(a:\sigma))(0) \quad (\text{initial output})$$

$$f_a(\sigma) = (f(a:\sigma))' \quad (\text{stream function derivative})$$

Basic idea: Mimic in syntax.

- ▶ Note: For $\sigma \in 2^\omega$:
 $0:\sigma = \llbracket \mathbf{X} \times \mathbf{s} \rrbracket(\sigma)$
 $1:\sigma = \llbracket \mathbf{1} + \mathbf{X} \times \mathbf{s} \rrbracket(\sigma)$

Mealy Machine of Function Expressions

- ▶ We define Mealy structure $\xi: FExpr \rightarrow (2 \times FExpr)^2$.

- ▶ Recall, in final Mealy machine (causal functions):

$$f[a] = (f(a:\sigma))(0) \quad (\text{initial output})$$

$$f_a(\sigma) = (f(a:\sigma))' \quad (\text{stream function derivative})$$

Basic idea: Mimic in syntax.

- ▶ Note: For $\sigma \in 2^\omega$:
 $0:\sigma = \llbracket \mathbf{X} \times \mathbf{s} \rrbracket(\sigma)$
 $1:\sigma = \llbracket \mathbf{1} + \mathbf{X} \times \mathbf{s} \rrbracket(\sigma)$

- ▶ Define: $0:\mathbf{s} := \mathbf{X} \times \mathbf{s}$, and
 $1:\mathbf{s} := \mathbf{1} + \mathbf{X} \times \mathbf{s}$
 $E(a:\mathbf{s}) :=$ result of substituting $a:\mathbf{s}$ for \mathbf{s} in E

- ▶ Lemma: $\llbracket E(a:\mathbf{s}) \rrbracket(\sigma) = \llbracket E \rrbracket(a:\sigma)$ for all $\sigma \in 2^\omega$.

Mealy Machine of Function Expressions

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

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

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

Algebraic Semantics is Final Morphism

Theorem:

For all $E \in FExpr$ and all $a \in 2$: $\llbracket E \rrbracket[a] = E[a]$, and
 $\llbracket E_a \rrbracket = \llbracket E \rrbracket_a$.

I.e., the algebraic semantics $\llbracket - \rrbracket: \langle FExpr, \xi \rangle \rightarrow \langle \Phi, \phi \rangle$ is a Mealy morphism, hence $\llbracket - \rrbracket$ equals final Mealy morphism.

$$\begin{array}{ccc} T(FExpr) & \xrightarrow{\tau \llbracket - \rrbracket} & T(\Phi) \\ \mu \downarrow & & \downarrow \alpha_{2adic} \\ FExpr & \xrightarrow{\llbracket - \rrbracket} & \Phi \\ \xi \downarrow & & \downarrow \phi \\ (2 \times FExpr)^2 & \xrightarrow{(2 \times \llbracket - \rrbracket)^2} & (2 \times \Phi)^2 \end{array}$$

Distributive (GSOS) Law?

We have compatibility maps ρ_{FExpr} and ρ_Φ such that:

$$\begin{array}{ccccc} \Sigma FExpr & \xrightarrow{\underline{\mu}} & FExpr & \xrightarrow{\xi} & (2 \times FExpr)^2 \\ \Sigma \langle \xi, id \rangle \downarrow & & & & \uparrow (2 \times \mu)^2 \\ \Sigma((2 \times FExpr)^2 \times FExpr) & \xrightarrow{\rho_{FExpr}} & & & (2 \times T_\Sigma(FExpr))^2 \end{array}$$

and

$$\begin{array}{ccccc} \Sigma \Phi & \xrightarrow{\alpha_{2adic}} & \Phi & \xrightarrow{\phi} & (2 \times \Phi)^2 \\ \Sigma \langle \phi, id \rangle \downarrow & & & & \uparrow (2 \times \alpha_{2adic})^2 \\ \Sigma((2 \times \Phi)^2 \times \Phi) & \xrightarrow{\rho_\Phi} & & & (2 \times T_\Sigma(\Phi))^2 \end{array}$$

Distributive (GSOS) Law?

For example:

(notation: for $\delta \in (2 \times X)^2$ write $\delta(a) = \langle \delta_1(a), \delta_2(a) \rangle$, $a \in 2$)

$$\rho_{FExpr}: \Sigma((2 \times FExpr)^2 \times FExpr) \rightarrow (2 \times T_\Sigma(FExpr))^2$$

$$\rho_{FExpr}: \langle \langle \delta, E \rangle, \langle \gamma, F \rangle \rangle \xrightarrow{\text{product}} \\ \lambda a \in 2. \langle \delta_1(a) \wedge \gamma_1(a), \\ (\delta_2(a) \times F(a:s)) + \iota(\delta_1(a)) \times \gamma_2(a) \rangle$$

$$\rho_\Phi: \Sigma((2 \times \Phi)^2 \times \Phi) \rightarrow (2 \times T_\Sigma(\Phi))^2$$

$$\rho_\Phi: \langle \langle \delta, f \rangle, \langle \gamma, g \rangle \rangle \xrightarrow{\text{product}} \\ \lambda a \in 2. \langle \delta_1(a) \wedge \gamma_1(a), \\ (\delta_2(a) \times g(a:-)) + [\delta_1(a)] \times \gamma_2(a) \rangle$$

Note: the map $\sigma \mapsto g(a:\sigma)$ is causal, when g is.

Distributive (GSOS) Law?

For example:

(notation: for $\delta \in (2 \times X)^2$ write $\delta(a) = \langle \delta_1(a), \delta_2(a) \rangle, a \in 2$)

$$\rho_{FExpr}: \Sigma((2 \times FExpr)^2 \times FExpr) \rightarrow (2 \times T_\Sigma(FExpr))^2$$

$$\rho_{FExpr}: \langle \langle \delta, E \rangle, \langle \gamma, F \rangle \rangle \xrightarrow{\text{product}} \\ \lambda a \in 2. \langle \delta_1(a) \wedge \gamma_1(a), \\ (\delta_2(a) \times F(a:s)) + \iota(\delta_1(a)) \times \gamma_2(a) \rangle$$

$$\rho_\Phi: \Sigma((2 \times \Phi)^2 \times \Phi) \rightarrow (2 \times T_\Sigma(\Phi))^2$$

$$\rho_\Phi: \langle \langle \delta, f \rangle, \langle \gamma, g \rangle \rangle \xrightarrow{\text{product}} \\ \lambda a \in 2. \langle \delta_1(a) \wedge \gamma_1(a), \\ (\delta_2(a) \times g(a:-)) + [\delta_1(a)] \times \gamma_2(a) \rangle$$

Note: the map $\sigma \mapsto g(a:\sigma)$ is causal, when g is.

What should $F(a:s)$ and $f(a:-)$ be in an arbitrary Mealy machine?

Distributive (GSOS) Law

Solution (sketch):

- ▶ Extend Σ with an operation $a \triangleright - : x \mapsto a \triangleright x$ for each bit $a \in 2$.
- ▶ $a \triangleright x$ should be thought of as a Mealy machine which has already seen an a (a is stored in some internal buffer).
- ▶ $a \triangleright -$ is specified by GSOS rule:

$$\frac{x \xrightarrow{a|b} y}{a \triangleright x \xrightarrow{c|b} c \triangleright y}$$

- ▶ In $\langle FExpr, \xi \rangle$: $a \triangleright E = E(a:s)$.
- ▶ In $\langle \Gamma, \gamma \rangle$: $a \triangleright f = f(a:-)$.

Many thanks to Bartek Klin for helping me solve this problem!

Conclusion

Summary:

Regular expressions and DFA-functor $2 \times (-)^A$	GSOS law (cf. [Jacobs, 2006])
Arithmetic bitstream expressions and bitstream functor $2 \times (-)$	GSOS law (see also Falk Bartels, PhD Thesis.)
Arithmetic function expressions and Mealy functor $(2 \times (-))^2$	GSOS law, but only after extending signature

Questions:

- ▶ F -expressions and F -coalgebras ? (in next talk by Marcello).
- ▶ Is notion of Mealy machine with buffer more generally useful?