Regular Expressions for Polynomial Coalgebras

Marcello Bonsangue[1,2], Jan Rutten[1,3] and **Alexandra Silva**[1]

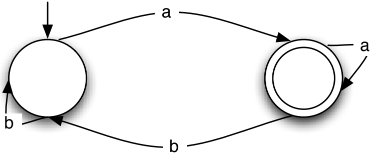
[1]Centrum voor Wiskunde en Informatica
[2]LIACS - Leiden University
[3]Vrije Universiteit Amsterdam

CoCoCo 2008

Pisa, January 21-23, 2008

Regular Expressions

- Well known for classical automata
- They provide a concise description of the accepted language
- Automaton $\leftrightsquigarrow$ Regular expressions



$$b^* a(a + bb^* a)^*$$

Polynomial coalgebras

- Generalizations of deterministic automata
- Polynomial coalgebras: set of states $S + t \; : \; S \to GS$

$$G ::= Id \mid B \mid G \times G \mid G + G \mid G^A$$

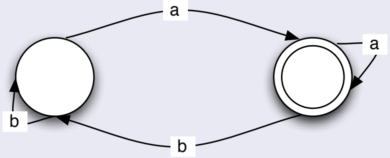
Polynomial coalgebras

- Generalizations of deterministic automata
- Polynomial coalgebras: set of states S + $t : S \to GS$

$$G ::= Id \mid B \mid G \times G \mid G + G \mid G^A$$

$G = 2 \times Id^A$ – Deterministic automata

Set of states S + trans. function $t : S \to 2 \times S^A$



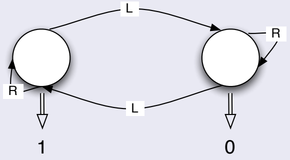
Polynomial coalgebras

- Generalizations of deterministic automata
- Polynomial coalgebras: set of states S + $t \ : \ S \to GS$

$$G ::= Id \mid B \mid G \times G \mid G + G \mid G^A$$

$G = Id \times A \times Id$ – Binary tree automata

Set of states S + trans. function $t \ : \ S \to S \times A \times S$



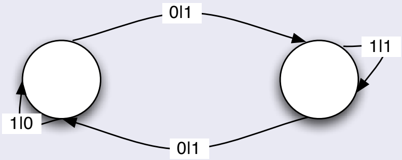
Polynomial coalgebras

- Generalizations of deterministic automata
- Polynomial coalgebras: set of states S + $t \; : \; S \to GS$

$$G ::= Id \mid B \mid G \times G \mid G + G \mid G^A$$

$G = (B \times Id)^A$ – Mealy machines

Set of states S + trans. function $t \; : \; S \to (B \times S)^A$



The story

- Logic for Mealy machines

  [BRS08] *Coalgebraic Logic and Synthesis of Mealy Machines*. FoSSaCS 2008

- Generalization to polynomial coalgebras

  [BRS08-2] *Regular expressions for polynomial coalgebras*. Submitted

The story

Part I

Logic for Mealy machines

$\xRightarrow{\text{intuition}}$

Part II

Regular expressions for polynomial coalgebras

Part I

Motivation

Design of Sequential systems

- (Binary) Mealy machines $\leftrightarrow$ digital circuits
- There is no formal way of specifying Mealy machines
- Typically they are "defined" in a natural language, such as English

  Source of ambiguities

- We need a formal way of specifying Mealy machines

Motivation

Design of Sequential systems

- (Binary) Mealy machines $\leftrightarrow$ digital circuits
- There is no formal way of specifying Mealy machines
- Typically they are "defined" in a natural language, such as English

  Source of ambiguities
- We need a formal way of specifying Mealy machines

Motivation

Design of Sequential systems

- (Binary) Mealy machines $\leftrightarrow$ digital circuits
- There is no formal way of specifying Mealy machines
- Typically they are "defined" in a natural language, such as English

Source of ambiguities

- We need a formal way of specifying Mealy machines

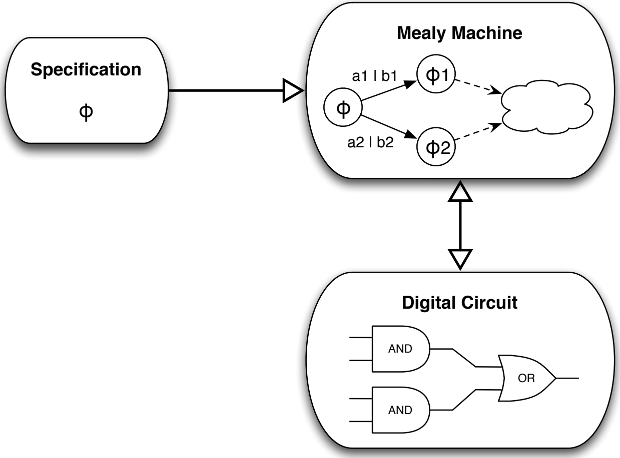
Motivation

Design of Sequential systems

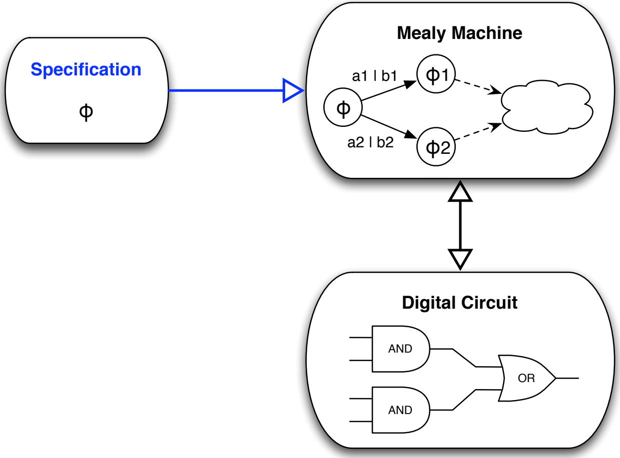
- (Binary) Mealy machines $\leftrightarrow$ digital circuits
- There is no formal way of specifying Mealy machines
- Typically they are “defined” in a natural language, such as English

  Source of ambiguities
- We need a formal way of specifying Mealy machines

What will we show?



What will we show?



But first...

What do we mean by **Binary Mealy Machine?**

- Mealy machines = **Deterministic Mealy machines**
- Mealy machine = set of states S + transition function f

$$\begin{array}{lcl} f & : & S \to (B \times S)^A \\ f(s)(a) & = & < b, s' > \end{array}$$

$$s \xrightarrow{a|b} s'$$

  A is the input alphabet and B is the output alphabet
- Binary Mealy machines : $A = 2$ and

$$B = \mathcal{B} = \begin{array}{ccc} & \top & \\ 1 & & 0 \\ & \bot & \end{array}$$

But first…

What do we mean by **Binary Mealy Machine?**

- Mealy machines = **Deterministic Mealy machines**
- Mealy machine = set of states S + transition function f

$$\begin{array}{lcl} f & : & S \to (B \times S)^A \\ f(s)(a) & = & < b, s' > \end{array}$$

$$s \xrightarrow{a|b} s'$$

A is the input alphabet and B is the output alphabet

- Binary Mealy machines : $A = 2$ and

$$B = \mathcal{B} = \begin{tikzcd} & \top \arrow[dl, no head] \arrow[dr, no head] & \\ 1 \arrow[dr, no head] & & 0 \arrow[dl, no head] \\ & \bot & \end{tikzcd}$$

Information order

$$\mathcal{B} = \begin{tikzcd} & \top \arrow[dl, dash] \arrow[dr, dash] & \\ 1 & & 0 \\ & \bot \arrow[ul, dash] \arrow[ur, dash] & \end{tikzcd}$$

- $\top$ – *abstraction* (under-specification)
- $\bot$ – *inconsistency* (over-specification)
- 0 and 1 – concrete output values.

Mealy automata are coalgebras

Observation:

A Mealy machine is a coalgebra of the functor

$$\begin{array}{rcl} M & : & Set \to Set \\ M(X) & = & (B \times X)^A \end{array}$$

(Almost) for free:

- Notion of (bi)simulation : equivalence between states, minimization
- Semantics in terms of final coalgebra – causal functions
- Specification language for Mealy machines

Mealy automata are coalgebras

Observation:

A Mealy machine is a coalgebra of the functor

$$\begin{aligned} M &: Set \to Set \\ M(X) &= (B \times X)^A \end{aligned}$$

(Almost) for free:

- Notion of (bi)simulation : equivalence between states, minimization
- Semantics in terms of final coalgebra – causal functions
- Specification language for Mealy machines

A coalgebraic logic for Mealy machines

Based on the work by Marcello and Alexander Kurz, we derive a specification language for $M(X) = (B \times X)^A$:

A coalgebraic logic for Mealy machines

Based on the work by Marcello and Alexander Kurz, we derive a specification language for $M(X) = (B \times X)^A$:

$$\phi ::= tt \quad | \quad \phi \wedge \phi \quad | \quad a(\phi) \quad | \quad a \downarrow b \quad | \quad x \quad | \quad \nu x.\psi$$

ψ guarded

A coalgebraic logic for Mealy machines

Based on the work by Marcello and Alexander Kurz, we derive a specification language for $M(X) = (B \times X)^A$:

$$\phi ::= tt \quad | \quad \phi \wedge \phi \quad | \quad a(\phi) \quad | \quad a \downarrow b \quad | \quad x \quad | \quad \nu x.\psi$$

ψ guarded

A coalgebraic logic for Mealy machines

Based on the work by Marcello and Alexander Kurz, we derive a specification language for $M(X) = (B \times X)^A$:

$$\phi ::= tt \quad | \quad \phi \wedge \phi \quad | \quad a(\phi) \quad | \quad a \downarrow b \quad | \quad x \quad | \quad \nu x.\psi$$

ψ guarded

Remark Simple but expressive language: Every finite Mealy machine corresponds to a finite formula in our language.

Simple properties

- Output 0 at each input of 1

$$1 \downarrow 0$$

- Output 0 at each input of two consecutive 1's

$$\nu x.(1(1 \downarrow 0 \wedge 1(x) \wedge 0(x)) \wedge 0(x))$$

- Output 0 at each second input of 1

$$\nu x.(0(x) \wedge 1(\nu y.0(y) \wedge 1 \downarrow 0 \wedge 1(x))$$

Simple properties

- Output 0 at each input of 1

$$1 \downarrow 0$$

- Output 0 at each input of two consecutive 1's

$$\nu x.(1(1 \downarrow 0 \wedge 1(x) \wedge 0(x)) \wedge 0(x))$$

- Output 0 at each second input of 1

$$\nu x.(0(x) \wedge 1(\nu y.0(y) \wedge 1 \downarrow 0 \wedge 1(x))$$

Simple properties

- Output 0 at each input of 1

$$\nu x.(1 \downarrow 0 \wedge 1(x) \wedge 0(x))$$

- Output 0 at each input of two consecutive 1's

$$\nu x.(1(1 \downarrow 0 \wedge 1(x) \wedge 0(x)) \wedge 0(x))$$

- Output 0 at each second input of 1

$$\nu x.(0(x) \wedge 1(\nu y.0(y) \wedge 1 \downarrow 0 \wedge 1(x))$$

Simple properties

- Output 0 at each input of 1

$$\nu x.(1 \downarrow 0 \wedge 1(x) \wedge 0(x))$$

- Output 0 at each input of two consecutive 1's

$$\nu x.(1(1 \downarrow 0 \wedge 1(x) \wedge 0(x)) \wedge 0(x))$$

- Output 0 at each second input of 1

$$\nu x.(0(x) \wedge 1(\nu y.0(y) \wedge 1 \downarrow 0 \wedge 1(x)))$$

Simple properties

- Output 0 at each input of 1

$$\nu x.(1 \downarrow 0 \wedge 1(x) \wedge 0(x))$$

- Output 0 at each input of two consecutive 1's

$$\nu x.(1(1 \downarrow 0 \wedge 1(x) \wedge 0(x)) \wedge 0(x))$$

- Output 0 at each second input of 1

$$\nu x.(0(x) \wedge 1(\nu y.0(y) \wedge 1 \downarrow 0 \wedge 1(x))$$

Simple properties

- Output 0 at each input of 1

$$\nu x.(1 \downarrow 0 \wedge 1(x) \wedge 0(x))$$

- Output 0 at each input of two consecutive 1's

$$\nu x.(1(1 \downarrow 0 \wedge 1(x) \wedge 0(x)) \wedge 0(x))$$

- Output 0 at each second input of 1

$$\nu x.(0(x) \wedge 1(\nu y.0(y) \wedge 1 \downarrow 0 \wedge 1(x))$$

Simple properties

- Output 0 at each input of 1

$$\nu x.(1 \downarrow 0 \wedge 1(x) \wedge 0(x))$$

- Output 0 at each input of two consecutive 1's

$$\nu x.(1(1 \downarrow 0 \wedge 1(x) \wedge 0(x)) \wedge 0(x))$$

- Output 0 at each second input of 1

$$\nu x.(0(x) \wedge 1(\nu y.0(y) \wedge 1 \downarrow 0 \wedge 1(x))$$

Formulae are coalgebras

$$\lambda \ : \ L \to (B \times L)^A$$

Formulae are coalgebras

$$\lambda \ : \ L \to (B \times L)^A$$

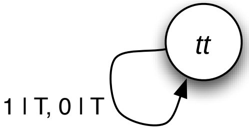
$$\phi ::= tt \quad | \quad \phi \wedge \phi \quad | \quad a(\phi) \quad | \quad a \downarrow b \quad | \quad x \quad | \quad \nu x.\psi$$

Formulae are coalgebras

$$\lambda \ : \ L \to (B \times L)^A$$

$$\phi ::= tt \quad | \quad \phi \wedge \phi \quad | \quad a(\phi) \quad | \quad a \downarrow b \quad | \quad x \quad | \quad \nu x.\psi$$

$$\lambda(tt)(a) = < \top, tt >$$

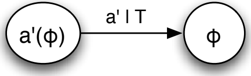


Formulae are coalgebras

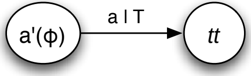
$$\lambda \;:\; L \rightarrow (B \times L)^A$$

$$\phi ::= tt \quad | \quad \phi \wedge \phi \quad | \quad a(\phi) \quad | \quad a \downarrow b \quad | \quad x \quad | \quad \nu x.\psi$$

$$\lambda(a'(\phi))(a) = \begin{cases} <\top, \phi> & a = a' \\ <\top, tt> & otherwise \end{cases}$$



or

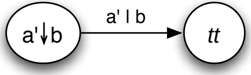


Formulae are coalgebras

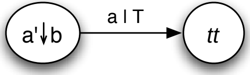
$$\lambda \; : \; L \rightarrow (B \times L)^A$$

$$\phi ::= tt \quad | \quad \phi \wedge \phi \quad | \quad a(\phi) \quad | \quad a \downarrow b \quad | \quad x \quad | \quad \nu x.\psi$$

$$\lambda(a' \downarrow b)(a) = \begin{cases} < b, tt > & a = a' \\ < \top, tt > & otherwise \end{cases}$$



or



Formulae are coalgebras

$$\lambda \ : \ L \to (B \times L)^A$$

$$\phi ::= tt \quad | \quad \phi \wedge \phi \quad | \quad a(\phi) \quad | \quad a \downarrow b \quad | \quad x \quad | \quad \nu x.\psi$$

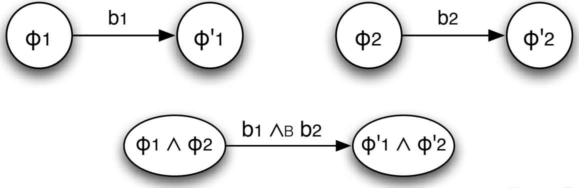
$$\lambda(\phi_1 \wedge \phi_2)(a) = \lambda(\phi_1)(a) \sqcap \lambda(\phi_2)(a)$$

Formulae are coalgebras

$$\lambda \ : \ L \to (B \times L)^A$$

$$\phi ::= tt \quad | \quad \phi \wedge \phi \quad | \quad a(\phi) \quad | \quad a \downarrow b \quad | \quad x \quad | \quad \nu x.\psi$$

$$\lambda(\phi_1 \wedge \phi_2)(a) = \lambda(\phi_1)(a) \sqcap \lambda(\phi_2)(a)$$

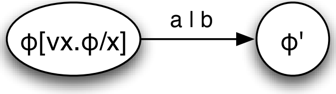


Formulae are coalgebras

$$\lambda \ : \ L \to (B \times L)^A$$

$$\phi ::= tt \quad | \quad \phi \wedge \phi \quad | \quad a(\phi) \quad | \quad a \downarrow b \quad | \quad x \quad | \quad \nu x.\psi$$

$$\lambda(\nu x.\phi)(a) = \lambda(\phi[\nu x.\phi/x])(a)$$

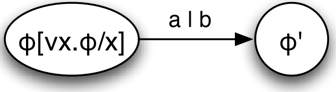


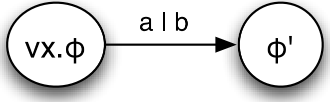
Formulae are coalgebras

$$\lambda \ : \ L \to (B \times L)^A$$

$$\phi ::= tt \quad | \quad \phi \wedge \phi \quad | \quad a(\phi) \quad | \quad a \downarrow b \quad | \quad x \quad | \quad \nu x.\psi$$

$$\lambda(\nu x.\phi)(a) = \lambda(\phi[\nu x.\phi/x])(a)$$





Why a coalgebra structure?

Two advantages

1. Semantics

$$\begin{tikzcd} L \arrow[r, "[\![\cdot]\!]"] \arrow[d, "\lambda"'] & \Gamma \arrow[d, "\gamma"] \\ (B \times L)^A \arrow[r] & (B \times \Gamma)^A \end{tikzcd}$$

2. Satisfaction relation in terms of simulation

$$s \models \phi \Leftrightarrow s \lesssim \phi$$

Why a coalgebra structure?

Two advantages

1. Semantics

$$\begin{tikzcd} L \arrow[r, "[\![\cdot]\!]"] \arrow[d, "\lambda"'] & \Gamma \arrow[d, "\gamma"] \\ (B \times L)^A \arrow[r] & (B \times \Gamma)^A \end{tikzcd}$$

2. Satisfaction relation in terms of simulation

$$s \models \phi \Leftrightarrow s \lesssim \phi$$

Why a coalgebra structure?

Two advantages

1. Semantics

$$\begin{tikzcd} L \arrow[r, "[\![\cdot]\!]"] \arrow[d, "\lambda"'] & \Gamma \arrow[d, "\gamma"] \\ (B \times L)^A \arrow[r] & (B \times \Gamma)^A \end{tikzcd}$$

2. Satisfaction relation in terms of simulation

$$s \models \phi \Leftrightarrow s \lesssim \phi$$

Logic is expressive

Theorem

1. *For all states s, s' of a Mealy machine (S, f),*

$$s \sim s' \text{ iff } \forall_{\phi \in L}. s \models \phi \Leftrightarrow s' \models \phi.$$

2. *If S is finite then there exists for any $s \in S$ a formula ϕ_s such that $s \sim \phi$.*

We also want:

For every formula ϕ construct a **finite** Mealy machine (S, f) such that $\exists_{s \in S}\ s \sim \phi$.

Logic is expressive

Theorem

1. *For all states s, s' of a Mealy machine (S, f),*

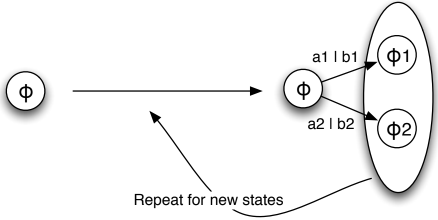
$$s \sim s' \text{ iff } \forall_{\phi \in L}. s \models \phi \Leftrightarrow s' \models \phi.$$

2. *If S is finite then there exists for any $s \in S$ a formula ϕ_s such that $s \sim \phi$.*

We also want:
For every formula ϕ construct a **finite** Mealy machine (S, f) such that $\exists_{s \in S}\ s \sim \phi$.

But. . .

Easy answer: Apply λ repeatedly!



- λ will not deliver a finite automata.

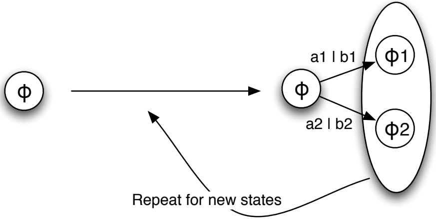
$$\phi = \nu x.1(x \wedge (\nu y.1(y)))$$

$$\lambda(\phi)(1) = < \top, \phi \wedge (\nu y.1(y)) >$$
$$\lambda(\phi \wedge (\nu y.1(y)))(1) = < \top, \phi \wedge (\nu y.1(y)) \wedge (\nu y.1(y)) >$$
$$\vdots$$

We need normalization!

But. . .

Easy answer: Apply λ repeatedly!



- λ will not deliver a finite automata.

$$\phi = \nu x.1(x \wedge (\nu y.1(y)))$$

$$\lambda(\phi)(1) = < \top, \phi \wedge (\nu y.1(y)) >$$
$$\lambda(\phi \wedge (\nu y.1(y)))(1) = < \top, \phi \wedge (\nu y.1(y)) \wedge (\nu y.1(y)) >$$
$$\vdots$$

We need normalization!

Normalization

$$
\begin{array}{lcl}
norm(tt) & = & tt \\
norm(a(\phi)) & = & a(norm(\phi)) \\
norm(a \downarrow b) & = & a \downarrow b \\
norm(\phi_1 \wedge \phi_2) & = & conj(rem(flatten(norm(\phi_1) \wedge norm(\phi_2)))) \\
norm(\nu x.\phi) & = & \nu x.norm(\phi)
\end{array}
$$

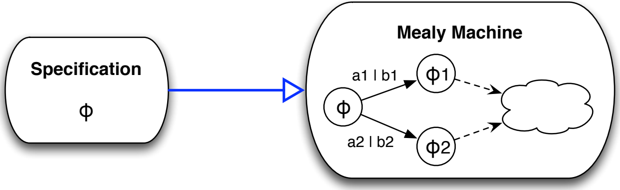
$$norm(a(tt) \wedge a \downarrow b \wedge tt \wedge a \downarrow b) = norm(a(tt) \wedge a \downarrow b)$$

Normalization

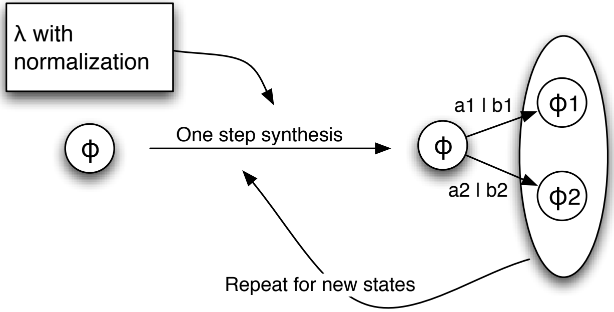
$$\begin{array}{lcl} norm(tt) & = & tt \\ norm(a(\phi)) & = & a(norm(\phi)) \\ norm(a \downarrow b) & = & a \downarrow b \\ norm(\phi_1 \wedge \phi_2) & = & conj(rem(flatten(norm(\phi_1) \wedge norm(\phi_2)))) \\ norm(\nu x.\phi) & = & \nu x.norm(\phi) \end{array}$$

$$norm(a(tt) \wedge a \downarrow b \wedge tt \wedge a \downarrow b) = norm(a(tt) \wedge a \downarrow b)$$

Synthesis



Synthesis


λ with
normalization
ϕ
One step synthesis
ϕ
a1 | b1
a2 | b2
ϕ1
ϕ2
Repeat for new states

One-step synthesis

$$
\begin{array}{lcl}
\delta & : & L \rightarrow (B \times L)^A \\
\delta(tt) & = & < \top, tt > \\
\delta(a'(\phi))(a) & = & \begin{cases} < \top, norm(\phi) > & a = a' \\ < \top, tt > & otherwise \end{cases} \\
\delta(a' \downarrow b)(a) & = & \begin{cases} < b, tt > & a = a' \\ < \top, tt > & otherwise \end{cases} \\
\delta(\phi_1 \wedge \phi_2)(a) & = & \delta(\phi_1)(a) \sqcap \delta(\phi_2)(a) \\
\delta(\nu x.\phi)(a) & = & < b, norm(\phi') > \\
 & & where < b, \phi' >= \delta(\phi[\nu x.\phi/x])(a)
\end{array}
$$

$$
< b_1, \phi_1 > \quad \sqcap \quad < b_2, \phi_2 > = < b_1 \wedge_B b_2, norm(\phi_1 \wedge \phi_2) >
$$

Examples

- $\phi = 1 \downarrow 0 \wedge (\nu x.1(x))$

$$
\begin{aligned}
\delta(\phi)(0) &= <\top, tt> \\
\delta(\phi)(1) &= \delta(1 \downarrow 0)(1) \sqcap \delta(\nu x.1(x))(1) \\
&= <0, tt> \sqcap <\top_B, \nu x.1(x)> \\
&= <0, \nu x.1(x)>
\end{aligned}
$$

Examples

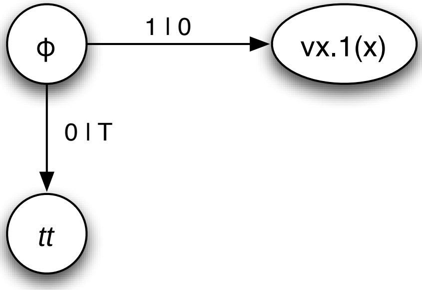
- $\phi = 1 \downarrow 0 \wedge (\nu x.1(x))$

$$
\begin{aligned}
\delta(\phi)(0) &= \; < \top, tt > \\
\delta(\phi)(1) &= \; \delta(1 \downarrow 0)(1) \sqcap \delta(\nu x.1(x))(1) \\
&= \; < 0, tt > \sqcap < \top_B, \nu x.1(x) > \\
&= \; < 0, \nu x.1(x) >
\end{aligned}
$$

Examples

- $\phi = 1 \downarrow 0 \wedge (\nu x.1(x))$

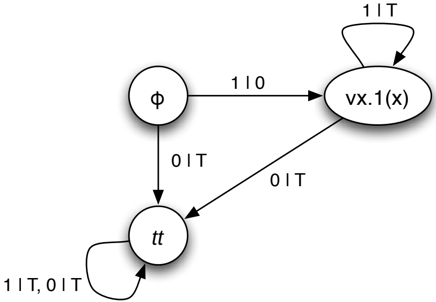
$$
\begin{array}{rcl}
\delta(\phi)(0) & = & < \top, tt > \\
\delta(\phi)(1) & = & \delta(1 \downarrow 0)(1) \sqcap \delta(\nu x.1(x))(1) \\
 & = & < 0, tt > \sqcap < \top_B, \nu x.1(x) > \\
 & = & < 0, \nu x.1(x) >
\end{array}
$$



Examples

- $\phi = 1 \downarrow 0 \wedge (\nu x.1(x))$

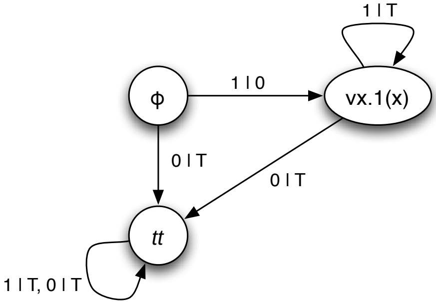
$$
\begin{aligned}
\delta(\phi)(0) &= <\top, tt> \\
\delta(\phi)(1) &= \delta(1 \downarrow 0)(1) \sqcap \delta(\nu x.1(x))(1) \\
&= <0, tt> \sqcap <\top_B, \nu x.1(x)> \\
&= <0, \nu x.1(x)>
\end{aligned}
$$



Examples

- $\phi = 1 \downarrow 0 \wedge (\nu x.1(x))$

$$\begin{array}{rcl} \delta(\phi)(0) & = & < \top, tt > \\ \delta(\phi)(1) & = & \delta(1 \downarrow 0)(1) \sqcap \delta(\nu x.1(x))(1) \\ & = & < 0, tt > \sqcap < \top_B, \nu x.1(x) > \\ & = & < 0, \nu x.1(x) > \end{array}$$



Remark: Not minimal.

Examples

- $\phi_2 = \nu x.(1(1 \downarrow 0) \wedge 1(x))$

$$
\begin{aligned}
\delta(\phi_2)(0) &= <\top, tt> \\
\delta(\phi_2)(1) &= \delta(1(1 \downarrow 0))(1) \sqcap \delta(\nu x.1(x))(1) \\
&= <\top, 1 \downarrow 0> \sqcap <\top, \nu x.1(x)> \\
&= <\top, 1 \downarrow 0 \wedge (\nu x.1(x))>
\end{aligned}
$$

Examples

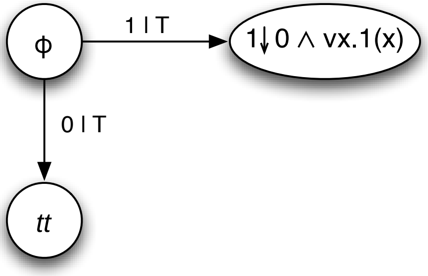
- $\phi_2 = \nu x.(1(1 \downarrow 0) \wedge 1(x))$

$$\begin{array}{rcl}
\delta(\phi_2)(0) & = & < \top, tt > \\
\delta(\phi_2)(1) & = & \delta(1(1 \downarrow 0))(1) \sqcap \delta(\nu x.1(x))(1) \\
& = & < \top, 1 \downarrow 0 > \sqcap < \top, \nu x.1(x) > \\
& = & < \top, 1 \downarrow 0 \wedge (\nu x.1(x)) >
\end{array}$$

Examples

- $\phi_2 = \nu x.(1(1 \downarrow 0) \wedge 1(x))$

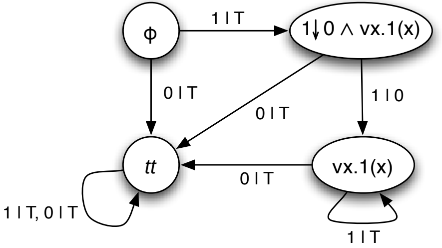
$$
\begin{array}{rcl}
\delta(\phi_2)(0) & = & < \top, tt > \\
\delta(\phi_2)(1) & = & \delta(1(1 \downarrow 0))(1) \sqcap \delta(\nu x.1(x))(1) \\
 & = & < \top, 1 \downarrow 0 > \sqcap < \top, \nu x.1(x) > \\
 & = & < \top, 1 \downarrow 0 \wedge (\nu x.1(x)) >
\end{array}
$$



Examples

- $\phi_2 = \nu x.(1(1 \downarrow 0) \wedge 1(x))$

$$
\begin{array}{rcl}
\delta(\phi_2)(0) & = & < \top, tt > \\
\delta(\phi_2)(1) & = & \delta(1(1 \downarrow 0))(1) \sqcap \delta(\nu x.1(x))(1) \\
& = & < \top, 1 \downarrow 0 > \sqcap < \top, \nu x.1(x) > \\
& = & < \top, 1 \downarrow 0 \wedge (\nu x.1(x)) >
\end{array}
$$



Examples

- $\phi_3 = \nu x.1(x \wedge (\nu y.1(y) \wedge 1 \downarrow 0))$

$$\begin{aligned}\delta(\phi_3)(0) &= <\top, tt> \\ \delta(\phi_3)(1) &= <\top, \phi_3 \wedge (\nu y.1(y) \wedge 1 \downarrow 0)>\end{aligned}$$

Examples

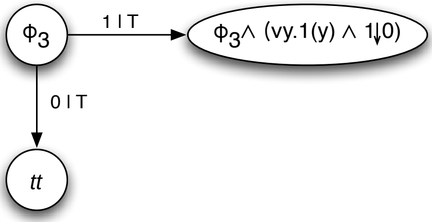
- $\phi_3 = \nu x.1(x \wedge (\nu y.1(y) \wedge 1 \downarrow 0))$

$$
\begin{array}{rcl}
\delta(\phi_3)(0) & = & < \top, tt > \\
\delta(\phi_3)(1) & = & < \top, \phi_3 \wedge (\nu y.1(y) \wedge 1 \downarrow 0) >
\end{array}
$$

Examples

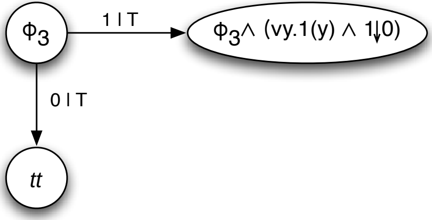
- $\phi_3 = \nu x.1(x \wedge (\nu y.1(y) \wedge 1 \downarrow 0))$

$$
\begin{aligned}
\delta(\phi_3)(0) &= \; <\top, tt> \\
\delta(\phi_3)(1) &= \; <\top, \phi_3 \wedge (\nu y.1(y) \wedge 1 \downarrow 0)>
\end{aligned}
$$



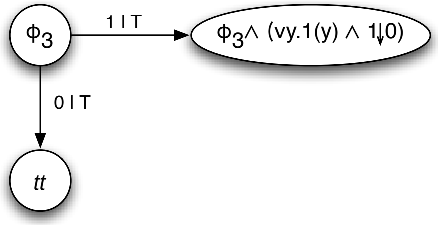
Examples (cont.)

$$\delta(\phi_3 \wedge (\nu y.1(y) \wedge 1 \downarrow 0))(1)$$



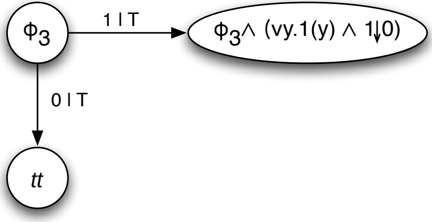
Examples (cont.)

$$\begin{array}{rl} & \delta(\phi_3 \wedge (\nu y.1(y) \wedge 1 \downarrow 0))(1) \\ = & \delta(\phi_3)(1) \sqcap \delta(\nu y.1(y) \wedge 1 \downarrow 0)(1) \end{array}$$



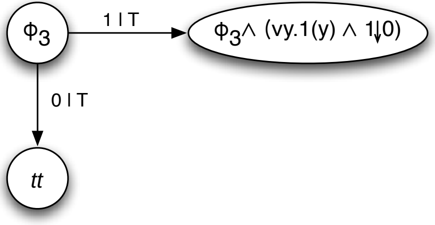
Examples (cont.)

$$\begin{array}{rl} & \delta(\phi_3 \wedge (\nu y.1(y) \wedge 1 \downarrow 0))(1) \\ = & \delta(\phi_3)(1) \sqcap \delta(\nu y.1(y) \wedge 1 \downarrow 0)(1) \\ = & < \top, \phi \wedge (\nu y.1(y) \wedge 1 \downarrow 0) > \sqcap < 0, \nu y.1(y) \wedge 1 \downarrow 0 > \end{array}$$



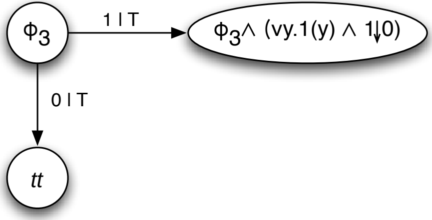
Examples (cont.)

$$\begin{array}{rl} & \delta(\phi_3 \wedge (\nu y.1(y) \wedge 1 \downarrow 0))(1) \\ = & \delta(\phi_3)(1) \sqcap \delta(\nu y.1(y) \wedge 1 \downarrow 0)(1) \\ = & < \top, \phi \wedge (\nu y.1(y) \wedge 1 \downarrow 0) > \sqcap < 0, \nu y.1(y) \wedge 1 \downarrow 0 > \\ = & < 0, \mathit{norm}(\phi_3 \wedge (\nu y.1(y) \wedge 1 \downarrow 0) \wedge (\nu y.1(y) \wedge 1 \downarrow 0)) > \end{array}$$



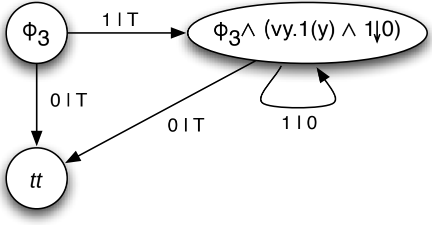
Examples (cont.)

$$
\begin{array}{rl}
& \delta(\phi_3 \wedge (\nu y.1(y) \wedge 1 \downarrow 0))(1) \\
= & \delta(\phi_3)(1) \sqcap \delta(\nu y.1(y) \wedge 1 \downarrow 0)(1) \\
= & < \top, \phi \wedge (\nu y.1(y) \wedge 1 \downarrow 0) > \sqcap < 0, \nu y.1(y) \wedge 1 \downarrow 0 > \\
= & < 0, norm(\phi_3 \wedge (\nu y.1(y) \wedge 1 \downarrow 0) \wedge (\nu y.1(y) \wedge 1 \downarrow 0)) > \\
= & < 0, \phi_3 \wedge (\nu y.1(y) \wedge 1 \downarrow 0) >
\end{array}
$$



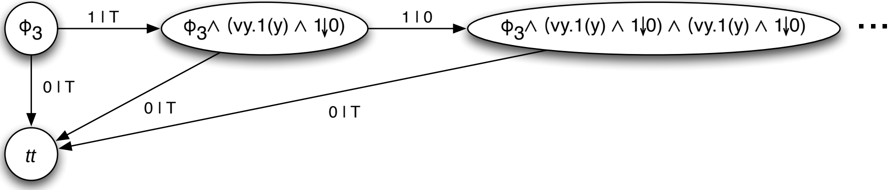
Examples (cont.)

$$
\begin{array}{rl}
& \delta(\phi_3 \wedge (\nu y.1(y) \wedge 1 \downarrow 0))(1) \\
= & \delta(\phi_3)(1) \sqcap \delta(\nu y.1(y) \wedge 1 \downarrow 0)(1) \\
= & < \top, \phi \wedge (\nu y.1(y) \wedge 1 \downarrow 0) > \sqcap < 0, \nu y.1(y) \wedge 1 \downarrow 0 > \\
= & < 0, \mathit{norm}(\phi_3 \wedge (\nu y.1(y) \wedge 1 \downarrow 0) \wedge (\nu y.1(y) \wedge 1 \downarrow 0)) > \\
= & < 0, \phi_3 \wedge (\nu y.1(y) \wedge 1 \downarrow 0) >
\end{array}
$$



Examples (cont.)

$$\begin{array}{rl} & \delta(\phi_3 \wedge (\nu y.1(y) \wedge 1 \downarrow 0))(1) \\ = & \delta(\phi_3)(1) \sqcap \delta(\nu y.1(y) \wedge 1 \downarrow 0)(1) \\ = & < \top, \phi \wedge (\nu y.1(y) \wedge 1 \downarrow 0) > \sqcap < 0, \nu y.1(y) \wedge 1 \downarrow 0 > \\ = & < 0, \mathit{norm}(\phi_3 \wedge (\nu y.1(y) \wedge 1 \downarrow 0) \wedge (\nu y.1(y) \wedge 1 \downarrow 0)) > \\ = & < 0, \phi_3 \wedge (\nu y.1(y) \wedge 1 \downarrow 0) > \end{array}$$



Conclusions (Part I)

- Coalgebraic approach : bisimulation and logics
- New logic for Mealy machines
- Synthesis algorithm that produces a finite machine

End of Part I

Part II

Generalizing

Specification Language + Synthesis for systems of type

$$S \to (B \times S)^A$$

Generalizing

Specification Language + Synthesis for systems of type

$$S \to GS$$

Generalizing

Specification Language + Synthesis for systems of type

$$S \to GS$$

$$G ::= Id \mid B \mid G \times G \mid G + G \mid G^A$$

B semilattice

Generalizing

Specification Language + Synthesis for systems of type

$$S \to GS$$

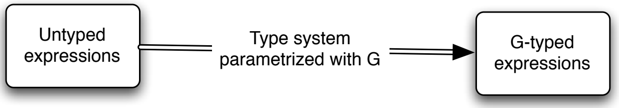
$$G ::= Id \mid B \mid G \times G \mid G + G \mid G^A$$

B semilattice

Notation: We will write $F \lhd G$ whenever F is an ingredient of G.

Consequences

- Everything will be parameterized in *G*
- Specification Language



Untyped expressions

$$\varepsilon \quad ::= \quad \emptyset \mid x \mid \varepsilon \oplus \varepsilon \mid \mu x.\gamma \mid b \mid l(\varepsilon) \mid r(\varepsilon) \mid l[\varepsilon] \mid r[\varepsilon] \mid a(\varepsilon)$$

Untyped expressions

$$\varepsilon \quad ::= \quad \emptyset \mid x \mid \varepsilon \oplus \varepsilon \mid \mu x.\gamma \mid b \mid l(\varepsilon) \mid r(\varepsilon) \mid l[\varepsilon] \mid r[\varepsilon] \mid a(\varepsilon)$$

B

Untyped expressions

$$\varepsilon \quad ::= \quad \emptyset \mid x \mid \varepsilon \oplus \varepsilon \mid \mu x.\gamma \mid b \mid l(\varepsilon) \mid r(\varepsilon) \mid l[\varepsilon] \mid r[\varepsilon] \mid a(\varepsilon)$$

GxG

Untyped expressions

$$\varepsilon \quad ::= \quad \emptyset \mid x \mid \varepsilon \oplus \varepsilon \mid \mu x.\gamma \mid b \mid l(\varepsilon) \mid r(\varepsilon) \mid \underbrace{l[\varepsilon] \mid r[\varepsilon]}_{\text{G+G}} \mid a(\varepsilon)$$

Untyped expressions

$$\varepsilon \quad ::= \quad \emptyset \mid x \mid \varepsilon \oplus \varepsilon \mid \mu x.\gamma \mid b \mid l(\varepsilon) \mid r(\varepsilon) \mid l[\varepsilon] \mid r[\varepsilon] \mid a(\varepsilon)$$

G^A

Type System

$$\frac{F \neq F_1 + F_2}{\vdash \emptyset \ : \ F \lhd G} \qquad \frac{\vdash \varepsilon_1 \ : \ F \lhd G \quad \vdash \varepsilon_2 \ : \ F \lhd G}{\vdash \varepsilon_1 \oplus \varepsilon_2 \ : \ F \lhd G} \qquad \frac{}{\vdash x \ : \ F \lhd G}$$

$$\frac{}{\vdash b \ : \ B \lhd G} \qquad \frac{\vdash \varepsilon \ : \ G \lhd G}{\vdash \varepsilon \ : \ Id \lhd G} \qquad \frac{\vdash \varepsilon \ : \ F_1 \lhd G}{\vdash l(\varepsilon) \ : \ F_1 \times F_2 \lhd G}$$

$$\frac{\vdash \varepsilon \ : \ F_1 \lhd G}{\vdash l[\varepsilon] \ : \ F_1 + F_2 \lhd G} \qquad \frac{\vdash \varepsilon \ : \ F_2 \lhd G}{\vdash r[\varepsilon] \ : \ F_1 + F_2 \lhd G} \qquad \frac{\vdash \varepsilon \ : \ F \lhd G}{\vdash a(\varepsilon) \ : \ F^A \lhd G}$$

Type System

$$\frac{F \neq F_1 + F_2}{\vdash \emptyset \ : \ F \lhd G} \qquad \frac{\vdash \varepsilon_1 \ : \ F \lhd G \quad \vdash \varepsilon_2 \ : \ F \lhd G}{\vdash \varepsilon_1 \oplus \varepsilon_2 \ : \ F \lhd G} \qquad \frac{}{\vdash x \ : \ F \lhd G}$$

$$\frac{}{\vdash b \ : \ B \lhd G} \qquad \frac{\vdash \varepsilon \ : \ G \lhd G}{\vdash \varepsilon \ : \ Id \lhd G} \qquad \frac{\vdash \varepsilon \ : \ F_1 \lhd G}{\vdash l(\varepsilon) \ : \ F_1 \times F_2 \lhd G}$$

$$\frac{\vdash \varepsilon \ : \ F_1 \lhd G}{\vdash l[\varepsilon] \ : \ F_1 + F_2 \lhd G} \qquad \frac{\vdash \varepsilon \ : \ F_2 \lhd G}{\vdash r[\varepsilon] \ : \ F_1 + F_2 \lhd G} \qquad \frac{\vdash \varepsilon \ : \ F \lhd G}{\vdash a(\varepsilon) \ : \ F^A \lhd G}$$

$$Exp_{F \lhd G} = \{\varepsilon \in Exp \ \mid \vdash \varepsilon \ : \ F \lhd G\}\,.$$

$Exp_G = Exp_{G \lhd G}$.

Exp_G has a coalgebra structure

Goal: $\lambda_G \ : \ Exp_G \to G(Exp_G)$

$$\lambda_{F \triangleleft G} \ : \ Exp_{F \triangleleft G} \to F(Exp_G)$$

Exp_G has a coalgebra structure

Goal: $\lambda_G \ : \ Exp_G \to G(Exp_G)$

$$\lambda_{F \lhd G} \ : \ Exp_{F \lhd G} \to F(Exp_G)$$

Exp_G has a coalgebra structure

Goal: $\lambda_G \; : \; Exp_G \to G(Exp_G)$

$$\lambda_{F \triangleleft G} \; : \; Exp_{F \triangleleft G} \to F(Exp_G)$$

$$\lambda_{F \triangleleft G}(\emptyset) = ?$$

Exp_G has a coalgebra structure

Goal: $\lambda_G \; : \; Exp_G \to G(Exp_G)$

$$\lambda_{F \lhd G} \; : \; Exp_{F \lhd G} \to F(Exp_G)$$

$$\lambda_{F \lhd G}(\emptyset) = ?$$

$$\lambda_{F \lhd G}(\emptyset) : F(Exp_G)$$

Exp_G has a coalgebra structure

Goal: $\lambda_G \; : \; Exp_G \rightarrow G(Exp_G)$

$$\lambda_{F \lhd G} \; : \; Exp_{F \lhd G} \rightarrow F(Exp_G)$$

$$\lambda_{F \lhd G}(\emptyset) = ?$$

$$\lambda_{F \lhd G}(\emptyset) : F(Exp_G)$$

$$\lambda_{F \lhd G}(\emptyset) = Empty_{F \lhd G} : F(Exp_G)$$

$Empty_{F \lhd G} \ : \ F(Exp_G)$

$$Empty_{Id \lhd G} \ : \ Exp_G$$
$$Empty_{Id \lhd G} = \emptyset$$

But recall:

$$\frac{F \neq F_1 + F_2}{\vdash \emptyset \ : \ F \lhd G}$$

So:

$$Empty_{Id \lhd G} \ : \ 1 + Exp_G$$
$$Empty_{Id \lhd G} = \begin{cases} \emptyset & G \neq G_1 + G_2 \\ \star & otherwise \end{cases}$$

Thus :

$$Empty_{F \lhd G} \ : \ F_\star(Exp_G), \qquad F_\star = 1 + F$$

$Empty_{F \lhd G} \; : \; F(Exp_G)$

$$Empty_{Id \lhd G} \; : \; Exp_G$$
$$Empty_{Id \lhd G} = \emptyset$$

But recall:

$$\frac{F \neq F_1 + F_2}{\vdash \emptyset \; : \; F \lhd G}$$

So:

$$Empty_{Id \lhd G} \; : \; 1 + Exp_G$$
$$Empty_{Id \lhd G} = \begin{cases} \emptyset & G \neq G_1 + G_2 \\ \star & otherwise \end{cases}$$

Thus :

$$Empty_{F \lhd G} \; : \; F_\star(Exp_G), \qquad F_\star = 1 + F$$

$Empty_{F \lhd G} \; : \; F(Exp_G)$

$$\begin{array}{l} Empty_{Id \lhd G} \; : \; Exp_G \\ Empty_{Id \lhd G} = \emptyset \end{array}$$

But recall:

$$\frac{F \neq F_1 + F_2}{\vdash \emptyset \; : \; F \lhd G}$$

So:

$$\begin{array}{l} Empty_{Id \lhd G} \; : \; 1 + Exp_G \\ Empty_{Id \lhd G} = \begin{cases} \emptyset & G \neq G_1 + G_2 \\ \star & otherwise \end{cases} \end{array}$$

Thus :

$$Empty_{F \lhd G} \; : \; F_\star(Exp_G), \qquad F_\star = 1 + F$$

$Empty_{F \lhd G} \;:\; F(Exp_G)$

$$
\begin{array}{l}
Empty_{Id \lhd G} \;:\; Exp_G \\
Empty_{Id \lhd G} = \emptyset
\end{array}
$$

But recall:

$$
\frac{F \neq F_1 + F_2}{\vdash \emptyset \;:\; F \lhd G}
$$

So:

$$
\begin{array}{l}
Empty_{Id \lhd G} \;:\; 1 + Exp_G \\
Empty_{Id \lhd G} = \begin{cases} \emptyset & G \neq G_1 + G_2 \\ \star & otherwise \end{cases}
\end{array}
$$

Thus :

$$
Empty_{F \lhd G} \;:\; F_\star(Exp_G), \qquad F_\star = 1 + F
$$

$\mathit{Empty}_{F \lhd G} \ : \ F_\star(\mathit{Exp}_G)$

$$
\begin{aligned}
&\mathit{Empty}_{Id \lhd G} = \begin{cases} \emptyset & G \neq G_1 + G_2 \\ \star & \mathit{otherwise} \end{cases} \\
&\mathit{Empty}_{B \lhd G} = \perp_B \\
&\mathit{Empty}_{F_1 \times F_2 \lhd G} = < \mathit{Empty}_{F_1 \lhd G}, \mathit{Empty}_{F_2 \lhd G} > \\
&\mathit{Empty}_{F_1 + F_2 \lhd G} = \star \\
&\mathit{Empty}_{F^A \lhd G} = \lambda a. \mathit{Empty}_{F \lhd G}
\end{aligned}
$$

$\lambda_{F \lhd G} \ : \ Exp_{F \lhd G} \rightarrow F_\star(Exp_G)$

$$
\begin{array}{l}
\lambda_{F \lhd G}(\emptyset) = Empty_{F \lhd G} \\
\lambda_{F_1 \times F_2 \lhd G}(l(\varepsilon)) = < \lambda_{F_1 \lhd G}(\varepsilon), Empty_{F_2 \lhd G} > \\
\lambda_{F_1 \times F_2 \lhd G}(r(\varepsilon)) = < Empty_{F_1 \lhd G}, \lambda_{F_2 \lhd G}(\varepsilon) > \\
\lambda_{F_1 + F_2 \lhd G}(l[\varepsilon]) = \kappa_1(\lambda_{F_1 \lhd G}(\varepsilon)) \\
\lambda_{F_1 + F_2 \lhd G}(r[\varepsilon]) = \kappa_2(\lambda_{F_2 \lhd G}(\varepsilon)) \\
\vdots
\end{array}
$$

Regular expressions

$$R_{F \lhd G} = \{\varepsilon \in \mathit{Exp}_{F \lhd G} \mid \lambda_{F \lhd G}(\varepsilon) \neq \star\}$$

Regular expressions

$$R_{F \lhd G} = \{\varepsilon \in Exp_{F \lhd G} \mid \lambda_{F \lhd G}(\varepsilon) \neq \star\}$$

Real coalgebra structure

$$\lambda_{F \lhd G} \ : \ R_{F \lhd G} \to F(R_{F \lhd G})$$

Regular expressions

$$R_{F \lhd G} = \{\varepsilon \in Exp_{F \lhd G} \mid \lambda_{F \lhd G}(\varepsilon) \neq \star\}$$

Real coalgebra structure

$$\lambda_{F \lhd G} \; : \; R_{F \lhd G} \to F(R_{F \lhd G})$$

Advantage: We can define a notion of satisfaction

$$s \not\models^{G} \varepsilon \Leftrightarrow \varepsilon \lesssim s$$

Language expressive w.r.t bisimulation

Theorem

Let G be a polynomial functor G and (S, g) a G-coalgebra.

1. *For all states $s, s' \in S$, $s \sim s'$ if and only if* $\forall \varepsilon \in R_G \,.\, s \models \varepsilon \Leftrightarrow s' \models \varepsilon$.
2. *If S is* finite *then there exists for any $s \in S$ an expression $\varepsilon_s \in R_G$ such that $\varepsilon_s \sim s$.*

Proof system

$$
\begin{array}{ll}
(\textit{Idempotency}) & \varepsilon \oplus \varepsilon = \varepsilon \\
(\textit{Commutativity}) & \varepsilon_1 \oplus \varepsilon_2 = \varepsilon_2 \oplus \varepsilon_1 \\
(\textit{Associativity}) & \varepsilon_1 \oplus (\varepsilon_2 \oplus \varepsilon_3) = (\varepsilon_1 \oplus \varepsilon_2) \oplus \varepsilon_3 \\
(\textit{Empty}) & \emptyset \oplus \varepsilon = \varepsilon
\end{array}
$$

Proof system (cont.)

$$(B-\emptyset) \qquad \emptyset = \bot_B \qquad (B-\oplus) \qquad b_1 \oplus b_2 = b_1 \vee_B b_2$$

$$(\times-\emptyset-L) \quad l(\emptyset)=\emptyset \qquad (\times-\oplus-L) \quad l(\varepsilon_1 \oplus \varepsilon_2) = l(\varepsilon_1) \oplus l(\varepsilon_2)$$
$$(\times-\emptyset-R) \quad r(\emptyset)=\emptyset \qquad (\times-\oplus-R) \quad r(\varepsilon_1 \oplus \varepsilon_2) = r(\varepsilon_1) \oplus r(\varepsilon_2)$$

$\cancel{l[\emptyset] = \emptyset}$ $\qquad$ $\cancel{r[\emptyset] = \emptyset}$

$$(+-\oplus-R) \quad r[\varepsilon_1 \oplus \varepsilon_2] = r[\varepsilon_1] \oplus r[\varepsilon_2]$$
$$(+-\oplus-L) \quad l[\varepsilon_1 \oplus \varepsilon_2] = l[\varepsilon_1] \oplus l[\varepsilon_2]$$

Proof system (cont.)

$$(B-\emptyset) \quad \emptyset = \bot_B \qquad (B-\oplus) \quad b_1 \oplus b_2 = b_1 \vee_B b_2$$

$$(\times-\emptyset-L) \quad l(\emptyset) = \emptyset \qquad (\times-\oplus-L) \quad l(\varepsilon_1 \oplus \varepsilon_2) = l(\varepsilon_1) \oplus l(\varepsilon_2)$$
$$(\times-\emptyset-R) \quad r(\emptyset) = \emptyset \qquad (\times-\oplus-R) \quad r(\varepsilon_1 \oplus \varepsilon_2) = r(\varepsilon_1) \oplus r(\varepsilon_2)$$

$$\cancel{l[\emptyset] = \emptyset} \qquad \cancel{r[\emptyset] = \emptyset}$$

$$(+-\oplus-R) \quad r[\varepsilon_1 \oplus \varepsilon_2] = r[\varepsilon_1] \oplus r[\varepsilon_2]$$
$$(+-\oplus-L) \quad l[\varepsilon_1 \oplus \varepsilon_2] = l[\varepsilon_1] \oplus l[\varepsilon_2]$$

Soundness

Theorem (Soundness)

The above equational system is sound, that is, for every polynomial functor G, if $\vdash^G \varepsilon_1 = \varepsilon_2$ *then* $\varepsilon_1 \sim_G \varepsilon_2$.

Completeness (modal fragment)

Theorem (Completeness)

For every polynomial functor G and modal regular expressions $\varepsilon_1, \varepsilon_2 \in R_G$, for all G-coalgebra (S, f), if $\varepsilon_1 \sim_G \varepsilon_2$ then $\vdash \varepsilon_1 = \varepsilon_2$.

Synthesis

- Normalization easily generalizes
- Synthesis is as before : $\lambda +$ *normalization*

Theorem (Kleene)

1. *If S is* finite *then there exists for any $s \in S$ an expression $\varepsilon_s \in R_G$ such that $\varepsilon_s \sim s$.*
2. *For all $\varepsilon \in R_G$, there exists a finite G-coalgebra (S, g) such that $\exists_{s \in S}\, s \sim \varepsilon$.*

Synthesis

- Normalization easily generalizes
- Synthesis is as before : $\lambda +$ *normalization*

Theorem (Kleene)

1. *If S is* finite *then there exists for any $s \in S$ an expression $\varepsilon_s \in R_G$ such that $\varepsilon_s \sim s$.*
2. *For all $\varepsilon \in R_G$, there exists a finite G-coalgebra (S, g) such that $\exists_{s \in S}\, s \sim \varepsilon$.*

Conclusions and Future work

Conclusions

- Language of regular expressions for polynomial coalgebras
- Equational system sound and complete (for the modal fragment)
- Generalization of Kleene theorem

Future work

- Move from **Set** to **pSet** : better definiton of regularity
- Enlarge the class of functors treated: add $\mathcal{P}$
- Completeness of the full logic
- Model checking

Conclusions and Future work

Conclusions

- Language of regular expressions for polynomial coalgebras
- Equational system sound and complete (for the modal fragment)
- Generalization of Kleene theorem

Future work

- Move from **Set** to **pSet** : better definiton of regularity
- Enlarge the class of functors treated: add $\mathcal{P}$
- Completeness of the full logic
- Model checking

End of Part II