Coalgebraic Logics and Synthesis for Mealy Machines

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

FoSSaCS 2008

Budapest, April 2008

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

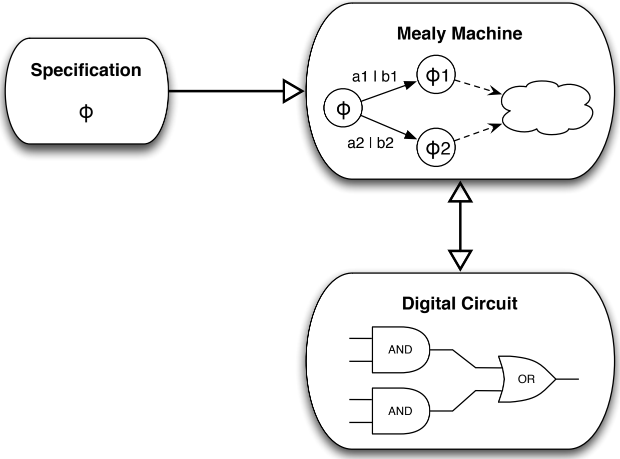
Design of Sequential systems

- (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

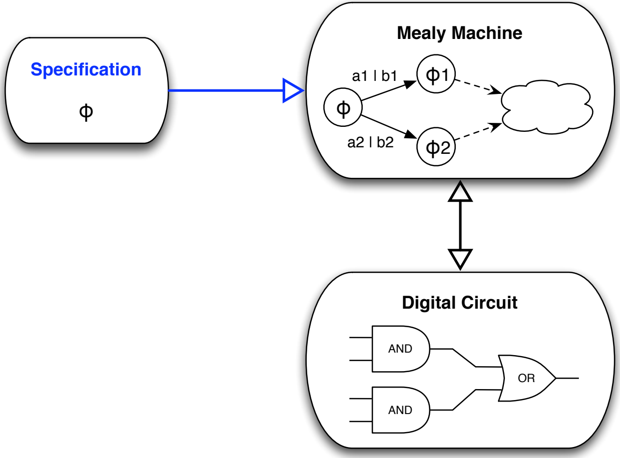
Source of ambiguities

- 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

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

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

A is the input alphabet and B is the output alphabet

- Binary Mealy machines : $A = 2$ and

$B = \mathcal{B} =$

But first...

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

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

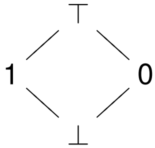
$$\begin{array}{lcl} f & : & S \rightarrow (B \times S)^A \\ f(s)(a) & = & < b, s' > \end{array}$$

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

  A is the input alphabet and B is the output alphabet

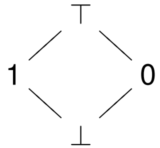
- Binary Mealy machines : $A = 2$ and

$B = \mathcal{B} =$



Information order

$\mathcal{B} =$



- $\top$ – *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

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

(Almost) for free:

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

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 Specification Language for Mealy machines

We derive a specification language for $M(X) = (B \times X)^A$:

A Specification Language for Mealy machines

We derive a specification language for $M(X) = (B \times X)^A$:

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

ψ guarded

A Specification Language for Mealy machines

We derive a specification language for $M(X) = (B \times X)^A$:

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

ψ guarded

A Specification Language for Mealy machines

We derive a specification language for $M(X) = (B \times X)^A$:

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

ψ guarded

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

A Specification Language for Mealy machines

We derive a specification language for $M(X) = (B \times X)^A$:

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

ψ guarded

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

And . . . What can we do now?

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

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

So far. . .

- Mealy machines are coalgebras
- Specification language for Mealy Machines
- But we want synthesis!
- Key observation : set of formulas has a coalgebraic structure!

So far. . .

- Mealy machines are coalgebras
- Specification language for Mealy Machines
- But we want synthesis!
- Key observation : set of formulas has a coalgebraic structure!

So far. . .

- Mealy machines are coalgebras
- Specification language for Mealy Machines
- But we want synthesis!
- Key observation : set of formulas has a coalgebraic structure!

So far...

- Mealy machines are coalgebras
- Specification language for Mealy Machines
- But we want synthesis!
- Key observation : set of formulas has a coalgebraic structure!

Formulae are coalgebras

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

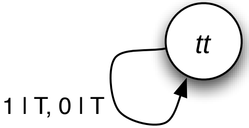
$$L \ni \phi ::= tt \quad | \quad \phi \wedge \phi \quad | \quad a(\phi) \quad | \quad a \downarrow b \quad | \quad x \quad | \quad \nu x.\psi$$

Formulae are coalgebras

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

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

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



Formulae are coalgebras

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

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

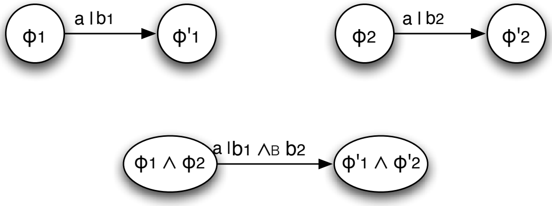
$$\lambda(\phi_1 \wedge \phi_2)(a) = \lambda(\phi_1)(a) \sqcap \lambda(\phi_2)(a)$$

Formulae are coalgebras

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

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

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

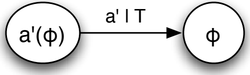


Formulae are coalgebras

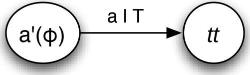
$$\lambda \; : \; L \to (B \times L)^A$$

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

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



or

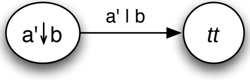


Formulae are coalgebras

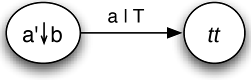
$$\lambda \ : \ L \to (B \times L)^A$$

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

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



or

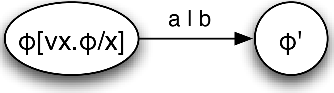


Formulae are coalgebras

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

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

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

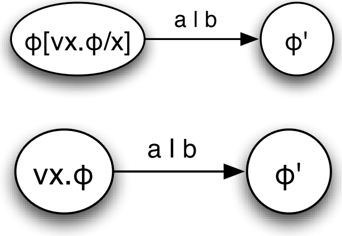


Formulae are coalgebras

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

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

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



Why a coalgebra structure?

Two advantages

1. Semantics

$$\begin{array}{ccc} L & \xrightarrow{[\![\cdot]\!]} & \Gamma \\ \lambda \downarrow & & \downarrow \gamma \\ (B \times L)^A & \longrightarrow & (B \times \Gamma)^A \end{array}$$

2. Satisfaction relation in terms of simulation

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

Why a coalgebra structure?

Two advantages

1. Semantics

$$\begin{array}{ccc} L & \xrightarrow{[\![\cdot]\!]} & \Gamma \\ \lambda \downarrow & & \downarrow \gamma \\ (B \times L)^A & \longrightarrow & (B \times \Gamma)^A \end{array}$$

2. Satisfaction relation in terms of simulation

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

Why a coalgebra structure?

Two advantages

1. Semantics

$$\begin{array}{ccc} L & \xrightarrow{[\![\cdot]\!]} & \Gamma \\ \lambda \downarrow & & \downarrow \gamma \\ (B \times L)^A & \longrightarrow & (B \times \Gamma)^A \end{array}$$

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 synthesis:

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),*

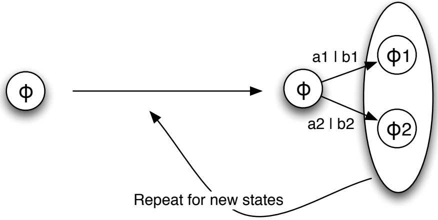
$$s \sim s' \text{ iff } \forall_{\phi \in L}. s \models \phi \Leftrightarrow s' \models \phi.$$

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 synthesis:
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 automaton.

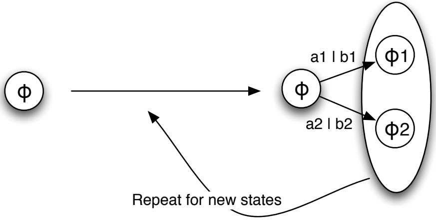
$$\phi = \nu x.1(x \wedge (\nu y.1(y)))$$

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

We need normalization!

But. . .

Easy answer: Apply λ repeatedly!



- λ will not deliver a finite automaton.

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

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

We need normalization!

Normalization

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

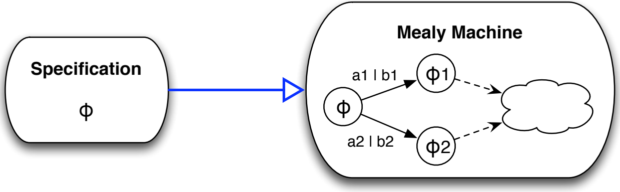
$$norm(a(tt) \wedge a \downarrow b \wedge tt \wedge a \downarrow b) = norm(a(tt) \wedge a \downarrow b)$$

Normalization

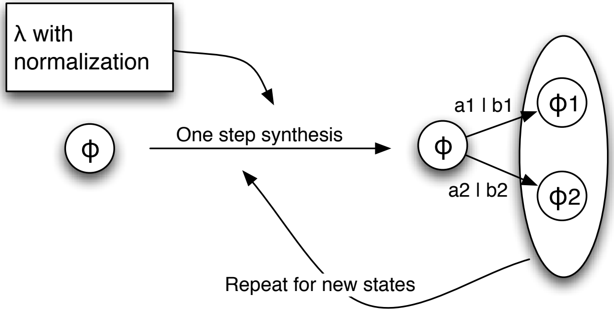
$$
\begin{array}{lcl}
norm(tt) & = & tt \\
norm(a(\phi)) & = & a(norm(\phi)) \\
norm(a \downarrow b) & = & a \downarrow b \\
norm(\phi_1 \wedge \phi_2) & = & conj(rem(flatten(norm(\phi_1) \wedge norm(\phi_2)))) \\
norm(\nu x.\phi) & = & \nu x.norm(\phi)
\end{array}
$$

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

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

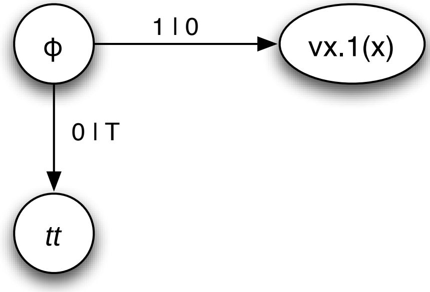
- $\phi = 1 \downarrow 0 \wedge (\nu x.1(x))$

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

Examples

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

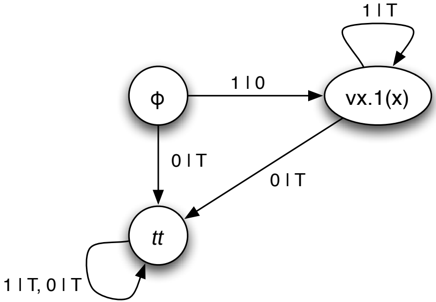
$$
\begin{aligned}
\delta(\phi)(0) &= <\top, tt> \\
\delta(\phi)(1) &= \delta(1 \downarrow 0)(1) \sqcap \delta(\nu x.1(x))(1) \\
&= <0, tt> \sqcap <\top_B, \nu x.1(x)> \\
&= <0, \nu x.1(x)>
\end{aligned}
$$



Examples

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

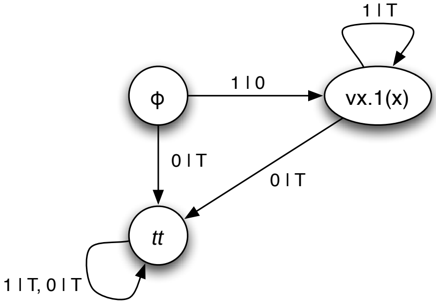
$$\begin{aligned}
\delta(\phi)(0) &= \ < \top, tt > \\
\delta(\phi)(1) &= \ \delta(1 \downarrow 0)(1) \sqcap \delta(\nu x.1(x))(1) \\
&= \ < 0, tt > \sqcap < \top_B, \nu x.1(x) > \\
&= \ < 0, \nu x.1(x) >
\end{aligned}$$



Examples

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

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



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

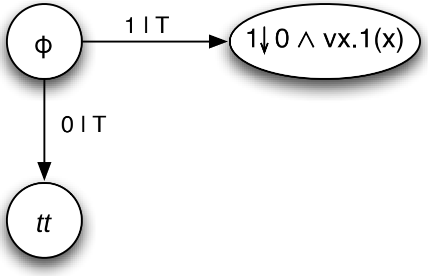
- $\phi_2 = \nu x.(1(1 \downarrow 0) \wedge 1(x))$

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

Examples

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

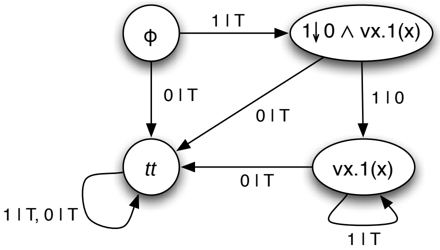
$$
\begin{array}{rcl}
\delta(\phi_2)(0) & = & < \top, tt > \\
\delta(\phi_2)(1) & = & \delta(1(1 \downarrow 0))(1) \sqcap \delta(\nu x.1(x))(1) \\
& = & < \top, 1 \downarrow 0 > \sqcap < \top, \nu x.1(x) > \\
& = & < \top, 1 \downarrow 0 \wedge (\nu x.1(x)) >
\end{array}
$$



Examples

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

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



Examples

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

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

Examples

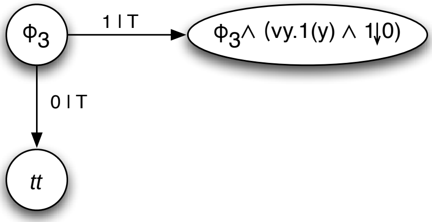
- $\phi_3 = \nu x.1(x \wedge (\nu y.1(y) \wedge 1 \downarrow 0))$

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

Examples

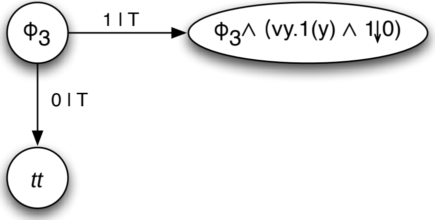
- $\phi_3 = \nu x.1(x \wedge (\nu y.1(y) \wedge 1 \downarrow 0))$

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



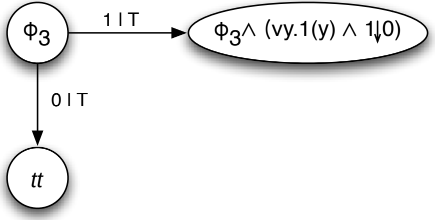
Examples (cont.)

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



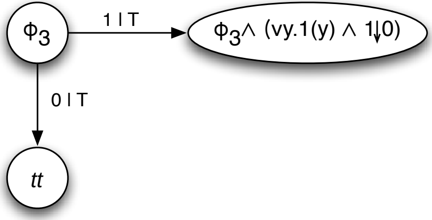
Examples (cont.)

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



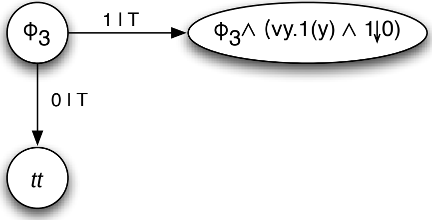
Examples (cont.)

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



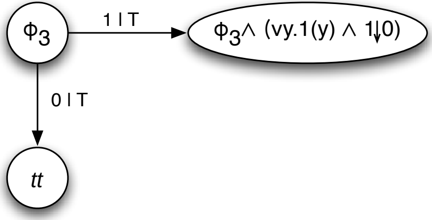
Examples (cont.)

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



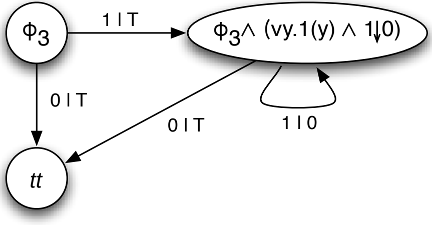
Examples (cont.)

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



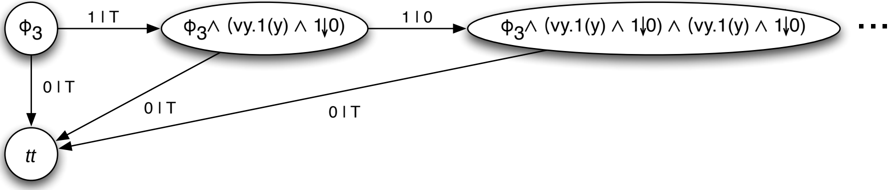
Examples (cont.)

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



Examples (cont.)

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



Conclusions

Conclusions

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

Future Work

- The logic has exactly the operators needed to represent all Mealy machines
- Similar to regular expressions for deterministic automata
- Can we generalize this approach to other types of coalgebras?

Towards Regular expressions for polynomial coalgebras

Conclusions

Conclusions

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

Future Work

- The logic has exactly the operators needed to represent all Mealy machines
- Similar to regular expressions for deterministic automata
- Can we generalize this approach to other types of coalgebras?

Towards Regular expressions for polynomial coalgebras