(Concrete) Coalgebraic logics and synthesis of 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

A|C seminar, May 2007

Motivation

- Design of hardware circuits
- English is not a very precise language
- We need a more precise description

Motivation

- Design of hardware circuits

Typical (simple) properties

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

- English is not a very precise language
- We need a more precise description

Motivation

- Design of hardware circuits
- English is not a very precise language
- We need a more precise description

Motivation

- Design of hardware circuits
- English is not a very precise language

Ambiguities

Does *Output 0 at the each input of 1* mean that when the input is 0 you should output 1?

- We need a more precise description

Motivation

- Design of hardware circuits
- English is not a very precise language
- We need a more precise description

What will we show?

We will show . . .

- . . . mealy machines as coalgebras
- . . . the notion of bisimulation
- . . . a logic for Mealy machines
- . . . satisfaction relation vs bisimulation
- . . . semantics of formulas
- . . . translation from mealy machines to formulas
- . . . synthesis of Mealy machines

Mealy Machines – Basic definitions/facts

Mealy machine = set of states S + transition function f

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

- A is the input alphabet
- B is the output alphabet and we require B boolean algebra (**Why?**)

Some notation

Graphical representation:

$$f(s)(a) = < b, s' > \Leftrightarrow \; s \xrightarrow{a|b} s'$$

Splitting f into components:

$$f(s) = < s[a], s_a >$$

($s[a]$ is the *(initial) output on input a* and s_a the *next state on input a*)

Some notation

Graphical representation:

$$f(s)(a) = < b, s' > \Leftrightarrow \; s \xrightarrow{a|b} s'$$

Splitting f into components:

$$f(s) = < s[a], s_a >$$

($s[a]$ is the *(initial) output on input a* and s_a the *next state on input a*)

A first basic example

A=B=2 (binary machine)

$$\begin{tikzcd} s_1 \arrow[loop above, "0|0"] \arrow[r, "1|1"] & s_2 \arrow[loop above, "1|0{,}0|1"] \end{tikzcd}$$

(This machine calculates the two’s complement of a binary number)

Mealy automata are coalgebras

Observation:

A Mealy machine is a coalgebra of the functor

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

Bisimulation

Definition (Bisimulation for Mealy)

Let (S, f) and (T, g) be two Mealy machines. We call a relation $R \subseteq S \times T$ a *bisimulation* if for all $(s, t) \in S \times T$ and $a \in A$

$$s \, R \, t \quad \Rightarrow \quad (\, s[a] = t[a] \;\text{ and }\; s_a \, R \, t_a \,)$$

Example

$$\begin{tikzcd} s_1 \arrow[loop left, "0|0"] \arrow[r, "1|1"] & s_2 \arrow[r, "1|0{,}0|1"] & s_3 \arrow[loop right, "1|0{,}0|1"] \end{tikzcd}$$

- s_2 and s_3 are bisimilar
- Bisimulation is very important for minimization

Final coalgebra

Recall:

- $A^\omega = \{ \sigma \mid \sigma : \{0, 1, 2, \ldots\} \to A \}$
- $a : \sigma = (a, \sigma(0), \sigma(1), \sigma(2), \ldots)$
- $\sigma' = (\sigma(1), \sigma(2), \sigma(3), \ldots)$
- $f : A^\omega \to B^\omega$ *causal*

Final coalgebra

Now, define:

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

and $\gamma(f)(a) =< f[a], f_a >$, with:

$$f[a] = f(a : \sigma)(0) \qquad f_a(\sigma) = f(a : \sigma)'$$

- Γ is a Mealy coalgebra…
- and it is final, meaning:

$$\begin{tikzcd} S \arrow[r, "\exists! h"] \arrow[d, "\alpha"'] & \Gamma \arrow[d, "\gamma"] \\ (B \times S)^A \arrow[r, "(id \times h)^A"'] & (B \times \Gamma)^A \end{tikzcd}$$

- $(h(S), \gamma)$ is the minimization of (S, α)

Final coalgebra

Now, define:

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

and $\gamma(f)(a) =< f[a], f_a >$, with:

$$f[a] = f(a:\sigma)(0) \qquad f_a(\sigma) = f(a:\sigma)'$$

- Γ is a Mealy coalgebra...
- and it is final, meaning:

$$\begin{tikzcd} S \arrow[r, "\exists! h"] \arrow[d, "\alpha"'] & \Gamma \arrow[d, "\gamma"] \\ (B \times S)^A \arrow[r, "(id \times h)^A"'] & (B \times \Gamma)^A \end{tikzcd}$$

- $(h(S), \gamma)$ is the minimization of (S, α)

Final coalgebra

Now, define:

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

and $\gamma(f)(a) = < f[a], f_a >$, with:

$$f[a] = f(a : \sigma)(0) \qquad f_a(\sigma) = f(a : \sigma)'$$

- Γ is a Mealy coalgebra...
- and it is final, meaning:

$$\begin{tikzcd} S \arrow[r, "\exists ! h"] \arrow[d, "\alpha"'] & \Gamma \arrow[d, "\gamma"] \\ (B \times S)^A \arrow[r, "(id \times h)^A"'] & (B \times \Gamma)^A \end{tikzcd}$$

- $(h(S), \gamma)$ is the minimization of (S, α)

End of preliminaries

A coalgebraic logic for Mealy machines

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

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

A coalgebraic logic for Mealy machines

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

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

Examples

- Output 0 at the each input of 1

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

- Output 0 at the 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

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

Examples

- Output 0 at the each input of 1

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

- Output 0 at the 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

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

Examples

- Output 0 at the each input of 1

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

- Output 0 at the 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

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

Examples

- Output 0 at the each input of 1

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

- Output 0 at the 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

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

Examples

- Output 0 at the each input of 1

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

- Output 0 at the 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

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

Examples

- Output 0 at the each input of 1

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

- Output 0 at the 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

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

Satisfaction relation

$$
\begin{array}{lll}
s \models_\eta tt & & \text{for all } s \\
s \models_\eta a(\phi) & \textit{iff} & s_a \models_\eta \phi \\
s \models_\eta a \downarrow b & \textit{iff} & s[a] = b \\
s \models_\eta \phi_1 \wedge \phi_2 & \textit{iff} & s \models_\eta \phi_1 \textit{ and } s \models_\eta \phi_2 \\
s \models_\eta \neg\phi & \textit{iff} & s \not\models_\eta \phi \\
s \models_\eta x & \textit{iff} & s \in \eta(x) \\
s \models_\eta \nu x.\phi & \textit{iff} & \exists T \subseteq S. s \in T \textit{ and } \forall t \in T. t \models_{\eta[T/x]} \phi
\end{array}
$$

$\models$ coincides with $\sim$

Theorem

The above logic is expressive for bisimulation, that is, for all states s, s' of a Mealy automaton (S, f) with S finite

$$s \sim s' \quad \textit{iff} \quad \forall \phi.\, s \models_\eta \phi \Leftrightarrow s' \models_\eta \phi$$

Proof (sketch)

$(\Rightarrow)$

By induction on ϕ.

$(\Leftarrow)$

$R = \{(s_w, s'_w) \mid w \in A^*\}$ is a bisimulation.

Formulas are Mealy coalgebras

$$L \xrightarrow[\alpha]{} (B \times L)^A$$

$$\alpha(\phi) = < \phi[a], \phi_a >$$

We define *initial output* and *derivative* for formulas.

Semantics

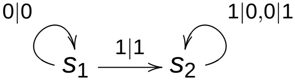
The Mealy coalgebra structure on L provides us (by finality) a natural semantics:

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

$$\llbracket \phi \rrbracket [a] = \phi[a] \quad \text{and} \quad \llbracket \phi \rrbracket_a = \llbracket \phi_a \rrbracket$$

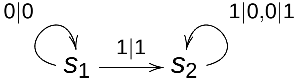
It assigns to every formula ϕ a causal stream function $\llbracket \phi \rrbracket : A^\omega \to B^\omega$.

From Mealy to Formulas



We can easily prove that $s_1 \models \phi_1$ and $s_2 \models \phi_2$.

From Mealy to Formulas



We can easily prove that $s_1 \models \phi_1$ and $s_2 \models \phi_2$.

Synthesis

Fragment from initial logic:

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

And now the idea is:

Formula $\phi \longrightarrow$ Normalized formula $\longrightarrow$ Mealy machine

Synthesis

Fragment from initial logic:

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

And now the idea is:

Formula $\phi \longrightarrow$ Normalized formula $\longrightarrow$ Mealy machine

Normalized formulas

$$\phi ::= \left(\bigwedge_I a_i(\phi_i) \wedge a_i \downarrow b_i \right) \wedge \left(\bigwedge_K x_k \right) \wedge \left(\bigwedge_L \nu x_l . \phi_l \right) .$$

with ϕ_l guarded.

Normalization process

1. Make the formula guarded (Vardi)
2. Replace single occurrences of $a(\phi)$ by $a(\phi) \wedge a \downarrow \top$
3. Replace single occurrences of $a \downarrow b$ by $a(tt) \wedge a \downarrow b$

One-step synthesis

Non-recursive part

$$\delta_1(\bigwedge_I a_i(\phi_i) \wedge a_i \downarrow b_i, a)$$

$$= \begin{cases} (< \bigwedge_M b_m, \bigwedge_M \phi_m >) & \exists_{M \subseteq I} \{a_m \mid m \in M\} = \{a\} \\ < \top, tt > & otherwise \end{cases}$$

Recursive part

$$\delta_2(\bigwedge_L \nu x_l.\phi_l, a) = \begin{cases} < \top, tt > & L = \emptyset \\ \bigwedge_L \delta(\phi_l[\nu x_l.\phi_l / x_l])(a) & otherwise \end{cases}$$

One-step synthesis

Non-recursive part

$$\delta_1(\bigwedge_I a_i(\phi_i) \wedge a_i \downarrow b_i, a)$$

$$= \begin{cases} (< \bigwedge_M b_m, \bigwedge_M \phi_m >) & \exists_{M \subseteq I} \{a_m \mid m \in M\} = \{a\} \\ < \top, tt > & \textit{otherwise} \end{cases}$$

Recursive part

$$\delta_2(\bigwedge_L \nu x_l.\phi_l, a) = \begin{cases} < \top, tt > & L = \emptyset \\ \bigwedge_L \delta(\phi_l[\nu x_l.\phi_l / x_l])(a) & \textit{otherwise} \end{cases}$$

One-step synthesis

Non-recursive part

$$\delta_1(\bigwedge_I a_i(\phi_i) \wedge a_i \downarrow b_i, a)$$

$$= \begin{cases} (< \bigwedge_M b_m, \bigwedge_M \phi_m >) & \exists_{M \subseteq I} \{a_m \mid m \in M\} = \{a\} \\ < \top, tt > & otherwise \end{cases}$$

Recursive part

$$\delta_2(\bigwedge_L \nu x_l.\phi_l, a) = \begin{cases} < \top, tt > & L = \emptyset \\ \bigwedge_L \delta(\phi_l[\nu x_l.\phi_l / x_l])(a) & otherwise \end{cases}$$

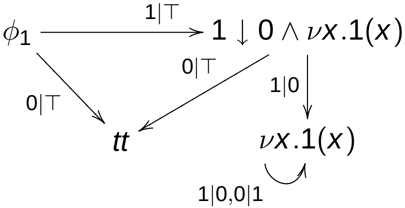
Examples

- $\phi_1 = 1(1 \downarrow 0) \wedge (\nu x.1(x))$

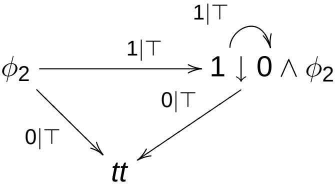
- $\phi_2 = \nu x.(1(1 \downarrow 0) \wedge 1(x))$

Examples

- $\phi_1 = 1(1 \downarrow 0) \wedge (\nu x.1(x))$



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



Conclusions and future work

Conclusions

- Coalgebraic approach : bisimulation and logics
- New logic for Mealy machines
- Synthesis algorithm

Future work

- Make the logic more *user-friendly* (syntatic sugar)
- Study more expressive logics : non-deterministic Mealy automata
- Calculate complexity of the synthesis algorithm

Conclusions and future work

Conclusions

- Coalgebraic approach : bisimulation and logics
- New logic for Mealy machines
- Synthesis algorithm

Future work

- Make the logic more *user-friendly* (syntatic sugar)
- Study more expressive logics : non-deterministic Mealy automata
- Calculate complexity of the synthesis algorithm