Synthesis of Mealy Machines Using Derivatives

Helle Hvid Hansen\textsuperscript{1,2} \quad David Costa\textsuperscript{2} \quad Jan Rutten\textsuperscript{1,2}

\textsuperscript{1}Section Theoretical Computer Science  
Vrije Universiteit Amsterdam (VUA)

\textsuperscript{2}SEN3  
Centrum voor Wiskunde en Informatica (CWI)

March 25, 2006 / Coalgebraic Methods in Computer Science
Main reference:

- Mealy machines are coalgebras.
- Abstract implementation via final Mealy coalgebra.
- Symbolic computation of derivatives.

Our contributions:
- Implementation (Haskell),
- Decision procedure for deciding equivalence of expressions,
- Results on rational 2-adic stream functions
  - automaton size,
  - complexity.
Outline

1. Mealy Machines and Coalgebras
2. Synthesis Using Derivatives (I)
   - Main Idea
3. Arithmetic Bitstream Specifications
   - 2-Adic Bitstream Algebra
   - Mod-2 Bitstream Algebra
4. Synthesis Using Derivatives (II)
   - Deciding Equivalence
5. Rational 2-Adic Functions
6. Mealy Synthesis Tool (incl. Demo)
Mealy Machines

- Finite State Machine (FSM): states, I/O-transitions:

![Diagram](attachment: FSM_diagram.png)

In state $s$, reading input $a$, the machine produces an output $b$, and moves to a new state $t$.

- Example:
Mealy Coalgebras

- **Mealy coalgebra (input in $A$, output in $B$):**
  
  Set-functor $M(X) = (B \times X)^A$

  $\delta : S \rightarrow (B \times S)^A$

  $s \mapsto (o_s, d_s)$

  $o : S \times A \rightarrow B$ is the output function,

  $d : S \times A \rightarrow Q$ is the next-state function.

- **Mealy homomorphism** $h : (S, \delta) \rightarrow (U, \rho)$:

  for all $s, t \in S$:

  $s \xrightarrow{a|b} t$ iff $h(s) \xrightarrow{a|b} h(t)$.

- **Mealy machine** $(S, \delta, s_0) = \text{(finite) Mealy coalgebra} \ + \ \text{initial state.}$
**Causal Behaviour**

- **Behaviour of** $(S, \delta, s_0)$: deterministic transformation of input stream to output stream (transducer).

- Behaviour of a state $s_0$ in $(S, \delta)$: $\text{Beh}(s_0) : A^\omega \rightarrow B^\omega$.

  For $\sigma = (a_0, a_1, a_2, \ldots) \in A^\omega$, $\text{Beh}(s_0)(\sigma) = (b_0, b_1, b_2, \ldots) \in B^\omega$

  where

  $S_0 \xrightarrow{a_0|b_0} S_1 \xrightarrow{a_1|b_1} \ldots \xrightarrow{a_{k-1}|b_{k-1}} S_k \xrightarrow{a_k|b_k} S_{k+1} \ldots$

  $\text{Beh}(s_0) : A^\omega \rightarrow B^\omega$ is causal ($b_n$ depends only on $a_0, \ldots, a_n$).
Let \( \Gamma = \{ f : A^\omega \rightarrow B^\omega \mid f \text{ causal} \} \).

Theorem[Rutten]: \( \Gamma \) carries a final Mealy coalgebra structure.

For \( f \in \Gamma \) define \( \pi(f) = \langle o_f, d_f \rangle : A \rightarrow B \times \Gamma \) as

- initial output of \( f \) (on input \( a \)):
  \[
o_f(a) := f[a] = f(a : \sigma)(0) = \text{head } f(a : \sigma)
  \]

- (stream function) derivative of \( f \) (on input \( a \)):
  \[
d_f(a) := f_a : A^\omega \rightarrow B^\omega
  \]
  \[
  \sigma \mapsto f(a : \sigma)' = \text{tail } f(a : \sigma)
  \]
Abstract Implementations

- States \( s, t \) in \((S, \delta)\) are equivalent \((s \sim t)\) iff \(\text{Beh}(s) = \text{Beh}(t)\).
- Mealy machine \((S, \delta, s_0)\) implements \(f : A^\omega \to B^\omega\) iff \(\text{Beh}(s_0) = f\).

Generating Abstract Implementations

Let \(\langle f \rangle\) be the subcoalgebra generated by \(f\) in \((\Gamma, \pi)\);
i.e., \(f \in \langle f \rangle \subseteq \Gamma\) is minimal, derivative-closed.
Then by finality of \((\Gamma, \pi)\):
- \(\langle f \rangle\) implements \(f\). (Because \(\text{Beh}(f) = f\).)
- \(\langle f \rangle\) is a minimal-state implementation.
  If \((S, s)\) implements \(f\), then \(\text{Beh}[\langle s \rangle] = \langle f \rangle\).
Synthesis of Mealy Machines Using Derivatives

Basic Idea:
Symbolic computation of $\langle f \rangle$.
- Specify causal stream functions algebraically (language $\mathcal{L}$).
- Define output and derivatives of expressions (cf. Brzozowski).

$$\mathcal{L} \xrightarrow{\langle o,d \rangle} (B \times \mathcal{L})^A.$$  

- $\theta \in \mathcal{L}$ defines/specifies a function $Beh(\theta) = f_\theta : A^\omega \rightarrow B^\omega$.
- Compute representation of $\langle f_\theta \rangle$ as least fixpoint.

Questions:
- Realisability: is $\langle f \rangle$ finite? *Rational bitstream functions.*
- Equivalence: how to decide $\theta \sim \phi$? *Normal forms.*
Synthesis of Mealy Machines Using Derivatives

Basic Idea:
Symbolic computation of \( \langle f \rangle \).
- Specify causal stream functions algebraically (language \( \mathcal{L} \)).
- Define output and derivatives of expressions (cf. Brzozowski).

\[
\mathcal{L} \xrightarrow{\langle o,d \rangle} (B \times \mathcal{L})^A.
\]

- \( \theta \in \mathcal{L} \) defines/specifies a function \( \text{Beh}(\theta) = f_\theta : A^\omega \rightarrow B^\omega \).
- Compute representation of \( \langle f_\theta \rangle \) as least fixpoint.

Questions:
- Realisability: is \( \langle f \rangle \) finite? *Rational bitstream functions.*
- Equivalence: how to decide \( \theta \sim \phi \)? *Normal forms.*
Outline

1. Mealy Machines and Coalgebras
2. Synthesis Using Derivatives (I)
   - Main Idea
3. Arithmetic Bitstream Specifications
   - 2-Adic Bitstream Algebra
   - Mod-2 Bitstream Algebra
4. Synthesis Using Derivatives (II)
   - Deciding Equivalence
5. Rational 2-Adic Functions
6. Mealy Synthesis Tool (incl. Demo)
Arithmetic Bitstream Specifications

Semantic Domain:
- Bitstreams $2^\omega$ (i.e. $A = B = 2 = \{0, 1\}$),
- Bitstream functions $f : 2^\omega \rightarrow 2^\omega$.

Arithmetic Bitstream Expressions:
- Constants $[0] = (0, 0, 0, 0, \ldots)$,
  $[1] = (1, 0, 0, 0, \ldots)$,
  $X = (0, 1, 0, 0, \ldots)$,
  $X^n = (0, \ldots, 0, 1, 0, 0, \ldots)$;
- Bitstream variable $\sigma$;
- Arithmetic operations: addition, multiplication, minus, division.
  - 2-Adic: infinitary binary arithmetic.
  - Mod-2: infinitary modulo-2 arithmetic.
Stream Behaviour of Expressions

Mimic semantics:

Recall in final Mealy coalgebra:

$$o_f(a) = f(a : \sigma)(0)$$

$$d_f(a) = f(a : \sigma)'$$

– Instantiating $\sigma$ with bit $a \in 2$ (later),
– Stream behaviour (head, tail).

Stream behaviour of constants:

<table>
<thead>
<tr>
<th>initial value (head)</th>
<th>derivative (tail)</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="0">0</a> = 0</td>
<td>[0]' = [0]</td>
</tr>
<tr>
<td><a href="0">1</a> = 1</td>
<td>[1]' = [0]</td>
</tr>
<tr>
<td>X(0) = 0</td>
<td>X' = [1]</td>
</tr>
<tr>
<td>X^n(0) = 0</td>
<td>(X^n)' = X^{n-1}, n \geq 1, (X^0 = [1])</td>
</tr>
</tbody>
</table>
Mimic semantics:

Recall in final Mealy coalgebra:

\[ o_f(a) = f(a : \sigma)(0) \]
\[ d_f(a) = f(a : \sigma)' \]

– Instantiating \( \sigma \) with bit \( a \in 2 \) (later),
– Stream behaviour (head, tail).

Stream behaviour of constants:

<table>
<thead>
<tr>
<th>initial value (head)</th>
<th>derivative (tail)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(<a href="0">0</a> = 0)</td>
<td>([0]' = [0])</td>
</tr>
<tr>
<td>(<a href="0">1</a> = 1)</td>
<td>([1]' = [0])</td>
</tr>
<tr>
<td>(X(0) = 0)</td>
<td>(X' = [1])</td>
</tr>
<tr>
<td>(X^n(0) = 0)</td>
<td>((X^n)' = X^{n-1}, n \geq 1, (X^0 = [1]))</td>
</tr>
</tbody>
</table>
2-Adic Bitstream Algebra

$2^\omega$ as the 2-adic integers:

$$(a_0, a_1, a_2, \ldots) \in 2^\omega \iff a_0 + a_1 2 + a_2 2^2 + a_3 2^3 + \ldots = \sum_{i=0}^{\infty} a_i 2^i.$$

2-adic operations (signature $\Sigma_{2adic} = \{+\times,-,\}/\}$):

- infinitary binary addition (carry-bits propagate indefinitely):
  
  E.g. $(1, 0, 0, \ldots) + (1, 1, 1, \ldots) = (0, 0, 0, \ldots)$.

- stream differential equation:

$$(\alpha + \beta)(0) = \alpha(0) \oplus \beta(0)$$

$$(\alpha + \beta)' = (\alpha' + \beta') + [\alpha(0) \land \beta(0)]$$
2-Adic Bitstream Algebra

2\omega as the 2-adic integers:

\((a_0, a_1, a_2, \ldots) \in 2\omega \iff a_0 + a_1 2 + a_2 2^2 + a_3 2^3 + \ldots = \sum_{i=0}^{\infty} a_i 2^i.\)

2-adic operations (signature \(\Sigma_{2adic} = \{+, \times, -, /\}\)):

Cauchy product:

\((\alpha \times \beta)(n) = \sum_{i=0}^{n} \alpha(i) \land \beta(n-i) \quad (2\text{-adic sum}).\)

Stream differential equation:

\begin{align*}
(\alpha \times \beta)(0) &= \alpha(0) \land \beta(0) \\
(\alpha \times \beta)' &= \alpha' \times \beta + [\alpha(0)] \times \beta'
\end{align*}
2-Adic Bitstream Algebra

$2^{\omega}$ as the 2-adic integers:

$$(a_0, a_1, a_2, \ldots) \in 2^{\omega} \iff \sum_{i=0}^{\infty} a_i 2^i = a_0 + a_1 2 + a_2 2^2 + a_3 2^3 + \ldots.$$)

2-adic operations (signature $\Sigma_{2adic} = \{+, \times, -, /\}$):

infinitary two’s complement:

E.g. $-(1, 0, 0, \ldots) = (1, 1, 1, \ldots)$.

stream differential equation:

$$(-\alpha)(0) = \alpha(0)$$
$$(-\alpha)' = -\alpha' + [\alpha(0)]$$
2-Adic Bitstream Algebra

2^\omega as the 2-adic integers:

\[(a_0, a_1, a_2, \ldots) \in 2^\omega \iff a_0 + a_12 + a_22^2 + a_32^3 + \ldots = \sum_{i=0}^{\infty} a_i 2^i.\]

2-adic operations (signature \(\Sigma_{2adic} = \{+, \times, -, /\}\)):

multiplicative inverse: \(\beta \times (1/\beta) = [1]\) (well-def’d if \(\beta(0) = 1\)).

division:

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

stream differential equation:

\[\frac{d}{dt} \left(\frac{\alpha}{\beta}\right)(0) = \frac{\alpha(0)}{\beta(0)}\]
\[\frac{d}{dt} \left(\frac{\alpha}{\beta}\right) = \frac{\alpha' - [\alpha(0)] \times \beta'}{\beta}\]
2-Adic Bitstream Algebra

2ω as the 2-adic integers:

\[(a_0, a_1, a_2, \ldots) \in 2^{\omega} \iff a_0 + a_1 2 + a_2 2^2 + a_3 2^3 + \ldots = \sum_{i=0}^{\infty} a_i 2^i.\]

2-adic operations (signature \(\Sigma_{2adic} = \{+, \times, -, /\}\)):

- \((2^{\omega}, +, \times, -, /, [0], [1])\) is integral domain, (i.e. commutative ring with no zero divisors).
- Since \(X = (0, 1, 0, 0, \ldots) \iff 2\), we have for \(\alpha \in 2^{\omega}:\)
  - \(\alpha + \alpha \sim X \times \alpha,\)
  - \(X^n + X^n \sim X^{n+1}.\)
Mod-2 Bitstream Algebra

$2^\omega$ as formal power series over $\mathbb{Mod}_2$-ring $(2, \oplus, \land, id, 0, 1)$:

$$(a_0, a_1, a_2, \ldots) \in 2^\omega \iff \sum_{n=0}^{\infty} a_n x^n$$
Mod-2 Bitstream Algebra

$2^\omega$ as formal power series over $\mathbb{Mod}_2$-ring $(2, \oplus, \wedge, id, 0, 1)$:

$$(a_0, a_1, a_2, \ldots) \in 2^\omega \iff \sum_{n=0}^{\infty} a_n x^n$$

Mod-2 operations (signature $\Sigma_{mod2} = \{\oplus, \otimes, \ominus, \oslash\}$):

- bitwise mod-2 sum: $(\alpha \oplus \beta)(n) = \alpha(n) \oplus \beta(n)$,
- cauchy product w.r.t. mod-2 sum.
- minus $= id$: $\ominus \alpha = \alpha$ (since $\alpha \oplus \alpha = (0, 0, 0, \ldots)$),
- multiplicative inverse: $1 \oslash \alpha$
- fraction: $\alpha \oslash \beta = \alpha \otimes (1 \oslash \beta)$ (well-def’d when $\alpha(0) = 1$).
Mod-2 Bitstream Algebra

$2^\omega$ as formal power series over \( \mathbb{Mod2}\text{-}\text{ring} (2, \oplus, \wedge, id, 0, 1) \):

\[
(a_0, a_1, a_2, \ldots) \in 2^\omega \iff \sum_{n=0}^{\infty} a_n x^n
\]

Mod-2 operations (signature \( \Sigma_{mod2} = \{\oplus, \otimes, \ominus, \oslash\} \)):

- \((2^\omega, \oplus, \otimes, \ominus, \oslash, [0], [1]) \) is integral domain in which \( \alpha \oplus \alpha \sim [0] \) (nilpotent).
- operations definable using stream differential equations.
Outline

1. Mealy Machines and Coalgebras
2. Synthesis Using Derivatives (I)
   - Main Idea
3. Arithmetic Bitstream Specifications
   - 2-Adic Bitstream Algebra
   - Mod-2 Bitstream Algebra
4. Synthesis Using Derivatives (II)
   - Deciding Equivalence
5. Rational 2-Adic Functions
6. Mealy Synthesis Tool (incl. Demo)
Compute output and derivative on input \( a \in 2 \):

**Instantiating with bit \( a \in 2 \):**

Recall in final Mealy coalgebra:

\[
\begin{align*}
o_f(a) &= f(a : \sigma)(0) \\
d_f(a) &= f(a : \sigma)'
\end{align*}
\]

We have for all \( \alpha \in 2^\omega \):

\[
\begin{align*}
0 : \alpha &= X \times \alpha = X \otimes \alpha \\
1 : \alpha &= [1] + X \times \alpha = [1] \oplus X \otimes \alpha
\end{align*}
\]

For \( \theta(\sigma) \in \Sigma_{2\text{adic}}(\sigma) \):

\[
\begin{align*}
o_\theta(0) &= (\theta[X \times \sigma])(0) \\
o_\theta(1) &= (\theta[[1] + X \times \sigma])(0) \\
d_\theta(0) &= (\theta[X \times \sigma])' \\
d_\theta(1) &= (\theta[[1] + X \times \sigma])'
\end{align*}
\]

Similarly, for \( \theta \in \Sigma_{\text{mod}_2}(\sigma) \).
Deciding Equivalence of Expressions

Normal forms in integral domains:

Normal form of $\theta \in \text{Expr}_\Sigma(\sigma)$ is $\text{NF}(\theta) = \frac{p}{q}$ where $p, q$ in distributive polynomial normal form (PNF).

Reducing Constant Coefficients:

- **2-Adic:** Use numeric interpretation: signed, binary expansion. E.g.

  $$X \times ([1] - X^2) \iff 2(1 - 4) = -6$$

  $$\text{NF}(X \times ([1] - X^2)) = -(X + X^2).$$

- **Mod-2:** Use ring laws and nilpotency: E.g.

  $$X \otimes ([1] \ominus X^2) \oplus X = X \ominus X^3 \oplus X = X^3.$$
Deciding Equivalence of Expressions

Normal forms in integral domains:

Normal form of $\theta \in \text{Expr}_\Sigma(\sigma)$ is $\text{NF}(\theta) = \frac{p}{q}$ where $p, q$ in distributive polynomial normal form (PNF).

Example

$$\theta = \frac{[1]}{[1] + X} + \sigma + [1] \leadsto \frac{([1] + [1] + X) + ([1] + X) \cdot \sigma}{[1] + X}$$

- Normal form in $2$-adic algebra: $X^2 + \frac{([1] + X) \times \sigma}{[1] + X}$
- Normal form in mod-$2$ algebra: $X \oplus \frac{([1] \oplus X) \otimes \sigma}{[1] + X}$
Deciding Equivalence of Expressions

Equivalence in integral domains:

Given two expressions in normal form, $\theta_1 = \frac{p_1}{q_1}$ and $\theta_2 = \frac{p_2}{q_2}$,

$$\theta_1 \sim \theta_2 \text{ iff } \frac{p_1}{q_1} \sim \frac{p_2}{q_2}$$

$$\text{iff } PNF(p_1 \times q_2) = PNF(p_2 \times q_1)$$

Complexity:

- computing $PNF(\theta)$ is exponential: $2^{O(|\theta|)}$.
- deciding $\theta \sim \phi$ is exponential: $2^{O(|\theta| + |\phi|)}$. 
Synthesis Example: $\theta(\sigma) = \frac{6 \times \sigma}{9}$

\[
\theta(\sigma) = \frac{(X + X^2) \times \sigma}{[1] + X^3} \quad \sim \sim \sim \quad \frac{6 \times \sigma}{9}
\]

\[
\theta = \frac{6\sigma}{9} = \frac{(X+X^2) \times (\sigma)}{1+X^3}
\]

\[
\theta_1 = \frac{3+6\sigma}{9} = \frac{(1+X)+(X+X^2) \times (\sigma)}{1+X^3}
\]

\[
\theta_{10} = \frac{-3+6\sigma}{9} = \frac{-(1+X)+(X+X^2) \times (\sigma)}{1+X^3}
\]

\[
\theta_{100} = \frac{-6+6\sigma}{9} = \frac{-(X+X^2)+(X+X^2) \times (\sigma)}{1+X^3}
\]

\[
\theta_{1001} = \frac{6+6\sigma}{9} = \frac{(X+X^2)+(X+X^2) \times (\sigma)}{1+X^3}
\]
Rational 2-adic function:

\[ f(\sigma) = \frac{n}{m} \times \sigma \]

where \( n, m \) are constant, polynomial (i.e. integers), \( m \) odd.

Example specification:

\[ \theta(\sigma) = \frac{1 - X^2}{1 - X + X^3} \times \sigma \left( -\frac{3}{7} \times \sigma \right) \]

Note: In \( NF(\theta) = \frac{p \times \sigma}{q} \), denominator \( q \) is constant

- all derivatives have denominator \( q \),
- equivalence of derivatives in linear time,
- (syntactic equality of numerators).
Rational 2-Adic Automaton Size and Complexity

Theorem:

For rational 2-adic stream function $f(\sigma) = \frac{n}{m} \times \sigma$,

$$\text{AutSize} \left( \frac{n}{m} \times \sigma \right) \leq \begin{cases} \frac{|n| + |m|}{\gcd(n, m)} - 1 & \text{if } \frac{n}{m} > 0 \\ \frac{|n| + |m|}{\gcd(n, m)} & \text{if } \frac{n}{m} \leq 0 \end{cases}$$

Conjecture (based on experiments): Upper bound is also lower bound.

Corollary: A Mealy machine implementing a rational 2-adic specification $\theta(\sigma) = \frac{n}{m} \times \sigma$ can be constructed in time $2^{O(|\theta|)}$. 
Mealy Synthesis Tool (Haskell)

Input specifications $\theta$ in:
- 2-adic arithmetic
- mod-2 arithmetic

Output:
- LaTeX-document: automaton states and transitions,
- DOT-file: graph representation of automaton

Source code, executable, documentation:
http://www.cwi.nl/~costa/diffcal
Tool Demo
Related Work

Derivative Constructions

- Raney (1958): stream function derivative (semantics only).
- Brzozowski (1964): DFA from regex.
- Redziejowski (1999): \(\omega\)-DFA from \(\omega\)-regex (related to Safra).
Related Work

Logic Synthesis

- Büchi\&Landweber (1969): S1S.
- Pnueli\&Rosner (1989): LTL.
- Kupferman\&Vardi (2004):
  - distributed synthesis from LTL and CTL specs.
  - imperfect information.

- Uses theory of automata, alternation, games:
  - determinization,
  - constructive non-emptiness test.
Concluding

Summary

- Mealy synthesis using derivatives: direct symbolic construction of deterministic automaton.
- 2-Adic and Mod-2 equivalence is decidable.
- Synthesis for rational (2-adic) functions:
  - EXPTIME in size $|\theta|$ of specification $\theta$,
  - Automaton size.
- Haskell tool available.

Future work:

- Extend approach to more general types of transducers and stream functions,
- Closer investigation of logic vs coalgebraic synthesis,
- Develop coinductive stream languages.