Behavioural differential equations and coinduction for binary trees

An exercise on coalgebraic reasoning

Alexandra Silva

[1]CWI, The Netherlands

ACG, October 2006

Motivation

- Previous work by Jan:

  Behavioural differential equations: a coinductive calculus of streams, automata, and power series

  Elements of stream calculus (an extensive exercise in coinduction)

  showed that coinduction and behavioural differential equations are effective for stream calculus
- We want to investigate if the same approach is effective for other infinite structures, *e.g.* infinite binary trees

What will we show?

We will show how to...

- ... define infinite binary trees coalgebraically
- ... define bisimulations for infinite binary trees
- ... develop a calculus for binary trees *à la formal power series*
- ... define infinite binary trees through *behavioural differential equations*
- ... calculate closed expressions for infinite binary trees

Binary trees coalgebraically

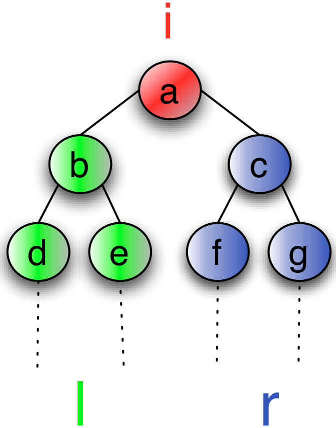
Final coalgebra for $FX = X \times A \times X$:

$$T_A \xrightarrow{<l,i,r>} T_A \times A \times T_A$$

Binary trees coalgebraically

Final coalgebra for $FX = X \times A \times X$:

$$T_A \xrightarrow{<l,i,r>} T_A \times A \times T_A$$



Definition principle

$< l, i, r >$ constitutes a final coalgebra structure on the set T_A.

$$\Downarrow$$

$$\left\{ \begin{array}{lcl} i(f(x)) & = & \cdots \\ l(f(x)) & = & \cdots \\ r(f(x)) & = & \cdots \end{array} \right.$$ has a unique solution.

Definition principle

$< l, i, r >$ constitutes a final coalgebra structure on the set T_A.

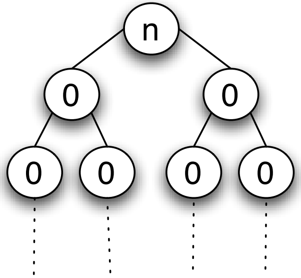
$$\Downarrow$$

$$\left\{ \begin{array}{lcl} i(f(x)) & = & \cdots \\ l(f(x)) & = & \cdots \\ r(f(x)) & = & \cdots \end{array} \right. \text{ has a unique solution.}$$

Examples

$$const : \mathbb{R} \longrightarrow T_{\mathbb{R}}$$

n $\longmapsto$

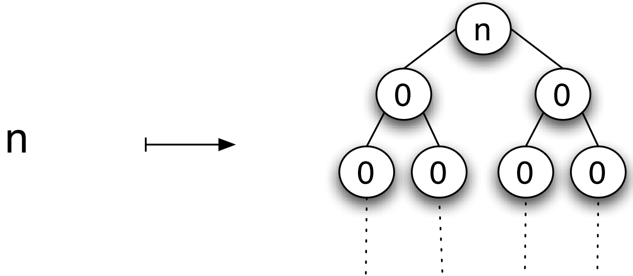


is totally defined by

$$
\begin{aligned}
i(const(n)) &= n \\
l(const(n)) &= r(const(n)) = const(0)
\end{aligned}
$$

Examples

$$const : \mathbb{R} \longrightarrow T_{\mathbb{R}}$$



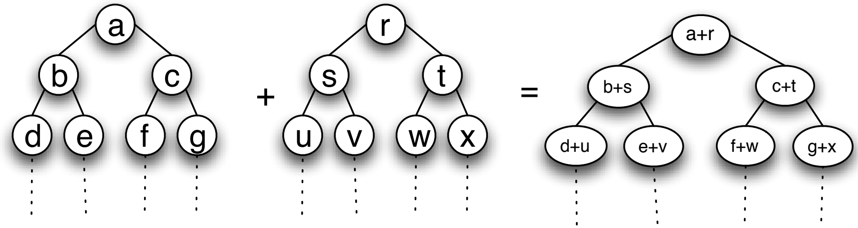
is totally defined by

$$\begin{aligned} i(const(n)) &= n \\ l(const(n)) &= r(const(n)) = const(0) \end{aligned}$$

Examples

The operation

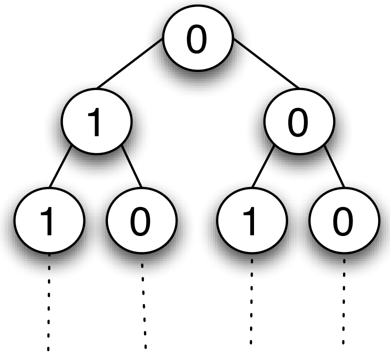
$$+ : \quad T_{\mathbb{R}} \quad \times \quad T_{\mathbb{R}} \longrightarrow T_{\mathbb{R}}$$



is totally determined by

$$\begin{aligned} i(\sigma + \tau) &= i(\sigma) + i(\tau) \\ l(\sigma + \tau) &= l(\sigma) + l(\tau) \\ r(\sigma + \tau) &= r(\sigma) + r(\tau) \end{aligned}$$

Examples



$$\begin{aligned} i(\sigma) &= 0 \\ l(\sigma) &= \sigma + const(1) \\ r(\sigma) &= \sigma \end{aligned}$$

Bisimulation and coinduction

Define:
A bisimulation on T_A is a relation $R \subseteq T_A \times T_A$ such that for every $(\sigma, \tau) \in R$:

1. $i(\sigma) = i(\tau)$
2. $(r(\sigma), r(\tau)) \in R$
3. $(l(\sigma), l(\tau)) \in R$

Theorem (Coinduction)

For all trees σ and τ in T_A if $\sigma \sim \tau$ then $\sigma = \tau$

In order to prove the equality of two trees σ and τ is enough to establish the existence of a bisimulation R s.t. $(\sigma, \tau) \in R$.

Bisimulation and coinduction

Define:
A bisimulation on T_A is a relation $R \subseteq T_A \times T_A$ such that for every $(\sigma, \tau) \in R$:

1. $i(\sigma) = i(\tau)$
2. $(r(\sigma), r(\tau)) \in R$
3. $(l(\sigma), l(\tau)) \in R$

Theorem (Coinduction)

For all trees σ and τ in T_A if $\sigma \sim \tau$ then $\sigma = \tau$

In order to prove the equality of two trees σ and τ is enough to establish the existence of a bisimulation R s.t. $(\sigma, \tau) \in R$.

Bisimulation and coinduction

Define:
A bisimulation on T_A is a relation $R \subseteq T_A \times T_A$ such that for every $(\sigma, \tau) \in R$:

1. $i(\sigma) = i(\tau)$
2. $(r(\sigma), r(\tau)) \in R$
3. $(l(\sigma), l(\tau)) \in R$

Theorem (Coinduction)

For all trees σ and τ in T_A if $\sigma \sim \tau$ then $\sigma = \tau$

In order to prove the equality of two trees σ and τ is enough to establish the existence of a bisimulation R s.t. $(\sigma, \tau) \in R$.

Examples

First, let us prove that:

$$const(n_1 + n_2) = const(n_1) + const(n_2),\ n_1, n_2 \in \mathbb{R}$$

Examples

$$f \text{ linear} \Rightarrow map_f(\sigma + \tau) = map_f(\sigma) + map_f(\tau)$$

where

- f linear $\Rightarrow f(x + y) = f(x) + f(y)$
- map_f is defined as

$$\begin{aligned} i(map_f(\sigma)) &= f(i(\sigma)) \\ l(map_f(\sigma)) &= map_f(l(\sigma)) \\ r(map_f(\sigma)) &= map_f(r(\sigma)) \end{aligned}$$

Examples

$$f\ linear \Rightarrow map_f(\sigma + \tau) = map_f(\sigma) + map_f(\tau)$$

where

- $f\ linear \Rightarrow f(x+y) = f(x) + f(y)$
- map_f is defined as

$$\begin{aligned} i(map_f(\sigma)) &= f(i(\sigma)) \\ l(map_f(\sigma)) &= map_f(l(\sigma)) \\ r(map_f(\sigma)) &= map_f(r(\sigma)) \end{aligned}$$

Examples

$$f \textit{ linear} \Rightarrow \textit{map}_f(\sigma + \tau) = \textit{map}_f(\sigma) + \textit{map}_f(\tau)$$

where

- $f \textit{ linear} \Rightarrow f(x + y) = f(x) + f(y)$
- $\textit{map}_f$ is defined as

$$\begin{array}{rcl} i(\textit{map}_f(\sigma)) & = & f(i(\sigma)) \\ l(\textit{map}_f(\sigma)) & = & \textit{map}_f(l(\sigma)) \\ r(\textit{map}_f(\sigma)) & = & \textit{map}_f(r(\sigma)) \end{array}$$

Formal power series

Recall: A formal power series is a function $\sigma : X^* \to k$ where X is the set of variables (or input symbols) and k is a semiring.

For A semiring, the set T_A is a formal power series over X=2 (**Why?**), *i.e*,

$$T_A = \{\sigma | \sigma : 2^* \to A\}$$

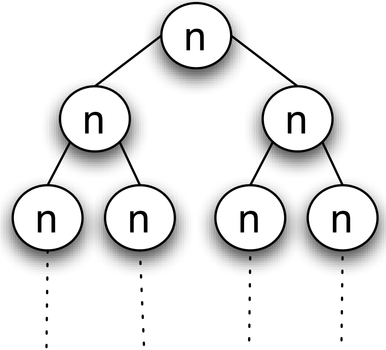
Behavioural Differential Equations

The formal definition of $\sigma \in T_A$ is now expressed in terms of a *behavioural differential equation*.

$$
\begin{array}{rcll}
\sigma(\varepsilon) & = & c & \textit{initial value} \\
\sigma_L & = & \textit{left\_exp} & \textit{left derivative} \\
\sigma_R & = & \textit{right\_exp} & \textit{right derivative}
\end{array}
$$

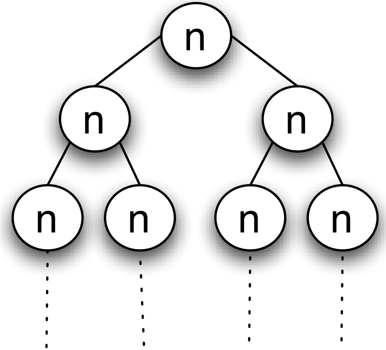
- $2 = \{L, R\}$

Examples



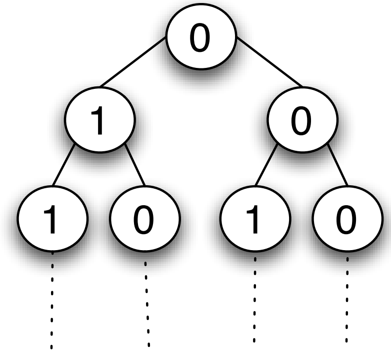
$$\sigma(\varepsilon) = n$$
$$\sigma_L = \sigma$$
$$\sigma_R = \sigma$$

Examples



$$\sigma(\varepsilon) = n$$
$$\sigma_L = \sigma$$
$$\sigma_R = \sigma$$

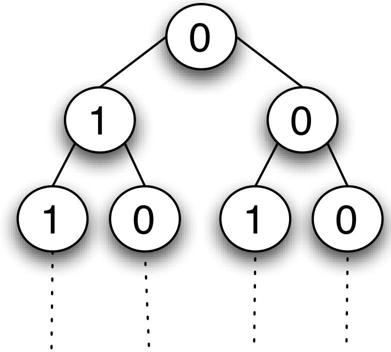
Examples



$$\begin{aligned}\sigma(\varepsilon) &= 0\\ \sigma_L &= \sigma + [1]\\ \sigma_R &= \sigma\end{aligned}$$

Note: [1] denotes *const*(1)

Examples

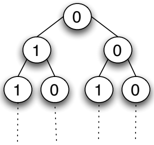


$$
\begin{aligned}
\sigma(\varepsilon) &= 0 \\
\sigma_L &= \sigma + [1] \\
\sigma_R &= \sigma
\end{aligned}
$$

Note: [1] denotes *const*(1)

Derivatives

Recall that we have previously defined



as

$$
\begin{array}{rcl}
i(\sigma) & = & 0 \\
l(\sigma) & = & \sigma + const(1) \\
r(\sigma) & = & \sigma
\end{array}
$$

which resembles the definition with derivatives and is due to the fact that for all $\sigma \in T_A$:

$$
\begin{array}{rcl}
\sigma(\varepsilon) & = & i(\sigma) \\
\sigma_L & = & l(\sigma) \\
\sigma_R & = & r(\sigma)
\end{array}
$$

Operations on trees

From formal power series we inherit several definitions of operations:

Name	Sum	Product
Initial value	$(\sigma+\tau)(\varepsilon)=\sigma(\varepsilon)+\tau(\varepsilon)$	$(\sigma \times \tau)(\varepsilon) = \sigma(\varepsilon) \times \tau(\varepsilon)$
Left der.	$(\sigma + \tau)_L = \sigma_L + \tau_L$	$(\sigma \times \tau)_L = \sigma_L \times \tau + \sigma(\varepsilon) \times \tau_L$
Right der	$(\sigma + \tau)_R = \sigma_R + \tau_R$	$(\sigma \times \tau)_R = \sigma_R \times \tau + \sigma(\varepsilon) \times \tau_R$

Fundamental Theorem

For all infinite binary trees $\sigma \in T_A$:

$$\sigma = \sigma(\varepsilon) + L \times \sigma_L + R \times \sigma_R$$

where

$$\begin{array}{cc} L(\varepsilon) = 0 & R(\varepsilon) = 0 \\ L_L = [1] & R_L = [0] \\ L_R = [0] & R_R = [1] \end{array}$$

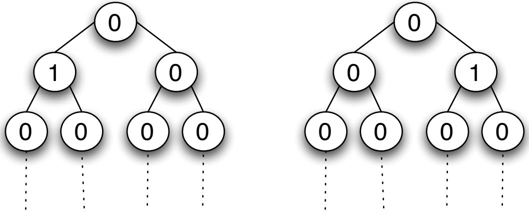
Fundamental Theorem

For all infinite binary trees $\sigma \in T_A$:

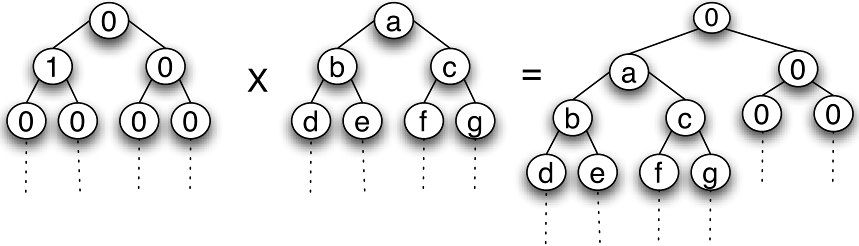
$$\sigma = \sigma(\varepsilon) + L \times \sigma_L + R \times \sigma_R$$

where

$$L(\varepsilon) = 0 \qquad R(\varepsilon) = 0$$
$$L_L = [1] \qquad R_L = [0]$$
$$L_R = [0] \qquad R_R = [1]$$



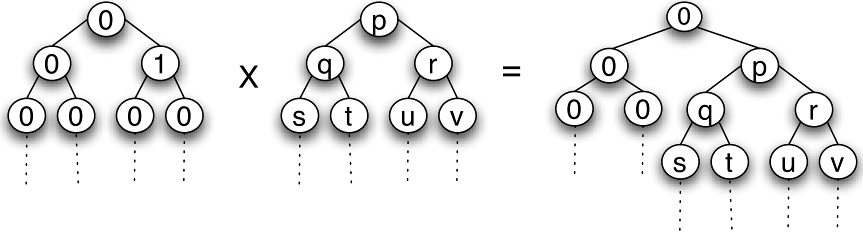
$L \times \sigma_L$



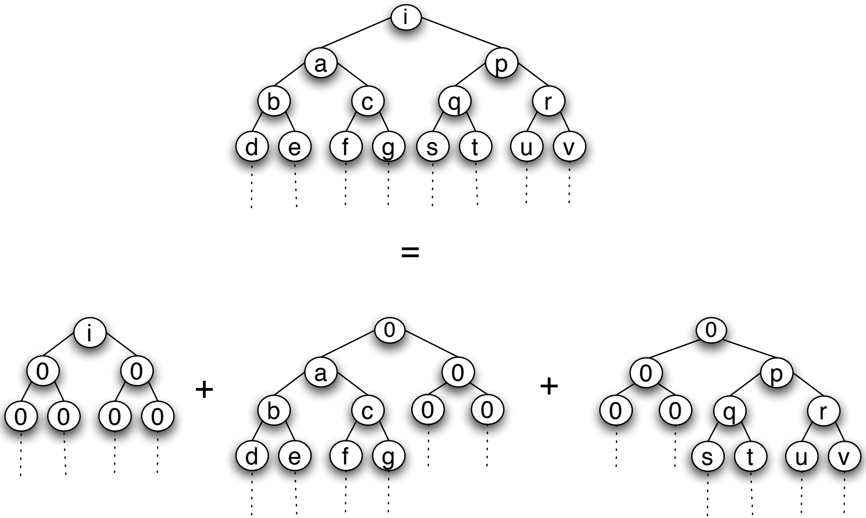
Why?

$R \times \sigma_R$

Similarly:



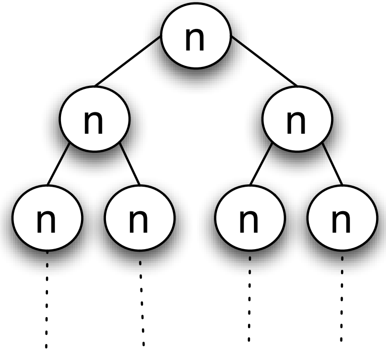
$\sigma = \sigma(\varepsilon) + L \times \sigma_L + R \times \sigma_R$



But. . . What can we do with this theorem?

?

Examples Revisited



$$\begin{aligned} \sigma(\varepsilon) &= n \\ \sigma_L &= \sigma \\ \sigma_R &= \sigma \end{aligned}$$

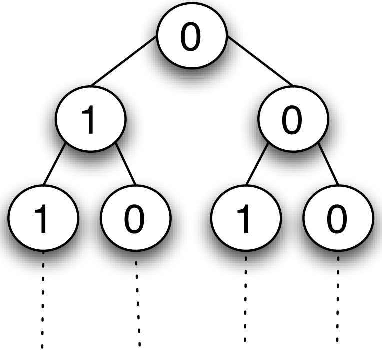
Inverse operation

The inverse of a tree – σ^{-1} – is defined as

$$\begin{aligned} \sigma^{-1}(\varepsilon) &= (\sigma(\varepsilon))^{-1} \\ (\sigma^{-1})_L &= (\sigma(\varepsilon))^{-1} \times \sigma_L \times \sigma^{-1} \\ (\sigma^{-1})_R &= (\sigma(\varepsilon))^{-1} \times \sigma_R \times \sigma^{-1} \end{aligned}$$

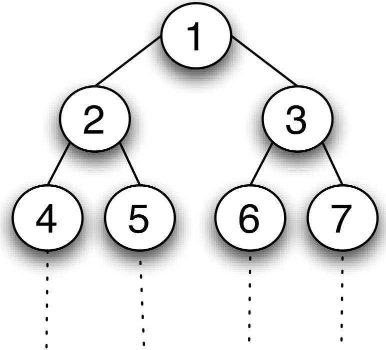
so that $\sigma \times \sigma^{-1} = 1$.

Examples Revisited



$$\begin{aligned} \sigma(\varepsilon) &= 0 \\ \sigma_L &= \sigma + [1] \\ \sigma_R &= \sigma \end{aligned}$$

The natural numbers



Substitution

Conclusions

- Coinductive definitions and bisimulations are a systematic way to reason about infinite structures and operations on them
- Behavioural differential equations are effective to represent (regular) infinite binary trees
- Closed expressions constitute a nice representation of trees (only involving constants)

Future work

- Behavioural differential equations are closely related to lazy functional programming implementations.
- Coinduction gives a systematic way of reasoning about such programs.
- In particular, we would like to study the relation between closed expressions and elimination of corecursion
- We would also like to understand better the class of rational trees