Centrum voor Wiskunde en Informatica

REPORT*RAPPORT*

SEN

Software Engineering

Software ENgineering

Behavioural differential equations and coinduction for binary trees

A.M. Silva, J.J.M.M. Rutten

REPORT SEN-R0702 MAY 2007

Centrum voor Wiskunde en Informatica (CWI) is the national research institute for Mathematics and Computer Science. It is sponsored by the Netherlands Organisation for Scientific Research (NWO).
CWI is a founding member of ERCIM, the European Research Consortium for Informatics and Mathematics.

CWI's research has a theme-oriented structure and is grouped into four clusters. Listed below are the names of the clusters and in parentheses their acronyms.

Probability, Networks and Algorithms (PNA)

Software Engineering (SEN)

Modelling, Analysis and Simulation (MAS)

Information Systems (INS)

P.O. Box 94079, 1090 GB Amsterdam (NL)
Kruislaan 413, 1098 SJ Amsterdam (NL)
Telephone +31 20 592 9333
Telefax +31 20 592 4199

ISSN 1386-369X

Behavioural differential equations and coinduction for binary trees

ABSTRACT
We study the set T_A of infinite binary trees with nodes labelled in a semiring A from a coalgebraic perspective. We present coinductive definition and proof principles based on the fact that T_A carries a final coalgebra structure. By viewing trees as formal power series, we develop a calculus where definitions are presented as behavioural differential equations. We present a general format for these equations that guarantees the existence and uniqueness of solutions. Although technically not very difficult, the resulting framework has surprisingly nice applications, which is illustrated by various concrete examples.

2000 Mathematics Subject Classification: 68P05, 68Q10, 68Q85
1998 ACM Computing Classification System: E.1, F.1, F.3
Keywords and Phrases: Binary trees, Coalgebra, Coinduction, Differential equations, Formal power series

Behavioural Differential Equations and Coinduction for Binary Trees

Alexandra Silva[1]* and Jan Rutten[1,2]

[1] Centrum voor Wiskunde en Informatica (CWI)
[2] Vrije Universiteit Amsterdam (VUA)
{ams,janr}@cwi.nl

Abstract. We study the set T_A of infinite binary trees with nodes labelled in a semiring A from a coalgebraic perspective. We present coinductive definition and proof principles based on the fact that T_A carries a final coalgebra structure. By viewing trees as formal power series, we develop a calculus where definitions are presented as behavioural differential equations. We present a general format for these equations that guarantees the existence and uniqueness of solutions. Although technically not very difficult, the resulting framework has surprisingly nice applications, which is illustrated by various concrete examples.

1 Introduction

Infinite data structures are often used to model problems and computing solutions for them. Therefore, reasoning tools for such structures have become more and more relevant. Coalgebraic techniques turned out to be suited for proving and deriving properties of infinite systems.

In [6], a coinductive calculus of formal power series was developed. In close analogy to classical analysis, the definitions were presented as behavioural differential equations and properties were proved in a calculational (and very natural) way. This approach has shown to be quite effective for reasoning about streams [6,7] and it seems worthwhile to explore its effectiveness for other data structures as well.

In this paper, we shall take a coalgebraic perspective on a classical data structure – infinite binary trees, and develop a similar calculus, using the fact that binary trees are a particular case of formal power series.

The contributions of the present paper are: a coinductive calculus, based on the notion of derivative, to define and to reason about trees and functions on trees; a set of illustrative examples and properties that show the usefulness and expressiveness of such calculus; the formulation of a general format that guarantees the existence and uniqueness of solutions of behavioural differential equations.

Infinite trees arise in several forms in other areas. Formal tree series (functions from trees to an arbitrary semiring) have been studied in [3], closely related to distributive Σ-algebras. The work presented in this paper is completely different since we are concerned with infinite binary trees and not with formal series over trees. In [5], infinite trees appear as generalisations of infinite words and an extensive study of tree automata and topological aspects of trees is made. We have not yet addressed the relation of our work with automata theory. Here we emphasize coinductive definition and proof principles for defining and reasoning about (functions on) trees.

At the end of the paper, in Section 6, the novelty of our approach is discussed further. Also several directions for further applications are mentioned there.

2 Trees and coinduction

We introduce the set T_A of infinite node-labelled binary trees, show that T_A satisfies a coinduction proof principle and illustrate its usefulness.

* Partially supported by the Fundação para a Ciência e a Tecnologia, Portugal, under grant number SFRH/BD/27482/2006.

The set T_A of infinite node-labelled binary trees, where to each node is assigned a value in A, is the final coalgebra for the functor $FX = X \times A \times X$ and can be formally defined by:

$$T_A = \{t \mid t : \{L, R\}^* \to A\}$$

The set T_A carries a final coalgebra structure consisting of the following function:

$$\begin{aligned} \langle l, i, r\rangle \ &: \ T_A \to T_A \times A \times T_A \\ t &\mapsto \langle \lambda w.t(Lw), t(\varepsilon), \lambda w.t(Rw)\rangle \end{aligned}$$

where l and r return the left and right subtrees, respectively, and i gives the label of the root node of the tree. Here, ε denotes the empty word and, for $b \in \{L, R\}$, bw denotes the word resulting from prefixing the word w with the letter b.
These definitions of the set T_A and the respective coalgebra map may not seem obvious. The follow reasoning justifies its correctness:

- It is well known from the literature [4] that the final system for the functor $G(X) = A \times X^B$ is (A^{B^*}, π), where
$$\begin{aligned} \pi \ &: \ A^{B^*} \to A \times (A^{B^*})^B \\ \pi(\phi) &= \langle \phi(\varepsilon), \lambda b \ v. \ \phi(bv)\rangle \end{aligned}$$
- The functor F is isomorphic to $H(X) = A \times X^2$.
- Therefore, the set A^{2^*} is the final coalgebra for the functor F. Considering $2 = \{L, R\}$ we can derive the definition of $\langle l, i, r\rangle$ from the one presented above for π.

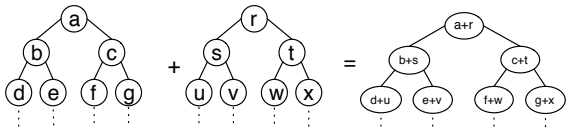
The fact that T_A is a final coalgebra means that for any arbitrary coalgebra $\langle lt, o, rt\rangle : U \to U \times A \times U$, there exists a unique $f : U \to T_A$, such that the following diagram commutes:

$$\begin{array}{ccc} U & \overset{\exists! f}{- - - - - - - - - \succ} & T_A \\ {\scriptstyle \langle lt,o,rt\rangle}\downarrow & & \downarrow{\scriptstyle \langle l,i,r\rangle} \\ U \times A \times U & \xrightarrow[f \times id_A \times f]{} & T_A \times A \times T_A \end{array}$$

The existence part of this statement gives us a *coinductive definition principle*. Every triplet of functions $lt : U \to U$, $o : U \to A$ and $rt : U \to U$ defines a function $h : U \to T_A$, such that:

$$i(h(x)) = o(x) \qquad l(h(x)) = h(lt(x)) \qquad r(h(x)) = h(rt(x))$$

We will see a more general formulation of this principle in section 3, where the right hand side of the above equations will be more general.
Taking $A = \mathbb{R}$ we present the definition of the elementwise sum as an example.



By the definition principle presented above, taking $o(\langle \sigma, \tau\rangle) = i(\sigma) + i(\tau)$, $lt(\langle \sigma, \tau\rangle) = \langle l(\sigma), l(\tau)\rangle$ and $rt(\langle \sigma, \tau\rangle) = \langle r(\sigma), r(\tau)\rangle$ there is a unique function $+ : T_{\mathbb{R}} \times T_{\mathbb{R}} \to T_{\mathbb{R}}$ satisfying:

$$i(\sigma + \tau) = i(\sigma) + i(\tau) \qquad l(\sigma + \tau) = l(\sigma) + l(\tau) \qquad r(\sigma + \tau) = r(\sigma) + r(\tau)$$

Note that in the first equation above we are using $+$ to represent both the sum of trees and the sum of real numbers.
Now that we have explained the formal definition for the set T_A and how one can uniquely define functions into T_A, another important question is still to be answered: how do we prove equality on T_A? In order to prove that two trees σ and τ are equal it is necessary and sufficient to prove

$$\forall_{w \in \{L,R\}^*} \ \sigma(w) = \tau(w) \tag{1}$$

The use of induction on w (prove that $\sigma(\varepsilon) = \tau(\varepsilon)$ and that whenever $\sigma(w) = \tau(w)$ holds then $\sigma(aw) = \tau(aw)$ also holds, for $a \in \{L, R\}$) clearly is a correct method to establish the validity of (1). However, we will often encounter examples where there is not a general formula for $\sigma(w)$ and $\tau(w)$. Instead, we take a coalgebraic perspective on T_A and use the *coinduction proof principle* in order to establish equalities. This proof principle is based on

the notion of bisimulation. A bisimulation on T_A is a relation $S \subseteq T_A \times T_A$ such that, for all σ and τ in T_A,

$$(\sigma, \tau) \in S \Rightarrow \sigma(\varepsilon) = \tau(\varepsilon) \ \wedge \ (l(\sigma), l(\tau)) \in S \ \wedge \ (r(\sigma), r(\tau)) \in S$$

We will write $\sigma \sim \tau$ whenever there exists a bisimulation that contains (σ, τ). The relation $\sim$, called the bisimilarity relation, is the union of all bisimulations (one can easily check that the union of bisimulation is itself a bisimulation).
The following theorem expresses the proof principle mentioned above.

Theorem 1 (Coinduction). *For all trees σ and τ in T_A, if $\sigma \sim \tau$ then $\sigma = \tau$.*

Proof. Consider two trees σ and τ in T_A and let $S \subseteq T_A \times T_A$ be a bisimulation relation which contains the pair (σ, τ). The equality $\sigma(w) = \tau(w)$ now follows by induction on the length of w. We have that $\sigma(\varepsilon) = \tau(\varepsilon)$, because S is a bisimulation. If $w = Lw'$, then

$$\begin{aligned} \sigma(Lw') &= l(\sigma)(w') && \text{(Definition of } l\text{)} \\ &= l(\tau)(w') && (S \text{ is a bisimulation and induction hypothesis}) \\ &= \tau(Lw') && \text{(Definition of } l\text{)} \end{aligned}$$

Similarly, if $w = Rw'$, then $\sigma(Rw') = r(\sigma)(w') = r(\tau)(w') = \tau(Rw')$. Therefore, for all $w \in \{L, R\}^*$, $\sigma(w) = \tau(w)$. This proves $\sigma = \tau$.

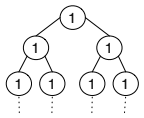
Thus, in order to prove that two trees are equal, it is sufficient to show that they are bisimilar. We shall see several examples of proofs by *coinduction* below.
As a first simple example, let us prove that the pointwise sum for trees of real numbers defined before is commutative. Let $S = \{\langle \sigma + \tau, \tau + \sigma \rangle \mid \sigma, \tau \in T_{\mathbb{R}}\}$. Since $i(\sigma + \tau) = i(\sigma) + i(\tau) = i(\tau + \sigma)$ and

$$\begin{array}{llll} l(\sigma + \tau) & = l(\sigma) + l(\tau) & S \ l(\tau) + l(\sigma) & = l(\tau + \sigma) \\ r(\sigma + \tau) & = r(\sigma) + r(\tau) & S \ r(\tau) + r(\sigma) & = r(\tau + \sigma) \end{array}$$

for any σ and τ, S is a bisimulation relation on $T_{\mathbb{R}}$. The commutativity property follows by coinduction.

3 Behavioural differential equations

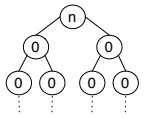
In this section, we shall view trees as formal power series. Following [6], coinductive definitions of operators into T_A and constant trees then take the shape of so-called *behavioural differential equations*. We shall prove a theorem guaranteeing the existence of a unique solution for a large family of systems of behavioural differential equations.
Formal power series are functions $\sigma : \mathcal{X}^* \to k$ from the set of words over an alphabet $\mathcal{X}$ to a semiring k. If A is a semiring, T_A, as defined in section 2, is the set of all formal power series over the alphabet $\{L, R\}$ with coefficients in A. In accordance with the general notion of *derivative* of formal power series [6] we shall write σ_L for $l(\sigma)$ and σ_R for $r(\sigma)$. We will often refer to σ_L, σ_R and $\sigma(\varepsilon)$ as the left derivative, right derivative and initial value of σ.
Following [6], we will develop a coinductive calculus of infinite binary trees. From now on coinductive definitions will have the shape of *behavioural differential equations*. Let us illustrate this approach by a simple example – the coinductive definition of a tree, called *one*, decorated with 1's in every node.



A formal definition of this tree consists the following behavioural differential equations:

differential equations	initial value
$one_L = one$ $one_R = one$	$one(\varepsilon) = 1$

The fact that there exists a unique tree that satisfies the above equations will follow from the theorem below, which presents a general format for behavioural differential equations guaranteeing the existence and uniqueness of solutions.
Behavioural differential equations will be used not just to define single constant trees but also functions on trees. We shall see examples below. Before we present the main result of this section, we need one more definition. We want to be able to view any element $n \in A$ as a tree (which we will denote by $[n]$):



The tree $[n]$ is formally defined as

$$\begin{aligned}[n](\varepsilon) &= n\\ [n](w) &= 0 \;\; w \neq \varepsilon\end{aligned}$$

Next we present a syntax describing the format of behavioural differential equations that we will consider. Let Σ be a set of function symbols, each with an arity $r(f) \geq 0$ for $f \in \Sigma$. (As usual we call f a constant if $r(f) = 0$.) Let $X = \{x_1, x_2, \ldots\}$ be a set of (syntactic) variables, and let $X_L = \{x_L \mid x \in X\}$, $X_R = \{x_R \mid x \in X\}$, $[X(\varepsilon)] = \{[x(\varepsilon)] \mid x \in X\}$ and $X(\varepsilon) = \{x(\varepsilon) \mid x \in X\}$ be sets of notational variants of them. The variables $x \in X$ will play the role of place holders for trees $\tau \in T_A$. Variables x_L, x_R, and $[x(\varepsilon)]$ will then act as place holders for the corresponding trees τ_L, τ_R and $[\tau(\varepsilon)]$ in T_A, while $x(\varepsilon)$ (without the square brackets) will correspond to τ's initial value $\tau(\varepsilon) \in A$. We call a behavioural differential equation for a function symbol $f \in \Sigma$ with arity $r = r(f)$ *well-formed* if it is of the form

differential equations	initial value
$\begin{aligned} f\,(\,x_1,\ldots,x_r\,)_L &= t_1\\ f\,(\,x_1,\ldots,x_r\,)_R &= t_2\end{aligned}$	$(\,f\,(\,x_1,\ldots,x_r\,)\,)\,(\varepsilon) = c(x_1(\varepsilon),\ldots,x_r(\varepsilon))$

where $c : A^r \to A$ is a given function, and where t_1 and t_2 are terms built out of function symbols in Σ and variables in $\{x_1, \ldots, x_r\}$ and their corresponding notational variants in X_L, X_R and $[X(\varepsilon)]$. A well-formed system of equations for Σ will then consist of one well-formed equation for each $f \in \Sigma$. A *solution* of such a system of equations is a set of tree functions

$$\tilde{\Sigma} = \{\tilde{f} : (T_A)^r \to T_A \mid f \in \Sigma\}$$

satisfying, for all $f \in \Sigma$ with arity r and for all $\tau_1, \ldots, \tau_r \in T_A$,

$$\left(\tilde{f}(\tau_1, \ldots, \tau_r)\right)(\varepsilon) = c(\tau_1(\varepsilon), \ldots, \tau_r(\varepsilon))$$

and

$$\left(\tilde{f}(\tau_1, \ldots, \tau_r)\right)_L = \tilde{t_1} \quad \text{and} \quad \left(\tilde{f}(\tau_1, \ldots, \tau_r)\right)_R = \tilde{t_2}$$

where the tree $\tilde{t_1} \in T_A$ (and similarly for $\tilde{t_2}$) is obtained from the term t_1 by replacing (all occurrences of) x_i by τ_i, $(x_i)_L$ by $(\tau_i)_L$, $(x_i)_R$ by $(\tau_i)_R$, and $[x_i(\varepsilon)]$ by $[\tau_i(\varepsilon)]$, for all $i = 1, \ldots, r$, and all function *symbols* $g \in \Sigma$ by their corresponding *function* $\tilde{g}$.

Theorem 2. *Let Σ be a set of function symbols. Every well-formed system of behavioural differential equations for Σ has precisely one solution of tree functions $\tilde{\Sigma}$.*

Proof. Please see appendix A.

Let us illustrate the generality of this theorem by mentioning a few examples of systems of differential equations that satisfy the format above. As a first example, take $\Sigma = \{one\}$ consisting of a single constant symbol (with arity 0) and $X = \emptyset$. We observe that the differential equations for *one* mentioned at the beginning of this section satisfy the format of the theorem. For a second example, let $\Sigma = \{+, \times\}$ with arities $r(+) = r(\times) = 2$ and let $X = \{\sigma, \tau\}$. Consider the following equations:

differential equations	initial value
$\begin{aligned}(\sigma+\tau)_L &= \sigma_L + \tau_L\\ (\sigma+\tau)_R &= \sigma_R + \tau_R\end{aligned}$	$(\sigma+\tau)(\varepsilon) = \sigma(\varepsilon) + \tau(\varepsilon)$

differential equations	initial value
$\begin{aligned}(\sigma\times\tau)_L &= (\sigma_L \times \tau) + ([\sigma(\varepsilon)] \times \tau_L)\\ (\sigma\times\tau)_R &= (\sigma_R \times \tau) + ([\sigma(\varepsilon)] \times \tau_R)\end{aligned}$	$(\sigma\times\tau)(\varepsilon) = \sigma(\varepsilon) \times \tau(\varepsilon)$

These equations define the operations of *sum* and *convolution product* of trees, to be further discussed in Section 4. Note that the right-hand side of the equation for $(\sigma \times \tau)_L$ (and similarly for $(\sigma \times \tau)_R$) is a good illustration of the general format: it is built from the functions $+$ and $\times$, applied to (a subset of) the variables on the left (τ), their derivatives (σ_L and τ_L), and their initial values viewed as trees ($[\sigma(\varepsilon)]$).
Clearly there are many interesting interesting instances of well-formed differential equations. Note, however, that the format *does* impose certain restrictions. The main point is that in the right-hand sides of the equations, only single L and R derivatives are allowed. The following is an example of a system of equations that is *not* well-formed and that does *not* have a unique solution.
Let $\Sigma = \{f\}$, with arity $r(f) = 1$, and let $X = \{\sigma\}$. The equations for f are

differential equations	initial value
$f(\sigma)_L = f(f(\sigma_{LL}))$ $f(\sigma)_R = [0]$	$f(\sigma)(\varepsilon) = \sigma(\varepsilon)$

Both $g(\sigma) = [\sigma(\varepsilon)] + (L \times [\sigma_{LL}(\varepsilon)])$ and $h(\sigma) = [\sigma(\varepsilon)] + (L \times [\sigma_{LL}(\varepsilon)] + L^2 \times (1 - L)^{-1})$ are solutions.
All the examples of systems of behavioural differential equations that will appear in the rest of this document fit into the format of Theorem 2. Therefore, we will omit proofs of the existence and uniqueness of solutions for those systems.
In the next section we will define operators on trees, based on some general operators on formal power series [6].

4 Tree calculus

In this section, we present operators on trees, namely sum, convolution product and inverse, and state some elementary properties, which we will prove using coinduction.
The sum of two trees is defined as the unique operator satisfying:

differential equations	initial value
$(\sigma + \tau)_L = \sigma_L + \tau_L$ $(\sigma + \tau)_R = \sigma_R + \tau_R$	$(\sigma + \tau)(\varepsilon) = \sigma(\varepsilon) + \tau(\varepsilon)$

Note that this is a generalisation of the sum on trees of real numbers defined in section 2 and that again we are overloading the use of $+$ to represent both sum on trees and sum on the elements of the semiring.
Sum satisfies some desired properties, easily proved by coinduction, such as commutativity or associativity:

Theorem 3. *For all σ, τ and ρ in T_A, $\sigma + 0 = \sigma$, $\sigma + \tau = \tau + \sigma$ and $\sigma + (\tau + \rho) = (\sigma + \tau) + \rho$.*

Here, we are using 0 as a shorthand for $[0]$. We shall use this convention (for all $n \in A$) throughout this document. We define the convolution product of two trees as the unique operation satisfying:

differential equations	initial value
$(\sigma \times \tau)_L = (\sigma_L \times \tau) + (\sigma(\varepsilon) \times \tau_L)$ $(\sigma \times \tau)_R = (\sigma_R \times \tau) + (\sigma(\varepsilon) \times \tau_R)$	$(\sigma \times \tau)(\varepsilon) = \sigma(\varepsilon) \times \tau(\varepsilon)$

Note that in the above definition we use $\times$ for representing both multiplication on trees and on the elements of the semiring. Following the convention mentioned above $\sigma(\varepsilon) \times \tau_L$ and $\sigma(\varepsilon) \times \tau_R$ are shorthands for $[\sigma(\varepsilon)] \times \tau_L$ and $[\sigma(\varepsilon)] \times \tau_R$. We shall also use the standard convention of writing $\sigma\tau$ for $\sigma \times \tau$.
The general formula to compute the value of σ according to a path given by the word $w \in \{L, R\}^*$ is:

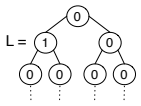
$$(\sigma \times \tau)(w) = \sum_{w = u \cdot v} \sigma(u) \times \tau(v)$$

where $\cdot$ denotes word concatenation.
To give the reader some intuition about this operation we will give a concrete example. Take A to be the Boolean semiring $\mathbb{B} = \{0, 1\}$, with operations $+ = \vee$ and $\times = \wedge$. Then, T_A corresponds to the languages over a two letter alphabet, and in this case the tree product operator coincides with language concatenation.
The following theorem states some familiar properties of this operator.

Theorem 4. *For all σ, τ, ρ in T_A and a, b in A, $\sigma \times 1 = 1 \times \sigma = \sigma$, $\sigma \times 0 = 0 \times \sigma = 0$, $\sigma \times (\tau + \rho) = (\sigma \times \tau) + (\sigma \times \rho)$, $\sigma \times (\tau \times \rho) = (\sigma \times \tau) \times \rho$ and $[a] \times [b] = [a \times b]$.*

Proof. An exercise in coinduction. In [7], these properties are proved for streams.

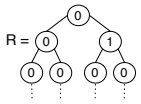
Note that the convolution product is not commutative. Before we present the inverse operation, let us introduce two (very useful) constants, which we shall call left constant L and right constant R. They will have an important role in the tree calculus that we are about to develop and will turn out to have interesting properties when interacting with the product operation. The left constant L is a tree with 0's in every node except in the root of the left branch where it has a 1:



It is formally defined as

$$L(w) = 1 \; if \; w = L$$
$$L(w) = 0 \; otherwise$$

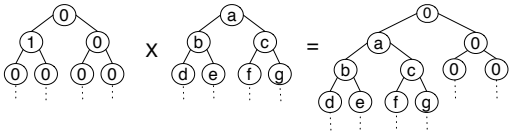
Similarly, the right constant R is only different from 0 in the root of its right branch:



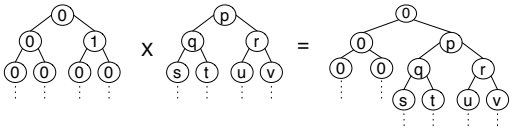
and is defined as

$$R(w) = 1 \; if \; w = R$$
$$R(w) = 0 \; otherwise$$

These constants have interesting properties when multiplied by an arbitrary tree. $L \times \sigma$ produces a tree whose root and right subtrees are null and the left branch is σ:



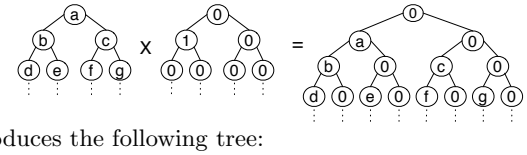
Dually, $R \times \sigma$ produces a tree whose root and left subtrees are null and the right branch is σ:



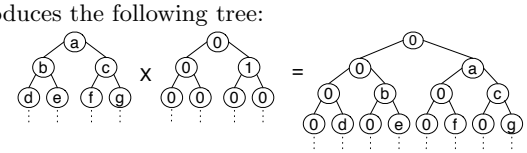
As before, if we see L and σ as languages and the product as concatenation, we can gain some intuition on the meaning of this operation. $L \times \sigma$ will prefix every word of σ with the letter L, meaning that no word starting by R will be an element of $L \times \sigma$, and thus, $L \times \sigma$ has a null right branch. Similar for $R \times \sigma$.
As we pointed out before, the product operation is not commutative. For example, $\sigma \times L \neq L \times \sigma$ and $\sigma \times R \neq R \times \sigma$. In fact, multiplying a tree σ on the right with L or R is interesting in itself. For instance, $\sigma \times L$ satisfies

$$(\sigma \times L)(w) = \begin{cases} \sigma(u) & w = uL \\ 0 & otherwise \end{cases}$$

which corresponds to the following transformation:



Similarly, $\sigma \times R$ produces the following tree:



Again, if you interpret these operations in the language setting, what is being constructed is the language that has all words of the form uL and uR, respectively, such that $\sigma(u) \neq 0$.

We define the inverse of a tree as the unique operator satisfying:

differential equations	initial value
$(\sigma^{-1})_L = \sigma(\varepsilon)^{-1} \times (-\sigma_L \times (\sigma^{-1}))$ $(\sigma^{-1})_R = \sigma(\varepsilon)^{-1} \times (-\sigma_R \times (\sigma^{-1}))$	$\sigma^{-1}(\varepsilon) = \sigma(\varepsilon)^{-1}$

We are using $-\sigma_L$ and $-\sigma_R$ as shorthands for $[-1] \times \sigma_L$ and $[-1] \times \sigma_R$, respectively. In this definition, we need to require A to be a ring, in order to have additive inverses. Moreover, the tree σ is supposed to have also a multiplicative inverse for its initial value.
The inverse of a tree has the usual properties:

Theorem 5. *For all σ and τ in T_A:*

$$\sigma^{-1} \text{ is the unique tree s.t. } \sigma \times \sigma^{-1} = 1 \tag{2}$$

$$(\sigma \times \tau)^{-1} = \tau^{-1} \times \sigma^{-1} \tag{3}$$

Proof. For the existence part of (2), note that

1. $(\sigma \times \sigma^{-1})(\varepsilon) = \sigma(\varepsilon) \times \sigma(\varepsilon)^{-1} = 1$
2. $(\sigma \times \sigma^{-1})_L = (\sigma_L \times \sigma^{-1}) + (\sigma(\varepsilon) \times (\sigma(\varepsilon)^{-1} \times (-\sigma_L \times \sigma^{-1}))) = 0$
3. $(\sigma \times \sigma^{-1})_R = (\sigma_R \times \sigma^{-1}) + (\sigma(\varepsilon) \times (\sigma(\varepsilon)^{-1} \times (-\sigma_R \times \sigma^{-1}))) = 0$

So, by uniqueness (using the behavioural differential equations that define 1) we have proved that $\sigma \times \sigma^{-1} = 1$. Now, for the uniqueness part of (2), suppose that there is a tree τ such that $\sigma \times \tau = 1$. We shall prove that $\tau = \sigma^{-1}$. Note that from the equality $\sigma \times \tau = 1$ we derive that

1. $\tau(\varepsilon) = \sigma(\varepsilon)^{-1}$
2. $\tau_L = \sigma(\varepsilon) \times (-\sigma_L \times \tau)$
3. $\tau_R = \sigma(\varepsilon) \times (-\sigma_R \times \tau)$

Thus, by uniqueness of solutions for systems of behavioural differential equations, $\tau = \sigma^{-1}$. For (3), note that $(\sigma \times \tau) \times \tau^{-1} \times \sigma^{-1} = \sigma \times (\tau \times \tau^{-1}) \times \sigma^{-1} = 1$. Therefore, using the uniqueness property of (2), $(\sigma \times \tau)^{-1} = \tau^{-1} \times \sigma^{-1}$.

5 Applications of tree calculus

We will illustrate the usefulness of our calculus by looking at a series of interesting examples. In order to compute closed formulae for trees we will be using the following identity:

$$\forall_{\sigma \in T_A}\ \sigma = \sigma(\varepsilon) + (L \times \sigma_L) + (R \times \sigma_R) \tag{4}$$

which can be easily proved by coinduction. We will now show how to use this identity to construct the closed formula for a tree.
Recall our first system of behavioural differential equations:

differential equations	initial value
$one_L = one$ $one_R = one$	$one(\varepsilon) = 1$

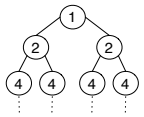
There we saw that the unique solution for this system was the tree with 1's in every node. Alternatively, we can compute the solution using (4) as follows.

$$\begin{aligned} & one && = one(\varepsilon) + (L \times one_L) + (R \times one_R) \\ \Leftrightarrow\ & one && = 1 + (L \times one) + (R \times one) \\ \Leftrightarrow\ & (1 - L - R)one && = 1 \\ \Leftrightarrow\ & one && = (1 - L - R)^{-1} \end{aligned}$$

Therefore, *one* can be represented by the (very compact) closed formula[3] $(1 - L - R)^{-1}$.
Let us see a few more examples. From now on we will work with $A = \mathbb{R}$, where we have the extra property: $[n] \times \sigma = \sigma \times [n]$, for all $n \in \mathbb{R}$ and $\sigma \in T_{\mathbb{R}}$.

[3] Note the similarity of this closed formula with the one obtained for the stream $(1, 1, \ldots)$ in [7]: $(1-X)^{-1}$.

The tree where every node at level k is labelled with the value 2^k, called *pow*,



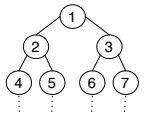
is defined by the following system:

differential equations	initial value
$pow_L = 2 \times pow$ $pow_R = 2 \times pow$	$pow(\varepsilon) = 1$

We proceed as before, applying (4):

$$\begin{aligned} & pow && = pow(\varepsilon) + (L \times pow_L) + (R \times pow_R) \\ \Leftrightarrow\ & pow && = 1 + (2L \times pow) + (2R \times pow) \\ \Leftrightarrow\ & (1 - 2L - 2R)pow && = 1 \\ \Leftrightarrow\ & pow && = (1 - 2L - 2R)^{-1} \end{aligned}$$

which gives us a nice closed formula for *pow*[4].
The tree with the natural numbers



is represented by the following system of differential equations:

differential equations	initial value
$nat_L = nat + pow$ $nat_R = nat + (2 \times pow)$	$nat(\varepsilon) = 1$

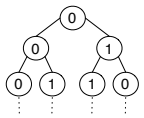
Applying identity (4):

$$\begin{aligned} & nat && = nat(\varepsilon) + (L \times nat_L) + (R \times nat_R) \\ \Leftrightarrow\ & nat && = 1 + (L \times (nat + pow)) + (R \times (nat + 2pow)) \\ \Leftrightarrow\ & (1 - L - R)nat && = 1 + L(1 - 2L - 2R)^{-1} + 2R(1 - 2L - 2R)^{-1} \\ \Leftrightarrow\ & (1 - L - R)nat && = (1 - L) \times (1 - 2L - 2R)^{-1} \\ \Leftrightarrow\ & nat && = (1 - L - R)^{-1} \times (1 - L) \times (1 - 2L - 2R)^{-1} \end{aligned}$$

The Thue-Morse sequence [1] can be obtained by taking the parities of the counts of 1's in the binary representation of non-negative integers. Alternatively, it can be defined by the repeated application of the substitution map $\{0 \to 01; 1 \to 10\}$:

$$0 \to 01 \to 0110 \to 01101001 \to \dots$$

We can encode this substitution map in a binary tree, called *thue*, which at each level k will have the first 2^k digits of the Thue-Morse sequence:



In this example, we take for A the Boolean ring $2 = \{0, 1\}$ (where $1 + 1 = 0$). The following system of differential equations defines *thue*:

differential equations	initial value
$thue_L = thue$ $thue_R = thue + one$	$thue(\varepsilon) = 0$

[4] Again, there is a strong similarity with streams: the closed formula for the stream $(1, 2, 4, 8, \dots)$ is $(1 - 2X)^{-1}$

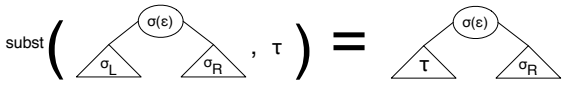
Note that $thue + one$ equals the (elementwise) complement of $thue$. Applying (4) to $thue$, we calculate:

$$\begin{aligned} & thue && = (L \times thue) + (R \times (thue + one)) \\ \Leftrightarrow\ & (1 - L - R) \times thue && = R \times one \\ \Leftrightarrow\ & thue && = (1 - L - R)^{-1} \times R \times one \end{aligned}$$

which then leads to the following pretty formula for $thue$:

$$thue = one \times R \times one$$

Let us present another example – a substitution operation, which given two trees σ and τ, replaces the left subtree of σ by τ.



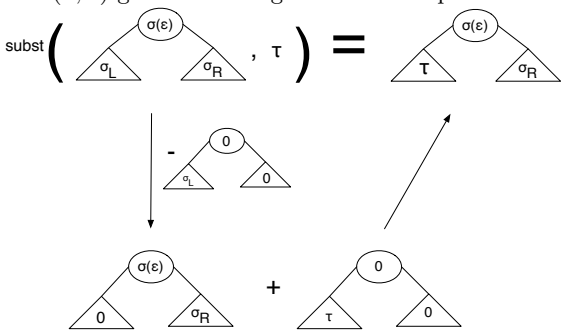
It is easy to see that the equations that define this operation are:

differential equations	initial value
$subst(\sigma,\tau)_L = \tau$ $subst(\sigma,\tau)_R = \sigma_R$	$subst(\sigma,\tau)(\varepsilon) = \sigma(\varepsilon)$

Then, we apply (4) and we reason:

$$\begin{aligned} & subst(\sigma,\tau) = \sigma(\varepsilon) + (L \times \tau) + (R \times \sigma_R) \\ \Leftrightarrow\ & subst(\sigma,\tau) = \sigma - (L \times \sigma_L) + (L \times \tau) \\ \Leftrightarrow\ & subst(\sigma,\tau) = \sigma - L(\sigma_L - \tau) \end{aligned}$$

Note that in the second step, we applied identity (4) to σ. Moreover, remark that the final closed formula for $subst(\sigma,\tau)$ gives us the algorithm to compute the substitution:



We can now wonder how to define a more general substitution operation that has an arbitrary path $P \in \{L,R\}^+$ as an extra argument and replaces the subtree of σ given by this path by τ. It seems obvious to define it as

$$subst(\sigma,\tau,P) = \sigma - P(\sigma_P - \tau)$$

where, in the right hand side, $P = a_1 a_2 \ldots a_n$ is interpreted as $a_1 \times a_2 \times \ldots \times a_n$ and the derivative σ_P is defined as

$$\sigma_P = \begin{cases} \sigma_\delta & P = \delta \\ (\sigma_\delta)_{P'} & P = \delta.P' \end{cases}$$

with δ being either L or R.
Let us check that our intuition is correct. First, we present the definition for this operation:

differential equations	initial value
$subst(\sigma,\tau,P)_\delta = \begin{cases} \tau & P = \delta \\ subst(\sigma_\delta,\tau,P') & P = \delta.P' \\ \sigma_\delta & P = \delta'.P' \end{cases}$	$subst(\sigma,\tau,P)(\varepsilon) = \sigma(\varepsilon)$

where $\delta' \neq \delta$. Now, observe that

$$R = \{\langle subst(\sigma,\tau,P), \sigma - P(\sigma_P - \tau)\rangle \mid \sigma,\tau \in T_{\mathbb{R}},\ P \in \{L,R\}^+\} \cup \{\langle \sigma,\sigma\rangle \mid \sigma \in T_{\mathbb{R}}\}$$

is a bisimulation relation because:

1. $(\sigma - P(\sigma_P - \tau))(\varepsilon) = \sigma(\varepsilon) = subst(\sigma,\tau,P)(\varepsilon)$

2. For $\delta \in \{L, R\}$,

$$\begin{aligned}(\sigma - P(\sigma_P - \tau))_\delta &= \sigma_\delta - P_\delta(\sigma_P - \tau) \\ &= \begin{cases} \tau & P = \delta \\ \sigma_\delta - P'((\sigma_\delta)_{P'} - \tau) & P = \delta.P' \\ \sigma_\delta & P = \delta'.P' \end{cases} \\ R & \begin{cases} \tau & P = \delta \\ subst(\sigma_\delta, \tau, P') & P = \delta.P' \\ \sigma_\delta & P = \delta'.P' \end{cases} \\ &= subst(\sigma, \tau, P)_\delta\end{aligned}$$

Therefore, by Theorem 1, $subst(\sigma, \tau, P) = \sigma - P(\sigma_P - \tau)$.
Using this formula we can now prove properties about this operation. For instance, one would expect that $subst(\sigma, \sigma_P, P) = \sigma$ and also $subst(subst(\sigma, \tau, P), \sigma_P, P) = \sigma$.
The first equality follows easily: $subst(\sigma, \sigma_P, P) = \sigma - P(\sigma_P - \sigma_P) = \sigma$. For the second one we have:

$$\begin{aligned}& subst(subst(\sigma, \tau, P), \sigma_P, P) \\ &= subst(\sigma - P(\sigma_P - \tau), \sigma_P, P) && \text{(Definition of } subst\text{)} \\ &= \sigma - P(\sigma_P - \tau) - P((\sigma - P(\sigma_P - \tau))_P - \sigma_P) && \text{(Definition of } subst\text{)} \\ &= \sigma - P(\sigma_P - \tau) - P(\tau - \sigma_P) && ((\sigma - P(\sigma_P - \tau))_P = \sigma_P - \sigma_P + \tau = \tau) \\ &= \sigma\end{aligned}$$

Remark that this operation is a standard example in introductory courses on algorithms and data structures. It is often presented either as a recursive expression (very much in the style of our differential equations) or as a contrived iterative procedure. This example shows that our compact formulae constitute a clear way of presenting algorithms and that they can be used to eliminate recursion. Moreover, the differential equations are directly implementable algorithms (in functional programming) and our calculus provides a systematic way of reasoning about such programs.

6 Discussion

We have modelled binary trees as formal power series and, using the fact that the latter constitute a final coalgebra, this has enabled us to apply some coalgebraic reasoning. Technically, none of this is very difficult. Rather, it is an application of well known coalgebraic insights. As is the case with many of such applications, it has the flavour of an exercise. At the same time, the result contains several new elements that have surprised us. Although technically Theorem 2 is an easy extension of a similar such theorem for streams, the resulting format for differential equations for trees is surprisingly general and useful. It has allowed us to define various non-trivial trees by means of simple differential equations, and to compute rather pleasant closed formulae for them. We have also illustrated that based on this, coinduction is a convenient proof method for trees. As an application, all of this is new, to the best of our knowledge. (Formal tree series, which have been studied extensively, may seem to be closely related but are not: here we are dealing with differential equations that characterise *single* trees.) In addition to the illustrations of the present differential calculus for trees, we see various directions for further applications: (i) The connection with (various types of) automata and the final coalgebra T_A of binary trees needs further study. For instance, every Moore automaton with input in $2 = \{L, R\}$ and output in A has a minimal representation in T_A. Details of these automata (and on weighted variants of them) are not complicated but have not been worked out here because of lack of space. (ii) The closed formula that we have obtained for the (binary tree representing the) Thue-Morse sequence suggests a possible use of coinduction and differential equations in the area of automatic sequences [2]. Typically, automatic sequences are represented by automata. The present calculus seems an interesting alternative, in which properties such as algebraicity of sequences can be derived from the tree differential equations that define them. (iii) Finally, the closed formulae that we obtain for tree substitution suggest many further applications of our tree calculus to (functional) programs on trees, including the analysis of their complexity.

References

1. J.-P. Allouche and J. Shallit. The ubiquitous Prouhet-Thue-Morse sequence. In C. Ding, T. Helleseth, and N. H., editors, *Sequences and their applications, Proceedings of SETA'98*, pages 1–16. Springer Verlag, 1999.
2. J.-P. Allouche and J. Shallit. *Automatic sequences: theory, applications, generalizations*. Cambridge University Press, 2003.
3. Z. Ésik and W. Kuich. Formal tree series. *Journal of Automata, Languages and Combinatorics*, 8(2):219–285, 2003.
4. E. G. Manes and M. A. Arbib. *Algebraic approaches to program semantics*. Springer-Verlag New York, Inc., New York, NY, USA, 1986.
5. D. Perrin and J.-E. Pin. *Infinite Words*, volume 141 of *Pure and Applied Mathematics*. Elsevier, 2004. ISBN 0-12-532111-2.
6. J. J. M. M. Rutten. Behavioural differential equations: a coinductive calculus of streams, automata, and power series. *Theor. Comput. Sci.*, 308(1-3):1–53, 2003.
7. J. J. M. M. Rutten. A coinductive calculus of streams. *Mathematical Structures in Computer Science*, 15(1):93–147, 2005.

A Proof of theorem 2

Proof (Proof of theorem 2). Consider a well-formed system of differential equations for Σ, as defined above. We define a set $\mathcal{T}$ of terms t by the following syntax:

$$t ::= \underline{\tau} \quad (\tau \in T_A) \;\mid\; f(t_1, \ldots, t_{r(f)}) \quad (f \in \Sigma)$$

where for every tree $\tau \in T_A$ the set $\mathcal{T}$ contains a corresponding term, denoted by $\underline{\tau}$, and where new terms are constructed by (syntactic) composition of function symbols from Σ with the appropriate number of argument terms. Next we turn $\mathcal{T}$ into an F-coalgebra by defining a function $\langle l, o, r\rangle : \mathcal{T} \to (\mathcal{T} \times A \times \mathcal{T})$ by induction on the structure of terms, as follows. First we define $o : \mathcal{T} \to A$ by

$$o(\underline{\tau}) = \tau(\varepsilon)$$
$$o\big(\, f(t_1, \ldots t_{r(f)})\,\big) = c\big(\, o(t_1), \ldots, o(t_{r(f)})\,\big)$$

(where c is the function used in the equations for f). Next we define $l : \mathcal{T} \to \mathcal{T}$ and $r : \mathcal{T} \to \mathcal{T}$ by $l(\underline{\tau}) = \underline{\tau_L}$ and $r(\underline{\tau}) = \underline{\tau_R}$, and by

$$l\,(\, f(t_1, \ldots t_r)\,) = \overline{t_1} \quad \text{and} \quad l\,(\, f(t_1, \ldots t_r)\,) = \overline{t_2}$$

Here the terms $\overline{t_1}$ and $\overline{t_2}$ are obtained from the terms t_1 and t_2 used in the equations for f, by replacing (every occurrence of) x_i by t_i, $(x_i)_L$ by $l(t_i)$, $(x_i)_R$ by $r(t_i)$, and $[x_i(\varepsilon)]$ by $[o(t)]$, for $i = 1, \ldots, r$. Because T_A is a final F-coalgebra, there exists a unique homomorphism $h : \mathcal{T} \to T_A$. We can use it to define tree functions $\tilde{f} : (T_A)^r \to T_A$, for every $f \in \Sigma$, by putting, for all $\tau_1, \ldots, \tau_r \in T_A$,

$$\tilde{f}(\tau_1, \ldots, \tau_r) = h\,\big(\, f(\underline{\tau_1}, \ldots, \underline{\tau_r})\,\big)$$

This gives us a set $\tilde{\Sigma}$ of tree functions. One can prove that it is a solution of our system of differential equations by coinduction, using the facts that $h(\underline{\tau}) = \tau$, for all $\tau \in T_A$, and

$$h\,(\, f(t_1, \ldots, t_r)\,) = \tilde{f}\,(\, h(t_1), \ldots, h(t_r)\,)$$

for all $f \in \Sigma$ and $t_i \in \mathcal{T}$. This solution is unique because, by finality of T_A, the homomorphism h is.