Information and Computation 208 (2010)578–593

Contents lists available at ScienceDirect

Information and Computation

journal homepage: www.elsevier.com/locate/ic

A coinductive calculus of binary trees

Alexandra Silva[a,*,1], Jan Rutten[a,b]

[a] *Centrum voor Wiskunde en Informatica (CWI), The Netherlands*
[b] *Vrije Universiteit Amsterdam (VUA), The Netherlands*

A R T I C L E   I N F O

Article history:
Received 23 November 2007
Revised 19 August 2008
Available online 4 January 2010

A B S T R A C T

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 [8], 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 [8,9] 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; the view of infinite binary trees as generalizations of other well-known data-structures, namely infinite streams and bi-infinite streams and a discussion of the notion of rational tree including a comparison with existing notions of rationality in the literature.

Infinite trees arise in several forms in other areas. Formal tree series (functions from trees to an arbitrary semiring) have been studied in [4], 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 power series over trees. In [6], 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.

[*] Corresponding author.
E-mail addresses: ams@cwi.nl (A. Silva), janr@cwi.nl (J. Rutten).
[1] Partially supported by the Fundação para a Ciência e a Tecnologia, Portugal, under Grant No. SFRH/BD/27482/2006.

At the end of the paper, in Section 8, 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.

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:

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

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 [5] that the final system for the functor $G(X) = A \times X^B$ is (A^{B^*}, π), where

$$\pi : A^{B^*} \to A \times (A^{B^*})^B$$
$$\pi(\phi) = \langle \phi(\varepsilon), \lambda b\, v.\, \phi(bv)\rangle$$

- 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:

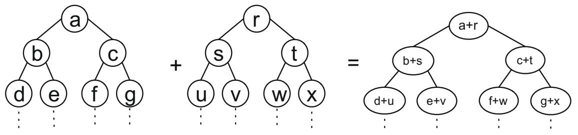
$$\begin{tikzcd} U \arrow[r, dashed, "\exists! f"] \arrow[d, "\langle lt{,}o{,}rt\rangle"'] & T_A \arrow[d, "\langle l{,}i{,}r\rangle"] \\ U \times A \times U \arrow[r, "f \times id_A \times f"'] & T_A \times A \times T_A \end{tikzcd}$$

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:

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

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:

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

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 definition of bisimulation presented above follows directly from instantiating the notion of F-bisimulation [7] to the functor $FX = X \times A \times X$.

Also in [7] a general formulation of the coinduction proof principle mentioned above is presented. The following theorem is the instantiation of such principle to infinite binary trees.

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

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

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 $\sigma = \tau$. $\square$

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

$$\begin{array}{ll} l(\sigma + \tau) = l(\sigma) + l(\tau) & Sl(\tau) + l(\sigma) = l(\tau + \sigma) \\ r(\sigma + \tau) = r(\sigma) + r(\tau) & Sr(\tau) + r(\sigma) = r(\tau + \sigma) \end{array}$$

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

Let us proceed with an apparently more complex example. For a function $f : \mathbb{R} \to \mathbb{R}$ with $f(a + b) = f(a) + f(b)$, $\forall_{a,b} \in \mathbb{R}$, we show that

$$map_f(\sigma + \tau) = map_f(\sigma) + map_f(\tau)$$

where map_f applies a function f to every node of a given tree and is defined coinductively as

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

Similarly to what we did before, let S be a relation defined as follows:

$$S = \{\langle map_f(\sigma + \tau), map_f(\sigma) + map_f(\tau) \rangle \mid \sigma, \tau \in T_{\mathbb{R}}\}$$

with f preserving sums, as described above. Because

$$\begin{aligned} i(map_f(\sigma + \tau)) = f(i(\sigma + \tau)) &= f(i(\sigma)) + f(i(\tau)) \\ &= i(map_f(\sigma)) + i(map_f(\tau)) \\ &= i(map_f(\sigma) + map_f(\tau)) \end{aligned}$$

and, for $t \in \{l, r\}$

$$\begin{array}{lll} t(map_f(\sigma+\tau)) & = map_f(t(\sigma+\tau)) & = map_f(t(\sigma)+t(\tau)) \\ & & S\; map_f(t(\sigma)) + map_f(t(\tau)) \\ & & = t(map_f(\sigma)) + t(map_f(\tau)) \\ & & = t(map_f(\sigma)+map_f(\tau)) \end{array}$$

we can conclude that S is a bisimulation. Therefore, the desired equality follows by coinduction.

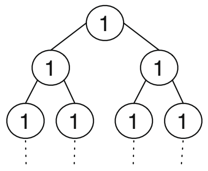
Although this example seemed more complex than the first, the final proof has a similar complexity. The bisimulation that witnesses the equality was constructed in a similar way and without great effort. This illustrates the power of proofs by coinduction – one can reduce the proof of laws about infinite structures to the construction of a relation that can be finitely described.

3. Behavioural differential equations

In this section, we shall view trees as formal power series. Following [8], 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 [8] 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 [8], 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.

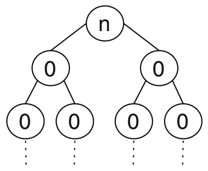


A formal definition of this tree consists the following behavioural differential equations:

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

$$\begin{array}{lll} [n](\varepsilon) & = & n \\ [n](w) & = & 0 \quad w \neq \varepsilon \end{array}$$

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 system of behavioural differential equations for a function symbol $f \in \Sigma$ with arity $r = r(f)$ *well-formed* if it is of the form

Differential equations	Initial value
$f(x_1,\ldots,x_r)_L = p$ $f(x_1,\ldots,x_r)_R = q$	$(f(x_1,\ldots,x_r))(\varepsilon) = c(x_1(\varepsilon),\ldots,x_r(\varepsilon))$

where $c : A^r \to A$ is a given function, and where p and q 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

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

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

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

where the tree $\tilde{p} \in T_A$ (and similarly for $\tilde{q}$) is obtained from the term p 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. 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. Note that $\mathcal{T}$ is disjoint from the set of terms p, q as described above. The latter have no constants, while the elements of $\mathcal{T}$ have no variables.

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

$$\begin{array}{lcl} o(\underline{\tau}) & = & \tau(\varepsilon) \\ o\left(f(t_1,\ldots t_{r(f)})\right) & = & c\left(o(t_1),\ldots,o(t_{r(f)})\right) \end{array}$$

(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

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

Here, the terms $\overline{p}$ and $\overline{q}$ are obtained from the terms p and q 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 $\underline{[o(t)]}$, 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 $\tau_1,\ldots,\tau_r \in T_A$,

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

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. □

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
$(\sigma + \tau)_L = \sigma_L + \tau_L$ $(\sigma + \tau)_R = \sigma_R + \tau_R$	$(\sigma + \tau)(\varepsilon) = \sigma(\varepsilon) + \tau(\varepsilon)$

Differential equations	Initial value
$(\sigma \times \tau)_L = (\sigma_L \times \tau) + ([\sigma(\varepsilon)] \times \tau_L)$ $(\sigma \times \tau)_R = (\sigma_R \times \tau) + ([\sigma(\varepsilon)] \times \tau_R)$	$(\sigma \times \tau)(\varepsilon) = \sigma(\varepsilon) \times \tau(\varepsilon)$

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

This system is not well-formed because in the right-hand side of the equation for $f(\sigma)_L$ the variable σ appears in a non-allowed notational variant $\sigma_L L$ (it should only appear as $\sigma, \sigma_L, \sigma_R$ or $\sigma(\varepsilon)$). 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, in each case there exists a unique solution.

In the next section, we will define operators on trees, based on some general operators on formal power series [8].

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.

Proof. Easy exercise in coinduction. The equalities follow, respectively, from the fact that the relations $\{\langle \sigma + 0, \sigma \rangle \mid \sigma \in T_A\}$, $\{\langle \sigma + \tau, \tau + \sigma \rangle \mid \sigma, \tau \in T_A\}$ and $\{\langle \sigma + (\tau + \rho), (\sigma + \tau) + \rho \rangle \mid \sigma, \tau, \rho \in T_A\}$ are bisimulations. □

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 $\sigma \times \tau$ according to a path given by a word $w \in \{L, R\}^*$ is given by $(\sigma \otimes \tau)(w)$ where:

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

where $\cdot$ denotes word concatenation. The fact that these two definitions of product coincide follows by coinduction because the following relation is a bisimulation:

$$S = \{\langle \sigma_1 \times \tau_1 + \cdots + \sigma_n \times \tau_n, \sigma_1 \otimes \tau_1 + \cdots + \sigma_n \otimes \tau_n \rangle \mid \sigma_i, \tau_i \in T_A\}$$

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, a tree $\tau \in T_A$ corresponds to a language $\mathcal{L}(\tau)$ over the alphabet $\{L, R\}$ given by

$$\mathcal{L}(\tau) = \{w \in \{L, R\}^* \mid \tau(w) = 1\} \tag{2}$$

The product of trees corresponds then to concatenation of languages:

$$\mathcal{L}(\tau \times \sigma) = \mathcal{L}(\tau) \times \mathcal{L}(\sigma)$$

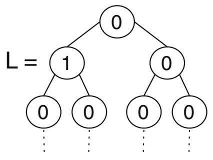
The following theorem states some familiar properties of the convolution product.

Theorem 4. *For all σ, τ, ρ in T_A and a, b in A*

$$\begin{array}{l} \sigma \times 1 = 1 \times \sigma = \sigma \\ \sigma \times 0 = 0 \times \sigma = 0 \\ \sigma \times (\tau + \rho) = (\sigma \times \tau) + (\sigma \times \rho) \\ \sigma \times (\tau \times \rho) = (\sigma \times \tau) \times \rho \\ [a] \times \sigma = \sigma \times [a], \textit{ if } A \textit{ is a commutative ring} \\ [a] \times [b] = [a \times b] \end{array}$$

Proof. An exercise in coinduction. In [9], these properties are proved for streams. □

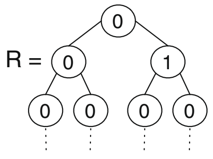
Note that the convolution product is not commutative. Before we present the inverse operation, let us introduce two (very useful) constants, which we shall call left constant L and right constant R. They will have an important role in the tree calculus that we are about to develop and will turn out to 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

$$\begin{array}{lcll} L(w) & = & 1 & \textit{if } w = L \\ L(w) & = & 0 & \textit{otherwise} \end{array}$$

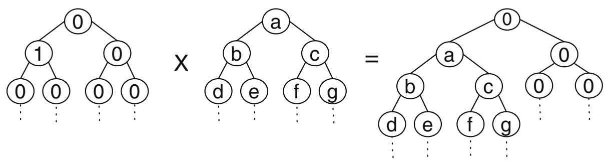
Similarly, the right constant R is only different from 0 in the root of its right branch:



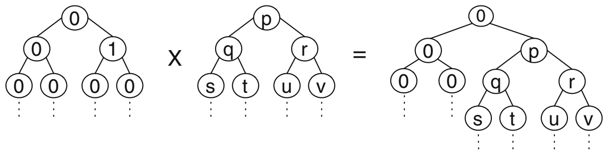
and is defined as

$$\begin{array}{lcll} R(w) & = & 1 & \textit{if } w = R \\ R(w) & = & 0 & \textit{otherwise} \end{array}$$

These constants have interesting properties when multiplied by an arbitrary tree. $L \times \sigma$ produces a tree whose root and right subtree 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:

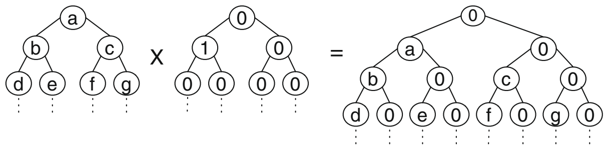


As before, if we see L and σ as languages and the product as concatenation, we can gain some intuition on the meaning of this operation. $L \times \sigma$ will prefix every word of σ with the letter L, meaning that no word starting by R will be an element of $L \times \sigma$, and thus, $L \times \sigma$ has a null right branch. Similar for $R \times \sigma$.

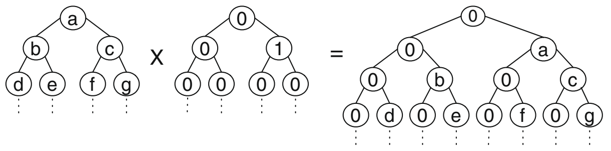
As we pointed out before, the product operation is not commutative. For example, $\sigma \times L \neq L \times \sigma$ and $\sigma \times R \neq R \times \sigma$. In fact, multiplying a tree σ on the right with L or R is interesting in itself. For instance, $\sigma \times L$ satisfies

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

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 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 :*

$$\sigma^{-1} \textit{ is the unique tree s.t. } \sigma \times \sigma^{-1} = 1 \tag{3}$$
$$(\sigma \times \tau)^{-1} = \tau^{-1} \times \sigma^{-1} \tag{4}$$

Proof. For the existence part of (3), 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 (3), 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 (4), 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 (3), $(\sigma \times \tau)^{-1} = \tau^{-1} \times \sigma^{-1}$. □

5. Applications of tree calculus

We will illustrate the usefulness of our calculus by looking at a series of interesting examples.

Throughout this section we will use different semirings. When we do not specify the semiring, the example is valid for an arbitrary semiring.

In order to compute closed formulae for trees we will be using the following theorem, that will enable us to solve behavioural differential equations in an algebraic manner.

Theorem 6. *For all $\sigma \in T_A$, $\sigma = \sigma(\varepsilon) + (L \times \sigma_L) + (R \times \sigma_R)$.*

Proof. The theorem follows by coinduction from the fact that

$$S = \{\langle \sigma, \sigma(\varepsilon) + (L \times \sigma_L) + (R \times \sigma_R)\rangle \mid \sigma \in T_A\} \cup \{(\sigma, \sigma) \mid \sigma \in T_A\}$$

is a bisimulation. □

We will now show how to use this theorem to construct a closed formula for a tree.

Recall our first system of behavioural differential equations:

Differential equations	Initial value
$one_L = one$ $one_R = one$	$one(\varepsilon) = 1$

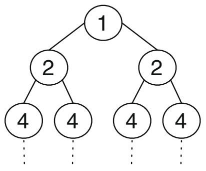
There we saw that the unique solution for this system was the tree with 1's in every node. Alternatively, we can compute the solution using Theorem 6 as follows:

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

Therefore, the tree *one* can be represented by the (very compact) closed formula $(1 - L - R)^{-1}$. Note the similarity of this closed formula with the one obtained for the stream $(1, 1, \ldots)$ in [9]: $(1 - X)^{-1}$.

Let us see a few more examples. In the following two examples we will work with $A = \mathbb{R}$.

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



is defined by the following system:

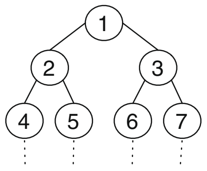
Differential equations	Initial value
$pow_L = 2 \times pow$ $pow_R = 2 \times pow$	$pow(\varepsilon) = 1$

We proceed as before, applying Theorem 6:

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

which gives us a nice closed formula for *pow*. Again, there is a strong similarity with streams: the closed formula for the stream $(1, 2, 4, 8, \ldots)$ is $(1-2X)^{-1}$.

The tree with the natural numbers



is represented by the following system of differential equations:

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

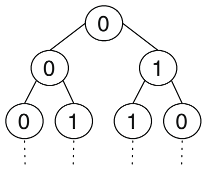
Applying Theorem 6, we have:

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

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 $\{0 \to 01;\ 1 \to 10\}$:

$$0 \to 01 \to 0110 \to 01101001 \to \ldots$$

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
$thue_L = thue$ $thue_R = thue + one$	$thue(\varepsilon) = 0$

Note that $thue + one$ equals the (elementwise) complement of *thue*. Applying Theorem 6 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*:

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

It is interesting to compare this formula with the *regular expression* that describes the corresponding *language* $\mathcal{L}(thue) \in \{L, R\}^*$ (cf. Eq. (2)), which is given by

$$\mathcal{L}(thue) = \{w \in \{L, R\}^* \mid thue(w) = 1\}$$

Putting

$$M = \mathcal{L}(thue), \quad N = \mathcal{L}(thue + one)$$

the above equations for the tree *thue* (together with Theorem 6) lead to the following *language* equation for M:

$$M = (L \times M) + (R \times N)$$

where $\times$ denotes language concatenation, $+$ denotes language union, and $1 = \{\varepsilon\}$.

Similarly, computing left and right derivatives for *thue* + *one* leads to a *language* equation for N:

$$N = (L \times N) + (R \times M) + 1$$

Solving these equations as usual – notably using

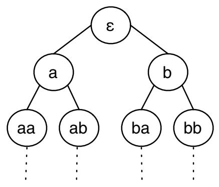
$$A = (B \times A) + C \quad \Rightarrow \quad A = B^* \times C$$

for languages A, B, C such that $\varepsilon \notin B$ — we find the following regular expression for M:

$$M = (L + (R \times L^* \times R))^* \times R \times L^*$$

Somehow the tree expression for *thue* above is simpler and nicer.

The Cantor space is the collection of all infinite sequences over a two element set. Typically, this set is {0, 1}, but to avoid confusion with the semiring units we will take $\{a, b\}$. The Cantor space can be represented as a tree:



In this example, we take for A the semiring of languages over a two-letter alphabet $2^{\{a,b\}^*}$, where $1 = \{\varepsilon\}$, $0 = \emptyset$, $+$ and $\times$ are, respectively, language union and concatenation. Note that each node of the above tree denotes in fact not a word but the language containing a singleton element.

The following system of differential equations defines *cantor*:

Differential equations	Initial value
$cantor_L = a \times cantor$ $cantor_R = b \times cantor$	$cantor(\varepsilon) = 1$

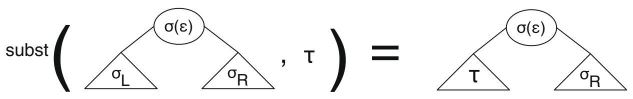
Applying Theorem 6 to *cantor*, we have:

$$\begin{array}{rlcl} & cantor & = & (L \times a \times cantor) + (R \times b \times cantor) \\ \Leftrightarrow & (1 - aL - bR) \times cantor & = & 1 \\ \Leftrightarrow & cantor & = & (1 - aL - bR)^{-1} \end{array}$$

which gives us a very compact and pleasant closed formula for *cantor*.

Note that in this example there are two alphabets at stake, the one denoting the tree branches $\{L, R\}$ and the one for the words in the language $\{a, b\}$. The interplay between this two alphabets is clearly reflected in the closed formula obtained.

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



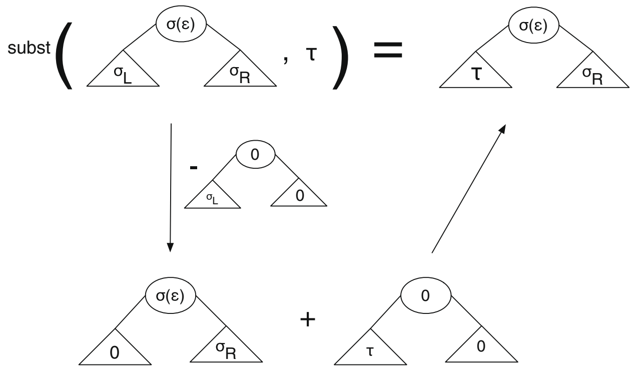
It is easy to see that the equations that define this operation are:

Differential equations	Initial value
$subst(\sigma,\tau)_L = \tau$ $subst(\sigma,\tau)_R = \sigma_R$	$subst(\sigma,\tau)(\varepsilon) = \sigma(\varepsilon)$

Then, we apply Theorem 6 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 Theorem 6 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, $P = a_1a_2\ldots a_n$ 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

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

is a bisimulation relation because:

(1) $(\sigma - P(\sigma_P - \tau))(\varepsilon) = \sigma(\varepsilon) = subst(\sigma,\tau,P)(\varepsilon)$
(2) For $\delta \in \{L,R\}$,

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

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

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

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

The first equality follows easily: $subst(\sigma,\sigma_P,P)=\sigma-P(\sigma_P-\sigma_P)=\sigma$.

For the second one we have:

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

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. Infinite trees as generalizations of (bi-)infinite streams

Infinite binary trees can be seen as generalizations of other well-known data structures. In this section, we will show how the sets of infinite and bi-infinite streams can be seen as special instances of T_A.

Let A be a semiring. The set of infinite streams over A is formally defined as

$$A^\omega=\{s\mid s:\omega\to A\}$$

The set A^ω carries a final coalgebra structure for the functor $GX=A\times X$ consisting of the following pair of functions:

$$\langle h,t\rangle : A^\omega\to A\times A^\omega,\;\; s\mapsto\left\langle s(0),s'\right\rangle$$

These assign to a stream $s=(s_0,s_1,s_2,\ldots)$ its initial value $s(0)=s_0\in A$ and its derivative $s'=(s_1,s_2,\ldots)\in A^\omega$.

We can now define the embedding of A^ω into T_A:

$$\begin{aligned}&f:A^\omega\to T_A\\ &f(s)(\varepsilon)=s(0)\\ &f(s)_L=f(s)_R=f(s')\end{aligned}$$

For a given stream s, $f(s)$ is a tree where every node at level k is labelled by $s(k)$. Defining an appropriate transition structure on A^ω for the functor $FX=X\times A\times X$, we can prove that f is a coalgebra homomorphism, *i.e*, the following diagram commutes:

$$\begin{tikzcd} A^\omega \arrow[r, "f"] \arrow[d, "\langle t,h,t\rangle"'] & T_A \arrow[d, "\langle l,i,r\rangle"] \\ A^\omega\times A\times A^\omega \arrow[r, "f\times id\times f"'] & T_A\times A\times T_A \end{tikzcd}$$

Thus, $S=f(A^\omega)\cong A^\omega$ is a subcoalgebra of T_A, *i.e.*, one can define a transition structure on S such that the inclusion map $i:S\to T_A$ is a coalgebra homomorphism. In this case, the above diagram shows that the required transition map on S is simply the restriction of $\langle l,i,r\rangle$ to S.

Moreover, S can also be characterised as the greatest subcoalgebra $\Box P$ contained in the following predicate P:

$$P=\{\sigma\in T_A\mid\sigma_L=\sigma_R\}$$

Proposition 1. $S=\Box P$.

Proof. The inclusion $S \subseteq \Box P$ follows from the fact that S is a subcoalgebra and that $S \subseteq P$:

$$\begin{array}{lll} \sigma \in S & \Leftrightarrow & \exists_{s \in A^\omega}\, \sigma = f(s) \\ & \Rightarrow & \sigma_L = f(s)_L = f(s)_R = \sigma_R \\ & \Leftrightarrow & \sigma \in P \end{array}$$

To prove the inclusion $\Box P \subseteq S$ let us first spell out what it means that $\sigma \in \Box P$:

$$\sigma \in \Box P \quad \Leftrightarrow \quad \sigma_w = \sigma'_w \quad \textit{for all } w, w' \in \{L,R\}^* \textit{ s.t. } |w| = |w'|$$

where $|\cdot|$ returns the length of a given word. This pointwise characterization of $\Box P$ comes from unfolding the definition of what it means to be a greatest subcoalgebra contained in P:

$$\begin{array}{lll} \sigma \in \Box P & \Leftrightarrow & \sigma \in P \textit{ and } \sigma_L, \sigma_R \in \Box P \\ & \Leftrightarrow & \sigma_L = \sigma_R \textit{ and } \sigma_L \in P, \sigma_R \in P \textit{ and } \sigma_{LL}, \sigma_{LR}, \sigma_{RL}, \sigma_{RR} \in \Box P \\ & \Leftrightarrow & \sigma_L = \sigma_R \textit{ and } \sigma_{LL} = \sigma_{LR} \textit{ and } \sigma_{RL} = \sigma_{RR} \textit{ and } \cdots \\ & \vdots & \end{array}$$

Formally, one still has prove that the set

$$P_1 = \left\{ \sigma \in T_A \mid \sigma_w = \sigma'_w \textit{ for all } w, w' \in \{L,R\}^* \textit{ s.t. } |w| = |w'| \right\}$$

is indeed the greatest subcoalgebra contained in P. The proof that $P_1 \subseteq P$ and that P_1 is a subcoalgebra follows easily from the definitions. Proving that is the greatest subcoalgebra follows also easily by induction on $|w|$.

Now, define $s \in A^\omega$, for a given $\sigma \in \Box P$, by $s(n) = \sigma(w)$, for any w such that $|w| = n$ and observe that $f(s) = \sigma$. Therefore, $\sigma \in S$. $\Box$

Next we give a similar such characterisation for bi-infinite streams.

The set of bi-infinite streams over A, for a given semiring A, is formally defined as

$$A^{\mathbb{Z}} = \{s \mid s : \mathbb{Z} \to A\}$$

The set $A^{\mathbb{Z}}$ has a dynamics given by the following three maps:

$$A^{\mathbb{Z}} \xrightarrow{\langle s_l, o, s_r \rangle} A^{\mathbb{Z}} \times A \times A^{\mathbb{Z}}$$

These assign to a bi-infinite stream $b = \left(\ldots, b_{-1}, \underline{b_0}, b_1, \ldots\right)$ its initial value $b(0) = b_0 \in A$, its *left shift* $s_l(b) = \left(\ldots b_{-1}, b_0, \underline{b_1}, b_2, \ldots\right) \in A^{\mathbb{Z}}$ and its *right shift* $s_l(b) = \left(\ldots b_{-2}, \underline{b_{-1}}, b_0, b_1, \ldots\right) \in A^{\mathbb{Z}}$. Note that the maps s_l and s_r have the property $s_l \circ s_r = s_r \circ s_l = id$.

We can now define the embedding of $A^{\mathbb{Z}}$ into T_A:

$$\begin{array}{l} g : A^{\mathbb{Z}} \to T_A \\ g(b)(\varepsilon) = b(0) \\ g(b)_L = g(s_l(b)) \\ g(b)_R = g(s_r(b)) \end{array}$$

The map g is a coalgebra homomorphism, *i.e*, the following diagram commutes:

$$\begin{tikzcd} A^{\mathbb{Z}} \arrow[r, "g"] \arrow[d, "<s_l, o, s_r>"'] & T_A \arrow[d, "<l, i, r>"] \\ A^{\mathbb{Z}} \times A \times A^{\mathbb{Z}} \arrow[r, "g \times id \times g"'] & T_A \times A \times T_A \end{tikzcd}$$

Thus, $B = f(A^{\mathbb{Z}}) \cong A^{\mathbb{Z}}$ is a subcoalgebra of T_A.

Moreover, B can be characterised as the greatest subcoalgebra $\Box Q$ contained in the following predicate Q:

$$Q = \{\sigma \in T_A \mid \sigma_{LR} = \sigma = \sigma_{RL}\}$$

Proposition 2. $B = \Box Q$.

Proof. The proof that $B = \Box Q$ is similar to the corresponding proof for infinite streams.

The inclusion $B \subseteq \Box Q$ follows from the fact that B is a subcoalgebra and that $B \subseteq Q$:

$$\begin{aligned} \sigma \in B \quad & \Leftrightarrow \quad \exists_{b \in A^{\mathbb{Z}}}\, \sigma = g(b) \\ & \Rightarrow \quad \sigma_{RL} = (g \circ s_r \circ s_l)(b) = g(b) = (g \circ s_l \circ s_r)(b) = \sigma_{LR} \\ & \Leftrightarrow \quad \sigma \in Q \end{aligned}$$

To prove the inclusion $\Box Q \subseteq B$ we spell out what it means that $\sigma \in \Box Q$:

$$\sigma \in \Box Q \quad \Leftrightarrow \quad \sigma_w = \sigma'_w \quad \textit{for all } w, w' \in \{L,R\}^* \textit{ s.t } |w|_a = |w'|_a\,,\ a \in \{L,R\}$$

where $|\cdot|_a$ returns the number of occurrences of a in a given word. As before, the formal proof that this pointwise characterization of $\Box Q$ is correct is omitted. The intuition behind it is similar to the one presented above for $\Box P$.

Now, define $b \in A^{\mathbb{Z}}$, for a given $\sigma \in \Box Q$, by $b(z) = \sigma(w)$, for any w such that $|w|_R - |w|_L = z$ and observe that $g(b) = \sigma$. Therefore, $\sigma \in B$. $\Box$

7. Rational binary trees

We introduce the family of rational trees. Rational trees are important because they are exactly the trees that can be represented by closed formulae. We compare our definition of rationality with existing notions. We prove that our definition of rationality is more expressive than the one presented in [6] and that it coincides with the one given for formal power series in [3].

All the examples presented so far are rational trees.

We define the set $\mathcal{R}$ of rational trees as the smallest subset of T_A (for a ring A) such that:

(1) $[n] \in \mathcal{R}$, for all $n \in A$
(2) $L, R \in \mathcal{R}$
(3) For all σ and τ in $\mathcal{R}$, $\sigma + \tau, \sigma \times \tau$ are also in $\mathcal{R}$
(4) For all σ in $\mathcal{R}$, such that $\sigma(\varepsilon)$ is invertible in A, σ^{-1} is also in $\mathcal{R}$

The expressions in $\mathcal{R}$ are given by the following grammar:

$$\sigma, \tau ::= [n], n \in A \mid L \mid R \mid \sigma + \tau \mid \sigma\tau \mid \sigma^{-1}\ (\sigma(\varepsilon) \text{ invertible in } A)$$

Next, we recall two existing notions of rationality from the literature.

Definition 1 ([6, p. 424])**.** A tree t is rational if it has only a finite number of different subtrees.

Our definition of rational is more general than this one. As an example take the tree *nat* of natural numbers. Obviously, it has an infinite number of different subtrees and it is still rational in our setting.

Definition 2 ([3, p. 6])**.** A formal power series is rational if it is an element of the rational closure of $k\langle X\rangle$.

$k\langle X\rangle$ is the set of polynomials (formal power series with finite support) over X with coefficients in k. By finite support we mean that for $\sigma \in k\langle X\rangle$ there is a finite number of words $w \in X^*$ such that $\sigma(w) \neq 0$. The rational closure of $k\langle X\rangle$ is the smallest set containing $k\langle X\rangle$ that is closed under the rational operations: sum, product, external products and star. There are two external product operations of k on $k\langle X\rangle$ defined, for $a \in k$, as $(a\sigma)(w) = a \times \sigma(w)$ and $(\sigma a)(w) = \sigma(w) \times a$. The star operator is defined as $\sigma^* = \sum_{n \geq 0} \sigma^n$. This definition is only valid for formal power series σ such that $\sigma(\varepsilon) = 0$ (because in this case the family $(\sigma^n)_{n\geq 0}$ is locally finite and summable [3]).

If we restrict definition 2 to formal power series over two variables, one can prove that the rational closure of $A\langle\{L,R\}\rangle$, which we will denote by $\mathcal{R}_{BR}$, is given by the following syntax:

$$\sigma, \tau ::= [n], n \in A \mid L \mid R \mid \sigma + \tau \mid \sigma\tau \mid \sigma^*\ (\sigma(\varepsilon) = 0)$$

The following theorem states the relation between this notion of rationality and ours.

Theorem 7. *The set of rational trees $\mathcal{R}$ coincides with the set of rational formal power series over two variables, as defined in [3].*

Proof. We will prove this theorem by induction on the syntax of the expressions. In fact, the syntax definitions for $\mathcal{R}$ and $\mathcal{R}_{BR}$ are very similar. They only differ in the use of star and inverse.

It easy to see that $[n], L, R, \sigma + \tau, \sigma\tau \in \mathcal{R} \Leftrightarrow [n], L, R, \sigma + \tau, \sigma\tau \in \mathcal{R}_{BR}$. Therefore, in order to conclude that $\mathcal{R}$ and $\mathcal{R}_{BR}$ are equal, we only have to show that:

$$\sigma^* \in \mathcal{R}_{BR} \Rightarrow \sigma^* \in \mathcal{R} \tag{5}$$

$$\sigma^{-1} \in \mathcal{R} \Rightarrow \sigma^{-1} \in \mathcal{R}_{BR} \tag{6}$$

For (5), observe that, if A is a ring then σ^* is the inverse of $1 - \sigma$ and, for $\sigma \in \mathcal{R}$, $(1-\sigma)^{-1} \in \mathcal{R}$. To see that σ^* is the inverse of $1 - \sigma$ note that

$$\sigma^*(1-\sigma) = \sigma^* - \sigma^*\sigma = \sigma^* - (\sigma^* - 1) = 1$$

Here, we use the fact that $\sigma^*\sigma = \sigma^+ = \sum_{n\geq 1}\sigma^n = \sigma^* - 1$ (for a more detailed proof we refer to [3]).

For (6), note that applying Theorem 6 to σ^{-1}, we have

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

Now, because $\left(\left(L \times -\sigma(\varepsilon)^{-1} \times \sigma_L\right) + \left(R \times -\sigma(\varepsilon)^{-1} \times \sigma_R\right)\right)(\varepsilon) = 0$, we know (using [3, Lemma 4.1]) that the solution for the equation $\sigma^{-1} = \sigma(\varepsilon)^{-1} + \left(\left(L \times -\sigma(\varepsilon)^{-1} \times \sigma_L\right) + \left(R \times -\sigma(\varepsilon)^{-1} \times \sigma_R\right)\right)\sigma^{-1}$ is $\sigma^{-1} = \left(\left(L \times -\sigma(\varepsilon)^{-1} \times \sigma_L\right) + \left(R \times -\sigma(\varepsilon)^{-1} \times \sigma_R\right)\right)^*\sigma(\varepsilon)^{-1}$, which is an element of $\mathcal{R}_{BR}$. $\square$

8. 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. It would also be interesting to study systematically the relation between tree expressions and, in the case $A = \{0, 1\}$, the regular expressions for the correspondent languages (we saw an example of this for the *thue* tree). (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.

Acknowledgments

We thank Clemens Kupke, Paulo Oliva and the anonymous referee for valuable suggestions and discussions.

References

[1] J.-P. Allouche, J. Shallit, The ubiquitous Prouhet–Thue–Morse sequence, in: C. Ding, T.N.H. Helleseth (Eds.), Sequences and Their Applications Proceedings of SETA'98, Springer, Berlin, 1999, pp. 1–16.
[2] J.-P. Allouche, J. Shallit, Automatic Sequences: Theory, Applications, Generalizations, Cambridge University Press, Cambridge, 2003..
[3] J. Berstel, C. Reutenauer, Rational Series and Their Languages, Springer, New York, NY, 1988..
[4] Z. Ésik, W. Kuich, Formal tree series, Journal of Automata, Languages and Combinatorics 8 (2) (2003) 219–285.
[5] E.G. Manes, M.A. Arbib, Algebraic Approaches to Program Semantics, Springer, New York, NY, 1986..
[6] D. Perrin, J.-E. Pin, Infinite Words, Pure and Applied Mathematics, vol. 141, Elsevier, Amsterdam, 2004, ISBN: 0-12-532111-2.
[7] J.J.M.M. Rutten, Universal coalgebra: a theory of systems, Theoretical Computer Science 249 (1) (2000) 3–80.
[8] J.J.M.M. Rutten, Behavioural differential equations: a coinductive calculus of streams, automata, and power series, Theoretical Computer Science 308 (1–3) (2003) 1–53.
[9] J.J.M.M. Rutten, A coinductive calculus of streams, Mathematical Structures in Computer Science 15 (1) (2005) 93–147.