

# Stream Differential Equations: concrete formats for coinductive definitions

Clemens Kupke<sup>1</sup>, Milad Niqui<sup>2</sup> and Jan Rutten<sup>2,3</sup>

<sup>1</sup>University of Oxford,

<sup>2</sup>Centrum Wiskunde & Informatica, Amsterdam,

<sup>3</sup>Radboud Universiteit Nijmegen

December 16, 2011

## Abstract

In this article we give an accessible introduction to stream differential equations, i.e., equations that take the shape of differential equations from analysis and that are used to define infinite streams. Furthermore we discuss a syntactic format for stream differential equations that ensures that any system of equations that fits into the format has a unique solution. It turns out that the stream functions that can be defined using our format are precisely the causal stream functions. Finally, we are going to discuss non-standard stream calculus that uses basic (co-)operations different from the usual head and tail operations in order to define and to reason about streams and stream functions.

## 1 Introduction

In recent years, coinduction has come to play an ever more important role in the theory of computing and computer science. Coinduction is a method for specifying and reasoning about infinite data types such as streams (i.e., infinite sequences) and, more generally all kinds of automata with circular behaviour. Coinduction is studied in various disciplines, including type theory [Coq94, Ber05, GM03] and functional programming [Gor94, Hin08], modal logic [Mos99], and automata theory [Rut98, KV08].

The notion of coinduction is dual to the classical, well-known notion of induction. This duality is best explained by the theory of *coalgebra* [JR97, Rut00]. Since the early nineties, coalgebra has become an active area of research in which one tries to understand all kinds of infinite data types, automata, transition systems and dynamical systems from a unifying perspective. The focus of coalgebra is on observable behaviour and one uses coinduction as a central methodology, both for behavioural specifications and to prove behavioural equivalences. As such, coalgebra builds on the pioneering work of Milner and Park [Mil80, Par81] on observational equivalence and bisimulation.

Classical definition and proof principles such as well-founded induction and recursion belong to mainstream mathematics, and they are common knowledge for many computer scientists as well. However, a common understanding of coinductive definition and proof principles is still lacking.

In ongoing research on coinduction, one tries to find suitable *formats* for the formulation of coinductive definition and proof principles. In the world of functional programming and

coalgebra, these formats have often been defined as the dual of familiar inductive notions such as iteration and (primitive) recursion, leading to very general and abstract schemes such as ( $\lambda$ -)coiteration and (primitive) corecursion (cf. [Bar03] and [Geu92, MD97, UV99]). Many of the existing formalisms for coinduction are often very general but too abstract for easy use, or concrete enough but limited in their applicability. In this paper, we shall therefore focus on one very concrete family of coinductive specifications called *behavioural differential equations* (BDEs) that seem to offer a good compromise between generality and practicality.

BDEs are a behavioural variant of classical differential equations of mathematical analysis. They were introduced in [Rut03], building on earlier work in automata theory [Brz64, Con71], formal power series [BR88, McI99], and coalgebra and logic [Rut98, PE98, Rut99]. They are particularly well suited for the specification of behaviour that involves combinations of both quantitative and qualitative aspects. The usefulness of BDEs is illustrated by many recent applications, including work in the following areas: automata and formal languages [Rut98], analytical differential equations [PE98, Rut05], denotational semantics [BG06], control of discrete event systems [KvS08], synthesis [Rut06, HR10], binary trees [SR10], stream processing networks [NR10], concurrent timed systems [Kom10], and language-based security protocols [MBG10].

## 1.1 Stream differential equations

In the present paper, we shall focus on a very specific but at the same time prototypical instance of BDEs, namely *stream differential equations* (SDEs) [Rut05]. Below we introduce the basic notions involved, discuss some examples, and give an outline of the remainder of this paper.

We define the set of all *streams* over a given set  $A$  by

$$A^\omega = \{\sigma \mid \sigma : \mathbb{N} \rightarrow A\}$$

We shall sometimes denote individual streams by

$$\sigma = (\sigma(0), \sigma(1), \sigma(2), \dots)$$

We define the *initial value* of  $\sigma$  by  $\sigma(0)$  and we define the *stream derivative*  $\sigma'$  of  $\sigma$  by

$$\sigma' = (\sigma(1), \sigma(2), \sigma(3), \dots)$$

We can view streams as states of an abstract machine, for which initial value and derivative together determine the *behaviour*: one can think of the initial value  $\sigma(0)$  as an (initial) observation on  $\sigma$ , after which it moves to a next state  $\sigma'$ . The initial value  $\sigma'(0)$  of  $\sigma'$ , which is equal to  $\sigma(1)$ , will then be the next observation on  $\sigma$ , and so on. In this manner, a stream presents all of its values one by one, by means of what is called lazy evaluation in functional programming.

The terminology of initial value and derivative – rather than the more common head and tail – has been chosen to emphasize a close analogy with classical differential equations from mathematical analysis: we can define streams and streams functions by means of so-called *stream differential equations* (SDEs). Let us look at a few examples, considering streams of natural numbers, that is, putting  $A = \mathbb{N}$ .

A trivial example of a stream differential equation is

$$\sigma' = \sigma \quad \sigma(0) = 1$$

which uniquely determines the stream  $\sigma = (1, 1, 1, \dots)$ . The following SDE defines the stream of Fibonacci numbers  $(0, 1, 1, 2, 3, 5, 8, \dots)$ :

$$\sigma'' = \sigma' + \sigma \quad \sigma(0) = 0 \quad \sigma(1) = 1 \quad (1)$$

Here  $+$  is element-wise addition of streams, which itself can also be defined by a BDE, as follows:

$$(\sigma + \tau)' = \sigma' + \tau' \quad (\sigma + \tau)(0) = \sigma(0) + \tau(0)$$

This example illustrates that SDEs can be used to define not only streams but also functions on streams.

The stream of Fibonacci numbers can be equivalently defined by the following recurrence relation:

$$\sigma(n+2) = \sigma(n+1) + \sigma(n) \quad \sigma(0) = 0 \quad \sigma(1) = 1$$

However, not for all SDEs such an equivalent definition by a recurrence relation exists. For instance, the following SDE defines the stream of the so-called *Hamming numbers*, consisting of all natural numbers of the form  $2^i 3^j 5^k$ , for  $i, j, k \geq 0$ , in increasing order (cf. [Dij81, Bar03]):

$$\sigma' = (2 \times \sigma) \parallel (3 \times \sigma) \parallel (5 \times \sigma) \quad \sigma(0) = 1 \quad (2)$$

Here  $(k \times \sigma)(n) = k \times (\sigma(n))$ , for all  $k = 2, 3, 5$  and all  $n \geq 0$ ; and the operator  $\parallel$  zips the three streams by always taking the smallest initial value first, removing doubles if necessary; a formal definition follows later, in Section 3. There is no obvious way to define the stream of Hamming numbers by means of a recurrence relation.

## 1.2 Solving SDEs

As with classical differential equations in mathematical analysis, a SDE can have the following three properties:

- (i) It is well-defined; that is, it has a unique solution.
- (ii) This solution is computable: we have an algorithm for computing all its finite approximations.
- (iii) The solution has a closed form: we can express it in terms of a given set of basic operations.

In most cases, property (iii) implies property (ii) which at its turn implies property (i). Although it might make sense to allow (nondeterministic) specifications with more than one solution, we shall typically view condition (i) as a basic requirement. For any kind of practical use, SDEs should better also satisfy (ii). In contrast, there will be only rare cases where we have a closed form (condition (iii)) for the solution of a SDE.

Let us illustrate these properties with a few examples. The equation  $\sigma' = \sigma$  without an initial solution obviously has many solutions, namely, every constant stream is a solution. A more interesting example is given by the following SDE:

$$c'' = \text{even}(c) \quad c(0) = 0 \quad c(1) = 0 \quad (3)$$

where the function `even` is defined by the following SDE:

$$\text{even}(\sigma)' = \text{even}(\sigma'') \quad \text{even}(\sigma)(0) = \sigma(0)$$

One can easily prove that

$$\text{even}(\sigma(0), \sigma(1), \sigma(2), \dots) = (\sigma(0), \sigma(2), \sigma(4), \dots)$$

We observe that the SDE (3) above for the stream  $c$  has many different solutions, such as  $(0, 0, 0, 0, 0, \dots)$  and  $(0, 0, 0, 0, 1, 1, 1, \dots)$ . So this is a not entirely trivial example<sup>1</sup> of a SDE that does not satisfy property (i).

As we will see in Section 3, the SDE defining the Hamming numbers (equation (2) above) has a unique solution. Furthermore, the SDE itself, together with the corresponding SDEs for the operations of merge and multiplication, can be used to compute arbitrary initial segments of the stream of all Hamming numbers. Thus this SDE satisfies (ii). As far as we can tell, however, there exist in the literature no proposal for a closed form, and so this stream does not seem to satisfy condition (iii).

An example of a stream satisfying all of (i), (ii) and (iii) is the stream of the Fibonacci numbers, defined by the SDE (1). As we shall see in Section 5, its unique solution can be expressed in stream calculus (in a way that is reminiscent of generating functions) as

$$\frac{X}{1 - X - X^2} = (0, 1, 1, 2, 3, 5, 8, 13, \dots)$$

In spite of the above and other similar such examples, there hardly exist *general criteria* that allow one to decide whether a SDE satisfies any of the properties above. This is in contrast with mathematical analysis, where one studies various well-defined classes of differential equations such as linear and non-linear equations, homogeneous and non-homogeneous equations, and the like.

### 1.3 Overview

In the remainder of this paper, we shall present partial answers to some of the questions about SDEs raised above. In Section 3, we shall sketch a solution method with which a unique solution for the SDE defining the Hamming numbers (equation (2)) can be constructed. This so-called syntactic method will then be generalised, in Section 4, where we shall introduce a general *format* for a large class of SDEs. Moreover, we shall prove that the class of stream functions one can define with these SDEs contains precisely all so-called *causal* stream functions. In Section 5, we present yet another format, called *linear* SDEs. For linear SDEs, we are able to construct unique solutions, which can be presented by *rational* expressions. The stream of Fibonacci numbers that we saw above is an example. Thus the class of linear SDEs is an example where properties (i) well-definedness, (ii) computability, and (iii) solutions have a closed form, are all satisfied. Finally, in Section 6, we shall illustrate the generality of the concept of SDEs by a discussion of what we have called non-standard stream calculus. Before all this, we first shall discuss a few further preliminaries, in Section 2.

### Related Work on Productivity

The ideas presented in this article, in particular in Section 4 on the syntactic format for defining causal stream functions, aim in a similar direction as the work on (data-oblivious) productivity of stream definitions in [EGH08, EGH<sup>+</sup>10]. The two main distinguishing features

---

<sup>1</sup>The example is due to Joerg Endrullis.

of our approach are on the one hand, that we are using coalgebraic techniques. On the other hand, we are ensuring productivity by devising a syntactic format such that *any definition in this format* of a stream or a stream function will be productive. This is in contrast to the approach in the cited papers where procedures are described that allow to decide, for a certain class of stream definitions, whether or not a given stream definition is productive. A feature of the coalgebraic framework is that the obtained results for streams can be easily generalised to other infinite structures such as infinite  $n$ -ary trees. Furthermore the theory can be developed parametric in the choice of basic (co-)operations - something we hint at in this article in Section 6 on non-standard stream calculus. These generalisations are discussed in detail in [KR10].

## 2 Preliminaries

We recall the definition of the set of *streams* over a set  $A$ :

$$A^\omega = \{\sigma \mid \sigma : \mathbb{N} \rightarrow A\}$$

where  $\mathbb{N}$  is the set of natural numbers including 0. We already defined the *stream derivative* of a stream  $\sigma = (\sigma(0), \sigma(1), \sigma(2), \dots)$  by

$$\sigma' = (\sigma(1), \sigma(2), \sigma(3), \dots)$$

and the *initial value* of  $\sigma$  by  $\sigma(0)$ .

A *stream system* over a set  $A$  is a pair  $(X, (o, t))$  consisting of a set  $X$  of states and a pair of maps consisting of an output (or observation) map  $o : X \rightarrow A$  and a transition (or next state) map  $t : X \rightarrow X$ . A *homomorphism* of stream systems  $(X, (o_X, t_X))$  and  $(Y, (o_Y, t_Y))$  is a map  $f : X \rightarrow Y$  such that, for all  $x \in X$ ,

$$o_Y(f(x)) = o_X(x) \quad \text{and} \quad t_Y(f(x)) = f(t_X(x))$$

Defining the maps of initial value  $\iota : A^\omega \rightarrow A$  and derivative  $\delta : A^\omega \rightarrow A^\omega$  by

$$\iota(\sigma) = \sigma(0) \quad \delta(\sigma) = \sigma'$$

turns the set of streams into a stream system

$$(A^\omega, (\iota, \delta))$$

It has the universal property of being *final* in the family of all stream systems over  $A$ : for every stream system  $(X, (o, t))$  there exists a *unique* stream homomorphism  $f : X \rightarrow A^\omega$ . It is given, for  $x \in X$  and  $n \geq 0$ , by

$$f(x)(n) = o(t^n(x))$$

where  $t^0 = t$  and  $t^{k+1} = t \circ t^k$ .

As we will see later, the finality of the set of streams, that is, the unique existence of stream homomorphisms into  $A^\omega$ , will form an important instrument to prove the unique existence of solutions of many stream differential equations.

Also, the finality of  $A^\omega$  will allow us to prove the equality of streams. In order to formulate this so-called *coinduction* proof principle for streams, we need the following notion. A *bisimulation relation* between stream systems  $(X, (o_X, t_X))$  and  $(Y, (o_Y, t_Y))$  is a set  $R \subseteq X \times Y$  such that for all  $(x, y) \in R$ ,

$$o_X(x) = o_Y(y) \quad \text{and} \quad (t_X(x), t_Y(y)) \in R$$

We write  $x \sim y$  if there exists a bisimulation  $R$  with  $(x, y) \in R$ . If we take both  $X$  and  $Y$  to be the stream system  $A^\omega$ , we obtain the following definition. A bisimulation on  $A^\omega$  is a relation  $R \subseteq A^\omega \times A^\omega$  such that, for all  $(\sigma, \tau) \in R$ ,

$$\sigma(0) = \tau(0) \quad \text{and} \quad (\sigma', \tau') \in R$$

The *coinduction proof principle* allows us to prove the equality of two streams by establishing the existence of an appropriate bisimulation relation.

**Theorem 2.1 (coinduction proof principle)** *For all sets  $A$ , for all streams  $\sigma, \tau \in A^\omega$ ,*

$$\sigma \sim \tau \Rightarrow \sigma = \tau$$

In other words: in order to prove  $\sigma = \tau$  it suffices to construct a stream bisimulation relation  $R \subseteq A^\omega \times A^\omega$  such that  $(\sigma, \tau) \in R$ .

**Proof:** Let  $R \subseteq A^\omega \times A^\omega$  be a stream bisimulation relation. If  $(\sigma, \tau) \in R$  then one can show by an easy induction that  $\sigma(n) = \tau(n)$ , for all  $n \geq 0$ . For an alternative proof, based on the finality of  $A^\omega$ , consider functions  $o : R \rightarrow A$  and  $t : R \rightarrow R$  defined, for all  $(\sigma, \tau) \in R$ , by

$$o((\sigma, \tau)) = \sigma(0) = \tau(0) \quad t((\sigma, \tau)) = (\sigma', \tau')$$

We note that  $o$  and  $t$  are well-defined because  $R$  is a bisimulation. This turns  $R$  into a stream system  $(R, (o, t))$ . Because both the projections  $\pi_1 : R \rightarrow A^\omega$  and  $\pi_2 : R \rightarrow A^\omega$ , defined by

$$\pi_1((\sigma, \tau)) = \sigma \quad \pi_2((\sigma, \tau)) = \tau$$

are homomorphisms of stream systems, the finality of  $A^\omega$  implies that  $\pi_1 = \pi_2$ . This means that  $\sigma = \tau$ , for all  $(\sigma, \tau) \in R$ .

(We note that the converse of the theorem holds as well, because the identity relation on  $A$  is a stream bisimulation relation.) QED

### 3 A syntactic solution method

In this section, we will show how to construct a unique solution for the stream differential equation for the Hamming numbers [Dij81, Bar03], which we already encountered as equation (2) in Section 1.1. We will be using a *syntactic method* to construct a (unique) solution for this SDE. In Section 4, the same method will be used, more generally, for SDEs defining so-called causal stream functions.

For (notational) convenience, the stream  $\gamma$  of the Hamming numbers will in our definition consist of natural numbers having no other prime factors than 2 and 3. The original version,

which allows prime factors 2, 3, and 5, can be treated in precisely the same way. Here are the first elements of  $\gamma$ :

$$\begin{aligned}\gamma &= (2^0 3^0, 2^1 3^0, 2^0 3^1, 2^2 3^0, 2^1 3^1, 2^3 3^0, 2^0 3^2, 2^2 3^1, \dots) \\ &= (1, 2, 3, 4, 6, 8, 9, 12, \dots)\end{aligned}$$

We define  $\gamma$  by the stream differential equation

$$\gamma' = (2 \times \gamma) \parallel (3 \times \gamma) \quad \gamma(0) = 1 \quad (4)$$

The ordered merge operator

$$\parallel : \mathbb{N}^\omega \times \mathbb{N}^\omega \rightarrow \mathbb{N}^\omega$$

is itself defined by the stream function differential equation

$$(\sigma \parallel \tau)' = \begin{cases} \sigma' \parallel \tau & \text{if } \sigma(0) < \tau(0) \\ \sigma' \parallel \tau' & \text{if } \sigma(0) = \tau(0) \\ \sigma \parallel \tau' & \text{if } \sigma(0) > \tau(0) \end{cases} \quad (\sigma \parallel \tau)(0) = \begin{cases} \sigma(0) & \text{if } \sigma(0) < \tau(0) \\ \tau(0) & \text{if } \sigma(0) \geq \tau(0) \end{cases} \quad (5)$$

where, for  $k = 2, 3$ , the function  $k \times (-) : \mathbb{N}^\omega \rightarrow \mathbb{N}^\omega$  is given by

$$(k \times \sigma)' = k \times (\sigma') \quad (k \times \sigma)(0) = k \cdot \sigma(0) \quad (6)$$

We shall use the fact that the stream system  $(\mathbb{N}^\omega, (\iota, \delta >))$  is final, which we saw in Section 2, to construct a solution for the SDE (4) above. The uniqueness of this solution will then follow by coinduction. In fact, our construction will construct a solution for all of the differential equations above, including those for  $\parallel$  and  $k \times$ .

Let us suppose for a moment that there exist a stream  $\gamma$  and operators  $\parallel$  and  $k \times$  that satisfy the equations above. If we compute the first few repeated stream derivatives of  $\gamma$ , using the above equations:

$$\begin{aligned}\gamma' &= (2 \times \gamma) \parallel (3 \times \gamma) \\ \gamma'' &= (2 \times ((2 \times \gamma) \parallel (3 \times \gamma))) \parallel (3 \times \gamma) \\ \gamma''' &= (2 \times ((2 \times \gamma) \parallel (3 \times \gamma))) \parallel (3 \times ((2 \times \gamma) \parallel (3 \times \gamma)))\end{aligned}$$

then we see that they exist of expressions denoting repeated compositions of applications of the functions  $\parallel$ ,  $2 \times$  and  $3 \times$  to the stream  $\gamma$ .

Next we formally introduce the set of all such expressions as follows. In order to distinguish between syntax and semantics, we first introduce new syntactic names: the letter  $\mathbf{c}$  will be used to denote  $\gamma$ ; the term `merge` denotes  $\parallel$ ; the term `2times` denotes  $2 \times$ ; and the term `3times` denotes  $3 \times$ . Moreover, we introduce for every stream  $\sigma \in \mathbb{N}^\omega$  a syntactic term  $\underline{\sigma}$ . We inductively define the set `Term` of all terms by

$$\text{Term} \ni t ::= \mathbf{c} \mid \underline{\sigma} \ (\sigma \in \mathbb{N}^\omega) \mid \text{2times}(t) \mid \text{3times}(t) \mid \text{merge}(t_1, t_2)$$

In order to use the finality of the stream system  $(\mathbb{N}^\omega, (\iota, \delta))$ , we turn the set `Term` of terms into a stream system  $(\text{Term}, (o, n))$ . To this end, we define functions

$$o : \text{Term} \rightarrow \mathbb{N} \quad n : \text{Term} \rightarrow \text{Term}$$

by induction on the structure of the terms as follows. The definition of  $o : \mathbf{Term} \rightarrow \mathbb{N}$  is based on the initial values of the SDEs (4), (5), and (6):

$$\begin{aligned} o(\mathbf{c}) &= 1 \\ o(\mathbf{merge}(t_1, t_2)) &= \begin{cases} o(t_1) & \text{if } o(t_1) < o(t_2) \\ o(t_2) & \text{if } o(t_1) \geq o(t_2) \end{cases} \\ o(\mathbf{2times}(t)) &= 2 \cdot o(t) \\ o(\mathbf{3times}(t)) &= 3 \cdot o(t) \end{aligned}$$

And for  $\sigma \in \mathbb{N}^\omega$ , we put

$$o(\underline{\sigma}) = \sigma(0)$$

The definition of  $n : \mathbf{Term} \rightarrow \mathbf{Term}$  is similarly based on the SDEs above:

$$\begin{aligned} n(\mathbf{c}) &= \mathbf{merge}(\mathbf{2times}(\mathbf{c}), \mathbf{3times}(\mathbf{c})) \\ n(\underline{\sigma}) &= \underline{\sigma'} \\ n(\mathbf{merge}(t_1, t_2)) &= \begin{cases} \mathbf{merge}(n(t_1), t_2) & \text{if } o(t_1) < o(t_2) \\ \mathbf{merge}(n(t_1), n(t_2)) & \text{if } o(t_1) = o(t_2) \\ \mathbf{merge}(t_1, n(t_2)) & \text{if } o(t_1) > o(t_2) \end{cases} \\ n(\mathbf{2times}(t)) &= \mathbf{2times}(n(t)) \\ n(\mathbf{3times}(t)) &= \mathbf{3times}(n(t)) \end{aligned}$$

(We note that the definition of  $n$  uses the definition of  $o$ .)

Now there exists, by the finality of  $(\mathbb{N}^\omega, (\iota, \delta))$ , a unique homomorphism

$$f : (\mathbb{N}^\omega, (\iota, \delta)) \rightarrow (\mathbf{Term}, (o, n))$$

We use  $f$  to define  $\gamma$  and our operators as follows:

$$\begin{aligned} \gamma &= f(\mathbf{c}) \\ \sigma \parallel \tau &= f(\mathbf{merge}(\underline{\sigma}, \underline{\tau})) \\ 2 \times \sigma &= f(\mathbf{2times}(\underline{\sigma})) \\ 3 \times \sigma &= f(\mathbf{3times}(\underline{\sigma})) \end{aligned}$$

**Proposition 3.1** *The stream  $\gamma$  and the functions  $\parallel$ ,  $2 \times$  and  $3 \times$  are the unique solutions of the differential equations (4), (5) and (6) above.*

**Proof:** We prove that  $\gamma$  satisfies equation (4) and leave the other statements to the reader. For  $\gamma$ , we have  $\gamma(0) = f(\mathbf{c})(0) = o(\mathbf{c}) = 1$ , using for the second equality the fact that  $f$  is a homomorphism. Furthermore,

$$\begin{aligned} \gamma' &= f(\mathbf{c})' \\ &= f(n(\mathbf{c})) \quad [\text{because } f \text{ is a homomorphism}] \\ &= f(\mathbf{merge}(\mathbf{2times}(\mathbf{c}), \mathbf{3times}(\mathbf{c}))) \\ &= f(\mathbf{2times}(\mathbf{c})) \parallel f(\mathbf{3times}(\mathbf{c})) \\ &= (2 \times f(\mathbf{c})) \parallel (2 \times f(\mathbf{c})) \\ &= (2 \times \gamma) \parallel (3 \times \gamma) \end{aligned}$$

Here the fourth and fifth equalities use the fact that  $f$  is compositional. A proof of this, as well as the proof of the uniqueness of the solution is omitted here. It will follow from the more general results in Section 4. QED

## 4 A format for defining causal stream functions

In the previous section we discussed a concrete example of a syntactic method for solving stream differential equations. We are going to use the example from the previous section as a guideline for setting up a general scheme for defining causal stream functions.

### 4.1 The basic system

In this section we are considering the set  $A^\omega$  of infinite streams over a set  $A$ . Here  $(-)(0) : A^\omega \rightarrow A$  is the function that maps a stream  $\sigma = \sigma_0\sigma_1\cdots$  to its first element  $\sigma_0 \in A$  and  $(-)' : A^\omega \rightarrow A^\omega$  denotes the tail operation.

A signature  $\Sigma$  is a collection of constant and operation symbols, each of which has a fixed arity. The intended interpretation of a  $k$ -ary symbol  $\underline{f} \in \Sigma$  will be a stream function  $f : (A^\omega)^k \rightarrow A^\omega$ . The core ingredient of our format for defining stream functions is the set  $\mathcal{T}_\Sigma$  of  $\Sigma$ -terms.

**Definition 4.1** Let  $\Sigma$  be a signature. We define the set  $\mathcal{T}_\Sigma(\mathcal{X})$  of  $\Sigma$ -terms over a set  $\mathcal{X}$  of generators inductively as follows:

$$\mathcal{T}_\Sigma(\mathcal{X}) \ni t ::= x \in \mathcal{X} \mid \underline{f}(t, \dots, t), \underline{f} \in \Sigma,$$

and we put  $\mathcal{T}_\Sigma := \mathcal{T}_\Sigma(A^\omega)$ , where  $A^\omega := \{\underline{\tau} \mid \tau \in A^\omega\}$ .

Let  $\mathcal{S} = \{f\}_{f \in \Sigma}$  be a collection of operations such that for any  $k$ -ary operation symbol  $\underline{f}$ ,  $f : (A^\omega)^k \rightarrow A^\omega$  is a  $k$ -ary operation. We define an interpretation  $\mathcal{I}_\mathcal{S} : \mathcal{T}_\Sigma \rightarrow A^\omega$  by putting  $\mathcal{I}_\mathcal{S}(\underline{\tau}) := \tau$  and  $\mathcal{I}_\mathcal{S}(\underline{f}(t_1, \dots, t_k)) = f(\mathcal{I}_\mathcal{S}(t_1), \dots, \mathcal{I}_\mathcal{S}(t_k))$ .

The set of terms comes equipped with the standard notion of substitution.

**Definition 4.2** For terms  $t, s_1, \dots, s_k \in \mathcal{T}_\Sigma(\mathcal{X})$  and distinct variables  $x_1, \dots, x_k$  we define the term  $t[x_1 := s_1, \dots, x_k := s_k]$ , in which variable  $x_i$  has been substituted by  $s_i$  for each  $i \in \{1, \dots, k\}$ , inductively as usual:

$$\begin{aligned} x[x_1 := s_1, \dots, x_k := s_k] &= \begin{cases} s_i & \text{if } x = x_i \text{ for some } i \in \{1, \dots, k\} \\ x & \text{otherwise,} \end{cases} \\ \underline{f}(t_1, \dots, t_n)[x_1 := s_1, \dots, x_k := s_k] &= \underline{f}(t_1[x_1 := s_1, \dots, x_k := s_k], \dots, t_n[x_1 := s_1, \dots, x_k := s_k]) \end{aligned}$$

Furthermore we write  $t[x_i := s_i]$  as shorthand for  $t[x_1 := s_1, \dots, x_k := s_k]$  if the set of indices  $i$  is clear from the context.

**Definition 4.3** Let  $\Sigma$  be a signature and let  $\underline{f} \in \Sigma$  be a  $k$ -ary operation symbol. A *stream equation* for  $\underline{f}$  is a pair  $\langle i, d \rangle$  (of “initial” value  $i$  and “derivative”  $d$ ) where

$$\begin{aligned} i &: A^k \rightarrow A \\ d &: A^k \rightarrow \mathcal{T}_\Sigma(\{x_1, \dots, x_k, y_1, \dots, y_k\}) \end{aligned}$$

A *stream definition* is a set  $\mathcal{D}$  of stream equations containing one equation  $\langle i_f, d_f \rangle$  for each  $\underline{f} \in \Sigma$ .

A *solution* of a stream definition is family of functions  $\mathcal{S} = \{f\}_{\underline{f} \in \Sigma}$  such that for every  $k$ -ary operation symbol  $\underline{f}$  the function  $f$  is of type

$$f : (A^\omega)^k \rightarrow A^\omega$$

and such that for all  $\underline{f} \in \Sigma$  we have

$$\begin{aligned} f(\tau_1, \dots, \tau_k)(0) &= i_f(\tau_1(0), \dots, \tau_k(0)) \\ f(\tau_1, \dots, \tau_k)' &= \mathcal{I}_{\mathcal{S}} \left( d_f(\tau_1(0), \dots, \tau_k(0)) [x_i := \tau_i, y_i := \tau_i'] \right) \end{aligned}$$

We will now prove that every stream definition  $\mathcal{D}$  has a unique solution  $\mathcal{S}$ . First of all we define a candidate for such a solution.

**Definition 4.4** Let  $\Sigma$  be a signature and let  $\mathcal{D} = \{\langle i_f, d_f \rangle\}_{\underline{f} \in \Sigma}$  be a stream definition for  $\Sigma$ . We define a map  $\llbracket \_ \rrbracket_{\mathcal{D}} : \mathcal{T}_{\Sigma} \rightarrow A^\omega$  as the unique coalgebra morphism from  $(\mathcal{T}_{\Sigma}, \langle o, n \rangle)$  into the final stream coalgebra:

$$\begin{array}{ccc} \mathcal{T}_{\Sigma} & \xrightarrow{\llbracket \_ \rrbracket_{\mathcal{D}}} & A^\omega \\ \langle o, n \rangle \downarrow & & \downarrow \langle (\_)(0), (\_)' \rangle \\ A \times \mathcal{T}_{\Sigma} & \xrightarrow{\text{id}_A \times \llbracket \_ \rrbracket_{\mathcal{D}}} & A \times A^\omega \end{array}$$

Here  $o : \mathcal{T}_{\Sigma} \rightarrow A$  (“output”) and  $n : \mathcal{T}_{\Sigma} \rightarrow \mathcal{T}_{\Sigma}$  (“next”) are defined inductively by putting

$$\begin{aligned} o(\tau) &= \tau(0) \\ o(\underline{f}(t_1, \dots, t_k)) &= i_f(o(t_1), \dots, o(t_k)) \\ n(\tau) &= \tau' \\ n(\underline{f}(t_1, \dots, t_k)) &= d_f(o(t_1), \dots, o(t_k)) [x_i := t_i, y_i := n(t_i)] \end{aligned}$$

Furthermore we let  $\mathcal{S}(\mathcal{D}) := \{\hat{f}\}_{\underline{f} \in \Sigma}$  where

$$\hat{f}(\tau_1, \dots, \tau_k) := \llbracket \underline{f}(\tau_1, \dots, \tau_k) \rrbracket_{\mathcal{D}} \in A^\omega.$$

**Remark 4.5** A reader familiar with coalgebras might find it useful to see how the existence of a unique solution  $\mathcal{S}$  of  $\mathcal{D}$  can be obtained as a corollary of results on  $\lambda$ -coiteration and bialgebras [Bar03]. One can think of a stream equation for an  $n$ -ary  $\underline{f} \in \Sigma$  as a specification of a natural transformation  $\rho^{\underline{f}} : (\text{Id} \times (A \times \text{Id}))^n \rightarrow A \times \mathcal{T}_{\Sigma}(\_)$ . Following the argument in Corollary 5.7 of [Bar03] this uniquely determines for every  $\underline{f} \in \Sigma$  a function  $\delta_f : (A^\omega)^n \rightarrow A^\omega$  which precisely corresponds to the unique solution of the stream definition. We prefer to derive the unique existence of a solution without directly referring to the result in [Bar03], because we want to keep this chapter as elementary and self-contained as possible.

In order to show that  $\mathcal{S}(\mathcal{D})$  is in fact the unique solution of  $\mathcal{D}$ , we will prove that solutions of  $\mathcal{D}$  and coalgebra morphisms from  $(\mathcal{T}_{\Sigma}, \langle o, n \rangle)$  into the final stream coalgebra are closely related. To begin with, we need two technical lemmas about bisimilarity on the coalgebra  $(\mathcal{T}_{\Sigma}, \langle o, n \rangle)$ .

**Lemma 4.6** For all  $t \in \mathcal{T}_\Sigma$  we have  $t \sim \llbracket t \rrbracket_{\mathcal{D}}$  where  $\sim$  denotes bisimilarity on  $(\mathcal{T}_\Sigma, \langle o, n \rangle)$ .

**Proof:** We prove by induction on  $t$  that the relation

$$R := \{(\llbracket t \rrbracket_{\mathcal{D}}, t) \mid t \in \mathcal{T}_\Sigma\}$$

is a bisimulation. For the base case consider  $t = \tau$  for some  $\tau \in A^\omega$ . Then  $o(\llbracket \tau \rrbracket_{\mathcal{D}}) = \llbracket \tau \rrbracket_{\mathcal{D}}(0) = o(\tau)$  and thus the output function agrees on both terms as required. Furthermore we have

$$n(\llbracket \tau \rrbracket_{\mathcal{D}}) = \llbracket \tau \rrbracket'_{\mathcal{D}} = \llbracket n(\tau) \rrbracket_{\mathcal{D}} \quad \text{and} \quad \left( \llbracket n(\tau) \rrbracket_{\mathcal{D}}, n(\tau) \right) \in R$$

and thus  $(n(\llbracket \tau \rrbracket_{\mathcal{D}}), n(\tau))$  as required.

Consider now some  $\underline{g}(t_1, \dots, t_k) \in \mathcal{T}_\Sigma$  with  $(\llbracket \underline{g}(t_1, \dots, t_k) \rrbracket_{\mathcal{D}}, \underline{g}(t_1, \dots, t_k)) \in R$ . We calculate

$$\begin{aligned} o(\llbracket \underline{g}(t_1, \dots, t_k) \rrbracket_{\mathcal{D}}) &= \llbracket \underline{g}(t_1, \dots, t_k) \rrbracket_{\mathcal{D}}(0) = o(\underline{g}(t_1, \dots, t_k)) \\ n(\llbracket \underline{g}(t_1, \dots, t_k) \rrbracket_{\mathcal{D}}) &= \llbracket \underline{g}(t_1, \dots, t_k) \rrbracket'_{\mathcal{D}} = \llbracket n(\underline{g}(t_1, \dots, t_k)) \rrbracket_{\mathcal{D}} \end{aligned}$$

and thus  $(n(\llbracket \underline{g}(t_1, \dots, t_k) \rrbracket_{\mathcal{D}}), n(\underline{g}(t_1, \dots, t_k))) \in R$  as required. QED

The second lemma states that bisimilarity on  $(\mathcal{T}_\Sigma, \langle o, n \rangle)$  is a congruence.

**Lemma 4.7** Let  $t \in \mathcal{T}_\Sigma(\{z_j\}_{j \in J})$  be a term with variables in  $\{z_j\}_{j \in J}$  and let  $\{s_j\}_{j \in J}$  and  $\{\hat{s}_j\}_{j \in J}$  be two families of terms  $s_j, \hat{s}_j \in \mathcal{T}_\Sigma$  such that  $s_j \sim \hat{s}_j$  for all  $j \in J$ . Then

$$t[z_j := s_j] \sim t[z_j := \hat{s}_j].$$

**Proof:** We define a family  $\{Z_i\}_{i \in \mathbb{N}}$  of relations and denote the union of all  $Z_i$  by  $Z$ :

$$\begin{aligned} Z_0 &:= \sim \\ Z_{n+1} &:= \{(t, \hat{t}) \in (\mathcal{T}_\Sigma)^2 \mid \exists t' \in \mathcal{T}_\Sigma(X) \exists s_1, \dots, s_m, \hat{s}_1, \dots, \hat{s}_m \in \mathcal{T}_\Sigma \text{ s.t. } t = t'[x_j := s_j], \\ &\quad \hat{t} = t'[x_j := \hat{s}_j] \text{ and } (s_j, \hat{s}_j) \in Z_n \text{ for all } j \in \{1, \dots, m\}\} \\ Z &:= \bigcup_{i=1}^n Z_n \end{aligned}$$

Note that  $Z_{n_1} \subseteq Z_{n_2}$  whenever  $n_1 \leq n_2$ . We show that the relation  $Z$  is a bisimulation, ie.  $o(t) = o(\hat{t})$  and  $(n(t), n(\hat{t})) \in Z$  for all  $(t, \hat{t}) \in Z_n$  for some  $n \in \mathbb{N}$ . The proof proceeds by induction on  $n$  and on the size of the smallest  $t' \in \mathcal{T}_\Sigma(X)$  that is a ‘‘witness’’ for  $(t, \hat{t}) \in Z$ . The base case, that  $n = 0$  is trivial as  $Z_0$  relates all bisimilar pairs. Suppose now that  $(t, \hat{t}) \in Z_{m+1}$ . If  $t' = x$  for some variable  $x$ , the claim follows from the fact that  $(t, \hat{t}) \in Z_m$  and the inductive hypothesis.

Let  $t' = \underline{g}(t'_1, \dots, t'_k)$ . We first check that  $o(t) = o(\hat{t})$ :

$$\begin{aligned} o(t) &= o(t'[x_j := s_j]) = o(\underline{g}(t'_1, \dots, t'_k)[x_j := s_j]) \\ &= i_g(o(t'_1[x_j := s_j]), \dots, o(t'_k[x_j := s_j])) \\ &\stackrel{\text{I.H.}}{=} i_g(o(t'_1[x_j := \hat{s}_j]), \dots, o(t'_k[x_j := \hat{s}_j])) \\ &= \dots = o(\hat{t}) \end{aligned}$$

The condition that  $n(t) \sim n(t')$  is equally straightforward:

$$\begin{aligned} n(t) &= n(t'[x_j := s_j]) = n(g(t'_1, \dots, t'_k)[x_j := s_j]) \\ &= d_g(o(t'_1[x_j := s_j]), \dots, o(t'_k[x_j := s_j]))[x_i := t'_i[x_j := s_j], y_i := n(t'_i[x_j := s_j])] \\ &\stackrel{\text{I.H.}}{=} d_g(o(t'_1[x_j := \hat{s}_j]), \dots, o(t'_k[x_j := \hat{s}_j]))[x_i := t'_i[x_j := s_j], y_i := n(t'_i[x_j := s_j])] \end{aligned}$$

By definition we have  $(t'_i[x_j := s_j], t'_i[x_j := \hat{s}_j]) \in Z_{m+1}$  and by the inductive hypothesis we have  $(n(t'_i[x_j := s_j]), n(t'_i[x_j := \hat{s}_j])) \in Z$ , ie.  $(n(t'_i[x_j := s_j]), n(t'_i[x_j := \hat{s}_j])) \in Z_{m'}$  for some  $m' \in \mathbb{N}$ . It is now easy to check that indeed  $(n(t), n(\hat{t})) \in Z$  as required. QED

The next proposition establishes a correspondence between coalgebra morphisms from  $(\mathcal{T}_\Sigma, \langle o, n \rangle)$  into the final stream coalgebra and solutions of  $\mathcal{D}$ .

**Proposition 4.8** *Let  $\mathcal{D}$  be a stream definition for some signature  $\Sigma$  and let  $\mathcal{S} = \{f\}_{f \in \Sigma}$  be a family of functions containing for each  $k$ -ary symbol  $f \in \Sigma$  a function  $f : (A^\omega)^k \rightarrow A^\omega$ . Then  $\mathcal{S}$  is a solution of  $\mathcal{D}$  iff  $\mathcal{I}_\mathcal{S} : \mathcal{T}_\Sigma \rightarrow A^\omega$  is a coalgebra morphism from  $(\mathcal{T}_\Sigma, \langle o, n \rangle)$  into the final coalgebra  $(A^\omega, \langle (-)(0), (-)' \rangle)$ .*

**Proof:** Let  $\mathcal{S} = \{f\}_{f \in \Sigma}$  be a solution of  $\mathcal{D}$ . We will prove that for all  $t \in \mathcal{T}_\Sigma$  we have  $\mathcal{I}_\mathcal{S}(t)(0) = o(t)$  and  $\mathcal{I}_\mathcal{S}(t)' = \mathcal{I}_\mathcal{S}(n(t))$ , ie., that  $\mathcal{S}$  induces a coalgebra morphism  $\mathcal{I}_\mathcal{S}$ . The claim is proven by induction on  $t$ . For the base case consider  $t = \underline{\tau}$  for some  $\tau \in A^\omega$ . Then  $\mathcal{I}_\mathcal{S}(\underline{\tau})(0) = \tau(0) = o(\underline{\tau})$  and  $\mathcal{I}_\mathcal{S}(\underline{\tau})' = \tau' = \mathcal{I}_\mathcal{S}(\underline{\tau}') = \mathcal{I}_\mathcal{S}(n(\underline{\tau}))$  as required.

For the induction step consider  $t = \underline{g}(t_1, \dots, t_k) \in \mathcal{T}_\Sigma$ . We calculate

$$\begin{aligned} \mathcal{I}_\mathcal{S}(\underline{g}(t_1, \dots, t_k))(0) &= g(\mathcal{I}_\mathcal{S}(t_1), \dots, \mathcal{I}_\mathcal{S}(t_k))(0) \\ &= i_g(\mathcal{I}_\mathcal{S}(t_1)(0), \dots, \mathcal{I}_\mathcal{S}(t_k)(0)) \\ &\stackrel{\text{I.H.}}{=} i_g(o(t_1), \dots, o(t_k)) \stackrel{\text{Def. of } o}{=} o(\underline{g}(t_1, \dots, t_k)) \end{aligned}$$

and

$$\begin{aligned} \mathcal{I}_\mathcal{S}(\underline{g}(t_1, \dots, t_k))' &= g(\mathcal{I}_\mathcal{S}(t_1), \dots, \mathcal{I}_\mathcal{S}(t_k))' \\ &\stackrel{\mathcal{S} \text{ solution}}{=} \mathcal{I}_\mathcal{S} \left( d_g(\mathcal{I}_\mathcal{S}(t_1)(0), \dots, \mathcal{I}_\mathcal{S}(t_k)(0))[x_i := \underline{\mathcal{I}_\mathcal{S}(t_i)}, y_i := \underline{\mathcal{I}_\mathcal{S}(t_i)}'] \right) \\ &\stackrel{\text{I.H.}}{=} \mathcal{I}_\mathcal{S} \left( d_g(o(t_1), \dots, o(t_k))[x_i := \underline{\mathcal{I}_\mathcal{S}(t_i)}, y_i := \underline{\mathcal{I}_\mathcal{S}(n(t_i))}] \right) \\ &\stackrel{(*)}{=} \mathcal{I}_\mathcal{S} (d_g(o(t_1), \dots, o(t_k))[x_i := t_i, y_i := n(t_i)]) \\ &= \mathcal{I}_\mathcal{S}(n(\underline{g}(t_1, \dots, t_k))) \end{aligned}$$

where the equality  $(*)$  follows from the fact that for arbitrary  $t \in \mathcal{T}_\Sigma(\{x_j\}_{j \in J})$  and  $\{s_j\}_{j \in J} \subseteq \mathcal{T}_\Sigma$  we have  $\mathcal{I}_\mathcal{S}(t[x_j := s_j]) = \mathcal{I}_\mathcal{S}(t[x_j := \mathcal{I}_\mathcal{S}(s_j)])$ . This can be easily proven by induction on  $t$ . Therefore we finished the proof that  $\mathcal{I}_\mathcal{S}$  is a coalgebra morphism.

For the converse, suppose now that  $\mathcal{S}$  induces a coalgebra morphism  $\mathcal{I}_\mathcal{S}$ . We have to show that  $\mathcal{S}$  is a solution of  $\mathcal{D}$ . Consider some  $k$ -ary  $f \in \Sigma$  and let  $\tau_1, \dots, \tau_k \in A^\omega$ . We calculate:

$$\begin{aligned} f(\tau_1, \dots, \tau_k)(0) &= \mathcal{I}_\mathcal{S}(f(\underline{\tau_1}, \dots, \underline{\tau_k}))(0) \\ &\stackrel{\text{coal.mor}}{=} o(f(\tau_1, \dots, \tau_k)) = i_f(o(\tau_1), \dots, o(\tau_k)) \\ &= i_f(\tau_1(0), \dots, \tau_k(0)). \end{aligned}$$

and

$$\begin{aligned}
f(\tau_1, \dots, \tau_k)' &= \mathcal{I}_{\mathcal{S}}(\underline{f}(\tau_1, \dots, \tau_k))' \\
&\stackrel{\text{coal.mor}}{=} \mathcal{I}_{\mathcal{S}}(n(\underline{f}(\tau_1, \dots, \tau_k))) \\
&= \mathcal{I}_{\mathcal{S}}(d_f(o(\tau_1), \dots, o(\tau_k)))[x_i := \tau_i, y_i := \tau_i'] \\
&= \mathcal{I}_{\mathcal{S}}(d_f(\tau_1(0), \dots, \tau_k(0)))[x_i := \tau_i, y_i := \tau_i']
\end{aligned}$$

which proves that  $\mathcal{S}$  is indeed a solution of  $\mathcal{D}$ . QED

Finally we have all ingredients in place in order to prove that every stream definition has a unique solution.

**Theorem 4.9** *Let  $\mathcal{D}$  be a stream definition for some signature  $\Sigma$ . Then  $\mathcal{S}(\mathcal{D})$  is the unique solution of  $\mathcal{D}$ .*

**Proof:** In order to see that  $\mathcal{S}(\mathcal{D})$  is the unique solution of  $\mathcal{D}$  it suffices to show that

$$\llbracket t \rrbracket_{\mathcal{D}} = \mathcal{I}_{\mathcal{S}(\mathcal{D})}(t) \quad \text{for } t \in \mathcal{T}_{\Sigma}, \quad (7)$$

because  $\llbracket \_ \rrbracket_{\mathcal{D}}$  is a coalgebra morphism by definition. Thus  $\mathcal{I}_{\mathcal{S}(\mathcal{D})}$  is a coalgebra morphism which implies that  $\mathcal{S}(\mathcal{D})$  is a solution of  $\mathcal{D}$  by Prop. 4.8. Clearly such a solution has to be unique: any other solution  $\mathcal{S}'$  would induce by Prop. 4.8 a coalgebra morphism  $\mathcal{I}_{\mathcal{S}'} : \mathcal{T}_{\Sigma} \rightarrow A^{\omega}$  such that  $\mathcal{I}_{\mathcal{S}'} \neq \mathcal{I}_{\mathcal{S}} = \llbracket \_ \rrbracket_{\mathcal{D}}$  which contradicts the fact that  $\llbracket \_ \rrbracket_{\mathcal{D}}$  is the unique coalgebra morphism of that type.

We now prove (7) by induction on  $t$ . For  $t = \tau$  we obviously have  $\llbracket \tau \rrbracket_{\mathcal{D}} = \tau = \mathcal{I}_{\mathcal{S}(\mathcal{D})}(\tau)$ . Furthermore, for  $t = \underline{f}(t_1, \dots, t_k)$ , we have

$$\begin{aligned}
\llbracket \underline{f}(t_1, \dots, t_k) \rrbracket_{\mathcal{D}} &\stackrel{(*)}{=} \llbracket \underline{f}(\llbracket t_1 \rrbracket_{\mathcal{D}}, \dots, \llbracket t_k \rrbracket_{\mathcal{D}}) \rrbracket_{\mathcal{D}} \\
&= \hat{f}(\llbracket t_1 \rrbracket_{\mathcal{D}}, \dots, \llbracket t_k \rrbracket_{\mathcal{D}}) \\
&\stackrel{\text{I.H.}}{=} \hat{f}(\mathcal{I}_{\mathcal{S}(\mathcal{D})}(t_1), \dots, \mathcal{I}_{\mathcal{S}(\mathcal{D})}(t_k)) = \mathcal{I}_{\mathcal{S}(\mathcal{D})}(\underline{f}(t_1, \dots, t_k))
\end{aligned}$$

Here equality (\*) follows from the fact that by Lemma 4.6 we have  $t_i \sim \llbracket t_i \rrbracket_{\mathcal{D}}$  for all  $i \in \{1, \dots, k\}$  and thus  $\underline{f}(t_1, \dots, t_k) \sim \underline{f}(\llbracket t_1 \rrbracket_{\mathcal{D}}, \dots, \llbracket t_k \rrbracket_{\mathcal{D}})$  by Lemma 4.7. As  $\llbracket \_ \rrbracket_{\mathcal{D}}$  is a coalgebra morphism, it identifies bisimilar elements of  $\mathcal{T}_{\Sigma}$  which shows that (\*) holds. QED

We conclude the discussion of the syntactic format by looking at some examples.

**Example 4.10** 1. The function  $\text{zip} : A^{\omega} \times A^{\omega} \rightarrow A^{\omega}$  that maps two streams  $\sigma(0)\sigma(1)\sigma(2)\dots$  and  $\tau(0)\tau(1)\tau(2)\dots$  to the stream  $\text{zip}(\sigma, \tau) := \sigma(0)\tau(0)\sigma(1)\tau(1)\dots$  is defined by a stream equation  $\langle i, d \rangle$  given by

$$\begin{aligned}
i(a_1, a_2) &:= a_1 \\
d(a_1.a_2) &:= \underline{\text{zip}}(y_2, x_1)
\end{aligned}$$

for  $a_1, a_2 \in A$ . The equation expresses that the first element of  $\text{zip}(\sigma, \tau)$  is equal to the first element of  $\sigma$  and that the tail of  $\text{zip}(\sigma, \tau)$  is equal to  $\text{zip}(\tau, \sigma')$ .

2. The function `merge` from Section 3 can be defined within the syntactic format. Let  $\Sigma = \{\text{merge}, \underline{2\text{times}}, \underline{3\text{times}}\}$  be the signature. The equations, which are essentially the ones from Section 3 written in a different way, look now as follows:

$$\begin{aligned}
i_{2\text{times}}(a) &:= 2 \cdot a \\
i_{3\text{times}}(a) &:= 3 \cdot a \\
i_{\text{merge}}(a_1, a_2) &:= \begin{cases} a_1 & \text{if } a_1 < a_2 \\ a_2 & \text{otherwise.} \end{cases} \\
d_{2\text{times}}(a) &:= \underline{2\text{times}}(y) \\
d_{3\text{times}}(a) &:= \underline{3\text{times}}(y) \\
d_{\text{merge}}(a_1, a_2) &:= \begin{cases} \underline{\text{merge}}(y_1, x_2) & \text{if } a_1 < a_2 \\ \underline{\text{merge}}(y_1, y_2) & \text{if } a_1 = a_2 \\ \underline{\text{merge}}(x_1, y_2) & \text{if } a_1 > a_2 \end{cases}
\end{aligned}$$

## 4.2 Causal functions

In this section we are going to prove that our format for defining stream functions defines precisely the causal stream functions.

**Definition 4.11** A function  $f : (A^\omega)^k \rightarrow A^\omega$  is definable if there exists a signature  $\Sigma$  and a stream definition  $\mathcal{D}$  for  $\Sigma$  such that  $f$  is part of the unique solution of  $\mathcal{D}$ , ie., such that  $f \in \mathcal{S}(\mathcal{D})$ .

**Definition 4.12** Let  $\tau = \tau(0)\tau(1)\dots \in A^\omega$  and  $\sigma = \sigma(0)\sigma(1)\dots \in A^\omega$  be two streams. For a natural number  $n \in \mathbb{N}$  we say that  $\tau$  and  $\sigma$  are  $n$ -equivalent (Notation:  $\tau \equiv_n \sigma$ ) if  $\tau(i) = \sigma(i)$  for all  $i \in \{0, \dots, n\}$ . A stream function  $f : (A^\omega)^k \rightarrow A^\omega$  is *causal* if for all  $n \in \mathbb{N}$  and for all  $\tau_1, \dots, \tau_k, \sigma_1, \dots, \sigma_k \in A^\omega$  with  $\tau_i \equiv_n \sigma_i$  for  $i \in \{1, \dots, k\}$  we have  $f(\tau_1, \dots, \tau_k) \equiv_n f(\sigma_1, \dots, \sigma_k)$ .

In other words, a stream function  $f$  is causal if the first  $n$  elements of the output stream  $f(\tau_1, \dots, \tau_k)$  depend only on the first  $n$  elements of each of the input streams  $\tau_1, \dots, \tau_k$ . The set  $\mathbf{C}_k$  of causal stream functions of arity  $k \in \mathbb{N}$  form the carrier of a final coalgebra for the functor  $(A \times \_)^{A^k}$ . The first component  $\mathbf{C} \rightarrow (A^k \rightarrow A)$  of the structure map of the final coalgebra is given by

$$f \mapsto f[a_1, \dots, a_k] := f(a_1 : \alpha_1, \dots, a_k : \alpha_k)$$

where  $\alpha_1, \dots, \alpha_k \in A^\omega$  are arbitrary streams. The second component  $\mathbf{C} \rightarrow (A^k \rightarrow \mathbf{C}_k)$  is given by

$$f \mapsto f_{(a_1, \dots, a_k)} := \lambda x_1 \dots \lambda x_k (f(a_1 : x_1, \dots, a_k : x_k))$$

For the details on causal functions and final coalgebras see [HR10]. We are now ready to prove the key lemma that allows us to show that any stream function defined in our syntactic format is causal.

**Lemma 4.13** *Let  $\Sigma$  be a signature and let  $\mathcal{D}$  be a stream definition for  $\Sigma$ . Furthermore let  $t \in \mathcal{T}_\Sigma(\{z_1, \dots, z_l\})$  be a term with at most  $l$  variables and let  $\tau_1, \dots, \tau_l, \sigma_1, \dots, \sigma_l \in A^\omega$  be streams such that  $\tau_j \equiv_n \sigma_j$  for  $j \in \{1, \dots, l\}$  and some  $n \in \mathbb{N}$ . Then*

$$\llbracket t[z_j := \tau_j] \rrbracket_{\mathcal{D}} \equiv_n \llbracket t[z_j := \sigma_j] \rrbracket_{\mathcal{D}}.$$

**Proof:** The claim is proven by induction on  $n$  and the structure of  $t$ . For the case  $n = 0$  and  $t = z_j$  for some  $j' \in \{1, \dots, l\}$  we get  $\llbracket t[z_j := \tau_j] \rrbracket_{\mathcal{D}} = \llbracket \tau_{j'} \rrbracket_{\mathcal{D}} = \tau_{j'} \equiv_0 \sigma_{j'} = \llbracket t[z_j := \sigma_j] \rrbracket_{\mathcal{D}}$ .

In the case  $n = 0$  and  $t = \underline{g}(t_1, \dots, t_m)$  we calculate

$$\begin{aligned} \llbracket \underline{g}(t_1, \dots, t_m)[z_j := \tau_j] \rrbracket_{\mathcal{D}}(0) &= o\left(\underline{g}(t_1, \dots, t_m)[z_j := \tau_j]\right) \\ &= i_g\left(o(t_1[z_j := \tau_j]), \dots, o(t_m[z_j := \tau_j])\right) \\ &= i_g\left(\llbracket t_1[z_j := \tau_j] \rrbracket_{\mathcal{D}}(0), \dots, \llbracket t_m[z_j := \tau_j] \rrbracket_{\mathcal{D}}(0)\right) \\ &\stackrel{\text{I.H.}}{=} i_g\left(\llbracket t_1[z_j := \sigma_j] \rrbracket_{\mathcal{D}}(0), \dots, \llbracket t_m[z_j := \sigma_j] \rrbracket_{\mathcal{D}}(0)\right) \\ &= \llbracket \underline{g}(t_1, \dots, t_m)[z_j := \sigma_j] \rrbracket_{\mathcal{D}}(0) \end{aligned}$$

which implies  $\llbracket \underline{g}(t_1, \dots, t_m)[z_j := \tau_j] \rrbracket_{\mathcal{D}} \equiv_0 \llbracket \underline{g}(t_1, \dots, t_m)[z_j := \sigma_j] \rrbracket_{\mathcal{D}}$  as required.

Suppose now that  $\tau_j \equiv_{n+1} \sigma_j$  for  $j \in \{1, \dots, l\}$ . In the case  $t = z_{j'}$  it is again easy to see that  $\llbracket z_{j'}[z_j := \tau_j] \rrbracket_{\mathcal{D}} \equiv_{n+1} \llbracket z_{j'}[z_j := \sigma_j] \rrbracket_{\mathcal{D}}$ . In the case  $t = \underline{g}(t_1, \dots, t_m)$  we calculate:

$$\begin{aligned} \llbracket \underline{g}(t_1, \dots, t_m)[z_j := \tau_j] \rrbracket'_{\mathcal{D}} &\stackrel{\text{I-1} \text{ co. mor.}}{=} \llbracket n(\underline{g}(t_1, \dots, t_m)[z_j := \tau_j]) \rrbracket_{\mathcal{D}} \\ &= \llbracket d_g\left(o(t_i[z_j := \tau_j])\right) [x_i := t_i[z_j := \tau_j], y_i = n(t_i[z_j := \tau_j])] \rrbracket_{\mathcal{D}} \\ &= \llbracket d_g\left(o(t_i[z_j := \tau_j])\right) [x_i := \llbracket t_i[z_j := \tau_j] \rrbracket, y_i = \llbracket t_i[z_j := \tau_j] \rrbracket'] \rrbracket_{\mathcal{D}} \\ &\stackrel{(*)}{\equiv}_n \llbracket d_g\left(o(t_i[z_j := \sigma_j])\right) [x_i := \llbracket t_i[z_j := \sigma_j] \rrbracket, y_i = \llbracket t_i[z_j := \sigma_j] \rrbracket'] \rrbracket_{\mathcal{D}} \\ &\quad \vdots \\ &= \llbracket \underline{g}(t_1, \dots, t_m)[z_j := \sigma_j] \rrbracket'_{\mathcal{D}} \end{aligned}$$

Here the equivalence  $(*)$  follows from multiple uses of the inductive hypothesis: we have  $\llbracket t_i[z_j := \tau_j] \rrbracket \equiv_{n+1} \llbracket t_i[z_j := \sigma_j] \rrbracket$  and  $\llbracket t_i[z_j := \tau_j] \rrbracket' \equiv_n \llbracket t_i[z_j := \sigma_j] \rrbracket'$  by the inductive hypothesis on  $t$ . Therefore also  $(*)$  holds by the inductive hypothesis on  $n$ . QED

**Theorem 4.14** *A function  $f : (A^\omega)^k \rightarrow A^\omega$  is definable iff  $f$  is causal.*

**Proof:** The fact that every definable stream function is causal is almost immediate from Lemma 4.13 and we leave the details of the precise argument to the reader.

In order to prove the converse we are going to define all causal functions in one (rather big) stream definition. Let  $\Sigma_{\mathcal{C}} := \{\underline{f} \mid f : (A^\omega)^k \rightarrow A^\omega\}$  be the signature that contains for every causal function  $f$  a corresponding symbol  $\underline{f}$ . We define a stream definition  $\mathcal{D}_{\mathcal{C}}$  for  $\Sigma_{\mathcal{C}}$  by considering for each  $k$ -ary  $\underline{f} \in \Sigma$  the stream equation given by

$$\begin{aligned} i_{\underline{f}}(a_1, \dots, a_k) &:= f[a_1, \dots, a_k] \\ d_{\underline{f}}(a_1, \dots, a_k) &:= \underline{f}_{(a_1, \dots, a_k)}(x_1, \dots, x_k) \end{aligned}$$

for all  $a_1, \dots, a_k \in A$ . In order to see now that every causal function is definable it suffices to prove that  $\mathcal{S}_C := \{\underline{f}\}_{f \in C}$  is a solution of  $\mathcal{D}_C$ . This can be easily checked. We have

$$\begin{aligned} f(a_1 : \tau_1, \dots, a_k : \tau_k)(0) &= f[a_1, \dots, a_k] = i_f(a_1, \dots, a_k) \\ f(a_1 : \tau_1, \dots, a_k : \tau_k)' &= f_{(a_1, \dots, a_k)}(\tau_1, \dots, \tau_k) \\ &= \mathcal{I}_{\mathcal{S}_C}(d_f(a_1, \dots, a_k)[x_i := \tau_i]) \end{aligned}$$

which shows that  $\mathcal{S}_C$  fulfils the conditions for being the unique solution of  $\mathcal{D}_C$ . QED

The reader might feel a bit uneasy about the fact that in the proof of the theorem we used an infinite set of equations for defining a given causal function  $f : (A^\omega)^k \rightarrow A^\omega$ . First of all this was necessary, because we defined all causal functions simultaneously. Secondly, there are causal functions  $f$  that have an infinite number of distinct derivatives  $\{f_w \mid w \in (A^k)^*\}$  where  $f_w$  is defined by  $f_\epsilon = f$  and  $f_{w \cdot (a_1, \dots, a_k)} = (f_w)_{(a_1, \dots, a_k)}$ . For those causal functions  $f$ , that only have a finite number of derivatives, it is easy to modify the argument in the proof of Theorem 4.14 in order to obtain a definition for  $f$  that uses only  $\theta(f)$  stream equations, where  $\theta(f)$  is the number of derivatives of  $f$ .

## 5 Linear stream differential equations

In this section, we shall discuss the class of linear SDEs and show that their solutions consist of so-called rational streams. We begin by recalling a bit of elementary stream calculus from [Rut05, Rut08].

### 5.1 Stream calculus

If we assume that  $A$  has some algebraic structure, the set  $A^\omega$  of streams over  $A$  inherits (parts of) this structure [BR88]. More specifically, let us assume that

$$\langle A, +, \cdot, 0, 1 \rangle$$

is a semiring. Examples are the set  $2 = \{0, 1\}$  of the Booleans, with disjunction for plus and conjunction for times; and the sets of the natural or real numbers, with plus and times as usual.

For  $r \in A$ , we define the constant stream

$$[r] = (r, 0, 0, 0, \dots)$$

which we often denote simply as  $r$ . Another constant stream is

$$X = (0, 1, 0, 0, 0, \dots)$$

We define the *sum* of two streams  $\sigma, \tau \in A^\omega$  by the SDE

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

with initial value  $(\sigma + \tau)(0) = \sigma(0) + \tau(0)$ . The *convolution product* of  $\sigma$  and  $\tau$  is given by

$$(\sigma \times \tau)' = (\sigma' \times \tau) + ([\sigma(0)] \times \tau')$$

with initial value  $(\sigma \times \tau)(0) = \sigma(0) \cdot \tau(0)$ . Equivalently, we have for all  $\sigma, \tau \in A^\omega$  and  $n \geq 0$ ,

$$\begin{aligned}(\sigma + \tau)(n) &= \sigma(n) + \tau(n) \\ (\sigma \times \tau)(n) &= \sum_{i=0}^n \sigma(i) \cdot \tau(n-i)\end{aligned}$$

Now it can be shown that

$$\langle A^\omega, +, \times, [0], [1] \rangle$$

is a semi-ring [BR88].

One can compute a stream from its initial value and derivative by the so-called *fundamental theorem* of stream calculus [Rut05]: for all  $\sigma \in A^\omega$ ,

$$\sigma = \sigma(0) + (X \times \sigma') \tag{8}$$

(writing  $\sigma(0)$  for  $[\sigma(0)]$ ). As we shall see below, the fundamental theorem can be used to solve certain SDEs.

For the remainder of this section, we assume that  $A$  is a field: it has an operation  $-$  that is inverse to  $+$  (making of  $A$  a ring). Moreover, every nonzero element in  $A$  has a unique multiplicative inverse.

The operations of minus and multiplicative inverse on  $A$  can be carried over to  $A^\omega$ . For  $\sigma \in A^\omega$ , we define

$$-\sigma = (-\sigma(0), -\sigma(1), -\sigma(2), \dots)$$

Equivalently, we can define minus by the SDE

$$(-\sigma)' = -(\sigma')$$

with initial value  $(-\sigma)(0) = -\sigma(0)$ .

If  $\sigma(0) \neq 0$  then the stream  $\sigma$  has a (unique) multiplicative inverse  $\sigma^{-1}$  in  $A^\omega$ , satisfying  $\sigma^{-1} \times \sigma = [1]$ . It can be defined as the unique solution of the SDE

$$(\sigma^{-1})' = -[\sigma(0)^{-1}] \times \sigma' \times \sigma^{-1}$$

with initial value

$$\sigma^{-1}(0) = \sigma(0)^{-1}$$

As usual, we shall often write  $1/\sigma$  for  $\sigma^{-1}$  and  $\sigma/\tau$  for  $\sigma \times \tau^{-1}$ .

We call a stream  $\pi \in A^\omega$  *polynomial* if there are  $k \geq 0$  and  $a_i \in A$  such that

$$\pi = a_0 + a_1X + a_2X^2 + \dots + a_kX^k = (a_0, a_1, a_2, \dots, a_k, 0, 0, 0, \dots)$$

where we write  $a_iX^i$  for  $[a_i] \times X^i$  with  $X^i$  the  $i$ -fold product of  $X$  with itself. For instance,

$$1 + X + 3X^2 + 7X^3 = (1, 1, 3, 7, 0, 0, 0, \dots)$$

is a polynomial stream.

We call a stream  $\rho \in A^\omega$  *rational* if it is the quotient  $\rho = \sigma/\tau$  of two polynomial streams  $\sigma$  and  $\tau$  with  $\tau(0) \neq 0$ . The stream

$$\frac{2 - X}{(1 - X)^2} = (2, 3, 4, 5, 6, \dots)$$

is an example of a rational stream.

As we mentioned above, the fundamental theorem of stream calculus can be used to solve stream differential equations. Here we give an example of a single SDE; in the next subsection, we shall see how to deal with systems of SDEs. Consider the following SDE:

$$\sigma' = -(3 \times \sigma)$$

with initial value  $\sigma(0) = 1$ . Using the fundamental theorem, we calculate its solution as follows:

$$\begin{aligned} \sigma &= \sigma(0) + (X \times \sigma') \\ &= 1 + (X \times -(3 \times \sigma)) \\ &= 1 - (3X \times \sigma) \end{aligned}$$

This implies  $(1 + 3X) \times \sigma = 1$  whence

$$\sigma = \frac{1}{1 + 3X}$$

One can easily prove that  $\sigma = (1, -3, 9, -27, \dots)$ .

## 5.2 Solving linear systems of stream differential equations

Using some elementary linear algebra notation (matrices and vectors), we show next how to solve *linear* systems of stream differential equations. For notational convenience, we shall deal with linear systems of dimension 2, which can be straightforwardly generalised to systems of higher dimensions. They are given by the following data:

$$\begin{pmatrix} \sigma \\ \tau \end{pmatrix}' = M \times \begin{pmatrix} \sigma \\ \tau \end{pmatrix} \quad \begin{pmatrix} \sigma \\ \tau \end{pmatrix} (0) = N \quad (9)$$

where  $M$  is a  $2 \times 2$ -matrix and  $N$  is a  $1 \times 2$ -matrix over  $A$ :

$$M = \begin{pmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{pmatrix} \quad N = \begin{pmatrix} n_1 \\ n_2 \end{pmatrix}$$

for  $m_{ij}, n_i \in A$ . The above notation is really just a short hand for the following system of two stream differential equations:

$$\begin{aligned} \sigma' &= (m_{11} \times \sigma) + (m_{12} \times \tau) & \sigma(0) &= n_1 \\ \tau' &= (m_{21} \times \sigma) + (m_{22} \times \tau) & \tau(0) &= n_2 \end{aligned}$$

We can solve such a system of equations by using twice the fundamental theorem of stream calculus (equation (8) above), once for  $\sigma$  and once for  $\tau$ :

$$\begin{aligned} \sigma &= \sigma(0) + (X \times \sigma') \\ \tau &= \tau(0) + (X \times \tau') \end{aligned}$$

In matrix notation, the fundamental theorem looks like

$$\begin{pmatrix} \sigma \\ \tau \end{pmatrix} = \begin{pmatrix} \sigma \\ \tau \end{pmatrix} (0) + X \times \begin{pmatrix} \sigma \\ \tau \end{pmatrix}'$$

Next we can solve our linear system (9) above by happily calculating as follows:

$$\begin{aligned} \begin{pmatrix} \sigma \\ \tau \end{pmatrix} &= \begin{pmatrix} \sigma \\ \tau \end{pmatrix} (0) + X \times \begin{pmatrix} \sigma \\ \tau \end{pmatrix}' \\ &= N + X \times M \times \begin{pmatrix} \sigma \\ \tau \end{pmatrix} \end{aligned}$$

This leads to

$$(I - (X \times M)) \begin{pmatrix} \sigma \\ \tau \end{pmatrix} = N$$

where  $I$  and  $X \times M$  are given by

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad X \times M = \begin{pmatrix} m_{11} \times X & m_{12} \times X \\ m_{21} \times X & m_{22} \times X \end{pmatrix}$$

Finally, we can express the unique solution of our linear system of stream differential equations as follows:

$$\begin{pmatrix} \sigma \\ \tau \end{pmatrix} = (I - (X \times M))^{-1} \times N \quad (10)$$

The advantage of the matrix notations above now becomes clear: we can compute the inverse of the matrix

$$(I - (X \times M)) = \begin{pmatrix} 1 - (m_{11} \times X) & -(m_{12} \times X) \\ -(m_{21} \times X) & 1 - (m_{22} \times X) \end{pmatrix}$$

whose values are simple polynomial streams, by standard linear algebra.

Let us look at an example and consider the following system of SDEs:

$$\begin{aligned} \sigma' &= \tau \\ \tau' &= -\sigma + 2\tau \end{aligned}$$

with initial values  $\sigma(0) = 1$  and  $\tau(0) = 2$ . In other words,

$$\begin{pmatrix} \sigma \\ \tau \end{pmatrix} = (I - (X \times M))^{-1} \times N$$

with

$$M = \begin{pmatrix} 0 & 1 \\ -1 & 2 \end{pmatrix} \quad N = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

The solution of this system of SDEs is now given by

$$\begin{aligned} \begin{pmatrix} \sigma \\ \tau \end{pmatrix} &= (I - (X \times M))^{-1} \times N \\ &= \begin{pmatrix} 1 & -X \\ X & 1 - 2X \end{pmatrix}^{-1} \times \begin{pmatrix} 1 \\ 2 \end{pmatrix} \\ &= \begin{pmatrix} \frac{1-2X}{(1-X)^2} & \frac{X}{(1-X)^2} \\ \frac{-X}{(1-X)^2} & \frac{1}{(1-X)^2} \end{pmatrix} \times \begin{pmatrix} 1 \\ 2 \end{pmatrix} \end{aligned}$$

And so we find

$$\sigma = \frac{1}{(1-X)^2} \quad \tau = \frac{2-X}{(1-X)^2}$$

For another example, we return to the SDE defining the Fibonacci numbers (equation (1) in Section 1.1)

$$\sigma'' = \sigma' + \sigma \quad \sigma(0) = 0 \quad \sigma(1) = 1$$

This SDE is higher-order: it involves a second derivative. One can transform it in the usual fashion into a linear system of ordinary SDEs:

$$\begin{aligned} \sigma' &= \tau \\ \tau' &= \sigma + \tau \end{aligned}$$

with initial values  $\sigma(0) = 0$  and  $\tau(0) = 1$ . This system corresponds to the following values of  $M$  and  $N$ :

$$M = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \quad N = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

As before, the solution is given by

$$\begin{aligned} \begin{pmatrix} \sigma \\ \tau \end{pmatrix} &= (I - (X \times M))^{-1} \times N \\ &= \begin{pmatrix} 1 & -X \\ -X & 1 - X \end{pmatrix}^{-1} \times \begin{pmatrix} 0 \\ 1 \end{pmatrix} \\ &= \begin{pmatrix} \frac{1-X}{1-X-X^2} & \frac{X}{1-X-X^2} \\ \frac{X}{1-X-X^2} & \frac{1}{1-X-X^2} \end{pmatrix} \times \begin{pmatrix} 0 \\ 1 \end{pmatrix} \end{aligned}$$

It follows that the solution of our higher-order SDE is

$$\sigma = \frac{X}{1 - X - X^2}$$

We conclude this section with the observation that the solutions of linear systems of stream differential equations always consist of rational streams. This follows from the general formula for the solution – equation (10) above – and the fact that if a matrix has polynomial streams as entries then the entries of its inverse are rational streams.

## 6 Non-standard stream calculus

Coalgebras are usually studied “up-to-isomorphism”, e.g., one talks about *the* final coalgebra of a functor because it is determined uniquely up-to-isomorphism. When reasoning about a concrete type of coalgebras one then has a certain “canonical” representation of the final coalgebra in mind. For the stream functor  $A \times \_$  the final coalgebra is usually given by the set of infinite  $A$ -streams  $A^\omega$  together with the usual operations  $(\_)(0)$  and  $(\_)'$ . There are, however, infinitely many ways of turning  $A^\omega$  into the final stream coalgebra - we will discuss some of them in this section. Our aim is to convince the reader that each of these representations of  $A^\omega$  as final coalgebra is potentially interesting on its own as each of them yields a different syntactic definition format. We confine ourselves to giving a list of examples for both non-standard stream representations and some examples of definitions that make good use of them.

## 6.1 Non-standard coalgebraic stream representations

Most of the examples for non-standard representations of the set  $A^\omega$  make use of some algebraic operations on  $A$ . Let us look at some examples for the case  $A = \mathbb{R}$  where  $\mathbb{R}$  denotes the set of real numbers.

**Example 6.1** 1. We can supply the set  $\mathbb{R}^\omega$  of streams over the set  $\mathbb{R}$  of real numbers with a coalgebra structure as follows. For  $\sigma \in \mathbb{R}^\omega$  we define

$$\Delta\sigma = (\sigma(1) - \sigma(0), \sigma(2) - \sigma(1), \sigma(3) - \sigma(2), \dots)$$

(cf. [PE98, Rut05]). We claim that  $\mathbb{R}^\omega$  together with the map

$$\langle (-)(0), \Delta \rangle: \mathbb{R}^\omega \rightarrow \mathbb{R} \times \mathbb{R}^\omega \quad \sigma \mapsto \langle \sigma(0), \Delta\sigma \rangle$$

is a final  $\mathbb{R} \times \_$ -coalgebra.

2. Another coalgebra structure on  $\mathbb{R}^\omega$  is obtained by defining

$$\frac{d\sigma}{dX} = (\sigma(1), 2 \cdot \sigma(2), 3 \cdot \sigma(3), \dots)$$

for  $\sigma \in \mathbb{R}^\omega$ . Again  $(\mathbb{R}^\omega, \langle (-)(0), \frac{d}{dX} \rangle)$  is a final  $\mathbb{R} \times \_$ -coalgebra.

3. In a similar fashion lots of examples could be designed: Given a set  $A$  together with some operation  $o : A \times A \rightarrow A$ , we define

$$\Delta_o\sigma = (o(\sigma(0), \sigma(1)), o(\sigma(1), \sigma(2)), o(\sigma(2), \sigma(3)), \dots)$$

and we can see that  $A^\omega$  together with the map  $\langle (-)(0), \Delta_o \rangle: A^\omega \rightarrow A \times A^\omega$  is a final coalgebra provided that for any  $a \in A$  the map  $a' \mapsto o(a, a')$  has an inverse.

But the examples for non-standard stream representations are not limited to coalgebras for the stream functor  $A \times \_$  as the following two interesting examples show.

**Example 6.2** 1. We define two functions  $\text{Even} : A^\omega \rightarrow A^\omega$  and  $\text{odd} : A^\omega \rightarrow A^\omega$  by putting

$$\begin{aligned} \text{odd}(\sigma(0), \sigma(1), \sigma(2), \dots) &:= (\sigma(1), \sigma(3), \dots) \\ \text{Even}(\sigma(0), \sigma(1), \sigma(2), \dots) &:= (\sigma(2), \sigma(4), \dots). \end{aligned}$$

It is not difficult to see that  $A^\omega$  together with the map  $\langle (-)(0), \text{odd}, \text{Even} \rangle : A^\omega \rightarrow A \times A^\omega \times A^\omega$  is the final  $(A \times \_ \times \_)$ -coalgebra.

2. Alternatively, we can define a function  $\text{even} : A^\omega \rightarrow A^\omega$  by putting

$$\text{even}(\sigma) := (\sigma(0), \sigma(2), \sigma(4), \dots)$$

for  $\sigma \in A^\omega$ . In this case the set  $A^\omega$  together with the function  $\langle (-)(0), \text{even}, \text{odd} \rangle$  is not a final coalgebra, but a subcoalgebra of the final  $(A \times \_ \times \_)$ -coalgebra. It has been demonstrated in [KR10] that this representation as a subcoalgebra of the final coalgebra is sufficient for obtaining a syntactic definition format.

## 6.2 Examples of non-standard stream calculus

We now turn to the examples of stream differential equations that make use of the different representations of  $A^\omega$ . Essential in all examples is the fact that we have a “non-standard” representation of  $A^\omega$  as a (subcoalgebra of a) final coalgebra. Therefore we can use finality in order to define stream constants and functions using stream differential equations that employ these representations. This has been made formal in [KR10] where a definition format is described that generalises the one from Section 4 to arbitrary non-standard representations. Here we are only going to state the stream differential equations without formally defining the notion of a solution and without proving that such a solution is uniquely determined.

**Example 6.3** 1. An example of a differential equation using the representation from Example 6.1(1)) is the following:

$$\sigma(0) = 1, \quad \Delta\sigma = \sigma$$

It has as unique solution the stream  $\sigma = (2^0, 2^1, 2^2, \dots)$ . A closed expression for this solution can be computed using the following identity, which can be viewed as the fundamental theorem of the difference calculus: for all  $\tau \in \mathbb{R}^\omega$ ,

$$\tau = \frac{1}{1-X} \times (\tau_0 + X \times \Delta\tau)$$

Using this and the differential equation above, one obtains

$$\sigma = \frac{1}{1-2X} = (2^0, 2^1, 2^2, \dots)$$

2. The following is an example of an differential equation using the stream representation from Example 6.1(2)):

$$\sigma(0) = 1, \quad \frac{d\sigma}{dX} = \sigma$$

Again, it has a unique solution, which is now given by  $\sigma(n) = \frac{1}{n!}$ . (It is not obvious how to find a closed expression for  $\sigma$ .)

Two interesting examples that use that  $(\_) (0)$ , **even** and **odd** representation are provided by differential equations for the well-known Thue-Morse and Toeplitz sequence [AS99].

**Example 6.4** Let  $A = \mathbb{F}_2$  be the two-element Boolean field. We are going to define stream constants and operations on the set  $\mathbb{F}_2^\omega$ . The following definition should be read as one collection of stream differential equations in the style of our formal definition scheme from Section 4. The difference to Section 4 is that we do not use the  $(\_) (0)$ ,  $(\_)'$ -representation, but the  $(\_) (0)$ , **even** and **odd**-representation of streams. The following three equations on the left specify a function  $\text{inv} : \mathbb{F}_2^\omega \rightarrow \mathbb{F}_2^\omega$  that computes the pointwise Boolean inverse of a given stream  $\sigma$ , ie.,

$$\text{inv}(\sigma) = (1 - \sigma(0))(1 - \sigma(1))(1 - \sigma(2)) \dots$$

The three equations on the right specify the Thue-Morse sequence.

$$\begin{array}{ll} \text{inv}(x)(0) & = 1 - x(0) & \text{M}(0) & = 0 \\ \text{even}(\text{inv}(x)) & = \text{inv}(\text{even}(x)) & \text{even}(\text{M}) & = \text{M} \\ \text{odd}(\text{inv}(x)) & = \text{inv}(\text{odd}(x)) & \text{odd}(\text{M}) & = \text{inv}(\text{M}) \end{array}$$

Furthermore we can add six equations that specify the sequence  $\bar{1}$  that consists of 1's only and the Toeplitz-sequence.

$$\begin{array}{ll} \bar{1}(0) & = 1 \\ \text{even}(\bar{1}) & = \bar{1} \\ \text{odd}(\bar{1}) & = \bar{1} \end{array} \qquad \begin{array}{ll} \text{T}(0) & = 1 \\ \text{even}(\text{T}) & = \bar{1} \\ \text{odd}(\text{T}) & = \text{inv}(\text{T}) \end{array}$$

Recently we observed that the definability of the Thue-Morse and Toeplitz streams in non-standard stream calculus is an instance of a general result: Every  $k$ -automatic sequence in the sense of [AS99] can be defined using a finite number of equations in non-standard stream calculus. The details can be found in [KR11].

### 6.3 Coinduction for non-standard stream representations

The non-standard stream representations that we discussed are not only useful for definitions but also for coinductive proofs: If we represent  $A^\omega$  as a (sub)coalgebra of a final coalgebra, we can show that two streams  $\sigma_1, \sigma_2 \in A^\omega$  are equal by proving that they are bisimilar. For example, for showing that  $\sigma_1 = \sigma_2$  for  $\sigma_1, \sigma_2 \in A^\omega$ , it suffices to show that there is a relation  $R \subseteq A^\omega \times A^\omega$  with  $(\tau_1, \tau_2) \in R$  implies  $\tau_1(0) = \tau_2(0)$  and  $(\Delta(\tau_1), \Delta(\tau_2)) \in R$ . In this case we call  $R$  a  $\{(-)(0), \Delta\}$ -bisimulation. Similarly, we can use  $\{(-)(0), \frac{d}{dX}\}$ -bisimulations or  $\{(-)(0), \text{even}, \text{odd}\}$ -bisimulations for proving equalities between streams. We are now going to discuss two examples where this non-standard coinduction leads to simple proofs of non-trivial facts about streams.

As a preparation for the first example we define the following so-called *falling powers* of  $X$ , for all  $n \geq 0$ , by

$$X^n = X^n / (1 - X)^{n+1}$$

Note that  $\Delta X^0 = \Delta(1/(1 - X)) = 0$  and  $\Delta X^{n+1} = X^n$ . For a stream  $\sigma \in \mathbb{R}^\omega$  we define its  $n$ -th coordinate in the  $\Delta$ -representation by putting

$$r_n^\sigma = \left( \Delta^{(n)} \sigma \right) (0) \in \mathbb{R}$$

Now we can represent a given stream as the following sum of falling powers

$$\text{sum}(\sigma) = r_0^\sigma \times X^0 + r_1^\sigma \times X^1 + r_2^\sigma \times X^2 + \dots$$

The fact that  $\text{sum}(\sigma)$  is indeed a representation of  $\sigma$  has straightforward proof by  $\{(-)(0), \Delta\}$ -coinduction.

**Theorem 6.5** *For all  $\sigma \in \mathbb{R}^\omega$  we have  $\sigma = \text{sum}(\sigma)$ .*

**Proof:** We show that

$$R = \{ (\sigma, \text{sum}(\sigma)) \mid \sigma \in \mathbb{R}^\omega \}$$

is an  $\{h, \Delta\}$ -bisimulation. Clearly,

$$\sigma(0) = \text{sum}(\sigma)(0)$$

Furthermore we have

$$\begin{aligned}
\Delta \text{sum}(\sigma) &= \Delta (r_0^\sigma \times X^0 + r_1^\sigma \times X^1 + r_2^\sigma \times X^2 + \dots) \\
&= r_0^\sigma \times \Delta X^0 + r_1^\sigma \times \Delta X^1 + r_2^\sigma \times \Delta X^2 + \dots \\
&= r_1^\sigma \times X^0 + r_2^\sigma \times X^1 + r_3^\sigma \times X^2 + \dots \\
&= \text{sum}(\Delta\sigma)
\end{aligned}$$

where for the latter equality we use  $r_{n+1}^\sigma = r_n^{\Delta\sigma}$ . As a consequence, we have

$$(\Delta\sigma, \Delta \text{sum}(\sigma)) = (\Delta\sigma, \text{sum}(\Delta\sigma)) \in R$$

This proves that  $R$  is an  $\{(-)(0), \Delta\}$ -bisimulation. QED

The theorem above was first presented in [Rut05, Thm 11.1] using “ordinary”  $\{(-)(0), (-)'\}$ -coinduction. The reader is invited to check that the proof there is quite a bit more complicated which demonstrates the usefulness of non-standard stream representations.

Another example for non-standard coinduction builds on Example 6.4 where we defined the Thue-Morse sequence  $M$  and the Toeplitz sequence  $T$ . The statement we want to prove establishes a close connection between these sequences.

**Theorem 6.6** *We have  $T = \Delta(M) = M + M'$ .*

In order to prove the theorem we need a small collection of lemmas.

**Lemma 6.7** *The following holds for all streams  $\sigma, \tau \in 2^\omega$ :*

1.  $\sigma + \text{inv}(\sigma) = \bar{1}$ ,
2.  $\sigma + \sigma = \bar{0}$ ,
3.  $\text{inv}(\sigma) + \tau = \text{inv}(\sigma + \tau)$
4.  $\text{inv}(\bar{1}) = \bar{0}$ ,
5.  $\text{inv}(\text{inv}(\sigma)) = \sigma$ .

**Proof:** All claims have a simple coinductive proof. We put

$$\begin{aligned}
R := & \{(\sigma + \text{inv}(\sigma), \bar{1}) \mid \sigma \in 2^\omega\} \cup \{(\sigma + \sigma, \bar{0}) \mid \sigma \in 2^\omega\} \\
& \{(\text{inv}(\sigma) + \tau, \text{inv}(\sigma + \tau)) \mid \sigma, \tau \in 2^\omega\} \cup \{(\text{inv}(\bar{1}), \bar{0})\} \\
& \{(\text{inv}(\text{inv}(\sigma)), \sigma) \mid \sigma \in 2^\omega\}.
\end{aligned}$$

and show that  $R$  is a  $\{(-)(0), \text{even}, \text{odd}\}$ -bisimulation. Consider first  $(\sigma + \text{inv}(\sigma), \bar{1}) \in R$ , then  $(\sigma + \text{inv}(\sigma))(0) = \sigma(0) + \text{inv}(\sigma)(0) = \sigma(0) + 1 - \sigma(0) = 1 = \bar{1}(0)$ . Furthermore  $\text{even}(\sigma + \text{inv}(\sigma)) = \text{even}(\sigma) + \text{even}(\text{inv}(\sigma)) = \text{even}(\sigma) + \text{inv}(\text{even}(\sigma))$  and  $\text{even}(\bar{1}) = \bar{1}$  and thus  $(\text{even}(\sigma + \text{inv}(\sigma)), \text{even}(\bar{1})) \in R$ . Similarly we can show that  $(\text{odd}(\sigma + \text{inv}(\sigma)), \text{odd}(\bar{1})) \in R$ . The other cases can be verified analogously, which shows that  $R$  is indeed a bisimulation and that the claims of the lemma hold true. QED

We can now turn to the proof of Theorem 6.6.

**Proof:** In order to prove the theorem, we show that the following relation is a  $\{(-)(0), \text{even}, \text{odd}\}$ -bisimulation:

$$R := \{(T, M + M'), (\text{inv}(T), \text{inv}(M + M')), (\bar{1}, \bar{1}), (\bar{0}, \bar{0})\}$$

Throughout the proof we are using the following identities which can be easily seen to be true for any stream  $\sigma \in A^\omega$ :

$$\begin{aligned} \sigma'(0) &= \text{odd}(\sigma)(0) \\ \text{odd}(\sigma') &= \text{even}(\sigma)' \\ \text{even}(\sigma') &= \text{odd}(\sigma) \end{aligned}$$

Let us now see that  $R$  is indeed a bisimulation. It suffices to check the bisimulation condition for first two elements of  $R$  as it obviously is fulfilled for the pairs  $(\bar{1}, \bar{1})$  and  $(\bar{0}, \bar{0})$ .

We have  $(M + M')(0) = M(0) + M'(0) = 0 + \text{odd}(M)(0) = 0 + \text{inv}(M) = 0 + 1 = 1 = T(0)$ . Furthermore  $\text{even}(M + M') = \text{even}(M) + \text{even}(M') = M + \text{odd}(M) = M + \text{inv}(M) = \bar{1}$  where the last equality follows from Item 1 of Lemma 6.7. As  $\text{even}(T) = \bar{1}$  this shows that  $(\text{even}(T), \text{even}(M + M')) \in R$ . In addition to that we have  $\text{odd}(M + M') = \text{odd}(M) + \text{odd}(M') = \text{inv}(M) + \text{even}(M)' = \text{inv}(M) + M' = \text{inv}(M + M')$  where the last equality is a consequence of Item 3 of Lemma 6.7. On the other hand  $\text{odd}(T) = \text{inv}(T)$  which proves that  $(\text{odd}(T), \text{odd}(M + M')) \in R$ .

Let us now check the bisimulation conditions for the pair  $(\text{inv}(T), \text{inv}(M + M'))$ . We calculate  $\text{inv}(T)(0) = 1 - T(0) = 1 - 1 = 0$  and  $\text{inv}(M + M')(0) = \text{inv}(M)(0) + (M')(0) = (1 - M(0)) + M'(0) = 1 + \text{odd}(M)(0) = 1 + \text{inv}(M)(0) = 1 + 1 = 0$ . where the first equality is again a consequence of Item 3 of Lemma 6.7. For the  $\text{even}$ -part we compute  $\text{even}(\text{inv}(T)) = \text{inv}(\text{even}(T)) = \text{inv}(\bar{1}) = \bar{0}$  where the last equality is Item 4 of Lemma 6.7. Moreover we have

$$\begin{aligned} \text{even}(\text{inv}(M + M')) &= \text{even}(\text{inv}(M) + M') = \text{even}(\text{inv}(M)) + \text{even}(M') \\ &= \text{inv}(\text{even}(M)) + \text{odd}(M) = \text{inv}(M) + \text{inv}(M) = \bar{0} \end{aligned}$$

which follows from Item 2 of Lemma 6.7. This implies that the pair  $(\text{even}(\text{inv}(T)), \text{even}(\text{inv}(M + M')))$  is an element of  $R$  as required.

Finally, consider the  $\text{odd}$ -part of the bisimulation condition: On the one hand we have  $\text{odd}(\text{inv}(T)) = \text{inv}(\text{odd}(T)) = \text{inv}(\text{inv}(T)) = T$  where the last equality is a consequence of Item 5 of Lemma 6.7. On the other hand we have  $\text{odd}(\text{inv}(M) + M') = \text{odd}(\text{inv}(M)) + \text{odd}(M') = \text{inv}(\text{odd}(M)) + \text{even}(M)' = \text{inv}(\text{inv}(M)) + M' = M + M'$  where the last equality is again a consequence of Item 5 of Lemma 6.7. Therefore  $(\text{odd}(\text{inv}(T)), \text{odd}(\text{inv}(M) + M')) \in R$  as required. QED

## References

- [AS99] J.-P. Allouche and J. Shallit. The Ubiquitous Prouhet–Thue–Morse Sequence. In *Sequences and Their Applications: Proceedings of SETA '98*, pages 1–16. Springer, 1999.
- [Bar03] F. Bartels. Generalised coinduction. *Mathematical Structures in Computer Science*, 13(2):321–348, 2003.

- [Ber05] Y. Bertot. Filters on coinductive streams, an application to eratosthenes' sieve. In Pawel Urzyczyn, editor, *TLCA*, volume 3461 of *LNCS*, pages 102–115. Springer, 2005.
- [BG06] M. Boreale and F. Gadducci. Processes as formal power series: A coinductive approach to denotational semantics. *Theor. Comput. Sci.*, 360(1-3):440–458, 2006.
- [BR88] J. Berstel and C. Reutenauer. *Rational series and their languages*, volume 12 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, 1988.
- [Brz64] J.A. Brzozowski. Derivatives of regular expressions. *Journal of the ACM*, 11(4):481–494, 1964.
- [Con71] J.H. Conway. *Regular algebra and finite machines*. Chapman and Hall, 1971.
- [Coq94] T. Coquand. Infinite objects in type theory. In Henk Barendregt and Tobias Nipkow, editors, *Types for Proofs and Programs, International Workshop TYPES'93, Nijmegen, The Netherlands, May 24–28, 1993, Selected Papers*, volume 806 of *Lecture Notes in Comput. Sci.*, pages 62–78. Springer-Verlag, 1994.
- [Dij81] E.W. Dijkstra. Hamming's exercise in SASL. Hand-written note EWD792, University of Texas, 1981.
- [EGH<sup>+</sup>07] J. Endrullis, C. Grabmayer, D. Hendriks, A. Ishihara, and J. W. Klop. Productivity of Stream Definitions. In *Proc. Conf. on Fundamentals of Computation Theory (FCT 2007)*, number 4639 in *LNCS*, pages 274–287. Springer, 2007.
- [EGH08] J. Endrullis, C. Grabmayer, and D. Hendriks. Data-Oblivious Stream Productivity. In *Proc. Conf. on Logic for Programming Artificial Intelligence and Reasoning (LPAR 2008)*, number 5330 in *LNCS*, pages 79–96. Springer, 2008.
- [EGH<sup>+</sup>10] J. Endrullis, C. Grabmayer, D. Hendriks, A. Ishihara, and J. W. Klop. Productivity of Stream Definitions. *Theoretical Computer Science*, 411:765–782, 2010. Extended version of [EGH<sup>+</sup>07].
- [Geu92] H. Geuvers. Inductive and coinductive types with iteration and recursion. In *Proceedings of the 1992 Workshop on Types for Proofs and Programs, Bastad*, pages 193–217, 1992.
- [GM03] P. Di Gianantonio and M. Miculan. A Unifying Approach to Recursive and Co-recursive Definitions. In H. Geuvers and F. Wiedijk, editors, *Types for Proofs and Programs, Second International Workshop, TYPES 2002, Berg en Dal, The Netherlands, April 24-28, 2002, Selected Papers*, volume 2646 of *LNCS*, pages 148–161. Springer, 2003.
- [Gor94] A.D. Gordon. A tutorial on co-induction and functional programming. In *Glasgow functional programming workshop*, pages 78–95. Springer, 1994.
- [Hin08] R. Hinze. Functional pearl: streams and unique fixed points. In *Proceeding of the 13th ACM SIGPLAN international conference on Functional programming, ICFP 2008*, pages 189–200. ACM, 2008.

- [HR10] H.H. Hansen and J.J.M.M. Rutten. Symbolic synthesis of Mealy machines from arithmetic bitstream functions. *Scientific Annals of Computer Science*, 2010. To appear.
- [JR97] B. Jacobs and J. J. M. M. Rutten. A Tutorial on (Co)Algebras and (Co)Induction. *Bulletin of EATCS*, 62:222–259, 1997.
- [Kom10] J. Komenda. Coinduction in concurrent timed systems. In *Proceedings CMCS 2010*, ENTCS. Elsevier, 2010.
- [KR10] C. Kupke and J.J.M.M. Rutten. Complete sets of cooperations. *Information and Computation*, 208(12):1398–1420, 2010.
- [KR11] C. A. Kupke and J. J. M. M. Rutten. On The Final Coalgebra Of Automatic Sequences. CWI Technical Report SEN-1112, CWI, December 2011.
- [KV08] C. Kupke and Y. Venema. Coalgebraic automata theory: Basic results. *Logical Methods in Computer Science*, 4(4), 2008.
- [KvS08] J. Komenda and J.H. van Schuppen. Modular control of discrete-event systems with coalgebra. *IEEE Trans. Automatic Control*, 53:447–460, 2008.
- [MBG10] D. Clark M. Boreale and D. Gorla. A semiring-based trace semantics for processes with applications to information leakage analysis. In C. Calude and V. Sassone, editors, *Proc. of 6th International IFIP Conference on Theoretical Computer Science (IFIP-TCS 2010)*. Springer, 2010.
- [McI99] M.D. McIlroy. Power series, power serious. *Journal of Functional Programming*, 9:323–335, 1999.
- [MD97] L.S. Moss and N. Danner. On the foundations of corecursion. *Logic Journal of the IGPL*, 5(2), 1997.
- [Mil80] R. Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer-Verlag, 1980.
- [Mos99] L.S. Moss. Coalgebraic logic. *Ann. Pure Appl. Logic*, 96(1-3):277–317, 1999.
- [NR10] M. Niqui and J. Rutten. Sampling, splitting and merging in coinductive stream calculus. In *MPC*, pages 310–330, 2010.
- [Par81] D.M.R. Park. Concurrency and automata on infinite sequences. In P. Deussen, editor, *Proceedings 5th GI conference*, volume 104 of *Lecture Notes in Computer Science*, pages 167–183. Springer-Verlag, 1981.
- [PE98] D. Pavlović and M.H. Escardó. Calculus in Coinductive Form. In *Proc. Symp. on Logic in Computer Science (LICS 1998)*, pages 408–417, 1998.
- [Rut98] J.J.M.M. Rutten. Automata and coinduction (an exercise in coalgebra). In D. Sangiorgi and R. de Simone, editors, *Proceedings of CONCUR’98*, volume 1466 of *LNCS*, pages 194–218, 1998.

- [Rut99] J.J.M.M. Rutten. Automata, power series, and coinduction: taking input derivatives seriously (extended abstract). In J. Wiedermann, P. van Emde Boas, and M. Nielsen, editors, *Proceedings of ICALP'99*, volume 1644 of *LNCS*, pages 645–654, 1999.
- [Rut00] J.J.M.M. Rutten. Universal coalgebra: a theory of systems. *Theoretical Computer Science*, 249(1):3–80, 2000. Fundamental Study.
- [Rut03] J.J.M.M. Rutten. Behavioural Differential Equations: a Coinductive Calculus of Streams, Automata, and Power Series. *Theoretical Computer Science*, 308(1-3):1–53, 2003.
- [Rut05] J.J.M.M. Rutten. Algebra, bitstreams, and circuits. In *Proceedings of the Dresden Conference 2004 (AAA68)*, volume 16 of *Contributions to General Algebra*, pages 231–250. Verlag Johannes Heyn, 2005.
- [Rut06] J.J.M.M. Rutten. Algebraic specification and coalgebraic synthesis of Mealy automata. In *Proceedings of FACS 2005*, volume 160 of *ENTCS*, pages 305–319. Elsevier Science Publishers, 2006.
- [Rut08] J.J.M.M. Rutten. Rational streams coalgebraically. *Logical Methods in Computer Science*, 4(3), 2008.
- [SR10] A. Silva and J.J.M.M. Rutten. A coinductive calculus of binary trees. *Information and Computation*, 208(5):578–593, 2010.
- [UV99] T. Uustalu and V. Vene. Primitive (co)recursion and course-of-value (co)iteration, categorically. *Informatika, Lith. Acad. Sci.*, 10(1):5–26, 1999.