Centrum voor Wiskunde en Informatica

REPORT*RAPPORT*

SEN

Software Engineering

Software ENgineering

An application of coinductive stream calculus to signal flow graphs

J.J.M.M. Rutten

REPORT SEN-E0305 OCTOBER 23, 2003

CWI is the National Research Institute for Mathematics and Computer Science. It is sponsored by the Netherlands Organization for Scientific Research (NWO).
CWI is a founding member of ERCIM, the European Research Consortium for Informatics and Mathematics.

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

Probability, Networks and Algorithms (PNA)

Software Engineering (SEN)

Modelling, Analysis and Simulation (MAS)

Information Systems (INS)

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

ISSN 1386-369X

An application of coinductive stream calculus to signal flow graphs

ABSTRACT
This report contains a set of lecture notes that were used in the spring of 2003 for a mini course of six lectures on the subject of streams, coinduction and signal flow graphs. It presents an application of coinductive stream calculus (as introduced in Technical Report SEN-R0023, CWI, Amsterdam, 2000) to signal flow graphs. In comparison to existing approaches, which are usually based on Laplace and Z-transforms, the model presented in these notes is very elementary. From a didactical point of view, the formal treatment of flow graphs is interesting because it deals with two fundamental phenomena in the theory of computation: memory (in the form of register or delay elements) and infinite behaviour (in the form of feedback).

2000 Mathematics Subject Classification: 68Q10, 68Q55, 68Q85
1998 ACM Computing Classification System: F.1, F.3
Keywords and Phrases: streams; coinduction; coalgebra; signal flow graphs

An Application of Coinductive Stream Calculus to Signal Flow Graphs

J.J.M.M. Rutten

CWI, P.O. Box 94079, 1090 GB Amsterdam
Vrije Universiteit, De Boelelaan 1081a, 1081 HV Amsterdam

Email: janr@cwi.nl, URL: www.cwi.nl/∼janr

Contents

Preface **iii**

1 Streams and coinduction **1**
1.1 Streams 1
1.2 Stream differential equations 3
1.3 Proofs by coinduction 7
1.4 Examples from Haskell 11
1.5 Solutions of differential equations⋆ 14
1.6 Discussion⋆ 17

2 Basic stream calculus **19**
2.1 Sum and product 19
2.2 Polynomial streams 22
2.3 Derivatives of sum and product 24
2.4 The operation of inverse 29
2.5 Solving linear equations 33

3 Stream circuits **37**
3.1 Basic circuits 37
3.2 Circuit composition 39
3.3 Circuits with feedback loops 47
3.4 Raising rabbits 52
3.5 Circuits and rational streams 53

Preface

Infinite sequences or *streams* occur at many different places in both mathematics and computer science. For the latter, think of bit streams flowing through the chips of your computer, or through the ether carrying messages from your mobile telephone. More generally, signals in the theory of signal processing are commonly represented by streams of real numbers. Also functions on streams are relevant in that setting, as they are the building blocks for filters and converters (such as the digital to analog converter in cd-players). Yet another example in computer science where streams appear is dataflow, which studies networks consisting of nodes and channels, through which streams of data elements flow. Such stream-based dataflow models have recently also been used in the area of component-based software engineering, where a process or software component can be specified in terms of a relation on the streams of ingoing and outgoing messages (or data elements) at each of its communication ports.

Streams occur also in various areas of mathematics. Examples are the use of streams in analysis: the basic notion of limit is formulated in terms of streams, and the Taylor series of analytical functions (such as $(0, 1, 0, -1, 0, 1, 0, -1, \ldots)$ for $\sin(x)$) are streams; in combinatorics, streams are often defined by recurrence relations or difference equations, for instance representing the solution of counting problems (such as the stream of Fibonacci numbers $(1, 1, 2, 3, 5, 8, \ldots)$); streams are used to model trajectories in dynamical systems.

Many more examples exist. These notes intend to study streams as such, in principle independent of any of the afore mentioned areas, but often taking examples from some of them, mostly from computer science. We shall develop systematic ways for: (1) defining and specifying streams (and functions on streams); (2) reasoning about streams, notably proving equalities between them; and (3) implementing streams, in particular using some basic form of (stream) circuits. know as (signal) flow graphs in the world of signal processing.

All of this will involve a bit of elementary but not so standard math-

ematics, which will be explained in all detail along the way. In short, because streams form an infinite datatype, the mathematical techniques of algebraic specification that are traditionally used for finite data types, are not really appropriate. Instead, we shall use a new proof and definition principle called *coinduction* (which is based on a recently developed general theory of dynamical systems called *coalgebra*). Underlying the use of coinduction is the view of streams as dynamical entities, whose behaviour consists of the repeatedly offering of the next element of the stream. We can define streams by specifying their behaviour, and the equality of two streams can be established by proving that they have the same behaviour (in other words, that they are 'behaviourally' equivalent). Although this may sound a bit mysterious at first, we hope that much of this will become clear already in the first chapter.

Acknowledgements

Special thanks are due to Clemens Grabmayer, who has assisted me during the Spring 2003 course on streams and coinduction at the VUA, and who has made numerous suggestions for improvements of these notes. I am also very grateful to the following persons for comments, suggestions, and pointers to the literature: Falk Bartels, Franck van Breugel, Jan van Eijck, and Jan Willem Klop.

Amsterdam, Spring 2003

Jan Rutten

Chapter 1

Streams and coinduction

We introduce the set of streams (with elements in an arbitrary set A) and explain how to define streams and operations on streams by stream differential equations, and how to prove facts about streams by coinduction.

1.1 Streams

Let A be any set. We define the set A^ω of all streams over A as

$$A^\omega = \{\sigma \mid \sigma : \{0, 1, 2, \ldots\} \to A\}$$

In this chapter, we make no assumptions on A, but in some of the examples, we shall look at particular instantiations of A (such as the natural numbers); in Chapter 2, A will be the set $\mathbb{R}$ of real numbers.

For a stream σ, we call $\sigma(0)$ the *initial value* of σ. We define the *derivative* σ' of a stream σ, for all $n \geq 0$, by

$$\sigma'(n) = \sigma(n + 1)$$

Initial value and derivative are usually called *head* and *tail* but the present terminology helps us, as we shall see, to develop a *calculus* of streams in close analogy to classical calculus in analysis.

Although streams will be viewed and handled as single mathematical entities, it will at various moments be convenient to refer to the individual elements of which they are made. For this, we shall use the following notation:

$$\sigma \quad = \quad (\sigma(0), \sigma(1), \sigma(2), \ldots)$$

(Similarly, we shall write $\tau = (\tau(0), \tau(1), \tau(2), \ldots)$ and the like.) With this notation, the derivative of σ is given by

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

For any $n \geq 0$, $\sigma(n)$ is called the n-th *element* of σ. It can also be expressed in terms of higher-order stream derivatives, defined, for all $k \geq 0$, by

$$\sigma^{(0)} = \sigma, \quad \sigma^{(k+1)} = (\sigma^{(k)})'$$

(For higher-order derivatives of order two or three, both notations will be used: we shall occasionally write σ'', σ''' as well as $\sigma^{(2)}$, $\sigma^{(3)}$.)

Lemma 1.1 *The n-th element of a stream σ is given by*

$$\sigma(n) = \sigma^{(n)}(0)$$

□

Exercise 1.1 Prove Lemma 1.1 by showing by induction on n: for all $n \geq 0$ and for all $i \geq 0$,

$$\sigma(n + i) = \sigma^{(n)}(i)$$

□

We shall also use the following notation, which will be convenient when we want to compute the first few elements of a stream. For $a \in A$ and $\tau \in A^\omega$, we define

$$a : \tau = (a, \tau(0), \tau(1), \tau(2), \ldots)$$

With this notation, we have for any $\sigma \in A^\omega$,

$$\sigma = \sigma(0) : \sigma' = \sigma(0) : (\sigma'(0) : \sigma^{(2)}) = \sigma(0) : (\sigma(1) : \sigma^{(2)})$$

Leaving out the brackets, we have, more generally for any $n \geq 1$,

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

Example 1.2 Let $\sigma, \tau \in A^\omega$ be defined by

$$\sigma(n) = \begin{cases} a & \text{if } n \text{ is even} \\ b & \text{if } n \text{ is odd} \end{cases}$$

$$\tau(n) = \begin{cases} b & \text{if } n \text{ is even} \\ a & \text{if } n \text{ is odd} \end{cases}$$

Thus $\sigma = (a, b, a, b, a, b, \ldots)$ and $\tau = (b, a, b, a, b, a, \ldots)$. We have: $\sigma' = \tau$ and $\tau' = \sigma$. For a formal proof of $\sigma' = \tau$, first note that, for $n \geq 0$,

$$\sigma(n+1) = \tau(n)$$

(If n is even then $n+1$ is odd, and $\sigma(n+1) = b = \tau(n)$; similarly for the case that n is odd.) It follows that

$$\sigma'(n) = \sigma(n+1) = \tau(n)$$

for all $n \geq 0$. Thus $\sigma' = \tau$. Clearly, one proves $\tau' = \sigma$ in the same way. Note that we also have $\sigma'' = \sigma$. □

Exercise 1.2 Let $\sigma, \tau \in A^\omega$ be defined by, for all $n \geq 0$,

$$\sigma(n) = \begin{cases} a & \text{if } n = 3k \text{ for some } k \geq 0 \\ b & \text{if } n = 3k+1 \text{ for some } k \geq 0 \\ c & \text{if } n = 3k+2 \text{ for some } k \geq 0 \end{cases}$$

$$\tau(n) = \begin{cases} c & \text{if } n = 3k \text{ for some } k \geq 0 \\ a & \text{if } n = 3k+1 \text{ for some } k \geq 0 \\ b & \text{if } n = 3k+2 \text{ for some } k \geq 0 \end{cases}$$

Express how σ and τ are related using the operation of derivative. □

Exercise 1.3 Let $a, b \in A$. Give a formula for $\sigma(n)$, for arbitrary $n \geq 0$, in each of the following cases:

(i) $\sigma = (a, a, b, a, a, b, a, a, b, \ldots)$

(ii) $\sigma = (a, b, b, c, c, c, a, b, b, c, c, c, a, b, b, c, c, c, \ldots)$

(iii) $\sigma = (a, b, a, a, b, a, a, a, b, \ldots)$

□

1.2 Stream differential equations

A particularly convenient and succinct way of defining streams is by means of so-called *stream differential equations*. In analogy to differential equations from mathematical analysis, such as $f'(x) = f(x)$, $f(0) = 1$, which defines the function $f(x) = e^x$, stream differential equations specify streams (and functions on streams) in terms of their derivatives and initial values.

In case differential equations are not your favourite topic in mathematics, there is no reason to become worried at this point. No previous knowledge will be required and, more importantly, the theory of stream

differential equations is much simpler and very intuitive. We shall become somewhat more formal later in this chapter; for now, we explain the use of stream differential equations by a number of examples.

Example 1.3 Let $a \in A$ and consider the following stream differential equation:

derivative	initial value
$\sigma' = \sigma$	$\sigma(0) = a$

The intended interpretation of this equation is that there exists a *unique* stream σ with derivative $\sigma' = \sigma$ and initial value $\sigma(0) = a$. The differential equation should be read as a *definition* of this unique stream σ, which is called the *solution* of the differential equation. (In Section 1.5, we shall return to the question whether such a unique solution always exists.) Computing the first few elements of σ gives

$$\sigma = \sigma(0) : \sigma' = a : \sigma = a : a : \sigma = a : a : a : \sigma$$

One can easily prove by induction on n that the n-th derivative $\sigma^{(n)}$ of σ satisfies $\sigma^{(n)} = \sigma$. Since $\sigma(n) = \sigma^{(n)}(0)$, by Lemma 1.1 above, it follows that

$$\sigma(n) = \sigma^{(n)}(0) = \sigma(0) = a$$

In other words, $\sigma = (a, a, a, \ldots)$. □

Example 1.4 Let a, $b \in A$ and consider the following differential equation:

derivative	initial value
$\sigma'' = \sigma$	$\sigma(0) = a$, $\sigma'(0) = b$

Since this equation specifies σ in terms of its *second* derivative σ'' $(= \sigma^{(2)})$, we call it a *higher-order* differential equation. Note that it not only specifies the initial value of σ but also that of σ'. Computing again the first few values, we find

$$\sigma = \sigma(0) : \sigma' = a : \sigma'(0) : \sigma'' = a : b : \sigma = a : b : a : b : \sigma$$

One can easily prove that, for all $n \geq 0$,

$$\sigma^{(n)} = \begin{cases} \sigma & \text{if } n \text{ is even} \\ \sigma' & \text{if } n \text{ is odd} \end{cases}$$

Because $\sigma(n) = \sigma^{(n)}(0)$, by Lemma 1.1, it follows that

$$\sigma(n) = \begin{cases} a & \text{if } n \text{ is even} \\ b & \text{if } n \text{ is odd} \end{cases}$$

Thus $\sigma = (a, b, a, b, a, b, \ldots)$. Instead of a higher-order differential equation, we can also use a *system* of differential equations to define this stream σ:

derivative	initial value
$\sigma' = \tau$	$\sigma(0) = a$
$\tau' = \sigma$	$\tau(0) = b$

This system of equations defines two streams σ and τ, with σ as before and with $\tau = (b, a, b, a, b, a, \ldots)$. □

Exercise 1.4 Let $a, b, c \in A$. Define the following streams by means of a differential equation (or a system of differential equations):

(i) $\sigma = (a, b, c, a, b, c, a, b, c, \ldots)$

(ii) $\sigma = (a, a, b, a, a, b, a, a, b, \ldots)$

(iii) $\sigma = (a, b, b, c, c, c, a, b, b, c, c, c, a, b, b, c, c, c, \ldots)$

(iv) Try the same for an arbitrary stream σ.

□

Sofar we have used stream differential equations to define individual streams, such as $(a, b, a, b, a, b, \ldots)$. We use differential equations also for the definition of *functions* on streams. Here are again some examples.

Example 1.5 Consider the following differential equation:

derivative	initial value
$even(\sigma)' = even(\sigma'')$	$even(\sigma)(0) = \sigma(0)$

The intended meaning of this equation is that there exists *for every stream* σ a unique stream called $even(\sigma)$ such that $even(\sigma)' = even(\sigma'')$ and $even(\sigma)(0) = \sigma(0)$. This single equation is in fact an infinite system of equations, one for each $\sigma \in A^\omega$. All these infinitely many equations together define a function

$$even : A^\omega \to A^\omega$$

that assigns to a stream σ the unique stream $even(\sigma)$ specified by these equations. How does $even(\sigma)$, for a given stream σ, look like? Computing the first few values gives

$$\begin{aligned} even(\sigma) &= even(\sigma)(0) : even(\sigma)' \\ &= \sigma(0) : even(\sigma'') \\ &= \sigma(0) : even(\sigma'')(0) : even(\sigma'')' \\ &= \sigma(0) : \sigma(2) : even(\sigma^{(4)}) \end{aligned}$$

and so on. By induction on n, we can prove, for any $n \geq 0$, that

$$(even(\sigma))^{(n)} \quad = \quad even(\sigma^{(2n)}) \tag{1.1}$$

It follows that

$$\begin{aligned}(even(\sigma))(n) &= (even(\sigma))^{(n)}(0)\\ &= even(\sigma^{(2n)})(0)\\ &= \sigma^{(2n)}(0)\\ &= \sigma(2n)\end{aligned}$$

Thus the function *even* maps a stream σ to

$$even(\sigma) = (\sigma(0), \sigma(2), \sigma(4), \ldots)$$

as its name already suggested it would. □

Exercise 1.5 Prove identity (1.1) above. □

Exercise 1.6 Consider the following differential equation:

derivative	initial value
$odd(\sigma)' = odd(\sigma'')$	$odd(\sigma)(0) = \sigma(1)$

Show that it defines a function $odd : A^\omega \to A^\omega$ that maps a stream σ to the stream

$$odd(\sigma) = (\sigma(1), \sigma(3), \sigma(5), \ldots)$$

How are the functions *even* and *odd* related? □

Example 1.6 Let the function $zip : A^\omega \times A^\omega \to A^\omega$ be defined by the following differential equation (more precisely: system of differential equations, one for every σ and τ in A^ω):

derivative	initial value
$zip(\sigma, \tau)' = zip(\tau, \sigma')$	$zip(\sigma, \tau)(0) = \sigma(0)$

For given σ, $\tau \in A^\omega$, the first few values of the stream $zip(\sigma, \tau)$ are:

$$\begin{aligned}zip(\sigma, \tau) &= zip(\sigma, \tau)(0) : zip(\sigma, \tau)'\\ &= \sigma(0) : zip(\tau, \sigma')\\ &= \sigma(0) : zip(\tau, \sigma')(0) : zip(\tau, \sigma')'\\ &= \sigma(0) : \tau(0) : zip(\sigma', \tau')\\ &= \sigma(0) : \tau(0) : zip(\sigma', \tau')(0) : zip(\sigma', \tau')'\\ &= \sigma(0) : \tau(0) : \sigma(1) : zip(\tau', \sigma'')\end{aligned}$$

Continuing this way, we see (and can prove formally by induction):

$$zip(\sigma, \tau) = (\sigma(0), \tau(0), \sigma(1), \tau(1), \sigma(2), \tau(2), \ldots)$$

□

Exercise 1.7 (a) Define by means of a higher-order differential equation a function $double : A^\omega \to A^\omega$ that maps a stream σ to

$$double(\sigma) = (\sigma(0), \sigma(0), \sigma(1), \sigma(1), \sigma(2), \sigma(2), \ldots)$$

(b) How are the functions *double* and *zip* related? □

1.3 Proofs by coinduction

Recall the streams $(a, a, a, \ldots)$ and $(a, b, a, b, a, b, \ldots)$ defined in Examples 1.3 and 1.4, and let us call them $[a]$ and $[ab]$, respectively. They satisfy:

$$[a] = a : [a], \quad [ab] = a : b : [ab]$$

Now suppose that we want to prove the following equality:

$$even([ab]) = [a]$$

One could argue that this identity is so trivial that no proof is needed. But for one thing, we want to be really precise and formal, and furthermore, we shall see many other examples that are much more complicated. So recalling the definition of the function $even : A^\omega \to A^\omega$ from Example 1.5, we compute the first few values of $even([ab])$:

$$\begin{aligned} even([ab]) &= even([ab])(0) : even([ab])' \\ &= [ab](0) : even([ab]'') \\ &= a : even([ab]) \end{aligned}$$

Comparing this with $[a] = a : [a]$, we see that the initial values of both streams are equal: $even([ab])(0) = a = [a](0)$. If we can prove that their derivatives are equal too, then we can conclude that $even([ab]) = [a]$. Since $even([ab])' = even([ab])$ and $[a]' = [a]$, we see that in order to prove $even([ab]) = [a]$, we have to prove $even([ab]) = [a]$. Thus our reasoning seems to be trapped in a vicious circle.

Still, the two equalities $even([ab]) = a : even([ab])$ and $[a] = a : [a]$ taken together seem to leave no doubt about the validity of $even([ab]) = [a]$, since they allow us to prove that both streams agree on initial segments of arbitrary length.

Below we introduce a proof method, called coinduction, that does allow us to give a formal proof of identities such as the one above. It will be formulated in terms of the following notion.

Definition 1.7 A *bisimulation* on A^ω is a relation $R \subseteq A^\omega \times A^\omega$ such that, for all σ and τ in A^ω,

$$\text{if } \langle \sigma, \tau \rangle \in R \quad \text{then} \quad \begin{cases} (i) & \sigma(0) = \tau(0) \quad \text{and} \\ (ii) & \langle \sigma', \tau' \rangle \in R \end{cases}$$

(We shall sometimes write $\sigma \, R \, \tau$ for $\langle \sigma, \tau \rangle \in R$.) □

If there exists a bisimulation relation R with $\sigma \, R \, \tau$ then we write $\sigma \sim \tau$ and say that σ and τ are *bisimilar*. In other words, the bisimilarity relation $\sim$ is the union of all bisimulations:

$$\sim = \bigcup \{ R \subseteq A^\omega \times A^\omega \mid R \text{ is a bisimulation relation } \}$$

Lemma 1.8 *If $R \subseteq A^\omega \times A^\omega$ and $S \subseteq A^\omega \times A^\omega$ are bisimulations then both the union $R \cup S$ and the relational composition*

$$R \circ S = \{ \langle \sigma, \tau \rangle \in A^\omega \times A^\omega \mid \exists \rho \in A^\omega, \ \langle \sigma, \rho \rangle \in R \ \wedge \ \langle \rho, \tau \rangle \in S \ \}$$

of R and S are again bisimulation relations. □

Exercise 1.8 Prove Lemma 1.8. Also prove that the bisimilarity relation $\sim$ is itself a bisimulation relation. □

Theorem 1.9 [Coinduction] *For all $\sigma, \tau \in A^\omega$, if there exists a bisimulation relation $R \subseteq A^\omega \times A^\omega$ with $\langle \sigma, \tau \rangle \in R$, then $\sigma = \tau$. In other words,*

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

Proof: Consider two streams σ and τ and let $R \subseteq A^\omega \times A^\omega$ be a bisimulation on A^ω containing the pair $\langle \sigma, \tau \rangle$. It follows by induction on n that $\langle \sigma^{(n)}, \tau^{(n)} \rangle \in R$, for all $n \geq 0$, because R is a bisimulation. This implies, again because R is a bisimulation, that $\sigma^{(n)}(0) = \tau^{(n)}(0)$, for all $n \geq 0$. By Lemma 1.1, $\sigma(n) = \tau(n)$, for all $n \geq 0$. Now $\sigma = \tau$ follows. □

Exercise 1.9 Show that the converse of Theorem 1.9 also holds: for all $\sigma, \tau \in A^\omega$, if $\sigma = \tau$ then $\sigma \sim \tau$. [Hint: show that $\{ \langle \sigma, \sigma \rangle \mid \sigma \in A^\omega \}$ is a bisimulation relation on A^ω.] □

In order to prove the equality of two streams σ and τ, it is according to Theorem 1.9 sufficient to establish the existence of a bisimulation relation $R \subseteq A^\omega \times A^\omega$ with $\langle \sigma, \tau \rangle \in R$. This proof principle is called *coinduction* (and is sometimes also referred to as the *bisimulation proof method*). The coinduction proof principle can be seen as a systematic way of strenghtening the statement one is trying to prove: instead of proving only the single identity $\sigma = \tau$, one computes the smallest bisimulation relation R that contains the pair $\langle \sigma, \tau \rangle$. The construction of R is always the same, and amounts to the computation of the closure of $\{\langle \sigma, \tau \rangle\}$ under taking derivatives; at every stage of the construction of R, one should check that the initial values of the streams in newly added pairs are equal. By the coinduction proof principle, it follows that $\alpha = \beta$ for all pairs $\langle \alpha, \beta \rangle \in R$. Since $\langle \sigma, \tau \rangle \in R$, by the construction of R, it follows in particular that $\sigma = \tau$.

Example 1.10 We prove the equality $even([ab]) = [a]$, which we discussed at the beginning of this section, by coinduction. We define $R = \{\langle even([ab]), [a] \rangle\}$. In order to show that R is a bisimulation, we have to check for the pair $\langle even([ab], [a] \rangle$ that it satisfies the two bisimulation conditions of Definition 1.7:

$$(i)\ \ even([ab])(0) = [a](0) \quad \text{and} \quad (ii)\ \ \langle even([ab])', [a]' \rangle \in R$$

These follow from $even([ab])(0) = a = [a](0)$, $even([ab])' = even([ab])$, and $[a]' = [a]$. Now $even([ab]) = [a]$ follows by coinduction (Theorem 1.9). $\Box$

Example 1.11 We prove by coinduction: for all $\sigma, \tau \in A^\omega$,

$$even(zip(\sigma, \tau)) = \sigma$$

(The function *zip* was defined in Example 1.6.) In order to find a suitable bisimulation, we compute, for given σ and τ,

$$\begin{aligned} even(zip(\sigma,\tau)) &= even(zip(\sigma,\tau))(0) : even(zip(\sigma,\tau))' \\ &= zip(\sigma,\tau)(0) : even(zip(\sigma,\tau)'') \\ &= \sigma(0) : even(zip(\sigma',\tau')) \end{aligned}$$

Comparing this to $\sigma = \sigma(0) : \sigma'$ suggests the following definition for a relation $R \subseteq A^\omega \times A^\omega$:

$$R = \{\langle even(zip(\sigma,\tau)),\ \sigma \rangle \mid \sigma, \tau \in A^\omega\}$$

For any $\sigma, \tau \in A^\omega$, $even(zip(\sigma,\tau))(0) = \sigma(0)$, and $even(zip(\sigma,\tau))' = even(zip(\sigma',\tau'))$ implies $\langle even(zip(\sigma,\tau))',\ \sigma' \rangle \in R$. As a consequence, R is a bisimulation. The equality $even(zip(\sigma,\tau)) = \sigma$ now follows by coinduction. $\Box$

That was easy. We defined R simply as the set of all pairs that we wanted to prove equal. As it turned out, R was a bisimulation and the identity we were after followed by coinduction. This is the right way to start proofs by coinduction in general. Often, however, the relation R has to be extended further before it satisfies condition (ii) of the definition of bisimulation. In other words, the relation has to be closed under taking derivatives. Everytime that new pairs are added to the relation, conditions (i) and (ii) of the definition of bisimulation have to be checked again. The following example clearly illustrates what we mean.

Example 1.12 We prove by coinduction: for all $\sigma \in A^\omega$,

$$zip(even(\sigma),\, odd(\sigma)) = \sigma \tag{1.2}$$

(The function *odd* was introduced in Exercise 1.6.) As before, we begin by defining

$$R = \{\langle zip(even(\sigma),\, odd(\sigma)),\, \sigma\rangle \mid \sigma \in A^\omega\}$$

In order to check that R satisfies the bisimulation conditions (i) and (ii) of Definition 1.7, we compute:

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

For all pairs $\langle zip(even(\sigma),\, odd(\sigma)),\, \sigma\rangle \in R$, we see that the initial values agree:

$$zip(even(\sigma),\, odd(\sigma))(0) = \sigma(0)$$

but that the pair of derivatives:

$$\langle zip(even(\sigma),\, odd(\sigma))',\, \sigma'\rangle = \langle zip(odd(\sigma), even(\sigma'')),\, \sigma'\rangle$$

is *not* in R. Therefore we extend the relation R by including these latter pairs as well. Thus our second proposal for R now looks like

$$\begin{aligned} R &= \{\langle zip(even(\sigma),\, odd(\sigma)),\, \sigma\rangle \mid \sigma \in A^\omega\} \\ &\cup \{\langle zip(odd(\sigma),\, even(\sigma'')),\, \sigma'\rangle \mid \sigma \in A^\omega\} \end{aligned}$$

Next it should be checked whether the two bisimulation conditions (i) and (ii) are satisfied by all the newly added pairs $\langle zip(odd(\sigma),\, even(\sigma'')),\, \sigma'\rangle$. We compute:

$$\begin{aligned} & zip(odd(\sigma),\, even(\sigma'')) \\ &= zip(odd(\sigma), even(\sigma''))(0) : zip(odd(\sigma), even(\sigma''))' \\ &= \sigma(1) : zip(even(\sigma''),\, odd(\sigma'')) \end{aligned}$$

Condition (i) is satisfied, since

$$zip(odd(\sigma),\, even(\sigma''))(0) = \sigma(1) = \sigma'(0)$$

The pair of derivatives now *is* in R:

$$\langle zip(odd(\sigma),\, even(\sigma''))',\, \sigma''\rangle = \langle zip(even(\sigma''),\, odd(\sigma'')),\, \sigma''\rangle \in R$$

This concludes the construction of R and the proof that it is a bisimulation. Identity (1.2) now follows by coinduction. □

Exercise 1.10 Prove by coinduction that $odd(zip(\sigma, \tau)) = \tau$, for all $\sigma,\, \tau \in A^\omega$. □

1.4 Examples from Haskell

Stream differential equations can be viewed as recipes for the stepwise computation of streams. Here we illustrate this point further by a number of examples that are phrased in a style that is inspired by the functional programming language Haskell [HT]. Although we do not literally use Haskell syntax, it would be straightforward to transform any of the examples below into a corresponding Haskell program (which could actually be executed to verify 'by execution' any of the properties that we shall prove). In all of the examples, we set

$$A = \mathbb{N} = \{0, 1, 2, \ldots\}$$

and work with the set $\mathbb{N}^\omega$ of streams of natural numbers.

Example 1.13 Suppose we are given a function $f : \mathbb{N} \to \mathbb{N}$. It induces a function $map[f] : \mathbb{N}^\omega \to \mathbb{N}^\omega$ that is defined by the following stream differential equation:

derivative	initial value
$map[f](\sigma)' = map[f](\sigma')$	$map[f](\sigma)(0) = f(\sigma(0))$

Computing the first few elements of $map[f](\sigma)$, for a stream σ, gives:

$$\begin{aligned} map[f](\sigma) &= map[f](\sigma)(0) : map[f](\sigma)' \\ &= f(\sigma(0)) : map[f](\sigma') \\ &= f(\sigma(0)) : map[f](\sigma')(0) : map[f](\sigma')' \\ &= f(\sigma(0)) : f(\sigma(1)) : map[f](\sigma'') \end{aligned}$$

and so on. Thus $map[f](\sigma) = (f(\sigma(0)),\, f(\sigma(1)),\, f(\sigma(2)), \ldots)$. □

Exercise 1.11 Consider two functions $f : \mathbb{N} \to \mathbb{N}$ and $g : \mathbb{N} \to \mathbb{N}$ and let their composition be defined as usual:

$$g \circ f : \mathbb{N} \to \mathbb{N}, \quad n \mapsto g(f(n))$$

Prove by coinduction that $map[g](map[f](\sigma)) = map[g \circ f](\sigma)$, for all $\sigma \in \mathbb{N}^\omega$. □

Example 1.14 In order to look at some concrete examples, let $k \geq 0$ be a given natural number and consider the following function on the natural numbers:

$$+k : \mathbb{N} \to \mathbb{N}, \quad n \mapsto n + k$$

Taking $k = 1$, we can define the stream $nats \in \mathbb{N}^\omega$ as follows:

derivative	initial value
$nats' = map[+1](nats)$	$nats(0) = 0$

Let us check whether this defines indeed the stream $(0, 1, 2, \ldots)$, by computing the first few elements of $nats$:

$$\begin{aligned} nats &= nats(0) : nats' \\ &= 0 : map[+1](nats) \\ &= 0 : (map[+1](nats))(0) : map[+1](nats)' \\ &= 0 : 1 : map[+1](nats') \\ &= 0 : 1 : map[+1](map[+1](nats)) \\ &= 0 : 1 : map[+2](nats) \end{aligned}$$

and so on. (For the last equality, we used Exercise 1.11 and the fact that $(+1) \circ (+1) = +2$.) Similarly, we can define the stream $odds \in \mathbb{N}^\omega$ of the odd natural numbers by the following stream differential equation:

derivative	initial value
$odds' = map[+2](odds)$	$odds(0) = 1$

□

Exercise 1.12 Compute the first few elements of the stream $odds$ defined above. Give a definition of the stream $evens$ of the even natural numbers, again using the $map[f]$ construct. □

Example 1.15 For all functions $f : \mathbb{N} \to \mathbb{N}$ we define a function

$$iterate[f] : \mathbb{N} \to \mathbb{N}^\omega$$

that assigns to a natural number $a \in \mathbb{N}$ the unique stream $iterate[f](a)$ defined by the following differential equation:

derivative	initial value
$iterate[f](a)' = iterate[f](f(a))$	$iterate[f](a)(0) = a$

Computing the first few elements, we find:

$$\begin{aligned} iterate[f](a) &= iterate[f](a)(0) : iterate[f](a)' \\ &= a : iterate[f](f(a))) \\ &= a : f(a) : iterate[f](f(f(a))) \end{aligned}$$

and so on. We see that the stream $iterate[f](a)$ consists of the numbers obtained by iteratively applying the function f to the initial value a:

$$iterate[f](a) = (a, f(a), f(f(a)), \ldots)$$

As an example, we take for f the function $+1 : \mathbb{N} \to \mathbb{N}$ (which maps a natural number n to $n + 1$) and for a the natural number 0, and we put:

$$Nats = iterate[+1](0)$$

Computing its first few values, we see

$$\begin{aligned} Nats &= iterate[+1](0) \\ &= iterate[+1](0)(0) : iterate[+1](0)' \\ &= 0 : iterate[+1](1) \\ &= 0 : 1 : iterate[+1](2) \end{aligned}$$

and so on. Thus also this stream is equal to the stream $(0, 1, 2, \ldots)$ of natural numbers. As yet another example of a proof by coinduction, let us prove

$$Nats = nats$$

with $nats$ as defined in Example 1.14 above. Since we saw, for the first few elements of these streams, that

$$Nats = 0 : 1 : iterate[+1](2), \quad nats = 0 : 1 : map[+2](nats)$$

the following definition of a relation $R \subseteq \mathbb{N} \times \mathbb{N}$ suggests itself:

$$R = \{\langle Nats, nats\rangle\} \cup \{\langle iterate[+1](k),\ map[+k](nats)\rangle \mid k \in \mathbb{N}\}$$

It is now easy to show that R is a bisimulation relation, whence $Nats = nats$, by coinduction. □

Exercise 1.13 Let the stream $Odds$ of odd natural numbers be defined by $Odds = iterate[+2](1)$. Compute its first few elements and then prove $Odds = odds$ by coinduction (the latter stream was defined in Example 1.14). □

Exercise 1.14 Prove by coinduction that for all $f : \mathbb{N} \to \mathbb{N}$ and $a \in \mathbb{N}$,

$$map[f](iterate[f](a)) = iterate[f](f(a))$$

□

Exercise 1.15 (a) Compute the first few elements of $iterate[+2](2)$ and of $iterate[+3](3)$.

(b) Define a function $merge : \mathbb{N}^\omega \times \mathbb{N}^\omega \to \mathbb{N}^\omega$ that merges two streams σ and τ of natural numbers in such a way that if σ and τ are ordered then so is the stream $merge(\sigma, \tau)$.

(c) Compute the first few elements of

$$merge(iterate[+2](2), iterate[+3](3))$$

□

Exercise 1.16 For a given function $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, let the function

$$ZipWith[f] : \mathbb{N}^\omega \times \mathbb{N}^\omega \to \mathbb{N}^\omega$$

be defined, for any σ, $\tau \in \mathbb{N}^\omega$, by the following differential equation:

derivative:	$ZipWith[f](\sigma, \tau)' = ZipWith[f](\sigma', \tau')$
initial value:	$ZipWith[f](\sigma, \tau)(0) = f(\sigma(0), \tau(0))$

Taking for f the function $+ : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, compute the first few elements of $ZipWith[+](\sigma, \tau)$, for given σ and τ. Define the stream of natural numbers, using the function $ZipWith[+]$ and the constant stream $[1] = (1, 1, 1, \ldots)$ of ones. □

1.5 Solutions of differential equations*

We have seen many examples of how stream differential equations can be used for the definition of streams and stream functions. Here we give a sketch of a proof that such (systems of) stream differential equations have a unique solution. The proof will be based on the fact that the set A^ω of all streams can be turned into a so-called final stream automaton, a notion which is introduced first.

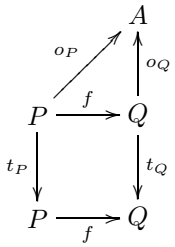
A *stream automaton* (also called stream *coalgebra*) is a triple

$$\langle Q,\, o_Q : Q \to A,\, t_Q : Q \to Q \rangle$$

(sometimes also denoted by $\langle Q, o_Q, t_Q \rangle$) consisting of a set Q of states, an output function $o_Q : Q \to A$, and a transition function $t_Q : Q \to Q$. For two stream automata $\langle P, o_P, t_P \rangle$ and $\langle Q, o_Q, t_Q \rangle$, a function $f : P \to Q$ is a *homomorphism*, denoted by

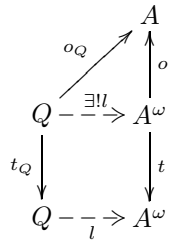
$$f : \langle P, o_P, t_P \rangle \to \langle Q, o_Q, t_Q \rangle$$

whenever, for all p in P, $o_P(p) = o_Q(f(p))$ and $f(t_P(p)) = t_Q(f(p))$. In other words, a function $f : P \to Q$ is a homomorphism if it makes both the triangle and the square in the diagram below commute:



The set A^ω of all streams can itself be turned into a stream automaton as follows. Defining $o : A^\omega \to A$ by $o(\sigma) = \sigma(0)$ and $t : A^\omega \to A^\omega$ by $t(\sigma) = \sigma'$, we obtain a stream automaton $\langle A^\omega, o, t \rangle$. It has the following universal property.

Theorem 1.16 *The automaton $\langle A^\omega, o, t \rangle$ is* final *among the family of all stream automata. That is, for any automaton $\langle Q, o_Q, t_Q \rangle$ there exists a unique homomorphism $l : \langle Q, o_Q, t_Q \rangle \to \langle A^\omega, o, t \rangle$:*



Proof: Let $\langle Q, o_Q, t_Q \rangle$ be a stream automaton and define $l : Q \to A^\omega$ as

$$l(q) = (o(q),\, o(t(q)),\, o(t(t(q))),\, \ldots)$$

for q in Q. It is straightforward to show that l is a homomorphism from $\langle Q, o_Q, t_Q \rangle$ to $\langle A^\omega, o, t \rangle$. For uniqueness, suppose f and g are homomorphisms from Q to A^ω. The equality of f and g follows by coinduction from

the fact that $R = \{\langle f(q), g(q)\rangle \mid q \in Q\}$ is a bisimulation on A^ω, which is proved next. Consider $\langle f(q), g(q)\rangle \in R$. Because f and g are homomorphisms, $o(f(q)) = o_Q(q) = o(g(q))$. Furthermore, $t(f(q)) = f(t_Q(q))$ and $t(g(q)) = g(t_Q(q))$. Because $\langle f(t_Q(q)), g(t_Q(q))\rangle \in R$, this shows that R is a bisimulation. Now $f = g$ follows by the coinduction proof principle Theorem 1.9. □

By the finality of the automaton $\langle A^\omega, o, t\rangle$ we can prove for many stream differential equations that they have a unique solution (thereby justifying their use as definitions). We present a few examples.

Example 1.17 Recall our first stream differential equation:

derivative	initial value
$\sigma' = \sigma$	$\sigma(0) = a$

Consider a stream automaton $\langle Q, o_Q, t_Q\rangle$ consisting of a singleton set $Q = \{q\}$ with $o_Q(q) = a$ and $t_Q(q) = q$. By Theorem 1.16, there exists a unique homomorphism $l : Q \to A^\omega$. We now define $\sigma = l(q)$. Because l is a homomorphism, it follows that σ is a solution of the differential equation: $\sigma' = t(\sigma) = t(l(q)) = l(t_Q(q)) = l(q) = \sigma$ and $\sigma(0) = o(\sigma) = o(l(q)) = o_Q(q) = a$. If τ is a stream with $\tau' = \tau$ and $\tau(0) = a$, then $\sigma = \tau$ follows, by the coinduction proof principle Theorem 1.9, from the fact that $\{\langle \sigma, \tau\rangle\}$ is a bisimulation relation on A^ω. Thus σ is the only solution of the differential equation. □

Example 1.18 In order to prove that the system of stream differential equations of Example 1.6 has a unique solution, and therefore uniquely determines the function $zip : A^\omega \times A^\omega \to A^\omega$, we consider a stream automaton $\langle Q, o_Q, t_Q\rangle$, with $Q = A^\omega \times A^\omega$ and with, for all $\langle \sigma, \tau\rangle \in Q$,

$$t_Q(\langle \sigma, \tau\rangle) = \langle \tau, \sigma'\rangle, \quad o_Q(\langle \sigma, \tau\rangle) = \sigma(0)$$

As before, there exists by Theorem 1.16, a unique homomorphism $l : Q \to A^\omega$. We now define $zip(\sigma, \tau) = l(\langle \sigma, \tau\rangle)$. Similar to the first example, it is not difficult to prove that zip is the unique function satisfying the above system of stream differential equations. □

The first example above involved a stream automaton with only one state; in the second example, an infinite automaton was used. One can, more generally, show the existence of a unique solution for more complicated systems of stream differential equations by using more complicated stream automata. All the stream differential equations that we shall encounter in these notes, can be shown to have a unique solution in this manner. We shall not present any details here and refer the interested reader to [Rut01].

1.6 Discussion*

In the language of category theory, any stream automaton (over the alphabet A) is a coalgebra of the functor that maps a set S to the set $A \times S$. The set of streams is a final coalgebra of this functor. The set of streams is just one example of an interesting final coalgebra. For many more examples and some of the basic elements of the theory of 'universal' coalgebra, see [JR97, Rut98, Rut99, Rut00]. The notion of bisimulation is due to Park and Milner [Par81, Mil80], who designed it as a notion of equivalence for a theory of concurrent processes. (It existed already before, under the name of p-relation, in the world of Kripke frames and modal logic [vB76].) Final coalgebras have been used as models for many dynamical systems at least since [AM80, MA86]. It was not until a categorical generalisation of the notion of bisimulation was introduced, by Aczel and Mendler in [AM89], that coinduction (both as a definition and as a proof principle) was taken more seriously. By now, there is a host of literature on many aspects of both theory and applications of coalgebra. Rather than trying to give an overview of the relevant literature here, we refer the reader to the proceedings of the annual international workshops on Coalgebraic Methods in Computer Science (CMCS), which started in 1998 (see [CMC]).

Chapter 2

Basic stream calculus

In this chapter, we study the set $\mathbb{R}^\omega$ of streams of real numbers. We shall introduce a number of constants and shall define the operations of sum, product, and inverse of streams. These constants and operations make of $\mathbb{R}^\omega$ a calculus with many pleasant properties. For instance, it will be possible to compute solutions of linear systems of equations.

2.1 Sum and product

Let σ and τ be two streams of real numbers. For notational convenience, we shall sometimes denote the elements of σ and τ by σ_n and τ_n instead of $\sigma(n)$ and $\tau(n)$:

$$\sigma = (\sigma_0, \sigma_1, \sigma_2, \ldots), \quad \tau = (\tau_0, \tau_1, \tau_2, \ldots)$$

Definition 2.1 We define the *sum* $\sigma + \tau$ of σ and τ by

$$\sigma + \tau = (\sigma_0 + \tau_0,\, \sigma_1 + \tau_1,\, \sigma_2 + \tau_2,\, \ldots)$$

(Note that we use the same symbol $+$ for both the sum of two streams and the sum of two real numbers.) We define the *convolution product* $\sigma \times \tau$ of σ and τ by

$$\sigma \times \tau = (\sigma_0 \cdot \tau_0,\, (\sigma_0 \cdot \tau_1) + (\sigma_1 \cdot \tau_0),\, (\sigma_0 \cdot \tau_2) + (\sigma_1 \cdot \tau_1) + (\sigma_2 \cdot \tau_0),\, \ldots)$$

That is, for any $n \geq 0$,

$$\begin{aligned} (\sigma \times \tau)(n) &= (\sigma_0 \cdot \tau_n) + (\sigma_1 \cdot \tau_{n-1}) + \cdots + (\sigma_{n-1} \cdot \tau_1) + (\sigma_n \cdot \tau_0) \\ &= \sum_{k=0}^{n} \sigma_k \cdot \tau_{n-k} \end{aligned}$$

In general, we shall simply say 'product' rather than 'convolution product'. Note that we use the symbol $\times$ for the multiplication of streams and the symbol $\cdot$ for the multiplication of real numbers. $\square$

Similar to the notation for the multiplication of real numbers (and functions), we shall write

$$\sigma^0 \equiv 1, \quad \sigma^{n+1} \equiv \sigma \times \sigma^n$$

for any $\sigma \in \mathbb{R}^\omega$ and $n \geq 0$. Note the distinction between this notation and the notation $\sigma^{(n)}$ for the n-th derivative of σ that was introduced in Chapter 1.

The sum of σ and τ consists of the stream of the sum of their elements. In order to understand the product $\sigma \times \tau$, think of a stream σ as a process producing its respective elements σ_0, σ_1, σ_2, and so on, one by one in an infinite sequence of time steps. The product $\sigma \times \tau$ can then be viewed as a kind of delayed computation of the elementwise product, in the following sense:

$$\begin{array}{rcl} \sigma \times \tau & = & (\sigma_0 \cdot \tau_0,\, \sigma_0 \cdot \tau_1,\, \sigma_0 \cdot \tau_2,\, \sigma_0 \cdot \tau_3,\, \ldots) \\ & + & (\quad 0 \quad ,\, \sigma_1 \cdot \tau_0,\, \sigma_1 \cdot \tau_1,\, \sigma_1 \cdot \tau_2,\, \ldots) \\ & + & (\quad 0 \quad ,\quad 0 \quad ,\, \sigma_2 \cdot \tau_0,\, \sigma_2 \cdot \tau_1,\, \ldots) \\ & + & \cdots \end{array}$$

Example 2.2 Here are some examples of the sum and product of a few simple streams:

$$\begin{array}{rcl} (1,1,1,\ldots) + (1,1,1,\ldots) & = & (2,2,2,\ldots) \\ (1,1,1,\ldots) \times (1,1,1,\ldots) & = & (1,2,3,\ldots) \\ (1,1,1,\ldots) \times (1,2,3,\ldots) & = & (1,3,6,10,\ldots) \\ (0,1,0,0,0,\ldots) + (0,1,0,0,0,\ldots) & = & (0,2,0,0,0,\ldots) \\ (0,1,0,0,0,\ldots) \times (0,1,0,0,0,\ldots) & = & (0,0,1,0,0,0,\ldots) \end{array}$$

$\square$

Exercise 2.1 (a) Compute, for arbitrary $\sigma \in \mathbb{R}^\omega$: $(1,1,1,\ldots) \times \sigma$.

(b) For $n \geq 0$, compute the sum of $(0,1,0,0,0,\ldots)$ with itself, n times.

(c) For $n \geq 0$, compute the product of $(0,1,0,0,0,\ldots)$ with itself, n times. $\square$

It will be convenient to define the operations of sum and product also for the combination of a real number r and a stream σ. This will allow us, for instance, to write $3 \times \sigma$ for $\sigma + \sigma + \sigma$. In order to define this formally, it will be convenient to view real numbers as streams in the following manner.

Definition 2.3 We define for every $r \in \mathbb{R}$ a stream $[r] \in \mathbb{R}^\omega$ by

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

Note that this defines in fact a function

$$[\ ] : \mathbb{R} \to \mathbb{R}^\omega, \quad r \mapsto [r]$$

which embeds the set of real numbers into the set of streams. □

This definition allows us to add and multiply real numbers r with streams σ, yielding:

$$\begin{aligned} [r] + \sigma &= (r, 0, 0, 0, \ldots) + \sigma \\ &= (r + \sigma_0,\, \sigma_1,\, \sigma_2,\, \sigma_3\ \ldots) \\ [r] \times \sigma &= (r, 0, 0, 0, \ldots) \times \sigma \\ &= (r \cdot \sigma_0,\, r \cdot \sigma_1,\, r \cdot \sigma_2,\, \ldots) \end{aligned}$$

Notation 2.4 For notational convenience, we shall very quickly start to simply write $r + \sigma$ for $[r] + \sigma$, and similarly $r \times \sigma$ for $[r] \times \sigma$. The context will always make clear whether the notation r has to be interpreted as the real number r of as the stream $[r]$. For multiplication, this difference is moreover made explicit by the use of two different symbols: $r \times \sigma$ always denotes the multiplication of streams (and hence r should be read as the stream $[r]$) and $r \cdot s$ always denotes the multiplication of real numbers. We shall also use the following convention:

$$\begin{aligned} -\sigma &\equiv [-1] \times \sigma \\ &= (-\sigma_0, -\sigma_1, -\sigma_2, \ldots) \end{aligned}$$

□

Next we present a few basic properties of our operators.

Proposition 2.5 *For all $r, s \in \mathbb{R}$ and $\sigma, \tau, \rho \in \mathbb{R}^\omega$,*

$$\begin{aligned} [r] + [s] &= [r + s] \\ \sigma + 0 &= \sigma \end{aligned}$$

$$\begin{aligned}
\sigma + \tau &= \tau + \sigma \\
\sigma + (\tau + \rho) &= (\sigma + \tau) + \rho \\
[r] \times [s] &= [r \cdot s] \\
0 \times \sigma &= 0 \\
1 \times \sigma &= \sigma \\
\sigma \times \tau &= \tau \times \sigma \\
\sigma \times (\tau + \rho) &= (\sigma \times \tau) + (\sigma \times \rho) \\
\sigma \times (\tau \times \rho) &= (\sigma \times \tau) \times \rho
\end{aligned}$$

Exercise 2.2 Prove the equalities given in Proposition 2.5. □

2.2 Polynomial streams

Particularly simple are those streams that from a certain point onwards are constant zero:

$$\sigma = (r_0, r_1, r_2, \ldots, r_n, 0, 0, 0, \ldots)$$

for $n \geq 0$ and $r_0, \ldots, r_n \in \mathbb{R}$. Using the following constant, we shall see that there is a very convenient way of denoting such streams.

Definition 2.6 We define the constant X by

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

□

Proposition 2.7 *For all $r \in \mathbb{R}$, $\sigma \in \mathbb{R}^\omega$, and $n \geq 0$:*

$$\begin{aligned}
r \times X &= (0, r, 0, 0, 0, \ldots) \\
X \times \sigma &= (0, \sigma_0, \sigma_1, \sigma_2, \ldots) \\
X^n &= (\underbrace{0, \ldots, 0}_{n \text{ times}}, 1, 0, 0, 0, \ldots)
\end{aligned}$$

Exercise 2.3 Prove Proposition 2.7. □

Example 2.8

$$\begin{aligned}
X \times X &= (0, 0, 1, 0, 0, 0, \ldots) \\
X^5 &= (0, 0, 0, 0, 0, 1, 0, 0, 0, \ldots) \\
7X &= 7 \times (0, 1, 0, 0, 0, \ldots)
\end{aligned}$$

$$\begin{aligned} & = (0,7,0,0,0,\ldots) \\ -8X^3 & = -8 \times (0,0,0,1,0,0,0,\ldots) \\ & = (0,0,0,-8,0,0,0,\ldots) \\ 2+3X & = (2,0,0,0,\ldots) + (0,3,0,0,0,\ldots) \\ & = (2,3,0,0,0,\ldots) \\ 2+3X-8X^3 & = (2,3,0,-8,0,0,0,\ldots) \end{aligned}$$

□

Exercise 2.4 (a) Compute $1-2X+5X^4+X^5$ and $1-X+X^2-X^3+X^4$.

(b) Write $(1,1,1,1,1,0,0,0,\ldots)$ using sum, product, and X.

(c) Same question for $(1,0,-1,0,1,0,-1,0,0,0,0,\ldots)$.

(d) Can you do the same for $(1,1,1,\ldots)$, the infinite stream of ones?

□

Definition 2.9 For all $n \geq 0$ and all $r_0,\ldots,r_n \in \mathbb{R}$:

$$r_0 + r_1X + r_2X^2 + \cdots + r_nX^n \quad = \quad (r_0,r_1,r_2,\ldots,r_n,0,0,0,\ldots)$$

Such streams are called *polynomial streams*. □

(The equality in Definition 2.9 is an immediate consequence of the definitions of sum, product, and X.) Let us stress the fact that although a polynomial stream such as

$$2+3X-8X^3$$

looks like a (polynomial) *function* $f(x) = 2+3x-8x^3$, for which x is a variable, it really is a *stream*, built from constant streams (2, 3, 8, and X), and the operations of sum and product. At the same time, it is true that we can calculate with polynomial streams in precisely the same way as we are used to compute with (polynomial) functions, as is illustrated by the following examples.

Example 2.10 In the computations below, we are using the basic properties of sum and product listed in Proposition 2.5.

$$\begin{aligned} (2-X)+(1+3X) & = 2+1-X+3X \\ & = 3+(-1+3)\times X \\ & = 3+2X \end{aligned}$$

$$
\begin{array}{rcl}
 & (= & (3,2,0,0,0,\ldots)\,) \\
(2-X)\times(1+3X) & = & ((2-X)\times 1) \;+\; ((2-X)\times 3X) \\
 & = & 2-X+6X-3X^2 \\
 & = & 2+5X-3X^2 \\
 & (= & (2,5,-3,0,0,0,\ldots)\,)
\end{array}
$$

□

Exercise 2.5 Compute the following streams:

(i) $(1+X)\times(1-X)$

(ii) $(1+7X^2)\times(-X+X^3+3X^4)$

(iii) $(1+X)^2$, $(1+X)^3$, etc.

□

Exercise 2.6 (a) Can you find a formula for the multiplication of two arbitrary polynomial streams:

$$(r_0+r_1X+r_2X^2+\cdots+r_nX^n)\times(s_0+s_1X+s_2X^2+\cdots+s_mX^m)$$

for $n,m\geq 0$, $r_0,\ldots,r_n,s_0,\ldots,s_m\in\mathbb{R}$?

(b) Can you find $\sigma\in\mathbb{R}^\omega$ such that $\sigma\times(1-X)=1$ $(=(1,0,0,0,\ldots))$?

□

2.3 Derivatives of sum and product

We show how to compute the derivatives of the sum and the product of two streams, and present some examples of stream differential equations (introduced in Section 1.2) that now involve the use of sum and product.

Theorem 2.11 *For all $\sigma,\tau\in\mathbb{R}^\omega$,*

$$
\begin{array}{rcl}
(\sigma+\tau)(0) & = & \sigma_0+\tau_0 \\
(\sigma+\tau)' & = & \sigma'+\tau' \\
(\sigma\times\tau)(0) & = & \sigma_0\cdot\tau_0 \\
(\sigma\times\tau)' & = & (\sigma_0\times\tau')+(\sigma'\times\tau)
\end{array}
$$

Proof: The first three equalities are immediate from Definition 2.1. For the last equality, we have

$$\begin{aligned}
&(\sigma \times \tau)' \\
&\quad = \quad ((\sigma_0 \cdot \tau_1) + (\sigma_1 \cdot \tau_0),\, (\sigma_0 \cdot \tau_2) + (\sigma_1 \cdot \tau_1) + (\sigma_2 \cdot \tau_0),\, \ldots) \\
&\quad = \quad (\sigma_0 \cdot \tau_1,\, \sigma_0 \cdot \tau_2,\, \sigma_0 \cdot \tau_3, \ldots) \;+\; (\sigma_1 \cdot \tau_0,\, (\sigma_1 \cdot \tau_1) + (\sigma_2 \cdot \tau_0), \ldots) \\
&\quad = \quad \sigma_0 \times (\tau_1, \tau_2, \tau_3, \ldots) \;+\; (\,(\sigma_1, \sigma_2, \sigma_3, \ldots) \times (\tau_0, \tau_1, \tau_2, \ldots)\,) \\
&\quad = \quad (\sigma_0 \times \tau') + (\sigma' \times \tau)
\end{aligned}$$

□

Remark 2.12 Sum and product of two streams σ and τ satisfy, in other words, the following stream differential equations:

derivative	initial value
$(\sigma + \tau)' = \sigma' + \tau'$	$(\sigma + \tau)(0) = \sigma_0 + \tau_0$
$(\sigma \times \tau)' = (\sigma_0 \times \tau') + (\sigma' \times \tau)$	$(\sigma \times \tau)(0) = \sigma_0 \cdot \tau_0$

These equalities can be seen as properties of the operations of sum and product, which were defined in Definition 2.1. Alternatively, they could also be taken as the definition of the operations of sum and product. From this definition, one could actually prove the equalities of Definition 2.1 as properties (which is omitted here). In short, Theorem 2.11 and Definition 2.1 can be seen as two different, but equivalent definitions. □

The following equalities will be particularly helpful in some of the calculation that will follow later.

Corollary 2.13 *For all $r \in \mathbb{R}$ and $\sigma \in \mathbb{R}^\omega$,*

$$\begin{aligned}
[r](0) &= r \\
[r]' &= [0] \\
X(0) &= 0 \\
X' &= [1] \\
([r] + \sigma)(0) &= r + \sigma_0 \\
([r] + \sigma)' &= \sigma' \\
([r] \times \sigma)(0) &= r \cdot \sigma_0 \\
([r] \times \sigma)' &= [r] \times \sigma' \\
(X \times \sigma)(0) &= 0 \\
(X \times \sigma)' &= \sigma
\end{aligned}$$

Proof: In order to avoid any confusion, we have, temporarily, denoted the stream interpretation of a real number r explicitly again by $[r]$. All of the above identities can be proved straightforwardly. For instance,

$$\begin{aligned} X' &= (0,1,0,0,0,\ldots)' \\ &= (1,0,0,0,\ldots) \\ &= [1] \\ &= 1 \end{aligned}$$

to mention one example. □

Example 2.14

$$\begin{aligned} (X \times X)' &= X \\ (X^{n+1})' &= X^n \\ (2 + 3X - 7X^3)' &= 3 - 7X^2 \\ (1 - X + X^2 - X^3 + X^4)' &= -1 + X - X^2 + X^3 \\ (r_0 + r_1 X + r_2 X^2 + \cdots + r_{n+1} X^{n+1})' &= r_1 + r_2 X + \cdots + r_{n+1} X^n \end{aligned}$$

□

It is clear from the above that taking the stream derivative of the product of two streams follows rules that are different from what we are used to in analysis. For (differentiable real-valued) functions $f(x)$ and $g(x)$, one has $(f \times g)' = (f' \times g) + (f \times g')$. (Here $f \times g$ is defined, for all x, by $(f \times g)(x) = f(x) \cdot g(x)$.) In particular, if $f(x) = r_0 + r_1 x + r_2 x^2 + \cdots + r_{n+1} x^{n+1}$, then $f'(x) = r_1 + 2r_2 x + \cdots + (n+1) r_{n+1} x^n$. We see that the rules for the computation of stream derivatives are, in fact, simpler.

Exercise 2.7 Compute the derivatives of the following streams:

(i) $\sigma = 7$

(ii) $\sigma = -5 - X^2 - X^6 + X^{100}$

(iii) $\sigma = 1 + X$, $\sigma = (1 + X)^2$, $\sigma = (1 + X)^3$.

Can you find a formula for the derivative of $\sigma = (1 + X)^n$, for arbitrary $n \geq 0$? (Use the formula for $(\sigma \times \tau)'$ from Theorem 2.11.) □

Exercise 2.8 Some of the proofs of the properties listed in Proposition 2.5 are a bit awkward. Using Theorem 2.11, we can alternatively prove some of these properties more conveniently by coinduction.

(a) Warming up: Prove $\sigma + \tau = \tau + \sigma$, for all $\sigma, \tau \in \mathbb{R}^\omega$, by coinduction.

(b) More difficult: For σ, τ, $\rho \in \mathbb{R}^\omega$, compute and compare the derivatives of $\sigma \times (\tau \times \rho)$ and $(\sigma \times \tau) \times \rho$. Define a bisimulation relation which allows you to prove the associativity of $\times$ by coinduction.

□

Using the properties above, we next look at some examples of streams defined by stream differential equations.

Example 2.15 Let $\sigma \in \mathbb{R}^\omega$ be defined by the following stream differential equation:

derivative	initial value
$\sigma' = 2 \times \sigma$	$\sigma(0) = 1$

We compute the first few elements of σ:

$$\begin{aligned}
\sigma &= \sigma(0) : \sigma' \\
&= 1 : (2 \times \sigma) \\
&= 1 : (2 \times \sigma)(0) : (2 \times \sigma)' \\
&= 1 : 2 : (2 \times \sigma') \quad \text{[using Corollary 2.13]} \\
&= 1 : 2 : (2^2 \times \sigma) \\
&= 1 : 2 : 2^2 : (2^3 \times \sigma)
\end{aligned}$$

and so on. We see: $\sigma = (1, 2, 2^2, 2^3, 2^4, \ldots)$. □

Exercise 2.9 Show that $\sigma = (1, 2, 2^2, 2^3, 2^4, \ldots)$ satisfies the differential equation of Example 2.15. □

Example 2.16 Let $\sigma, \tau \in \mathbb{R}^\omega$ be defined by the following equations:

derivative	initial value
$\sigma' = \sigma + \tau$	$\sigma(0) = 0$
$\tau' = \tau$	$\tau(0) = 1$

Clearly, $\tau = (1, 1, 1, \ldots)$. For σ, we compute

$$\begin{aligned}
\sigma &= \sigma(0) : \sigma' \\
&= 0 : (\sigma + \tau) \\
&= 0 : (\sigma + \tau)(0) : (\sigma + \tau)' \\
&= 0 : (\sigma(0) + \tau(0)) : (\sigma' + \tau') \\
&= 0 : 1 : (\sigma + 2\tau) \\
&= 0 : 1 : (\sigma + 2\tau)(0) : (\sigma + 2\tau)' \\
&= 0 : 1 : 2 : (\sigma + 3\tau)
\end{aligned}$$

and so on. We see: $\sigma = (0, 1, 2, 3, \ldots)$. □

Exercise 2.10 Let $\tau = (1, 1, 1, \ldots)$. Prove by coinduction, for all $\sigma \in \mathbb{R}^\omega$:

$$map[+1](\sigma) = \sigma + \tau$$

(The function $map[+1]$ was introduced in Example 1.14 for streams of natural numbers. The same definition works for streams of real numbers.) □

Example 2.17 Let $\sigma \in \mathbb{R}^\omega$ be defined by

derivative	initial value
$\sigma' = X \times \sigma$	$\sigma(0) = 1$

Then:

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

Thus $\sigma = (1, 0, 1, 0, 1, 0, \ldots)$. □

Exercise 2.11 (a) Compute the stream σ defined by

derivative	initial value
$\sigma' = X^2 \times \sigma$	$\sigma(0) = 1$

(b) Compute the stream σ defined by

derivative	initial value
$\sigma'' = \sigma + \sigma'$	$\sigma(0) = 0$, $\sigma'(0) = 1$

Does this sequence look familiar? (Think rabbits.)

(c) Compute the stream σ defined by

derivative	initial value
$\sigma'' = -\sigma$	$\sigma(0) = 0$, $\sigma'(0) = 1$

Does this sequence look familiar? (Think geometry.)

□

2.4 The operation of inverse

In Section 2.5, we shall solve linear equations in one unknown τ, such as

$$\tau \quad = \quad 1 + (X \times \tau) \tag{2.1}$$

(where, recall, $1 = (1, 0, 0, 0, \ldots)$ by Notation 2.3).

Exercise 2.12 Here is a down-to earth (but not very practical) way of solving equation (2.1). Let $\tau = (\tau_0, \tau_1, \tau_2, \ldots)$ and substitute this expression on both sides of the equation. From this compute the value of τ_0, then that of τ_1, and so on. □

Ideally, we would like to solve (2.1) by reasoning as follows:

$$\begin{aligned} \tau &= 1 + (X \times \tau) \\ &\Rightarrow \quad \tau - (X \times \tau) = 1 \\ &\Rightarrow \quad (1 - X) \times \tau = 1 \\ &\Rightarrow \quad \tau = \frac{1}{1 - X} \end{aligned}$$

Recall, however, that we are not dealing with functions but with streams. Therefore it is by no means obvious what we mean by the 'inverse' of $(1-X)$:

$$\frac{1}{1 - X} = \frac{1}{(1, -1, 0, 0, 0, \ldots)} = ?$$

Using stream differential equations, there turns out to be a very natural and convenient way to define the inverse of any stream σ with $\sigma(0) \neq 0$. Here are a few examples first.

Example 2.18 The inverse $\tau = \frac{1}{1-X}$ of the stream $1 - X$ should be such that $(1 - X) \times \tau = 1$. Equivalently, the stream τ should satisfy

$$\tau = 1 + (X \times \tau)$$

(which is the equation (2.1) that we started out with at the beginning of this section). A first observation is that this equation uniquely determines what the initial value of τ should be:

$$\begin{aligned} \tau(0) &= (1 + (X \times \tau))(0) \\ &= 1 \quad \text{[using Corollary 2.13]} \end{aligned}$$

Taking the stream derivative on both sides of the equation gives

$$\begin{aligned} \tau' &= (1 + (X \times \tau))' \\ &= \tau \quad \text{[again using Corollary 2.13]} \end{aligned}$$

We therefore see that τ must satisfy the following stream differential equation:

derivative	initial value
$\tau' = \tau$	$\tau(0) = 1$

(One can also show that if τ satisfies this differential equation, then $\tau = 1 + (X \times \tau)$.) We have seen this type of differential equation before, and we know how to solve it. Namely, $\tau = (1, 1, 1, \ldots)$ is its unique solution. Now we can define

$$\frac{1}{1 - X} = (1, 1, 1, \ldots)$$

□

Exercise 2.13 Check that $(1, 1, 1, \ldots) \times (1 - X) = 1$, using the definition of the product. □

Example 2.19 The inverse τ of the stream $1 - X^2$ should satisfy $(1 - X^2) \times \tau = 1$, which is equivalent to

$$\tau = 1 + (X^2 \times \tau)$$

This gives $\tau(0) = (1 + (X^2 \times \tau))(0) = 1$. For the derivative we find

$$\begin{aligned} \tau' &= (1 + (X^2 \times \tau))' \\ &= X \times \tau \quad \text{[using Corollary 2.13]} \end{aligned}$$

Thus τ is determined by the following equation:

derivative	initial value
$\tau' = X \times \tau$	$\tau(0) = 1$

This equation we recognize from Example 2.17, where the solution $\tau = (1, 0, 1, 0, 1, 0, \ldots)$ was found. Thus

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

□

Exercise 2.14 Compute the inverse of the following streams:

(i) $1 + X$

(ii) $1 - rX$ (for any $r \in \mathbb{R}$)

(iii) $(1 - X)^2$

(iv) $(1, 1, 1, \ldots)$

(v) $1 + X^2$

□

Exercise 2.15 Prove, for all $r \in \mathbb{R}$,

$$\frac{r}{1-X} = (r, r, r, \ldots)$$

Use coinduction to prove, for all $a, b \in \mathbb{R}$,

$$zip(\frac{a}{1-X}, \frac{b}{1-X}) = \frac{a+bX}{1-X}$$

(The operation *zip* was introduced in Example 1.6.) □

A similar procedure works for any stream σ with $\sigma(0) \neq 0$. From the requirement that

$$\sigma \times \frac{1}{\sigma} \quad = \quad 1 \tag{2.2}$$

one can deduce a stream differential equation as follows. Taking the initial value at both sides gives

$$\sigma(0) \cdot \frac{1}{\sigma}(0) = 1$$

which implies $\frac{1}{\sigma}(0) = \frac{1}{\sigma(0)}$. Taking the derivative of both sides gives

$$(\sigma_0 \times (\frac{1}{\sigma})') \; + \; (\sigma' \times \frac{1}{\sigma}) = 0$$

which implies

$$(\frac{1}{\sigma})' = -\frac{1}{\sigma(0)} \times \sigma' \times \frac{1}{\sigma}$$

This leads to the following definition.

Definition 2.20 We define the inverse $\frac{1}{\sigma}$ of a stream σ with $\sigma(0) \neq 0$ as the unique stream satisfying the following stream differential equation:

derivative	initial value
$(\frac{1}{\sigma})' = -\frac{1}{\sigma(0)} \times \sigma' \times \frac{1}{\sigma}$	$(\frac{1}{\sigma})(0) = \frac{1}{\sigma(0)}$

□

We shall use the following notational convention: for all $\sigma, \tau \in \mathbb{R}^\omega$ with $\sigma(0) \neq 0$,

$$\frac{\tau}{\sigma} \equiv \frac{1}{\sigma} \times \tau = \tau \times \frac{1}{\sigma}$$

This product is called the *quotient* of τ and σ.

As with sum and product, we can calculate with the operation of inverse in the same way as we compute with functions.

Proposition 2.21 *For all $\sigma, \tau \in \mathbb{R}^\omega$ with $\sigma(0) \neq 0 \neq \tau(0)$,*

$$\begin{aligned} \sigma \times \frac{1}{\sigma} &= 1 \\ \frac{1}{\sigma} \times \frac{1}{\tau} &= \frac{1}{\sigma \times \tau} \\ \frac{1}{\frac{1}{\sigma}} &= \sigma \end{aligned}$$

Proof: The first equality follows by coinduction (Theorem 1.9) from the fact that the following relation

$$\{\langle \sigma \times \frac{1}{\sigma}, 1\rangle \mid \sigma \in \mathbb{R}^\omega\} \cup \{\langle 0, 0\rangle\}$$

is a bisimulation relation. The second equation follows from the first one, and the last equation can again be proved by coinduction. (See [Rut01] for details.) □

Example 2.22

$$\begin{aligned} \frac{1}{(1-X)^2} &= \frac{1}{(1-X) \times (1-X)} \\ &= \frac{1}{1-X} \times \frac{1}{1-X} \\ &= (1,1,1,\ldots) \times (1,1,1,\ldots) \\ &= (1,2,3,\ldots) \\ \frac{1}{1-X^2} &= \frac{1}{(1-X) \times (1+X)} \\ &= \frac{1}{1-X} \times \frac{1}{1+X} \\ &= (1,1,1,\ldots) \times (1,-1,1,-1,1,-1,\ldots) \\ &= (1,0,1,0,1,0,\ldots) \end{aligned}$$

□

We conclude this section with the following definition.

Definition 2.23 The product of a polynomial stream and the inverse of a polynomial stream is called a *rational stream*. Equivalently, a stream σ is rational if there exist $n, m \geq 0$ and coefficients $r_0, \ldots, r_n, s_0, \ldots, s_m \in \mathbb{R}$ with $s_0 \neq 0$, such that

$$\sigma = \frac{r_0 + r_1 X + r_2 X^2 + \cdots + r_n X^n}{s_0 + s_1 X + s_2 X^2 + \cdots + s_m X^m}$$

□

Exercise 2.16 Prove that the sum, the product, and the inverse of rational streams are again rational. □

2.5 Solving linear equations

Next we show how to solve systems of linear equations. The solution of such systems can be computed in an algebraic manner, and will be expressed in terms of the constants and the operations of sum, product, and inverse. We shall only treat a few examples, which will be all that is needed later. (On the basis of these examples, however, it would not be very difficult to formulate and prove a more general result.)

Example 2.24 Consider the following system of equations:

$$\begin{aligned} \sigma &= 1 + (X \times \tau) \\ \tau &= X \times \sigma \end{aligned}$$

In order to find σ and τ, we compute as follows:

$$\begin{aligned} \sigma &= 1 + (X \times \tau) \\ &= 1 + (X \times X \times \sigma) \\ &= 1 + (X^2 \times \sigma) \end{aligned}$$

This implies

$$\begin{aligned} \sigma - (X^2 \times \sigma) &= 1 \\ (1 - X^2) \times \sigma &= 1 \end{aligned}$$

Thus

$$\begin{aligned} \sigma &= \frac{1}{1 - X^2} \\ \tau &= \frac{X}{1 - X^2} \end{aligned}$$

□

Example 2.25 Consider the following system of equations:

$$\begin{aligned} \sigma &= 1 + (X \times \tau) \\ \tau &= 2 + (X \times \sigma) \end{aligned}$$

In order to find σ and τ, we compute as before:

$$\begin{aligned} \sigma &= 1 + (X \times \tau) \\ &= 1 + (X \times (2 + (X \times \sigma))) \\ &= 1 + 2X + (X^2 \times \sigma) \end{aligned}$$

This implies

$$\sigma - (X^2 \times \sigma) = 1 + 2X$$
$$(1 - X^2) \times \sigma = 1 + 2X$$

Thus

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

□

Exercise 2.17 With σ and τ as is Example 2.25, what is the value of $\sigma(n)$ and $\tau(n)$, for $n \geq 0$? □

Exercise 2.18 Compute the solution of the following systems of equations:

(a)

$$\begin{aligned} \sigma &= 1 + (2X \times \sigma) - (X \times \tau) \\ \tau &= X \times \sigma \end{aligned}$$

(b)

$$\begin{aligned} \sigma &= 1 + (X \times \sigma) + (X \times \tau) \\ \tau &= X \times \sigma \end{aligned}$$

□

In conclusion of this section, we make the following general observation, the proof of which is omitted. The solutions of finite systems of linear equations such as the ones above, are always rational streams (cf. Definition 2.23). Conversely, any rational stream can be obtained as the solution of such a linear system of equations.

Chapter 3

Stream circuits

Certain functions from $\mathbb{R}^\omega$ to $\mathbb{R}^\omega$ can be represented by means of graphical networks that are built from a small number of basic ingredients. Such networks can be viewed as implementations of stream functions. We call them stream circuits; in the literature, they are also referred to as (signal) flow graphs. Using the basic stream calculus from Chapter 2, we shall give a formal but simple answer to the question precisely which stream functions can be implemented by such stream circuits.

3.1 Basic circuits

The circuits that we are about to describe, will generally have a number of *input ends* and a number of *output ends*. Here is an example of a simple circuit, consisting of one input and one output end:

The input end is denoted by the arrow shaft $\longmapsto\!\!\!\!-\!\!\!\!-$ and the output end is denoted by the arrow head $-\!\!\!\!-\!\!\!\!\rightarrow$. For streams $\sigma, \tau \in \mathbb{R}^\omega$, we shall write

$$\sigma \longmapsto \tau$$

and say that the circuit *inputs* the stream σ and *outputs* the stream τ. Writing the elements of these streams explicitly, this notation is equivalent to

$$(\sigma_0, \sigma_1, \sigma_2, \ldots) \longmapsto (\tau_0, \tau_1, \tau_2, \ldots)$$

which better expresses the intended operational behaviour of the circuit: It consists of an infinite sequence of actions, at time moments $0, 1, 2, \ldots$. At

each moment $n \geq 0$, the circuit simultaneously inputs the value $\sigma_n \in \mathbb{R}$ at its input end and outputs the value $\tau_n \in \mathbb{R}$ at its output end. In general, this value τ_n depends both on the value σ_n and on the values σ_i that have been taken as inputs at earlier time moments $i < n$. Note that this implies that circuits have memory.

Next we present the four basic types of circuits, out of which all other circuits in this chapter will be constructed.

(a) For every $a \in \mathbb{R}$, we define a circuit with one input and one output end, called an *a-multiplier*, for all $\sigma, \tau \in \mathbb{R}^\omega$, by

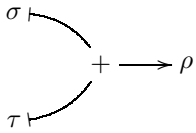
$$\begin{aligned} \sigma \vdash\!\!\xrightarrow{a} \tau \quad &\Longleftrightarrow \quad \tau_n = a \cdot \sigma_n, \ \text{ all } n \geq 0 \\ &\Longleftrightarrow \quad \tau = a \times \sigma \end{aligned}$$

This circuit takes, at any moment $n \geq 0$, a value σ_n at its input end, multiplies it with the constant a, and outputs the result $\tau_n = a \cdot \sigma_n$ at its output end. It defines, in other words, a function that assigns to an input stream σ the output stream $\tau = a \times \sigma$.

Occasionally, it will be more convenient to write the multiplying factor a as a super- or subscript of the arrow:

$$\vdash\!\!\xrightarrow{a} \ \equiv \ \vdash\!\!\overset{a}{\longrightarrow} \ \equiv \ \vdash\!\!\underset{a}{\longrightarrow}$$

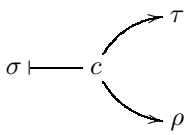
(b) The *adder* circuit has two input and one output ends, and is defined, for all $\sigma, \tau, \rho \in \mathbb{R}^\omega$ by



$$\begin{aligned} &\Longleftrightarrow \quad \rho_n = \sigma_n + \tau_n, \ \text{ all } n \geq 0 \\ &\Longleftrightarrow \quad \rho = \sigma + \tau \end{aligned}$$

At moment $n \geq 0$, the adder simultaneously inputs the values σ_n and τ_n at its input ends, and outputs their sum $\rho_n = \sigma_n + \tau_n$ at its output end.

(c) The *copier* circuit has one input and two output ends and is defined, for all $\sigma, \tau, \rho \in \mathbb{R}^\omega$, by



$$\begin{aligned} &\Longleftrightarrow \quad \tau_n = \sigma_n = \rho_n, \ \text{ all } n \geq 0 \\ &\Longleftrightarrow \quad \tau = \sigma = \rho \end{aligned}$$

At any moment $n \geq 0$, the copier inputs the value σ_n at its input end, and outputs two identical copies τ_n and ρ_n at its output ends.

(d) A *register* circuit has one input and one output end and is defined, for all $\sigma, \tau \in \mathbb{R}^\omega$, by

$$\begin{aligned} \sigma \vdash\!\!\!\!-R\!\!\!\!\rightarrow \tau \quad &\iff \quad \tau_0 = 0 \text{ and } \tau_n = \sigma_{n-1}, \text{ all } n \geq 1 \\ &\iff \quad \tau = (0, \sigma_0, \sigma_1, \sigma_2, \ldots) \end{aligned}$$

The register circuit can be viewed as consisting of a one-place memory cell that initially contains the value 0. The register starts its activity, at time moment 0, by outputting its value $\tau_0 = 0$ at its output end, while it simultaneously inputs the value σ_0 at its input end, which is stored in the memory cell. At any future time moment $n \geq 1$, the value $\tau_n = \sigma_{n-1}$ is output and the value σ_n is input and stored. (For obvious resaons, the register circuit is sometimes also called a *unit delay*.) Recalling the constant stream $X = (0, 1, 0, 0, 0, \ldots)$ from Definition 2.6, it is an immediate consequence of Proposition 2.7 that we have, for all $\sigma, \tau \in \mathbb{R}^\omega$,

$$\sigma \vdash\!\!\!\!-R\!\!\!\!\rightarrow \tau \quad \iff \quad \tau = X \times \sigma$$

3.2 Circuit composition

We can construct a larger circuit out of two smaller ones by connecting output ends of the first to input ends of the second. Rather than giving a fully general and formal definition, which is not very difficult but a bit tedious, we prefer to explain circuit composition by means of a number of examples. These will be sufficiently representative to teach the reader how to construct his or her own circuits.

Example 3.1 For the composition of a 2-multiplier and a 3-multiplier, we shall write

$$\vdash\!\!\!\!-2\!\!\!\!\rightarrow \circ \vdash\!\!\!\!-3\!\!\!\!\rightarrow$$

We call the connection point $\circ$ an (internal) *node* of the composed circuit. A computation step of this circuit, at any moment in time, consists of the simultaneous occurrence of the following actions: a value is input at the input end of the 2-register; it is multiplied by 2 and output at the output end of the 2-register; the result is input at the input end of the 3-register, is multiplied by 3 and is output at the output end of the 3-multiplier. More formally, and fortunately also more succinctly, we define the behaviour of the composed circuit, for all $\sigma, \tau \in \mathbb{R}^\omega$, by

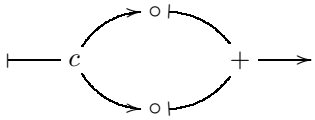
$$\begin{aligned} &\sigma \vdash\!\!\!\!-2\!\!\!\!\rightarrow \circ \vdash\!\!\!\!-3\!\!\!\!\rightarrow \tau \\ &\iff \quad \sigma \vdash\!\!\!\!-2\!\!\!\!\rightarrow \exists \rho \vdash\!\!\!\!-3\!\!\!\!\rightarrow \tau \\ &\iff \quad \exists \rho \in \mathbb{R}^\omega : \quad \sigma \vdash\!\!\!\!-2\!\!\!\!\rightarrow \rho \quad \text{and} \quad \rho \vdash\!\!\!\!-3\!\!\!\!\rightarrow \tau \end{aligned}$$

We shall consider all three of the above notations as equivalent. Combining the definitions of a 2- and 3-multiplier, we can in the above example easily compute how the output stream τ depends on the input stream σ:

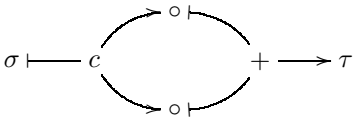
$$\begin{aligned}
&\sigma \vdash\!\!\xrightarrow{2}\ \circ \vdash\!\!\xrightarrow{3}\ \tau \\
\iff\ & \exists \rho \in \mathbb{R}^\omega : \quad \sigma \vdash\!\!\xrightarrow{2}\ \rho \quad \text{and} \quad \rho \vdash\!\!\xrightarrow{3}\ \tau \\
\iff\ & \exists \rho \in \mathbb{R}^\omega : \quad \rho = 2 \times \sigma \ \text{ and } \ \tau = 3 \times \rho \\
\iff\ & \tau = 6 \times \sigma
\end{aligned}$$

Note that the stream ρ is uniquely determined by the stream σ. The motivation for our notation "$\exists\rho$" is not so much to suggest that there might be more possible candidate streams for ρ, but rather to emphasise the fact that in order to express the output stream τ in terms of σ, we have to compute the value of the stream ρ in the middle. □

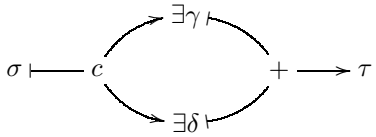
Example 3.2 We can compose circuits, more generally, with several output ends with circuits having a corresponding number of input ends, as in the following example:



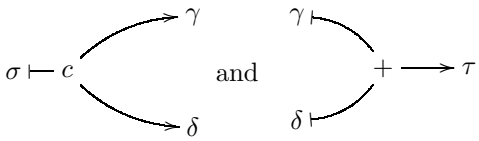
In this example, the behaviour of the resulting circuit is defined, for all $\sigma, \tau \in \mathbb{R}^\omega$, by



$\iff$



$\iff$

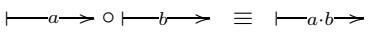
$\exists \gamma, \delta \in \mathbb{R}^\omega$: 

and

$$\begin{aligned}
\iff\ & \exists \gamma, \delta \in \mathbb{R}^\omega : \quad \sigma = \gamma = \delta \ \text{ and } \ \tau = \gamma + \delta \\
\iff\ & \tau = 2 \times \sigma
\end{aligned}$$

□

Exercise 3.1 (a) Prove that the following two circuits are equivalent, for all $a, b \in \mathbb{R}$:

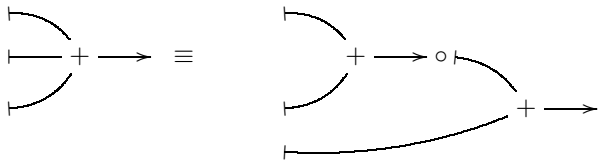


(b) Construct a number of different circuits that are all equivalent to a 1-multiplier. How many such circuits do there exist?

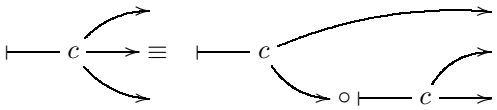
□

It will be convenient to have adders with more than two inputs and, similarly, copiers with more than two outputs.

Definition 3.3 We define a ternary adder as the composition of two binary adders as follows:

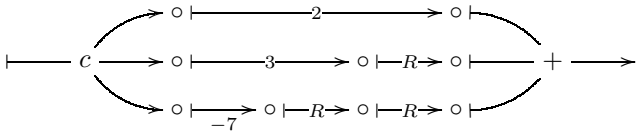


For input streams $\sigma, \tau, \rho \in \mathbb{R}^\omega$, it produces the output stream $\sigma + \tau + \rho$. We define a ternary copier by the following composition:



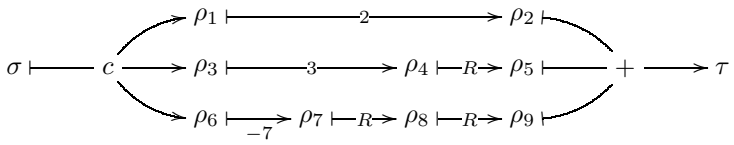
It takes one input stream and produces three identical copies as output streams. Adders and copiers with four or more inputs and outputs can be constructed in a similar fashion. □

Example 3.4 The following circuit combines (various instances of) all four basic circuit types:

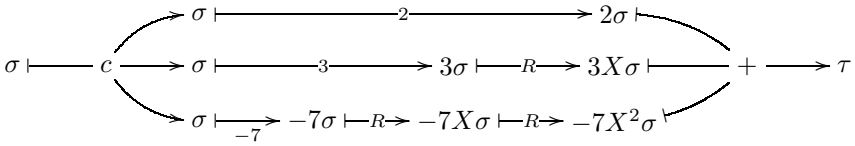


In order to express the output stream τ for a given input stream σ, we have to compute one intermediate stream for each of the (nine) internal nodes ∘

in the circuit above. In other words, we have to compute $\rho_1, \ldots, \rho_9 \in \mathbb{R}^\omega$ such that



Using the definitions of the basic circuits, and computing from left to right, we find:



(To save space, we have omitted the symbol $\times$ for multiplication.) We can now express the output stream τ in terms of the input stream σ as follows:

$$\begin{aligned} \tau &= (2 \times \sigma) + (3X \times \sigma) + (-7X^2 \times \sigma) \\ &= (2 + 3X - 7X^2) \times \sigma \end{aligned}$$

The circuit above computes, in other words, the following function on streams:

$$f : \mathbb{R}^\omega \to \mathbb{R}^\omega, \quad f(\sigma) = (2 + 3X - 7X^2) \times \sigma$$

If we supply the circuit with the input stream $\sigma = 1$ $(= (1, 0, 0, 0, \ldots))$ then the output stream is

$$\begin{aligned} \tau &= f(1) \\ &= 2 + 3X - 7X^2 \end{aligned}$$

□

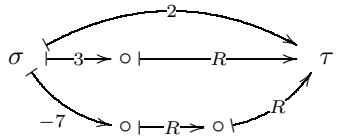
Definition 3.5 Let $f : \mathbb{R}^\omega \to \mathbb{R}^\omega$ be a stream function.

(a) If a circuit with one input end and one output end transforms every input stream $\sigma \in \mathbb{R}^\omega$ to an output stream $\tau = f(\sigma)$, then we say that the circuit *implements* (or: is an implementation of) the function f.

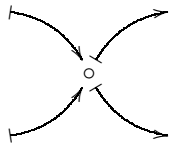
(b) We call the output stream $\tau = f(1)$, obtained on input $\sigma = 1$, the stream *generated* by the circuit.

□

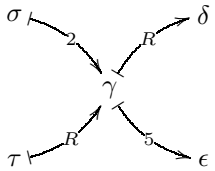
Convention 3.6 In order to reduce the size of the diagrams with which we depict stream circuits, it will often be convenient to leave the operations of copying and addition implicit. In this manner, we can, for instance, draw the circuit of Example 3.4 above as follows:



The (respective elements of the) stream σ gets copied along each of the three outgoing arrows. Similarly, the stream τ will be equal to the sum of the output streams of the three incoming arrows. This convention saves a lot of writing. Moreover, if we want to express τ in terms of σ, we now have only three internal streams to compute. If a node has both incoming and outgoing arrows, such as



then first the values of the output streams of the incoming arrows have to be added; then the resulting sum is copied and given as an input stream to each of the outgoing arrows. Consider for instance the circuit below. It has input streams σ and τ, an intermediate stream γ, and output streams δ and ϵ in $\mathbb{R}^\omega$:

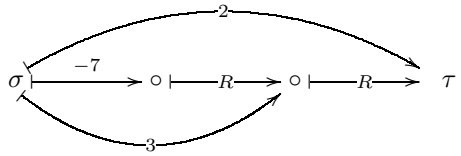


satisfying

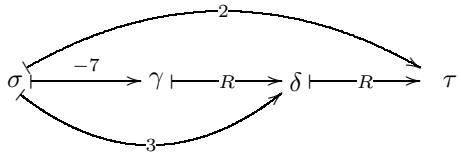
$$\begin{aligned}
\gamma &= 2\sigma + (X \times \tau) \\
\delta &= X \times \gamma \\
&= (2X \times \sigma) + (X^2 \times \tau) \\
\epsilon &= 5\gamma \\
&= 10\sigma + (5X \times \tau)
\end{aligned}$$

□

Example 3.7 Consider the following circuit:



(Note that this circuit contains one copier and two adders, which are invisible because we are using Convention 3.6.) In order to express τ in terms of σ, we have to compute two intermediate streams γ and δ such that:

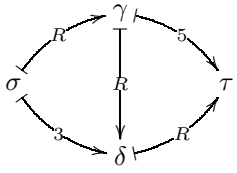


It follows that

$$\begin{aligned}
\gamma &= -7 \times \sigma \\
\delta &= (3 \times \sigma) + (X \times \gamma) \\
&= (3 \times \sigma) + (-7X \times \sigma) \\
\tau &= (2 \times \sigma) + (X \times \delta) \\
&= (2 \times \sigma) + X \times ((3 \times \sigma) + (-7X \times \sigma)) \\
&= (2 \times \sigma) + (3X \times \sigma) + (-7X^2 \times \sigma) \\
&= (2 + 3X - 7X^2) \times \sigma
\end{aligned}$$

We see that this circuit has the same stream function as the circuit of Example 3.4. An important difference is that the latter circuit contained three register circuits, whereas the present circuit only uses two. Thus it uses less memory. □

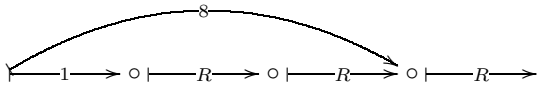
Example 3.8 We compute the stream function implemented by the following circuit, with input stream σ, output stream τ, and intermediate streams γ and δ:



We have:

$$\begin{aligned}\gamma &= X \times \sigma \\ \delta &= (3 \times \sigma) + (X^2 \times \sigma) \\ \tau &= (5 \times \gamma) + (X \times \delta) \\ &= (8X + X^3) \times \sigma\end{aligned}$$

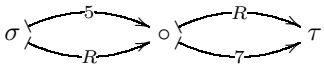
Thus the stream function implemented by this circuit is $f : \mathbb{R}^\omega \to \mathbb{R}^\omega$ with $f(\sigma) = (8X + X^3) \times \sigma$, for all $\sigma \in \mathbb{R}^\omega$. An equivalent circuit, implementing the same stream stream function, is given by:



□

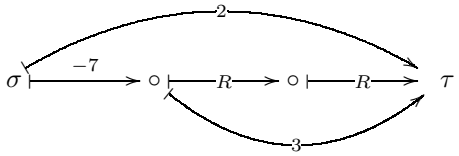
Exercise 3.2 Show that the two circuits of Example 3.8 both implement the same stream function. □

Exercise 3.3 (a) Draw a picture of the following circuit showing explicitly the adders and copiers that are used:

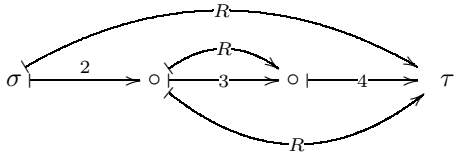


Then compute the stream function $f : \mathbb{R}^\omega \to \mathbb{R}^\omega$ that is implemented by this circuit.

(b) Compute the stream function $f : \mathbb{R}^\omega \to \mathbb{R}^\omega$ that is implemented by the following circuit:



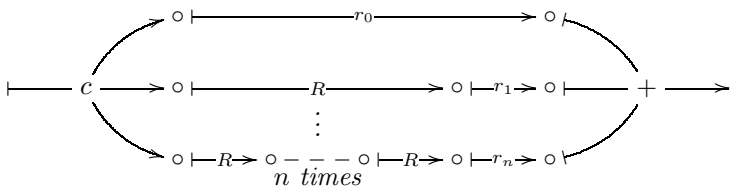
(c) Compute the stream function that is implemented by the following circuit:



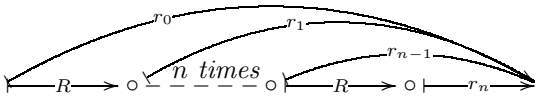
□

The following proposition characterizes which stream functions can be implemented by the type of circuits that we have been considering sofar.

Proposition 3.9 *For all $n \geq 0$ and $r_0, \ldots, r_n \in \mathbb{R}$, each of the following two circuits:*



and



implements the stream function $f : \mathbb{R}^\omega \to \mathbb{R}^\omega$ given, for all $\sigma \in \mathbb{R}^\omega$, by

$$f(\sigma) = \rho \times \sigma$$

where the stream ρ (generated by these circuits) is the polynomial

$$\rho = r_0 + r_1 X + r_2 X^2 + \cdots + r_{n-1} X^{n-1} + r_n X^n$$

Exercise 3.4 Prove Proposition 3.9. □

Exercise 3.5 (a) Give a circuit that generates the polynomial stream $1 - 3X + 5X^2$. Same question for $1 + X$.

(b) What is the stream function computed by the composition of these two circuits?

□

Exercise 3.6 (a) Consider the function $f : \mathbb{R}^\omega \to \mathbb{R}^\omega$ that assigns to every $\sigma \in \mathbb{R}^\omega$ the stream $f(\sigma) = \tau$ defined by $\tau_0 = \sigma_0$ and, for all $n \geq 1$, by

$$\tau_n = \sigma_{n-1} + \sigma_n$$

Construct a circuit that implements the function f.

(b) Can you do the same in case

$$\tau_n = \sigma_0 + \sigma_1 + \cdots + \sigma_n$$

for all $n \geq 0$?

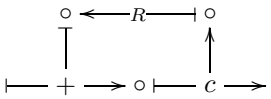
□

Exercise 3.7 Take any of the circuits that we have seen sofar. (In fact, take any circuit.) Reverse all the arrows, replace all adders by copiers, and replace all copiers by adders. Then show that both the original and the reversed circuit compute the same stream function. □

3.3 Circuits with feedback loops

The use of feedback loops in stream circuits increases their expressive power substantially. We shall again start with a few examples and then give a simple and precise characterization of all stream functions that can be implemented by circuits with feedback loops.

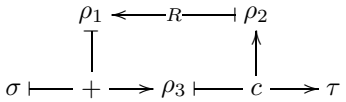
Example 3.10 Here is the simplest example of a circuit with feedback:



In spite of its simplicity, this circuit is already quite interesting. Before we give a formal computation of the stream function that this circuit implements, we give an informal description of its behaviour first. Assuming that we have an input stream $\sigma = (\sigma_0, \sigma_1, \sigma_2, \ldots)$, we compute the respective elements of the output stream $\tau = (\tau_0, \tau_1, \tau_2, \ldots)$. Recall that a register can be viewed as a one-place memory cell with initial value 0. At moment 0, our circuit begins its activity by inputting the first value σ_0 at its input end. The present value of the register, 0, is added to this and the result $\tau_0 = \sigma_0 + 0 = \sigma_0$ is the first value to be output. At the same time, this value σ_0 is copied and stored as the new value of the register. The next step consists of inputting the value σ_1, adding the present value of the register, σ_0, to it, and outputting the resulting value $\tau_1 = \sigma_0 + \sigma_1$. At the same time, this value $\sigma_0 + \sigma_1$ is copied and stored as the new value of the register. The next step will input σ_2 and output the value $\tau_2 = \sigma_0 + \sigma_1 + \sigma_2$. And so on. We find:

$$\tau = (\sigma_0, \sigma_0 + \sigma_1, \sigma_0 + \sigma_1 + \sigma_2, \ldots)$$

Next we show how the same answer can be obtained, more formally and more systematically, by applying a bit of basic stream calculus. As before, we try to express the output stream τ in terms of the input stream σ by computing the values of intermediate streams $\rho_1, \rho_2, \rho_3 \in \mathbb{R}^\omega$, corresponding to the three internal nodes of the circuit, such that



Note that the values of ρ_1, ρ_2, ρ_3 are mutually dependent because of the presence of the feedback loop: ρ_3 depends on ρ_1 which depends on ρ_2 which depends on ρ_3. Fortunately, we have used coinduction in Chapter 2 to develop a stream calculus that is precisely fit for this type of circularity. Unfolding the definitions of the basic circuits of which the above circuit is composed (one adder, one register, and one copier), we find the following system of equations:

$$\begin{aligned} \rho_1 &= X \times \rho_2 \\ \rho_3 &= \sigma + \rho_1 \\ \rho_2 &= \rho_3 \\ \tau &= \rho_3 \end{aligned}$$

We have learned in Section 2.5 how to solve such a system of equations:

$$\begin{aligned} \tau &= \rho_3 \\ &= \sigma + \rho_1 \\ &= \sigma + (X \times \rho_2) \\ &= \sigma + (X \times \tau) \end{aligned}$$

As a consequence, $\tau - (X \times \tau) = \sigma$, which is equivalent to $\tau = \frac{1}{1-X} \times \sigma$. Thus the stream function $f : \mathbb{R}^\omega \to \mathbb{R}^\omega$ that is implemented by the feedback circuit is given, for all $\sigma \in \mathbb{R}^\omega$, by

$$f(\sigma) = \frac{1}{1-X} \times \sigma$$

Somewhat surprisingly, maybe, we see that this function consists again of the convolution product of the argument σ and a constant stream $\frac{1}{1-X}$. The main difference with the examples in the previous sections is that in the present example this constant stream is no longer a polynomial stream, but the inverse of a polynomial stream. □

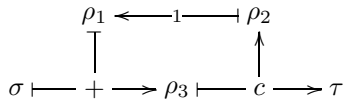
Exercise 3.8 We still have to check that the first informal and the second formal computation of the function implemented by the feedback circuit in Example 3.10 coincide. Prove, for all $\sigma \in \mathbb{R}^\omega$, that

$$\frac{1}{1-X} \times \sigma = (\sigma_0, \sigma_0 + \sigma_1, \sigma_0 + \sigma_1 + \sigma_2, \ldots)$$

□

Not every feedback loop gives rise to a circuit with a well-defined behaviour. Consider for instance the following circuit, with input stream σ,

output stream τ, and internal streams ρ_1, ρ_2, ρ_3:



In this circuit, we have replaced the register feedback loop of Example 3.10 by a 1-multiplier. If we try to compute the stream function of this circuit as before, we find the following system of equations:

$$\begin{aligned} \rho_1 &= 1 \times \rho_2 \\ \rho_3 &= \sigma + \rho_1 \\ \rho_2 &= \rho_3 \\ \tau &= \rho_3 \end{aligned}$$

This leads to $\rho_3 = \sigma + \rho_3$, which implies $\sigma = 0$. But σ is supposed to be an arbitrary input stream, so this does not make sense.

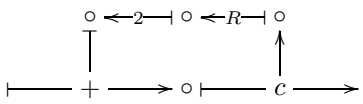
Problems like this can be avoided by assuming that circuits have the following property.

Assumption 3.11 From now on, we shall consider only circuits in which every feedback loop passes through at least one register circuit. □

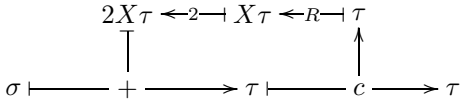
This formulation is admittedly informal, but as we shall see, it is precise enough for our present purposes. (Moreover, it can be made completely formal without too much effort.) Under Assumption 3.11, all circuits will have a well-defined behaviour. (Again, a formal proof is not difficult but is omitted here.)

Exercise 3.9 What happens if we replace the register circuit of Example 3.10 by a 2-multiplier (instead of the 1-multiplier in the example above)? □

Example 3.12 Consider the following circuit:



Taking an input stream σ and output stream τ and computing the values of the internal streams, we find



It follows that $\tau = \sigma + (2X \times \tau)$, thus

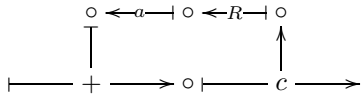
$$\tau \quad = \quad \frac{1}{1-2X} \times \sigma$$

Taking $\sigma = 1$, we see that the stream generated by this circuit is

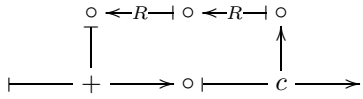
$$\frac{1}{1-2X} = (1, 2, 2^2, 2^3, \ldots)$$

□

Exercise 3.10 (a) For every $a \in \mathbb{R}$, compute the stream function implemented by the following circuit:

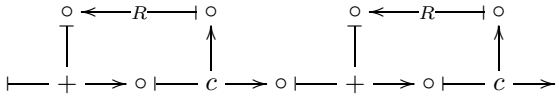


(b) The same question for

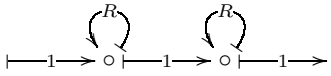


□

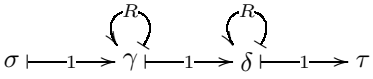
Example 3.13 Consider the following circuit:



Using (a variation of) Convention 3.6, we may omit all adders and copiers and use the following equivalent diagram:



One has to be a bit careful, though. Taking an input stream σ, an output stream τ, and internal streams γ and δ, we get



The tricky part is now to realise that, for instance, γ equals the sum of all the incoming arrows, which are two: one from σ and one from γ itself. Note that γ is copied along both outgoing arrows, including the one to itself. As a consequence, we get $\gamma = \sigma + (X \times \gamma)$. For δ, we have something similar. This leads to the following system of equations:

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

Using $\gamma = \frac{1}{1-X} \times \sigma$ and $\delta = \frac{1}{1-X} \times \gamma$, it then follows that

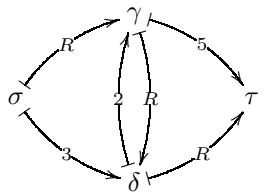
$$\begin{aligned} \tau &= \delta \\ &= \frac{1}{1-X} \times \gamma \\ &= \frac{1}{1-X} \times \frac{1}{1-X} \times \sigma \\ &= \frac{1}{(1-X)^2} \times \sigma \end{aligned}$$

The stream generated by this circuit, obtained by taking $\sigma = 1$, is

$$\frac{1}{(1-X)^2} = (1, 2, 3, \ldots)$$

□

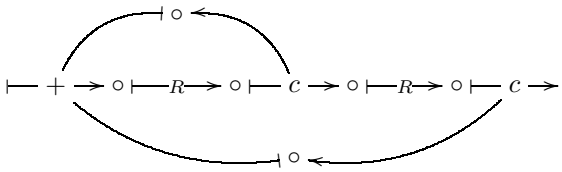
Exercise 3.11 Compute the stream function implemented by the following circuit:



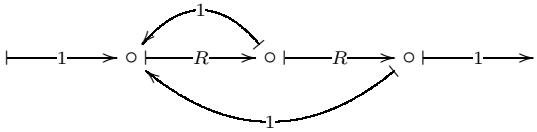
□

3.4 Raising rabbits

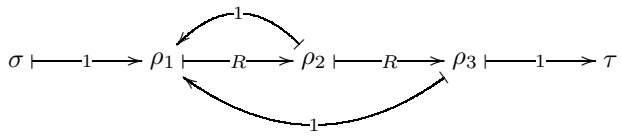
Before we formulate the main result of the present chapter, let us look at one more example. Consider the following circuit:



Leaving the adder and copiers implicit, this circuit is equivalent to



This little circuit describes the reproduction of rabbits in the following manner. Suppose we take $\sigma = 1$ as the input stream. The value 1 represents the arrival, at moment 0, of one pair of young rabbits, consisting of one male and one female. After one time unit – let us say a month – the rabbits are fully grown and reproduce, becoming the proud parents of one new pair of baby rabbits, which we assume to consist again of one male and one female. This reproduction is modelled in the circuit by the (implicit) copier after the first register. After yet another month, the original pair of rabbits reproduces again, giving birth to a second pair of young rabbits. The second reproduction is modelled by the copier after the second register. After that, the parent rabbits are old and they retire (and they do not reproduce anymore). There are two feedback loops in the circuit, because each of the two newly born rabbit pairs immediately starts behaving as their parents did before: after one month they reproduce and after yet another month they reproduce again. As a consequence, the value τ_n, for every $n \geq 0$, of the output stream $\tau = (\tau_0, \tau_1, \tau_2, \ldots)$ is equal to the total number of rabbit pairs that retire at moment n. Assuming now the input stream σ to be arbitrary, we can compute the stream τ as before, taking streams ρ_1, ρ_2, ρ_3 as intermediate streams:



This yields the following equations:

$$\begin{aligned} \rho_1 &= \sigma + \rho_2 + \rho_3 \\ \rho_2 &= X \times \rho_1 \\ \rho_3 &= X \times \rho_2 \\ \tau &= \rho_3 \end{aligned}$$

In order to compute τ, we first compute ρ_1:

$$\begin{aligned} \rho_1 &= \sigma + \rho_2 + \rho_3 \\ &= \sigma + (X \times \rho_1) + (X^2 \times \rho_1) \end{aligned}$$

This implies $\rho_1 = \frac{1}{1-X-X^2} \times \sigma$, whence

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

Returning to our rabbits above and taking $\sigma = 1$, we find

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

for the output stream generated by our circuit. Computing the first few values (see Exercise 3.12 below) gives

$$\tau = (0, 0, 1, 2, 3, 5, 8, 13, \ldots)$$

These numbers are very famous in mathematics and are called the *Fibonacci* numbers.

Exercise 3.12 In order to compute the first few elements of the stream $\tau = \frac{X^2}{1-X-X^2}$, show that τ satisfies the following differential equation:

derivative	initial value
$\tau'' = 1 + \tau + \tau'$	$\tau(0) = 0$, $\tau'(0) = 0$

Then prove that this implies $\tau_0 = \tau_1 = 0$, $\tau_2 = 1$, and for all $n \geq 1$: $\tau_{n+2} = \tau_{n+1} + \tau_n$. □

3.5 Circuits and rational streams

The following theorem characterizes which stream functions can be implemented by stream circuits.

Theorem 3.14 *(a) The stream function $f : \mathbb{R}^\omega \to \mathbb{R}^\omega$ implemented by a (finite) stream circuit, possibly containing feedback loops (that always pass through at least one register), is always of the form, for all $\sigma \in \mathbb{R}^\omega$:*

$$f(\sigma) = \rho \times \sigma$$

for some rational stream

$$\rho = \frac{r_0 + r_1 X + r_2 X^2 + \cdots + r_n X^n}{s_0 + s_1 X + s_2 X^2 + \cdots + s_m X^m}$$

with $n, m \geq 0$, $r_0, \ldots, r_n, s_0, \ldots, s_m \in \mathbb{R}$, and $s_0 \neq 0$.

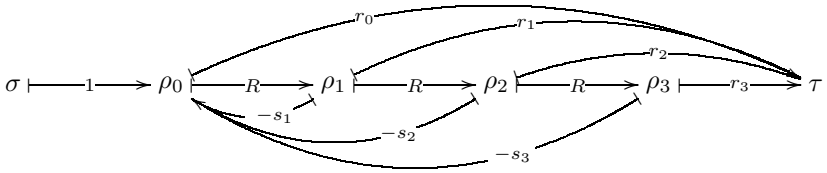
(b) Conversely, any such function can be implemented by a stream circuit.

Proof: We have seen many examples in the previous sections bearing witness to statement (a). A general proof would require a precise definition of stream circuit as well as a formalisation of Assumption 3.11, and is omitted here.

For (b), we treat only the special case that

$$\rho = \frac{r_0 + r_1 X + r_2 X^2 + r_3 X^3}{1 + s_1 X + s_2 X^2 + s_3 X^3}$$

where we have taken $n = m = 3$ and $s_0 = 1$. The general case can be treated similarly. We claim that the following circuit implements the function $f(\sigma) = \rho \times \sigma$ (all $\sigma \in \mathbb{R}^\omega$):



where we have denoted input and output streams by σ and τ, and intermediate streams by $\rho_0, \rho_1, \rho_2, \rho_3$. They satisfy the following equations:

$$\begin{aligned}
\rho_0 &= \sigma - (s_1 \times \rho_1) - (s_2 \times \rho_2) - (s_3 \times \rho_3) \\
\rho_1 &= X \times \rho_0 \\
\rho_2 &= X \times \rho_1 \\
\rho_3 &= X \times \rho_2 \\
\tau &= (r_0 \times \rho_0) + (r_1 \times \rho_1) + (r_2 \times \rho_2) + (r_3 \times \rho_3)
\end{aligned}$$

It follows that

$$\rho_0 = \sigma - (s_1 X \times \rho_0) - (s_2 X^2 \times \rho_0) - (s_3 X^3 \times \rho_0)$$

As a consequence, we have, for $i = 0, 1, 2, 3$, that

$$\rho_i \quad = \quad \frac{X^i}{1 + s_1 X + s_2 X^2 + s_3 X^3} \times \sigma$$

This implies

$$\tau \quad = \quad \frac{r_0 + r_1 X + r_2 X^2 + r_3 X^3}{1 + s_1 X + s_2 X^2 + s_3 X^3} \times \sigma$$

whereby the claim above is proved. □

Taking $\sigma = 1$ in Theorem 3.14 gives the following corollary.

Corollary 3.15 *A stream $\rho \in \mathbb{R}^\omega$ is rational if and only if it is generated by a (finite) stream circuit.* □

Exercise 3.13 Give for each of the streams below a circuit that generates it:

(i)

$$\frac{1 + X + X^2}{1 - X}$$

(ii)

$$\frac{X + X^3 + X^5}{1 + X^2 + X^4 + X^6}$$

(iii)

$$\frac{X^3}{1 - X - X^2}$$

□

Bibliography

[AM80] M.A. Arbib and E.G. Manes. Machines in a category. *Journal of Pure and Applied Algebra*, 19:9–20, 1980.

[AM89] P. Aczel and N. Mendler. A final coalgebra theorem. In D.H. Pitt, D.E. Ryeheard, P. Dybjer, A. M. Pitts, and A. Poigne, editors, *Proceedings category theory and computer science*, number 389 in Lecture Notes in Computer Science, pages 357–365. Springer-Verlag, 1989.

[CMC] CMCS'98-'02. Proceedings of the international workshop series coalgebraic methods in computer science. Available at URL: www.elsevier.nl/locate/entcs. Selected papers of these proceedings have been/are being published in the journals "Theoretical Computer Science" (Vol. 260 and 280), "Theoretical Informatics and Applications", "Mathematical Structures in Computer Science.".

[HT] The Haskell Team. http://www.haskell.org.

[JR97] Bart Jacobs and Jan Rutten. A tutorial on (co)algebras and (co)induction. *Bulletin of the EATCS*, 62:222–259, 1997. Available at URL: www.cwi.nl/∼janr.

[MA86] E.G. Manes and M.A. Arbib. *Algebraic approaches to program semantics*. Texts and monographs in computer science. Springer-Verlag, 1986.

[Mil80] R. Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer-Verlag, 1980.

[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.

[Rut98] J.J.M.M. Rutten. Automata and coinduction (an exercise in coalgebra). Report SEN-R9803, CWI, 1998. Available at URL: www.cwi.nl. Also in the proceedings of CONCUR '98, D. Sangiorgi and R. de Simone (eds.), LNCS 1466, Springer-Verlag, 1998, pp. 194–218.

[Rut99] J.J.M.M. Rutten. Automata, power series, and coinduction: taking input derivatives seriously (extended abstract). Report SEN-R9901, CWI, 1999. Available at URL: www.cwi.nl. Also in the proceedings of ICALP '99, J. Wiedermann, P. van Emde Boas, and M. Nielsen (eds.), LNCS 1644, Springer-Verlag, 1999, pp. 645–654.

[Rut00] J.J.M.M. Rutten. Universal coalgebra: a theory of systems. *Theoretical Computer Science*, 249(1):3–80, 2000.

[Rut01] J.J.M.M. Rutten. Elements of stream calculus (an extensive exercise in coinduction). In S. Brooks and M. Mislove, editors, *Proceedings of MFPS 2001*, volume 45 of *ENTCS*, pages 1–66. Elsevier Science Publishers, 2001. Also available as report SEN-R0120 at www.cwi.nl. To appear in Mathematical Structures in Computer Science.

[vB76] J. van Benthem. *Modal correspondence theory.* PhD thesis, University of Amsterdam, Amsterdam, 1976.