On Streams and Coinduction

Preface

The natural numbers are the most famous instance of an initial algebra and constitute the typical example for the explanation of induction. Similarly and dually, streams are the best known example of a final coalgebra and offer a perfect playground for the use of coinduction, both for definitions and for proofs. In these notes, we explain the basics of streams over an arbitrary alphabet in Chapter 1. Next we study the special case of streams over the reals in Chapter 2, and show how the (field) structure of the reals induces an equally rich structure on the set of streams. This leads to a calculus of streams, in close analogy to classical calculus in analysis, which is similar but more general than the calculus of generating functions. We apply the calculus of streams to the problem of solving differential equations in Chapter 3 and to various counting problems from enumerative combinatorics in Chapter 4. In Chapter 5, we apply streams and coinduction to a problem stemming from the world of computer science, more specifically, software engineering, by constructing a model for (software) component connectors. In Chapter A, the definitions of the most important operations on streams are summarized.

Chapter 1 is the basis for all subsequent chapters. Chapter 2 is needed for both Chapter 3 and Chapter 4 but not for Chapter 5, which depends solely on Chapter 1. In addition to the work by many other authors, which is discussed at the end of each chapter, the material of Chapters 1 through Chapter 4 is based on the following papers: [**35**–**40**]. Chapter 5 is based on very recent joint work with Farhad Arbab (CWI, Amsterdam), which appeared as: F. Arbab and J. J. M. M. Rutten, *A coinductive calculus of component connectors*, Report SEN-R0216, CWI, 2002.

Amsterdam, September 2002
Jan Rutten

Acknowledgments

I am very grateful to Prakash Panangaden for many years of inspiring discussions and meetings (starting in a discotheque in Tampere, Finland, in 1988), for organising the meeting 'Mathematical Models and Techniques for Analysing Systems' in the fall of 2002, and for asking me to present my work on streams and coinduction there. Many thanks are due to the following persons, for all what I have learned from them over the past years, through their papers, discussions, comments, and suggestions: Jiri Adamek, Farhad Arbab, Jaco de Bakker, Alexandru Baltag, Falk Bartels, Frank de Boer, Marcello Bonsangue,Franck van Breugel, Andrea Corradini, Philippe Darondeau, Jan van Eijk, Martin Escardó, H. Peter Gumm, Jan Heering, Furio Honsell, Bart Jacobs, Jan Willem Klop, Joost Kok, Jan Komenda, Clemens Kupke, Alexander Kurz, Elena Marchiori, Doug McIlroy, Wilfried Meyer Viol, Mike Mislove, Ugo Montanari, Larry Moss, Ataru Nakagawa, Maurice Nivat, Prakash Panangaden, Dusko Pavlović, Andy Pitts, Gordon Plotkin, John Power, Horst Reichel, Martin Rößiger, Jan van Schuppen, Nico Temme, Hendrik Tews, Daniele Turi, Erik de Vink, and Fer-Jan de Vries. I am very grateful to Franck van Breugel for many detailed comments on an earlier version of these notes.

CHAPTER 1

Streams and Coinduction

We introduce the set of streams (over a given set) and explain, using some elementary notions from universal coalgebra, how to define streams and operations on streams by coinduction, 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

$$A^\omega = \big\{\sigma \mid \sigma\colon \{0, 1, 2, \dots\} \to A\big\}.$$

In this chapter, no assumptions on A are made but later we shall look at specific choices for A. In particular, A will be the set $\mathbb{R}$ of real numbers in Chapter 2. 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:

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

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

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

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

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

Now the nth element of σ is also given by

$$\sigma(n) = \sigma^{(n)}(0). \tag{1}$$

1.2. Proofs by Coinduction

In order to conclude that two streams σ and τ are equal, it is both necessary and sufficient to prove

$$\forall n \geq 0, \quad \sigma(n) = \tau(n). \tag{2}$$

What general methods for establishing the validity of (2) are there? A straightforward *induction* on the natural number n (prove $\sigma(0) = \tau(0)$ and show that

$\sigma(n) = \tau(n)$ implies $\sigma(n+1) = \tau(n+1)$) may seem the obvious answer, but very often it is not. There will be numerous occasions where we will have no workable description or formula for $\sigma(n)$ and $\tau(n)$, and where, consequently, induction simply cannot be applied.

Instead, we shall take a coalgebraic perspective on A^ω, and use almost exclusively the proof principle of *coinduction*, which is formulated in terms of the following notion.

DEFINITION 1.2.1. 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, if $\langle \sigma, \tau \rangle \in R$ then

(i) $\sigma(0) = \tau(0)$ and
(ii) $\langle \sigma', \tau' \rangle \in R$.

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

One easily checks that unions and (relational) compositions of bisimulations are bisimulations again. 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*. This bisimilarity relation $\sim$ is the union of all bisimulations and, therewith, the greatest bisimulation.

THEOREM 1.2.2 (Coinduction). *For all* $\sigma, \tau \in A^\omega$,

$$\sigma \sim \tau \implies \sigma = \tau.$$

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 identity (1), $\sigma(n) = \tau(n)$, for all $n \geq 0$. Now $\sigma = \tau$ follows. □

The converse of the theorem above holds trivially, since $\{\langle \sigma, \sigma \rangle \mid \sigma \in A^\omega\}$ is a bisimulation relation on A^ω. The theorem can be used as a proof principle: in order to prove the equality of two streams σ and τ, it is sufficient to establish the existence of a bisimulation relation $R \subseteq A^\omega \times A^\omega$ with $\langle \sigma, \tau \rangle \in R$.

The above coinduction proof principle can be seen as a systematic way of strengthening the statement one is trying to prove: instead of proving only the single identity $\sigma = \tau$, one extends the set $\{\langle \sigma, \tau \rangle\}$ to a generally much larger set $R \subseteq A^\omega \times A^\omega$, with $\langle \sigma, \tau \rangle \in R$, such that R is a bisimulation relation. (As we shall see, the way such a larger set is constructed is always the same, and amounts to the computation of the closure of the original set under the taking of derivatives.) For such bisimulation relations, Theorem 1.2.2 states, once and for all, that for *all* pairs $\langle \alpha, \beta \rangle \in R$, we have $\alpha = \beta$. Since the proof of Theorem 1.2.2 is by induction, the theorem can thus be viewed as a systematic way of what sometimes is called *induction loading*.

The fact that the coinduction proof principle is proved by induction on the natural numbers, suggests that induction is a more fundamental principle than coinduction. We do not really address this issue here, but only remark that it is possible to develop a theory of so-called coinductive natural numbers. These consist of the ordinary natural numbers plus one extra element called infinity, and satisfy a basic coinduction proof principle from which the usual induction proof principle is derivable.

We shall see many examples of proofs by coinduction: one of the main reasons why we have become interested in A^ω at all, is that it offers a perfect playground

for demonstrating the use and usefulness of coinduction. First, however, we discuss a corresponding definition principle.

1.3. Definitions by Coinduction

We use the word coinduction also as a term for certain definitions, called behavioural differential equations. They are based on a universal property of A^ω, which we introduce next, using again some elementary universal coalgebra.

A *stream coalgebra* or *stream automaton* is a pair $Q = (Q, \langle o_Q, t_Q \rangle)$ consisting of a set Q of states, together with an output function $o_Q \colon Q \to A$, and a transition function $t_Q \colon Q \to Q$. For a state q, the output value $o_Q(q)$ can be viewed as its *observable behaviour*, and the transition value $t_Q(q)$ (the next state) can be viewed as its *dynamical behaviour*.

A *homomorphism*

$$f \colon (P, \langle o_P, t_P \rangle) \to (Q, \langle o_Q, t_Q \rangle)$$

between two stream automata is a behaviour preserving function $f \colon P \to Q$: for all p in P, $o_P(p) = o_Q\big(f(p)\big)$ and $f\big(t_P(p)\big) = t_Q\big(f(p)\big)$. In other words, a function $f \colon P \to Q$ is a homomorphism if the following diagram commutes:

$$\begin{array}{ccc} P & \xrightarrow{\ f\ } & Q \\ {\scriptstyle \langle o_P, t_P \rangle}\downarrow & & \downarrow{\scriptstyle \langle o_Q, t_Q \rangle} \\ A \times P & \xrightarrow[1_A \times f]{} & A \times Q \end{array} .$$

Here $(1_A \times f)(\langle a, p \rangle) = \langle a, f(p) \rangle$, for all $\langle a, p \rangle \in A \times P$. The set A^ω of all streams can itself be turned into a stream automaton as follows. Defining $o \colon A^\omega \to A$ by $o(\sigma) = \sigma(0)$ and $t \colon A^\omega \to A^\omega$ by $t(\sigma) = \sigma'$, we obtain a stream automaton $(A^\omega, \langle o, t \rangle)$. It has the following universal property.

THEOREM 1.3.1. *The automaton $(A^\omega, \langle o, t \rangle)$ is* final *among the family of all stream automata. That is, for any automaton $(Q, \langle o_Q, t_Q \rangle)$ there exists a unique homomorphism $l \colon (Q, \langle o_Q, t_Q \rangle) \to (A^\omega, \langle o, t \rangle)$:*

$$\begin{array}{ccc} Q & \overset{\exists! l}{\dashrightarrow} & A^\omega \\ {\scriptstyle \langle o_Q, t_Q \rangle}\downarrow & & \downarrow{\scriptstyle \langle o, t \rangle} \\ A \times Q & \underset{1_A \times l}{\dashrightarrow} & A \times A^\omega \end{array} .$$

PROOF. Let $(Q, \langle o_Q, t_Q \rangle)$ be a stream automaton and define $l \colon Q \to A^\omega$ as

$$l(q) = \big(o(q), o\big(t(q)\big), o\big(t\big(t(q)\big)\big), \dots\big)$$

for q in Q. It is straightforward to show that l is a homomorphism from $(Q, \langle o_Q, t_Q \rangle)$ to $(A^\omega, \langle o, t \rangle)$. 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, $o\big(f(q)\big) = o_Q(q) = o\big(g(q)\big)$. Furthermore, $t\big(f(q)\big) = f\big(t_Q(q)\big)$ and $t\big(g(q)\big) = g\big(t_Q(q)\big)$. Because $\big\langle f\big(t_Q(q)\big), g\big(t_Q(q)\big) \big\rangle \in R$, this shows that R is a bisimulation. Now $f = g$ follows by the coinduction proof principle Theorem 1.2.2. □

The finality of $(A^\omega, \langle o, t\rangle)$ provides the formal basis for *coinductive definitions* of (operators on) streams, which we call *behavioural differential equations*. Similarly to differential equations in analysis (such as $f(0) = 1$ and $f' = f$), such equations define streams by specifying their initial value and derivative, that is, their "behaviour". We present a few examples and shall become more general in later chapters. Consider the following behavioural differential equation:

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

(We see that a behavioural differential equation consists of two equalities: one specifying the derivative and one specifying the initial value.) It is clear that the stream $\sigma = (a, a, a, \dots)$ satisfies the equalities above, and that it is the only stream with this property. More formally, the unique existence of a solution of the behavioural differential equation can be established with the help of the finality of the set of streams. To this end, consider a stream automaton $(Q, \langle o_Q, t_Q\rangle)$ consisting of a singleton set $Q = \{q\}$ with $o_Q(q) = a$ and $t_Q(q) = q$. By Theorem 1.3.1, there exists a unique homomorphism

$$\begin{array}{ccc} Q & \xrightarrow{\;l\;} & A^\omega \\ {\scriptstyle \langle o_Q, t_Q\rangle}\downarrow & & \downarrow{\scriptstyle \langle o, t\rangle} \\ A \times Q & \xrightarrow[1_A \times l]{} & A \times A^\omega \end{array}.$$

We now define $\sigma = l(q)$. Because l is a homomorphism, it follows that σ is a solution of the differential equation: $\sigma' = t(\sigma) = t\big(l(q)\big) = l\big(t_Q(q)\big) = l(q) = \sigma$ and $\sigma(0) = o(\sigma) = o\big(l(q)\big) = o_Q(q) = a$. If τ is a stream with $\tau' = \tau$ and $\tau(0) = a$, then $\sigma = \tau$ follows, by the coinduction proof principle Theorem 1.2.2, 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.

As a second example, we consider the following system of behavioural differential equations (one for each σ and τ in A^ω):

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

.

In order to show that the above system of differential equations uniquely determines a binary operation $\mathrm{zip}\colon A^\omega \times A^\omega$, we consider a stream automaton $(Q, \langle o_Q, t_Q\rangle)$, consisting of $Q = A^\omega \times A^\omega$ 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 1.3.1, a unique homomorphism $l\colon Q \to A^\omega$. We now define $zip(\sigma, \tau) = l(< \sigma, \tau >)$. Similar to the first example, it is not difficult to prove that zip is the unique function satisfying the above system of behavioural differential equations. Note that zip could have been, equivalently, defined by the following "global" formula, for every $n \geq 0$:

$$\mathrm{zip}(\sigma, \tau)(2n) = \sigma(n), \quad \mathrm{zip}(\sigma, \tau)(2n + 1) = \tau(n).$$

Although it may seem more complicated in this particular example, we prefer to work with behavioural differential equations for two reasons. Firstly, they are more general. We shall see many examples of streams defined by behavioural differential equations for which there is no easy "global" formula such as the one for zip

above. Secondly, we can use the coinduction proof principle to reason about streams that have been defined coinductively, that is, by means of behavioural differential equations. Again, many convincing examples will follow.

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, solve more complicated systems of behavioural differential equations by using more complicated stream automata. Much more can be said about the classification of families of differential equations and their solutions. In these notes, however, we shall only work with examples, and we shall refer to the literature for general results.

1.4. Examples of Coinductive Proofs

In order to present a few simple examples of coinductive reasoning, we introduce two more operators on streams:

derivative	initial value
$\mathrm{even}(\sigma)' = \mathrm{even}(\sigma'')$	$\mathrm{even}(\sigma)(0) = \sigma(0)$
$\mathrm{odd}(\sigma)' = \mathrm{odd}(\sigma'')$	$\mathrm{odd}(\sigma)(0) = \sigma'(0)$

.

The well-definedness of these operators can be proved as before. The reader should have no trouble convincing himself that these operations satisfy the following identities:

$$\mathrm{even}(\sigma) = \big(\sigma(0), \sigma(2), \sigma(4), \dots\big),$$
$$\mathrm{odd}(\sigma) = \big(\sigma(1), \sigma(3), \sigma(5), \dots\big).$$

As an example, we prove the following identity by coinduction: for all $\sigma, \tau \in A^\omega$,

$$\mathrm{even}\big(\mathrm{zip}(\sigma, \tau)\big) = \sigma. \tag{3}$$

In order to apply the coinduction proof principle Theorem 1.2.2, we define

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

and prove that R is a bisimulation relation. We have to show that every pair $\langle \mathrm{even}\big(\mathrm{zip}(\sigma, \tau)\big), \sigma\rangle \in R$ satisfies the two bisimulation conditions of Definition 1.2.1:

(i) $\mathrm{even}\big(\mathrm{zip}(\sigma, \tau)\big)(0) = \sigma(0)$ and
(ii) $\big\langle \mathrm{even}\big(\mathrm{zip}(\sigma, \tau)\big)', \sigma'\big\rangle \in R$

For (i), note that $\mathrm{even}\big(\mathrm{zip}(\sigma, \tau)\big)(0) = \mathrm{zip}(\sigma, \tau)(0) = \sigma(0)$. For (ii),

$$\begin{aligned}\big\langle \mathrm{even}\big(\mathrm{zip}(\sigma, \tau)\big)', \sigma'\big\rangle &= \langle \mathrm{even}\big(\mathrm{zip}(\sigma, \tau)''\big), \sigma'\rangle \\ &= \big\langle \mathrm{even}\big(\mathrm{zip}(\tau, \sigma')'\big), \sigma'\big\rangle \\ &= \big\langle \mathrm{even}\big(\mathrm{zip}(\sigma', \tau')\big), \sigma'\big\rangle \in R.\end{aligned}$$

This concludes the proof that R is a bisimulation. Identity (3) then follows from the proof principle Theorem 1.2.2.

That was easy. We just defined our R as the set of all pairs that we wanted to prove equal. As it turned out, R was a bisimulation and the identity followed by coinduction. This is the right way to start coinductive proofs in general. Often, however, the relation has to be extended further before it satisfies condition (ii) of the definition of a bisimulation. In other words, the relation has to be closed under the taking of derivatives. At every such extension, condition (i), on initial values,

has to be checked again. The proof of the following identity, for all $\sigma \in A^\omega$, clearly illustrates what we mean:

$$\mathrm{zip}\big(\mathrm{even}(\sigma), \mathrm{odd}(\sigma)\big) = \sigma. \tag{4}$$

In order to apply coinduction, we begin again by defining

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

Next we check whether the pairs in R satisfy the bisimulation conditions (i) and (ii) of Definition 1.2.1. For (i), we have

$$\mathrm{zip}\big(\mathrm{even}(\sigma), \mathrm{odd}(\sigma)\big)(0) = \mathrm{even}(\sigma)(0) = \sigma(0).$$

For (ii), we compute

$$\begin{aligned}\big\langle \mathrm{zip}\big(\mathrm{even}(\sigma), \mathrm{odd}(\sigma)\big)', \sigma'\big\rangle &= \big\langle \mathrm{zip}\big(\mathrm{odd}(\sigma), \mathrm{even}(\sigma)'\big), \sigma'\big\rangle \\ &= \big\langle \mathrm{zip}\big(\mathrm{odd}(\sigma), \mathrm{even}(\sigma'')\big), \sigma'\big\rangle\end{aligned}$$

which is *not* in R. The way to overcome this problem is as simple as it is effective: we extend R by including these latter pairs as well. In other words, our second proposal for R now looks like

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

Next, the two bisimulation conditions should be checked for all the newly added pairs $\big\langle \mathrm{zip}\big(\mathrm{odd}(\sigma), \mathrm{even}(\sigma'')\big), \sigma'\big\rangle \in R$. For (i),

$$\mathrm{zip}\big(\mathrm{odd}(\sigma), \mathrm{even}(\sigma'')\big)(0) = \mathrm{odd}(\sigma)(0) = \sigma'(0).$$

Computing derivatives, for (ii), gives

$$\begin{aligned}\big\langle \mathrm{zip}\big(\,\mathrm{odd}(\sigma), \mathrm{even}(\sigma'')\big)', \sigma''\big\rangle &= \big\langle \mathrm{zip}\big(\mathrm{even}(\sigma''), \mathrm{odd}(\sigma)'\big), \sigma''\big\rangle \\ &= \big\langle \mathrm{zip}\big(\mathrm{even}(\sigma''), \mathrm{odd}(\sigma'')\big), \sigma''\big\rangle\end{aligned}$$

which *is* an element of R. This concludes both the construction of the relation R and, at the same time, the proof that it is in fact a bisimulation. Identity (4) now follows from the coinduction proof principle.

The above proof principle (and related principles) is sometimes referred to as the *bisimulation proof method*. It can prove identities in the way we saw above. But it can also be used to *disprove* an equality: one might encounter, at some stage of the construction of a bisimulation relation, a pair that has to be added in order to satisfy bisimulation condition (ii), but which does not satisfy condition (i). Then the conclusion is that all the pairs that have been considered so far, are *not* equal.

1.5. Coinduction and Greatest Fixed Points

There is yet another, in fact more classical, way of understanding bisimulations and the coinduction proof principle. Consider the set

$$\mathcal{P}(A^\omega \times A^\omega) = \{R \mid R \subseteq A^\omega \times A^\omega\},$$

of binary relations on A^ω, and the function

$$\Phi\colon \mathcal{P}(A^\omega \times A^\omega) \to \mathcal{P}(A^\omega \times A^\omega),$$

defined, for any $R \subseteq A^\omega \times A^\omega$, by

$$\Phi(R) = \{\langle \alpha, \beta\rangle \mid \alpha(0) = \beta(0) \wedge \langle \alpha', \beta'\rangle \in R\}.$$

As an immediate consequence of the definition of bisimulation, we have

$$R \text{ is a bisimulation} \iff R \subseteq \Phi(R).$$

Bisimulation relations are, in other words, *post-fixed points* of Φ. Consequently, the coinduction proof principle (Theorem 1.2.2) is equivalent to the following equality, where $\mathrm{id}_{A^\omega} = \{\langle \alpha, \alpha \rangle \mid \alpha \in A^\omega\}$:

$$\mathrm{id}_{A^\omega} = \cup\{R \mid R \subseteq \Phi(R)\}.$$

Since id_{A^ω} is itself a (bisimulation and thus a) post-fixed point, it is in fact the greatest fixed point of Φ. Therefore the above equality is an instance of the following well-known greatest fixed point theorem [**45**]. Let X be any set and let $\mathcal{P}(X) = \{V \mid V \subseteq X\}$ be the set of all its subsets. If $\Psi\colon \mathcal{P}(X) \to \mathcal{P}(X)$ is a monotone operator, that is, $R \subseteq S$ implies $\Psi(R) \subseteq \Psi(S)$ for all $R \subseteq X$ and $S \subseteq X$, then Ψ has a greatest fixed point $P = \Psi(P)$ satisfying

$$P = \cup\{R \mid R \subseteq \Psi(R)\}. \tag{5}$$

This equality can be used as a proof principle in the same way as Theorem 1.2.2: In order to prove that $R \subseteq P$, for any $R \subseteq X$, it suffices to show that R is a post-fixed point of Ψ, that is, $R \subseteq \Psi(R)$. We shall see further instances of this theorem (and related proof principles) in Chapter 5.

1.6. Other Coalgebras

In the language of category theory, a *coalgebra* of a functor $F : \mathrm{Set} \to \mathrm{Set}$ is a pair $\langle S, f\colon S \to F(S)\rangle$ consisting of a set S, called the carrier or state set of the coalgebra, and a function $f\colon S \to F(S)$ called the structure map of the coalgebra. Thus a stream automaton

$$Q \xrightarrow{\langle o,t \rangle} A \times Q$$

is a coalgebra of the functor $A \times (-)\colon \mathrm{Set} \to \mathrm{Set}$ (which maps a set V to the Cartesian product $A \times V$ and a function f to $1_A \times f$). This functor can be viewed as a special instance of the functor $A \times (-)^B\colon \mathrm{Set} \to \mathrm{Set}$, which in addition to the set A has a set B as a second parameter. This functor maps a set V to the set $A \times V^B$, where $V^B = \{f \mid f\colon B \to V\}$ is the set of all functions from B to V. A coalgebra of this functor looks like

$$Q \xrightarrow{\langle o,t \rangle} A \times Q^B$$

with $o\colon Q \to A$, as before, but now with $t\colon Q \to Q^B$. Such a coalgebra is an automaton with state space Q, with outputs in A, given for any $q \in Q$ by $o(q)$, and with inputs from B, leading, for any input $b \in B$, to a next state $t(q)(b) \in Q$. Also the functor $A \times (-)^B$ has an interesting final coalgebra, consisting of the set

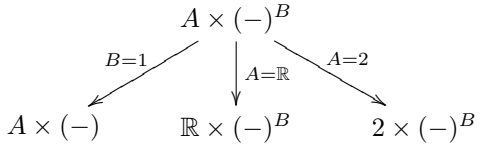
$$A^{B^*} = \{f \mid f\colon B^* \to A\}$$

of all functions from B^* to A (B^* is the set of finite sequences of elements of B). This set becomes particularly interesting if the set of outputs A is well-structured. Notably, if A is a *semi-ring* (which is, roughly speaking, a ring without an operation of subtraction) then A^{B^*} is the set of *formal power series* in many non-commutative variables from B and coefficients in A. Examples of such semi-rings are the set of real numbers (which is actually a ring) and the set $2 = \{0, 1\}$ of Boolean values.

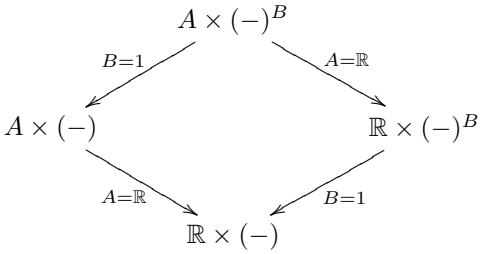
The corresponding final coalgebras are the set $\mathbb{R}^{B^*}$ of formal power series over B with coefficients in $\mathbb{R}$, and the set

$$\begin{aligned} 2^{B^*} &= \{f \mid f : B^* \to 2\} \\ &\cong \{L \mid L \subseteq B^*\} \end{aligned}$$

of all languages over B. The following diagram summarises all the functors that we have mentioned so far. In the diagram, arrows should be read as "specialises to":



The common specialisation of the left two functors to $\mathbb{R} \times (-)$:



turns out to have (almost) all the interesting properties of the general case. The corresponding final coalgebra, which is the set $\mathbb{R}^\omega$ of all streams of real numbers, will be the subject of Chapter 2.

1.7. Discussion

Streams are just one out of many (final) coalgebras. We saw already some examples in Section 1.6. For many more examples and some of the basic elements of the theory of "universal" coalgebra, see [**23, 38**]. The notion of bisimulation is due to Park and Milner [**31, 33**], who designed it as a notion of equivalence for a theory of concurrent processes. (Earlier, it already arose, under the name of p-relation, in the world of Kripke frames and modal logic [**46**].) Final coalgebras have been used as models for certain dynamical systems at least since [**6, 27**]. It was not until a categorical generalisation of the notion of bisimulation was introduced, by Aczel and Mendler in [**2**], that coinduction (both as a definition and as a proof principle) was taken more seriously. Notably, Aczel used it as the basis for a theory of non-well-founded sets in [**1**]; see also earlier related models by Forti and Honsell [**18**]. 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. The proceedings have been published in Elsevier's Electronic lecture Notes in Theoretical Computer Science (ENTCS) series, as volumes 11, 19, 33, 44, and 65. Formal power series, automata, and languages, have been extensively studied in the literature (see for instance [**11, 26**], to mention just two out of many references). A coalgebraic treatment of these structures can be found in [**35, 36**].

CHAPTER 2

Stream Calculus

We study the set $\mathbb{R}^\omega$ of streams of real numbers. Definitions will be given in terms of behavioural differential equations, and proofs by coinduction. We introduce a number of constants and operators, and develop a calculus of streams that can be viewed as a more general alternative for so-called generating functions. As a first application, we use the calculus to solve difference equations. More applications of this calculus will follow in subsequent chapters, on differential equations and counting problems.

2.1. Behavioural Differential Equations

We already saw some simple examples of behavioural differential equations in Section 1.3, which were shown to have unique solutions on the basis of the finality of the set of all streams. In what follows, we shall encounter many other, more complicated (systems of) equations. For two of them, we shall prove in the present section the existence of a unique solution in some detail. All other equations occurring in this chapter can be solved in precisely the same way.

We shall prove that for any two streams σ and τ, there exist unique streams $\sigma+\tau$ and $\sigma\otimes\tau$ that satisfy the following system of behavioural differential equations:

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

.

(We use $+$ both as an operation on streams and as the operation of addition of real numbers.) The rationale behind these equations will be given later, for $+$ in Section 2.2 and for $\otimes$ in Chapter 3. Here they merely serve as an example. Technically, the above system of equations is more complicated than the ones we saw earlier, in Chapter 1, because the derivative of $\sigma\otimes\tau$ is expressed in terms of not only $\otimes$ but also $+$. As a consequence, it is now not so easy to see in one glance how the solutions should look like. Still, the equations can be rather straightforwardly interpreted as recipes for the construction of the respective elements of the solutions. For instance,

$$\begin{aligned}
(\sigma\otimes\tau)(0) &= \sigma(0)\times\tau(0),\\
(\sigma\otimes\tau)(1) &= (\sigma\otimes\tau)'(0)\\
&= \big((\sigma'\otimes\tau) + (\sigma\otimes\tau')\big)(0)\\
&= \big(\sigma'(0)\times\tau(0)\big) + \big(\sigma(0)\times\tau'(0)\big)\\
&= \big(\sigma(1)\times\tau(0)\big) + \big(\sigma(0)\times\tau(1)\big)
\end{aligned}$$

$$\begin{aligned}(\sigma \otimes \tau)(2) &= (\sigma \otimes \tau)''(0)\\ &= \big((\sigma' \otimes \tau) + (\sigma \otimes \tau')\big)'(0)\\ &= \cdots\\ &= \big(\sigma(2) \times \tau(0)\big) + \big(2 \times \sigma(1) \times \tau(1)\big) + \big(\sigma(0) \times \tau(2)\big)\end{aligned}$$

and so on. This illustrates the algorithmic aspect of the behavioural differential equations. A formal proof of the unique existence of their solutions can again be based on the finality of $\mathbb{R}^\omega$, as follows.

THEOREM 2.1.1. *For all $\sigma, \tau \in \mathbb{R}^\omega$, there exist unique streams $\sigma + \tau$ and $\sigma \otimes \tau$ such that*

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

.

PROOF. We construct what could be called a syntactic stream automaton, whose states are given by expressions that include all the possible shapes that occur on the right side of the behavioural differential equations above. The solutions are then given by the unique homomorphism into $\mathbb{R}^\omega$. More precisely, let the set $\mathcal{E}$ of expressions be given by the following syntax:

$$E ::= \underline{\sigma} \mid E \mathbin{\underline{+}} F \mid E \mathbin{\underline{\otimes}} F.$$

The set $\mathcal{E}$ contains for every stream σ in $\mathbb{R}^\omega$ a constant symbol $\underline{\sigma}$. The operators that we are in the process of defining, are represented in $\mathcal{E}$ by a syntactic counterpart, denoted by $\mathbin{\underline{+}}$ and $\mathbin{\underline{\otimes}}$. The set $\mathcal{E}$ is next supplied with an automaton structure $(\mathcal{E}, \langle o_\mathcal{E}, t_\mathcal{E}\rangle)$ by defining functions $o_\mathcal{E} \colon \mathcal{E} \to \mathbb{R}$ and $t_\mathcal{E} \colon \mathcal{E} \to \mathcal{E}$ by induction on the syntactic structure of expressions. For the definition of $o_\mathcal{E}$ and $t_\mathcal{E}$ on the constant symbols $\underline{\sigma}$, the automaton structure $(\mathbb{R}^\omega, \langle o, t\rangle)$ on $\mathbb{R}^\omega$ is used:

$$t_\mathcal{E}(\underline{\sigma}) = \underline{t(\sigma)}, \quad o_\mathcal{E}(\underline{\sigma}) = o(\sigma).$$

(Recall from Section 1.3 that $o(\sigma) = \sigma(0)$ and $t(\sigma) = \sigma'$ for $\sigma \in \mathbb{R}^\omega$.) Thus the constant $\underline{\sigma}$ behaves in the automaton $\mathcal{E}$ precisely as the stream σ behaves in the automaton $\mathbb{R}^\omega$. (This includes $\mathbb{R}^\omega$ as a subautomaton in $\mathcal{E}$.) For composite expressions, the definitions of $o_\mathcal{E}$ and $t_\mathcal{E}$ literally follow the definition of the corresponding behavioural differential equations. Writing $E(0)$ for $o_\mathcal{E}(E)$ and E' for $t_\mathcal{E}(E)$, for any expression E in $\mathcal{E}$, we put

definition of $t_\mathcal{E}$	definition of $o_\mathcal{E}$
$(E \mathbin{\underline{+}} F)' = E' \mathbin{\underline{+}} F'$	$(E \mathbin{\underline{+}} F)(0) = E(0) + F(0)$
$(E \mathbin{\underline{\otimes}} F)' = (E' \mathbin{\underline{\otimes}} F) + (E \mathbin{\underline{\otimes}} F')$	$(E \mathbin{\underline{\otimes}} F)(0) = E(0) \times F(0)$

.

The above defines the functions $o_\mathcal{E}$ and $t_\mathcal{E}$ on all of $\mathcal{E}$. Because $\mathbb{R}^\omega$ is a final automaton, there exists, by Theorem 1.3.1, a unique homomorphism

$$\begin{array}{ccc} \mathcal{E} & \xrightarrow{\;l\;} & \mathbb{R}^\omega \\ {\scriptstyle \langle o_\mathcal{E}, t_\mathcal{E}\rangle}\downarrow & & \downarrow{\scriptstyle \langle o, t\rangle} \\ A \times \mathcal{E} & \xrightarrow[1_A \times l]{} & A \times \mathbb{R}^\omega \end{array}$$

which assigns to each expression E the stream $l(E)$ it represents. With the help of l, we now define operators

$$+\colon \mathbb{R}^\omega \times \mathbb{R}^\omega \to \mathbb{R}^\omega, \quad \otimes\colon \mathbb{R}^\omega \times \mathbb{R}^\omega \to \mathbb{R}^\omega$$

by putting, for all $\sigma, \tau \in \mathbb{R}^\omega$,

$$\sigma + \tau = l(\underline{\sigma} \mathbin{\underline{+}} \underline{\tau}), \quad \sigma \otimes \tau = l(\underline{\sigma} \mathbin{\underline{\otimes}} \underline{\tau}).$$

In order to show that these operators satisfy the behavioural differential equations above, one first notes the following facts:

(i) For all $E \in \mathcal{E} : l(E)(0) = E(0)$ and $\big(l(E)\big)' = l(E')$.
(ii) For all $\sigma \in \mathbb{R}^\omega : l(\underline{\sigma}) = \sigma$.
(iii) The homomorphism $l\colon \mathcal{E} \to \mathbb{R}^\omega$ is compositional. That is, for all expressions E and F in $\mathcal{E}$,
  (a) $l(E \mathbin{\underline{+}} F) = l(E) + l(F)$
  (b) $l(E \mathbin{\underline{\otimes}} F) = l(E) \otimes l(F)$.
  Note that the operators on the left are syntactic constructors of expressions, while on the right there are the operators on streams.

Fact (i) is equivalent to the fact that l is a homomorphism (recall that by definition $\sigma(0) = o(\sigma)$, $\sigma' = t(\sigma)$, for $\sigma \in \mathbb{R}^\omega$, and $E(0) = o_{\mathcal{E}}(E)$, $E' = t_{\mathcal{E}}(E)$, for $E \in \mathcal{E}$). Facts (ii) and (iii) can be proved by coinduction.

Using these three facts, it follows, for all $\sigma, \tau \in \mathbb{R}^\omega$, that

$(\sigma + \tau)' = \sigma' + \tau'$	$(\sigma + \tau)(0) = \sigma(0) + \tau(0)$
$(\sigma \otimes \tau)' = (\sigma' \otimes \tau) + (\sigma \otimes \tau')$	$(\sigma \otimes \tau)(0) = \sigma(0) \times \tau(0)$

.

For instance, we have

$$\begin{aligned}
(\sigma \otimes \tau)' &= \big(l(\underline{\sigma} \mathbin{\underline{\otimes}} \underline{\tau})\big)' \\
&= l\big((\underline{\sigma} \mathbin{\underline{\otimes}} \underline{\tau})'\big) && \text{[(i): } l \text{ is a homomorphism]} \\
&= l\left(\big((\underline{\sigma})' \mathbin{\underline{\otimes}} \underline{\tau}\big) \mathbin{\underline{+}} \big(\underline{\sigma} \mathbin{\underline{\otimes}} (\underline{\tau})'\big)\right) && \text{[by the definition of } t_{\mathcal{E}}\text{]} \\
&= l\big((\underline{\sigma'} \mathbin{\underline{\otimes}} \underline{\tau}) \mathbin{\underline{+}} (\underline{\sigma} \mathbin{\underline{\otimes}} \underline{\tau'})\big) && \text{[by the definition of } t_{\mathcal{E}}\text{]} \\
&= l(\underline{\sigma'} \mathbin{\underline{\otimes}} \underline{\tau}) + l(\underline{\sigma} \mathbin{\underline{\otimes}} \underline{\tau'}) && \text{[(iii): } l \text{ is compositional]} \\
&= \big(l(\underline{\sigma'}) \otimes l(\underline{\tau})\big) + \big(l(\underline{\sigma}) \otimes l(\underline{\tau'})\big) && \text{[(iii): } l \text{ is compositional]} \\
&= (\sigma' \otimes \tau) + (\sigma \otimes \tau') && \text{[(ii): } l \text{ is the identity on stream constants]}
\end{aligned}$$

and similarly for the other cases. This proves that the operators $+$ and $\otimes$ satisfy the behavioural differential equations above. Using coinduction, one can easily prove that they are the unique operators with this property. (The few details that have been omitted in the above proof can be found in [**37**, Section 13].) □

From now on, we shall without further ado introduce all kinds of streams and operators by means of behavioural differential equations. All of them can be justified in the same manner as above, by coinduction and finality of $\mathbb{R}^\omega$.

2.2. Basic Stream Calculus

Taking the existence of solutions of behavioural differential equations from now on for granted, much energy will be spent on the computation of closed formulas for such solutions, expressing them in terms of a basic set of stream constants and stream operators, which we introduce next.

First of all, we want to be able to view any real number as a constant stream. For $r \in \mathbb{R}$, let $[r]$ be the unique stream satisfying the following behavioural differential equation:

derivative	initial value
$[r]' = [0]$	$[r](0) = r$

.

The following definition is clearly equivalent:

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

Having once formally introduced the inclusion of the reals into the set of streams by means of this operator $[\ ]\colon \mathbb{R} \to \mathbb{R}^\omega$, we immediately observe that in stream calculus there will hardly be any chance of confusing the stream $[r]$ with the real number r it represents. Therefore we shall usually simply write r for $[r]$.

We introduce one more constant: Let X be the unique stream satisfying

derivative	initial value
$X' = [1]$	$\mathrm{X}(0) = 0$

.

The constant X plays a crucial role in stream calculus, and may be thought of as a formal variable. Again, there is a more explicit equivalent definition:

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

Next we repeat the definition of the operation of sum that we have already seen as an example in Section 2.1. The sum $\sigma + \tau$ of two streams σ and τ is defined as the unique stream satisfying:

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

.

(Note that we are using the same symbol $+$ both for the sum of streams and the sum of real numbers.) Alternatively and equivalently, the sum of $\sigma = (s_0, s_1, s_2, \dots)$ and $\tau = (t_0, t_1, t_2, \dots)$ is given by

$$\sigma + \tau = (s_0 + t_0, s_1 + t_1, s_2 + t_2, \dots).$$

For the motivation of the next operator, let us briefly look at the world of (real-valued) functions, which at more than one occasion, will be a source of inspiration for stream calculus. Consider a function $f\colon \mathbb{R} \to \mathbb{R}$ and a stream (of coefficients) $\sigma = (s_0, s_1, s_2, \dots)$ such that $f(x) = s_0 + s_1 x + s_2 x^2 + \cdots$. Writing $\bar{f}(x) = s_1 + s_2 x + s_3 x^2 + \cdots$, we have

$$f(x) = s_0 + \big(x \times \bar{f}(x)\big).$$

Let $g : \mathbb{R} \to \mathbb{R}$ and $\tau = (t_0, t_1, t_2, \dots)$ be another such function and stream, with $g(x) = t_0 + t_1 x + t_2 x^2 + \cdots$ and write, similarly, $g(x) = t_0 + \big(x \times \bar{g}(x)\big)$. Computing now the (elementwise) function product $f(x) \times g(x)$, one finds

$$\begin{aligned} f(x) \times g(x) &= \big(s_0 + \big(x \times \bar{f}(x)\big)\big) \times \big(t_0 + \big(x \times \bar{g}(x)\big)\big) \\ &= (s_0 \times t_0) + x \times \big(\bar{f}(x) \times g(x) + s_0 \times \bar{g}(x)\big) \end{aligned}$$

This is one way of motivating the following definition. Let the(*convolution*) *product* $\sigma \times \tau$ of two streams σ and τ be the unique stream satisfying:

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

.

(Note that we are using the same symbol $\times$ both for the product of streams and the product of real numbers. Further note that in the above definition, $\sigma(0) \times \tau'$ is a shorthand for $[\sigma(0)] \times \tau'$.) We shall use the following standard conventions: for all $n \geq 0$,

$$-\sigma \equiv [-1] \times \sigma, \quad \sigma\tau \equiv \sigma \times \tau, \quad \sigma^0 \equiv 1, \quad \sigma^{n+1} \equiv \sigma \times \sigma^n$$

On the basis of the analogy between function multiplication and stream multiplication, one might have expected

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

which generally is *not* valid. Apparently, stream differentiation does not quite correspond to function derivation in analysis. Rather, it corresponds to the above transformation of a function f into $\bar{f}$, an operation which is not usually present in analysis. Later we shall see variations on both the notion of stream derivative and the operator of multiplication, which have more familiar properties. Let it be noted, however, that in stream calculus, both the operations of stream derivative and of convolution product, with their non-standard properties, are of central importance.

There is also the following formula for the nth element of $\sigma \times \tau$, for any $n \geq 0$,

$$(\sigma \times \tau)(n) = \sum_{k=0}^{n} \sigma(n-k) \times \tau(k) \tag{6}$$

which could have been (and traditionally is) taken as an alternative definition of $\sigma \times \tau$. (Since it will play no role in what follows, a proof of this identity, which would not be too difficult, is omitted.) Note that stream product shares this property with the function product considered above, since

$$f(x) \times g(x) = s_0t_0 + (s_1t_0 + s_0t_1) \times x + (s_2t_0 + s_1t_1 + s_0t_2) \times x^2 + \cdots .$$

As we shall see time and again, coinductive reasoning directly in terms of the above behavioural differential equation is quite a bit simpler than the use of (6), because of the summation over the indices k and $(n-k)$ in the latter. Equally importantly, we shall see examples of other definitions, such as that of inverse below, where no simple formula for the nth element is known.

We introduce an operator on streams that is inverse to multiplication in the following sense. Given a stream σ, we look for a stream σ^{-1} such that $\sigma \times \sigma^{-1} = 1$. Constant streams $[r] = (r, 0, 0, 0, \dots)$ with $r \neq 0$, clearly have $[r]^{-1} = (r^{-1}, 0, 0, 0, \dots) = [r^{-1}]$ as their inverse, because

$$(r, 0, 0, 0, \dots) \times (r^{-1}, 0, 0, 0, \dots) = (1, 0, 0, 0, \dots).$$

Considering now an arbitrary stream σ with $\sigma(0) \neq 0$, and assuming for a second that σ^{-1} indeed existed, it would have to satisfy

$$0 = 1' = (\sigma \times \sigma^{-1})' = (\sigma' \times \sigma^{-1}) + \big(\sigma(0) \times (\sigma^{-1})'\big).$$

This implies

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

Multiplying (on the left) with the inverse of (the stream) $\sigma(0)$, one obtains

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

Thus the equation $1 = \sigma \times \sigma^{-1}$ determines what $(\sigma^{-1})'$ should be. It also determines the initial value $(\sigma^{-1})(0)$: taking the initial value on both sides of the equation (recall that 1 stands for the stream $[1] = (1, 0, 0, 0, \dots)$) gives

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

whence $(\sigma^{-1})(0) = \sigma(0)^{-1}$. What has happened is that we have derived from our "specification" $\sigma \times \sigma^{-1} = 1$ a behavioural differential equation, which we can now simply take as the definition of inverse. For all $\sigma \in \mathbb{R}^\omega$ with $\sigma(0) \neq 0$, let the (*multiplicative*) *inverse* σ^{-1} of σ be the unique stream satisfying the following equation:

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

.

Whenever we write σ^{-1}, we shall silently assume that $\sigma(0) \neq 0$. As usual, we write

$$\sigma^{-n} \equiv (\sigma^{-1})^n, \quad \frac{\sigma}{\tau} \equiv \sigma \times \tau^{-1}.$$

With this convention, we can write $(\sigma^{-1})'$ as

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

which is a bit easier to remember. For reasons similar to the case of stream multiplication, stream derivation of inverse behaves differently from what we are used to in analysis. In particular, $(\sigma^{-1})' = -\sigma' \times \sigma^{-2}$ generally does *not* hold.

Was our preference to use behavioural differential equations for the definition of the operators so far based on either obsession (in the case of the constants) or computational convenience (in the case of the product), in the case of inverse, it turns out to be essential. For inverse, no explicit formula such as (6) for product, is known (not in the literature, not to us):

$$\sigma^{-1}(n) = ?.$$

The best one can do is to define inverse by means of a recurrence relation as follows. The elements of σ^{-1} are given by $\sigma^{-1}(0) = \sigma(0)^{-1}$ and

$$\sigma^{-1}(n) = \sigma(0)^{-1} \sum_{k=0}^{n-1} \sigma(n-k) \times \sigma^{-1}(k)$$

for $n \geq 1$. This is an immediate consequence of the requirement that $\sigma \times \sigma^{-1} = 1$ and identity (6)) above. Note that this recurrence is of so-called unbounded order: the nth element $\sigma^{-1}(n)$ depends on all of the n preceding values $\sigma^{-1}(0)$ through $\sigma^{-1}(n-1)$. Reasoning in terms of such recurrences is not only extremely tedious but even next to impossible.

The good news is that in stream calculus, there is no need for a closed formula for $\sigma^{-1}(n)$. All our reasoning about inverse, in fact about all streams and stream operators, will be coinductive, calculating with stream derivatives as specified by behavioural differential equations.

The definition of the following operation on streams is again clearly inspired by analysis. For all streams σ and τ with $\tau(0) = 0$, the *composition* $\sigma \circ \tau$ is defined

as the unique stream satisfying the following behavioural differential equation:

derivative	initial value
$(\sigma \circ \tau)' = \tau' \times (\sigma' \circ \tau)$	$(\sigma \circ \tau)(0) = \sigma(0)$

.

We shall often write

$$\sigma \circ \tau \equiv \sigma(\tau).$$

The condition that $\tau(0) = 0$ is needed in the proof of Theorem 2.5.3 in Section 2.5, which shows that composition behaves as usual.

2.3. Coinduction-Up-To

Before proving a number of basic properties of the operators introduced above, we present another proof principle, called coinduction-up-to, which is a mild generalisation of the coinduction proof principle of Theorem 1.2.2.

Definition 2.3.1. A *bisimulation-up-to* is a relation $R \subseteq \mathbb{R}^\omega \times \mathbb{R}^\omega$ such that, for all σ and τ in $\mathbb{R}^\omega$, if $\sigma \; R \; \tau$ then

(i) $\sigma(0) = \tau(0)$ and
(ii) there exist $n \geq 0$, $\alpha_0, \ldots, \alpha_n, \beta_0, \ldots, \beta_n \in \mathbb{R}^\omega$, such that $\sigma' = \alpha_0 + \cdots + \alpha_n$ and $\tau' = \beta_0 + \cdots + \beta_n$ and, for all $0 \leq i \leq n$: either $\alpha_i = \beta_i$ or $\alpha_i \; R \; \beta_i$. The latter condition will usually be denoted by

$$\alpha_0 + \cdots + \alpha_n \; (\Sigma R) \; \beta_0 + \cdots + \beta_n.$$

Theorem 2.3.2. *If $\sigma \; R \; \tau$, for streams σ and τ and a bisimulation-up-to relation $R \subseteq \mathbb{R}^\omega \times \mathbb{R}^\omega$, then $\sigma = \tau$.*

Proof. Let $R \subseteq \mathbb{R}^\omega \times \mathbb{R}^\omega$ be a bisimulation-up-to. Let $\overline{R}$ be the smallest relation satisfying

(1) $R \subseteq \overline{R}$,
(2) $\{\langle \sigma, \sigma \rangle \mid \sigma \in \mathbb{R}^\omega\} \subseteq \overline{R}$,
(3) for all $\alpha_0, \alpha_1, \; \beta_0, \beta_1 \in \mathbb{R}^\omega$, if $\alpha_0 \; \overline{R} \; \beta_0$ and $\alpha_1 \; \overline{R} \; \beta_1$ then $(\alpha_0 + \alpha_1) \; \overline{R} \; (\beta_0 + \beta_1)$.

Using the fact that R is a bisimulation-up-to, one can easily prove by induction on the definition of $\overline{R}$ that the latter is an (ordinary) bisimulation relation on $\mathbb{R}^\omega$. Since $R \subseteq \overline{R}$, the theorem follows by (ordinary) coinduction, Theorem 1.2.2. □

For a simple but typical example, consider streams σ, τ and ρ such that

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

.

Then $\{\langle \sigma, \tau \rangle, \langle \sigma, \rho \rangle\}$ is a bisimulation-up-to and $\sigma = \tau = \rho$ follows by coinduction-up-to. Other examples can be found in the proof of Theorem 2.4.1.

2.4. Basic Properties

We prove a number of elementary properties of stream operators.

THEOREM 2.4.1. *For all $r, s \in \mathbb{R}$ and $\sigma, \tau, \rho \in \mathbb{R}^\omega$,*

$$\begin{gathered}
\sigma + 0 = \sigma, \\
\sigma + \tau = \tau + \sigma, \qquad (7) \\
\sigma + (\tau + \rho) = (\sigma + \tau) + \rho, \\
0 \times \sigma = 0, \\
1 \times \sigma = \sigma, \\
[r] \times [s] = [r \times s], \\
\sigma \times \tau = \tau \times \sigma, \qquad (8) \\
\sigma \times (\tau + \rho) = (\sigma \times \tau) + (\sigma \times \rho), \qquad (9) \\
\sigma \times (\tau \times \rho) = (\sigma \times \tau) \times \rho, \\
\sigma \times \sigma^{-1} = 1, \qquad (10) \\
(\sigma^{-1})^{-1} = \sigma, \qquad (11) \\
(\sigma \times \tau)^{-1} = \tau^{-1} \times \sigma^{-1}, \qquad (12) \\
(X \times \sigma)' = \sigma. \qquad (13)
\end{gathered}$$

PROOF. We shall prove only some of the above identities (for the others see [**39**]). For (7) let $R = \{\langle \sigma + \tau, \tau + \sigma \rangle \mid \sigma, \tau \in \mathbb{R}^\omega\}$. Since $(\sigma + \tau)(0) = \sigma(0) + \tau(0) = (\tau + \sigma)(0)$ and

$$\begin{aligned}
(\sigma + \tau)' &= (\sigma' + \tau') \\
&\; R \; (\tau' + \sigma') \\
&= (\tau + \sigma)'
\end{aligned}$$

for any σ and τ, R is a bisimulation relation on $\mathbb{R}^\omega$. Identity (7) now follows by coinduction (Theorem 1.2.2). Identity (8) is, on the basis of the non-symmetrical definition of stream product, somewhat surprising. It is an immediate consequence of identity (6). A proof by coinduction can be given too, but is slightly complicated (see [**39**]). For (9), let $R = \{\langle \sigma \times (\tau + \rho), (\sigma \times \tau) + (\sigma \times \rho) \rangle \mid \sigma, \rho, \tau \in \mathbb{R}^\omega\}$. We prove that R is a bisimulation-up-to. Initial values match and computing derivatives, we find

$$\begin{aligned}
\big(\sigma \times (\tau + \rho)\big)' &= \big(\sigma' \times (\tau + \rho)\big) + \big(\sigma(0) \times (\tau' + \rho')\big) \\
&\; (\Sigma R) \; \big((\sigma' \times \tau) + (\sigma' \times \rho)\big) + \big((\sigma(0) \times \tau') + (\sigma(0) \times \rho')\big) \\
&= \big((\sigma' \times \tau) + \big(\sigma(0) \times \tau'\big)\big) + \big((\sigma' \times \rho) + \big(\sigma(0) \times \rho'\big)\big) \\
&= \big((\sigma \times \tau) + (\sigma \times \rho)\big)'
\end{aligned}$$

with (ΣR) as defined in Definition 2.3.1. Thus, R is a bisimulation-up-to and identity (9) follows by coinduction-up-to (Theorem 2.3.2). For identity (10) note

that

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

and coinduction does the rest. For (11), note that

$$\begin{aligned}\big((\sigma^{-1})^{-1}\big)' &= -\big(\sigma^{-1}(0)\big)^{-1} \times (\sigma^{-1})' \times (\sigma^{-1})^{-1} \\ &= -\sigma(0) \times \big(-\sigma(0)^{-1} \times \sigma' \times \sigma^{-1}\big) \times (\sigma^{-1})^{-1} \\ &= \sigma' \times \big(\sigma^{-1} \times (\sigma^{-1})^{-1}\big) \\ &= \sigma' \qquad \text{[by identity (10)]}\end{aligned}$$

and use coinduction. For (12), note that

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

Multiplying both sides of the resulting equality with $(\sigma \times \tau)^{-1}$ and using identity (10) implies (12). □

2.5. Infinite Sum

As was already suggested by the comparison between function product and stream product, in Section 2.2, streams can be viewed as power series in one formal variable (namely, the constant stream X). In order to make this more precise, we need to introduce (countably) infinite sums of streams.

Let $\sigma_0, \sigma_1, \sigma_2, \dots$ be a sequence of streams such that, for all $k \geq 0$, $\sigma_0(k) + \sigma_1(k) + \sigma_2(k) + \cdots < \infty$. Such a sequence is called *summable*. We define:

derivative	initial value
$\Big(\sum_{n=0}^{\infty} \sigma_n\Big)' = \sum_{n=0}^{\infty}(\sigma_n')$	$\Big(\sum_{n=0}^{\infty} \sigma_n\Big)(0) = \sum_{n=0}^{\infty} \sigma_n(0)$

Often we shall also write

$$\sum_{n=0}^{\infty} \sigma_n = \sigma_0 + \sigma_1 + \sigma_2 + \cdots .$$

Infinite sum is as well-behaved as binary sum. For instance, we have

$$\left(\sum_{n=0}^{\infty} \sigma_n\right) \times \tau = \sum_{n=0}^{\infty} (\sigma_n \times \tau)$$

and similarly for the other basic properties of sum. For the proof of the theorem below, we shall need the following basic property, which easily follows from identity (13) by induction: for all $n \geq 0$,

$$(X^{n+1})' = X^n.$$

THEOREM 2.5.1. *For all streams* $\sigma = (s_0, s_1, s_2, \dots)$,

$$s_0, s_1 \times X^1, \quad s_2 \times X^2, \dots$$

is summable, and satisfies

$$\sigma = s_0 + (s_1 \times X^1) + (s_2 \times X^2) + \cdots .$$

PROOF. Using the defining equation for infinite sum; the fact that if a stream σ is summable, then also σ' is summable; and identity (2.5), the proof is straightforward by coinduction. □

This theorem is to be contrasted with the following way to represent streams $\sigma = (s_0, s_1, s_2, \dots)$, which is very common in mathematics: if $f\colon \mathbb{R} \to \mathbb{R}$ is a function and r is a positive real number such that for all $|x| < r$,

$$f(x) = s_0 + s_1 x + s_2 x^2 + \cdots$$

(that is, the infinite sum on the right exists and is equal to $f(x)$), then f is called a *generating function* for the stream σ. The characterisation of streams given in Theorem 2.5.1 above, is different in two respects. The infinite sum is not merely an *encoding* of σ, but is itself a stream (built from the constant streams $s_i = (s_i, 0, 0, 0, \dots)$, for all $i \geq 0$, the constant stream $X = (0, 1, 0, 0, 0, \dots)$, and the operators of sum and product), which is *equal* to σ. Moreover, this equality is not subject to a condition of summability, but simply holds for any σ. In particular, this also holds for streams such as $(0!, 1!, 2!, \dots)$, for which no generating function exists (because $0! + 1!x + 2!x^2 + \cdots$ diverges for every $x \neq 0$).

Here are some identities involving infinite sums.

THEOREM 2.5.2. *For all streams* $\sigma = (s_0, s_1, s_2, \dots)$,

$$\frac{1}{1-X} = 1 + X + X^2 + X^3 + \cdots , \tag{14}$$

$$X \times \sigma = (0, s_0, s_1, s_2, \dots), \tag{15}$$

$$\frac{1}{1+X} = 1 - X + X^2 - X^3 + \cdots , \tag{16}$$

$$\frac{1}{1-X^2} = 1 + X^2 + X^4 + \cdots , \tag{17}$$

$$\frac{1}{(1-X)^2} = 1 + 2X + 3X^2 + 4X^3 + \cdots , \tag{18}$$

$$\frac{1}{1-rX} = 1 + rX + r^2X^2 + r^3X^3 + \cdots , \tag{19}$$

$$\frac{X}{1+X^2} = X - X^3 + X^5 - X^7 + \cdots . \tag{20}$$

PROOF. All of these identities can be easily proved by coinduction. □

The following theorem shows that stream composition $\sigma \circ \tau = \sigma(\tau)$, introduced at the end of Section 2.2, behaves as it should.

THEOREM 2.5.3. *For all streams* $\sigma = (s_0, s_1, s_2, \dots)$, *and* τ *with* $\tau(0) = 0$,

$$\sigma(\tau) = s_0 + (s_1 \times \tau^1) + (s_2 \times \tau^2) + \cdots . \tag{21}$$

PROOF. Because $\tau(0) = 0$, we have $(\tau^{n+1})' = \tau' \times \tau^n$, for all $n \geq 0$. As a consequence,

$$\begin{aligned}\big(s_0 + (s_1 \times \tau^1) + (s_2 \times \tau^2) + \cdots\big)' &= (s_1 \times \tau') + (s_2 \times \tau' \times \tau) + (s_3 \times \tau' \times \tau^2) + \cdots \\ &= \tau' \times \big(s_1 + (s_2 \times \tau) + (s_3 \times \tau^2) + \cdots\big).\end{aligned}$$

For $\sigma = (s_0, s_1, s_2, \dots)$, let

$$R = \big\{\big\langle \rho \times \sigma(\tau), \rho \times \big(s_0 + (s_1 \times \tau^1) + (s_2 \times \tau^2) + \cdots\big)\big\rangle \mid \sigma, \tau, \rho \in \mathbb{R}^\omega\big\}.$$

Because

$$\begin{aligned}
\big(\rho \times \sigma(\tau)\big)' &= \big(\rho' \times \sigma(\tau)\big) + \big(\rho(0) \times \big(\tau' \times \sigma'(\tau)\big)\big) \\
&= \big(\rho' \times \sigma(\tau)\big) + \big(\rho(0) \times \tau'\big) \times \sigma'(\tau) \\
&\Sigma R\ \rho' \times \big(s_0 + (s_1 \times \tau^1) + (s_2 \times \tau^2) + \cdots\big) + \\
&\qquad \big(\rho(0) \times \tau'\big) \times \big(s_1 + (s_2 \times \tau^1) + (s_3 \times \tau^2) + \cdots\big) \\
&= \rho' \times \big(s_0 + (s_1 \times \tau^1) + (s_2 \times \tau^2) + \cdots\big) + \\
&\qquad \rho(0) \times \big(\tau' \times \big(s_1 + (s_2 \times \tau^1) + (s_3 \times \tau^2) + \cdots\big)\big) \\
&= \big(\rho \times \big(s_0 +, (s_1 \times \tau^1) + (s_2 \times \tau^2) + \cdots\big)\big)'.
\end{aligned}$$

R is a bisimulation-up-to. The theorem follows by coinduction-up-to. □

Here are a few useful instances and applications of Theorem 2.5.3:

$$\begin{aligned}
\sigma(X) &= \sigma, \\
\sigma(-X) &= s_0 - s_1 X + s_2 X^2 - s_3 X^3 + \cdots \\
&= (s_0, -s_1, s_2, -s_3, \dots), \\
\tfrac{1}{2}\big(\sigma(X) + \sigma(-X)\big) &= (s_0, 0, s_3, 0, \dots) \\
\sigma(X^2) &= s_0 + s_1 X^2 + s_2 X^4 + s_3 X^6 + \cdots . \qquad (22)
\end{aligned}$$

2.6. Solving Behavioural Differential Equations

We introduce the "Fundamental Theorem" of stream calculus, named after the corresponding theorem from analysis. As we discussed in Section 2.1, the *existence* of a (unique) solution of many behavioural differential equations, can be proved using the finality of $\mathbb{R}^\omega$. With the help of the Fundamental Theorem, we can actually compute an *expression* for such solutions in terms of the basic operators on streams, namely, constants, sum, product, and inverse. In particular, we show how to obtain such expressions for the solutions of (homogeneous or non-homogeneous) linear equations, possibly with non-constant coefficients. Later, these solutions will be used to solve difference equations and analytical differential equations of a similar kind.

Here is the Fundamental Theorem of stream calculus.

THEOREM 2.6.1. *For all $\sigma \in \mathbb{R}^\omega$: $\sigma = \sigma(0) + (X \times \sigma')$.*

PROOF. We first note that $(X \times \sigma)' = \sigma$, for all $\sigma \in \mathbb{R}^\omega$. Next, we define $R = \{\langle \sigma, \sigma\rangle \mid \sigma \in \mathbb{R}^\omega\} \cup \{\langle \sigma, \sigma(0) + (X \times \sigma')\rangle \mid \sigma \in \mathbb{R}^\omega\}$. Because

$$\begin{aligned}
&\sigma'\ R\ \sigma' \\
&\quad = \big(\sigma(0) + (X \times \sigma')\big)'.
\end{aligned}$$

R is a bisimulation (all other conditions that are to be checked are trivial). The theorem follows by coinduction. □

Theorem 2.6.1 is named after the fundamental theorem from analysis: for all (well-behaved) functions $f\colon \mathbb{R} \to \mathbb{R}$,

$$f(x) = f(0) + \int_0^x f'.$$

The exact correspondence between both theorems will be made precise later, after we have introduced Taylor series, in Definition 3.4.1 below. Theorem 2.6.1 tells us that we can obtain σ out of σ' by multiplication with the constant stream X (and addition with the initial value $\sigma(0)$). Multiplication with the constant stream X can therefore be viewed as a kind of stream integration. We observe that stream calculus has the pleasant property that every stream is both differentiable and integrable.

Next, we show how the Fundamental Theorem can be used to compute the solution of behavioural differential equations in stream calculus in an algebraic manner. We start with linear behavioural differential equations with constant coefficients:

$\sigma^{(n)} + r_{n-1}\sigma^{(n-1)} + \cdots + r_1\sigma^{(1)} + r_0\sigma = \tau$	derivative
$\sigma^{(k)}(0) = c_k, \quad 0 \leq k \leq n-1$	initial value

where $n \geq 1$, $\sigma^{(k)}$ denotes the kth stream derivative of σ, $r_0, \ldots, r_{n-1}$ are real numbers (interpreted as constant streams $[r_k]$), $c_0, \ldots, c_{n-1}$ are also real numbers, and τ is an arbitrary stream. If $\tau = 0$ the equation is called homogeneous, otherwise it is called non-homogeneous. The claim is that

(1) there exists a unique stream σ satisfying the equation;
(2) σ can be expressed in terms of τ, the coefficients $r_0, \ldots, r_{n-1}$, the initial values $c_0, \ldots, c_{n-1}$, the constant stream X, and the operators of plus, product and inverse.

Rather than giving a proof of this claim in its full generality, we prefer to treat a few special cases, being confident that they will bring the main idea across. Let us start with a simple example of a homogeneous equation:

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

.

In order to solve it we multiply the left and right side of the equation with X^2 (where 2 is the coefficient of the highest derivative):

$$(X^2 \times \sigma'') - (X^2 \times \sigma') - (X^2 \times \sigma) = 0.$$

Applying the Fundamental Theorem 2.6.1 to both σ and σ', we obtain, using $\sigma(0) = 0$ and $\sigma'(0) = 1$,

$$\sigma = X \times \sigma', \quad \sigma' = 1 + (X \times \sigma'')$$

implying $X^2 \times \sigma'' = -X + \sigma$ and $X^2 \times \sigma' = X \times \sigma$. Substituting this above gives

$$\begin{aligned} 0 &= (X^2 \times \sigma'') - (X^2 \times \sigma') - (X^2 \times \sigma) \\ &= -X + \sigma - (X \times \sigma) - (X^2 \times \sigma) \\ &= -X + (1 - X - X^2) \times \sigma \end{aligned}$$

yielding as the final outcome the following expression for the solution of the differential equation:

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

One could continue by manipulating this expression still further in order to obtain a so-called *closed formula* for the nth element of σ, expressing $\sigma(n)$ in terms of a formula on the natural numbers depending on (the variable) n. We shall come back to this in Section 2.7.

Here is another example, this time non-homogeneous:

derivative	initial value
$\sigma' - \sigma = (1-X)^{-1}$	$\sigma(0) = 1$

.

By Theorem 2.6.1, we obtain $X \times \sigma' = \sigma - 1$, whence

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

As a consequence, the following expression is obtained:

$$\sigma = \frac{1}{(1-X)^2}. \tag{23}$$

The following table contains some further examples which the reader may wish to use to test his or her stream-calculus abilities:

derivative	initial value
$\sigma' - 5\sigma = -2(1-2X)^{-1}$	$\sigma(0) = 3$
$\tau'' + r^2\tau = 1$	$\tau(0) = 1,\ \tau'(0) = 0$
$\rho'' - \rho' = 2 + 6X$	$\rho(0) = 1,\ \rho'(0) = 0$

.

The solutions of these equations are

$$\sigma = \frac{3-8X}{1-7X+10X^2}, \quad \tau = \frac{1+X+X^2}{1+r^2X^2}, \quad \rho = \frac{1-X+2X^2+6X^3}{1-X}.$$

The fact that the coefficients of the equations so far have been constants is by no means crucial. Here is the more general formulation:

$\sigma^{(n)} + (\rho_{n-1} \times \sigma^{(n-1)}) + \cdots + (\rho_0 \times \sigma) = \tau$	derivative
$\sigma^{(k)}(0) = c_k, 0 \leq k \leq n-1$	initial value

where now not only τ but also $\rho_0, \ldots, \rho_{n-1}$ are arbitrary streams. The claim is again that

(1) there exists a unique stream σ satisfying the equation;
(2) σ can be expressed in terms of the streams τ and $\rho_0, \ldots, \rho_{n-1}$, the initial values $c_0, \ldots, c_{n-1}$, the constant stream X, and the operators of plus, product and inverse.

The proof for arbitrary $n \geq 1$ is not more difficult (just more writing) than the proof for $n = 1$, which is presented next. Consider the equation

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

where ρ and τ are arbitrary streams. Calculating as before (multiplying both sides of the equation by X, invoking Theorem 2.6.1), we obtain for the solution of this equation the following expression:

$$\sigma = \frac{\sigma(0) + (X \times \tau)}{1 + (X \times \rho)}.$$

2.7. Solving Difference Equations

As an application of the techniques of Section 2.6, we next show how they can be used to solve linear difference equations (also called linear recurrence relations) with constant coefficients. The main idea is to transform difference equations, which can be seen as inductive definitions of streams, in a canonical fashion into behavioural differential equations. The heart of the matter is thus a systematic transformation of inductive stream definitions into coinductive ones.

More precisely, we consider non-homogeneous linear difference equations of order k with constant coefficients:

difference equation	initial values
$s_{n+k} + r_{k-1}s_{n+k-1} + \cdots + r_1 s_{n+1} + r_0 s_n = t_n$	$s_0, \ldots, s_{k-1}$

where $k \geq 1$ (the order of the equation), $n \geq 0$, the coefficients $r_0, \ldots, r_{n-1}$ are real numbers (the coefficient for s_{n+k} has been taken identical to 1 for convenience), and $t_0, t_1, t_2, \ldots$ is an arbitrary sequence of real numbers. Defining

$$\sigma = (s_0, s_1, s_2, \ldots), \quad \tau = (t_0, t_1, t_2, \ldots)$$

we set out to transform the above difference equation into a behavioural differential equation in the variable σ, by multiplying both sides of the equation with the constant stream X^n:

$$s_{n+k}X^n + r_{k-1}s_{n+k-1}X^n + \cdots + r_1 s_{n+1}X^n + r_0 s_n X^n = t_n X^n.$$

Next we take on both sides the infinite sum over all $n \geq 0$:

$$\sum_{n=0}^{\infty} s_{n+k}X^n + r_{k-1}\sum_{n=0}^{\infty} s_{n+k-1}X^n + \cdots + r_1 \sum_{n=0}^{\infty} s_{n+1}X^n + r_0 \sum_{n=0}^{\infty} s_n X^n = \sum_{n=0}^{\infty} t_n X^n.$$

Since, for any $i \geq 0$,

$$\sigma^{(i)} = s_i + s_{i+1}X^1 + s_{i+2}X^2 + \cdots$$

which is an immediate corollary of Theorem 2.5.1, we obtain

$$\sigma^{(k)} + r_{k-1}\sigma^{(k-1)} + \cdots + r_1\sigma^{(1)} + r_0\sigma = \tau$$

with initial values

$$\sigma(0) = s_0, \quad \sigma^{(1)}(0) = s_1, \ldots, \sigma^{(k-1)}(0) = s_{k-1}.$$

And so we are done, since we have learned in Section 2.6 how to solve this type of behavioural differential equation.

Let us look at a few examples. Here is maybe the most famous difference equation of all:

difference equation	initial value
$s_{n+2} - s_{n+1} - s_n = 0$	$s_0 = 0,\ s_1 = 1$

defining the Fibonacci numbers. The method above transforms it into

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

which we recognise from Section 2.6, where the solution was shown to be

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

This in principle answers the question as to what stream is defined by the difference equation we started with. A further question that is often raised is how the nth element $\sigma(n)$ looks like, ideally as a function of n. The method of partial fractions, well-known from the theory of generating functions, also works in stream calculus. Defining

$$r_+ = \frac{1+\sqrt{5}}{2}, \quad r_- = \frac{1-\sqrt{5}}{2}$$

which are the roots of $1 - X - X^2 = (1 - r_+X)(1 - r_-X)$, we have

$$\begin{aligned}
\sigma &= \frac{X}{1-X-X^2}, \\
&= \frac{1}{r_+ - r_-}\left(\frac{1}{1-r_+X} - \frac{1}{1-r_-X}\right) \\
&= \frac{1}{\sqrt{5}}\left(\sum_{n=0}^{\infty} r_+^n X^n - \sum_{n=0}^{\infty} r_-^n X^n\right) && \text{[by identity (19)]} \\
&= \sum_{n=0}^{\infty} \frac{1}{\sqrt{5}}(r_+^n - r_-^n)X^n
\end{aligned}$$

giving the familiar answer

$$\sigma(n) = \frac{1}{\sqrt{5}}(r_+^n - r_-^n).$$

Note that the entire game is played here inside the world of stream calculus, without any reference to (generating) functions. For a second example, non-homogeneous this time, consider

difference equation	initial values
$s_{n+1} - s_n = 1$	$s_0 = 1$

.

Our method transforms it into

derivative	initial value
$\sigma' - \sigma = (1-X)^{-1}$	$\sigma(0) = 1$

(using the fact that $(1-X)^{-1} = 1 + X + X^2 + \cdots$, identity (14)). For the solution we look again at Section 2.6:

$$\begin{aligned}
\sigma &= (1-X)^{-2} && \text{[identity (23)]} \\
&= 1 + 2X + 3X^2 + 4X^3 + \cdots && \text{[by identity (18)]} \\
&= (1, 2, 3, 4, \dots)
\end{aligned}$$

which comes as no surprise. Here is yet another example of a non-homogeneous equation:

difference equation	initial value
$s_{n+1} - 5s_n = -2^{n+1}$	$s_0 = 3$

.

Writing $-2^{n+1} = -2 \times 2^n$ and using identity (19), the following differential equation is obtained:

derivative	initial value
$\sigma' - 5\sigma = -2(1-2X)^{-1}$	$\sigma(0) = 3$

.

Using the method of partial fractions again, the solution found in Section 2.6 can be rewritten as

$$\begin{aligned}\sigma &= \frac{3-8X}{1-7X+10X^2} \\ &= \frac{2}{3}(1-2X)^{-1} + \frac{7}{3}(1-5X)^{-1}\end{aligned}$$

yielding, again by identity (19),

$$\sigma(n) = \frac{2}{3}2^n + \frac{7}{3}5^n.$$

2.8. Streams and Weighted Automata

In this section, we introduce the notion of weighted stream automaton, the transition diagrams of which can serve as pleasant graphical representations of streams and their successive derivatives.

An $\mathbb{R}$-*weighted stream automaton*, or *weighted automaton* for short, is a pair $Q = (Q, \langle o, t\rangle)$ consisting of a set Q of states, together with an output function $o\colon Q \to \mathbb{R}$, and a transition function $t\colon Q \to (Q \to_f \mathbb{R})$, where the latter set contains only functions of finite *support*:

$$Q \to_f \mathbb{R} = \{\phi\colon Q \to \mathbb{R} \mid \text{the set } \{q \in Q \mid \phi(q) \neq 0\} \text{ is finite }\}.$$

The output function o assigns to each state q in Q a real number $o(q)$ in $\mathbb{R}$. The transition function t assigns to a state q in Q a function $t(q)\colon Q \to_f \mathbb{R}$, which specifies for any state u in Q a real number $t(q)(u)$ in $\mathbb{R}$. This number can be thought of as the weight (cost, multiplicity, duration, etc.) with which the transition from q to u occurs. The following notation will be used: $q \xrightarrow{r} u$ denotes $t(q)(u) = r$ and $\underline{q}_r$ denotes $o(q) = r$. Moreover, for $t(q)(u) = 1$ we shall often simply write $q \to u$, and we write $\underline{q}$ when $o(q) = 1$, in which case q is called an *output* state. In pictures, we usually include only non-zero output values and only arrows with a non-zero label.

The *stream behaviour* $S(q) \in \mathbb{R}^\omega$ of a state q in a weighted automaton $(Q, \langle o, t\rangle)$, can be defined in two equivalent ways. First, there is the following formula, for any $k \geq 0$:

$$\begin{aligned}(24)\quad S(q)(k) = \sum\{l_1 \times \cdots \times l_k \times l \mid \Sigma q_0, \ldots, q_k\colon \\ q = q_0 \xrightarrow{l_1} \cdots \xrightarrow{l_k} q_k \text{ and } o(q_k) = l\}.\end{aligned}$$

This formula computes the kth element $S(q)(k)$ of the stream $S(q)$ as a weighted count of the number of paths (transition sequences) of length k that start in the state q (and end in a state with a non-zero output). It is precisely this aspect of counting paths that makes weighted automata very suitable for the representation of counting problems in general. This will be the subject of Chapter 4.

At the same time, formula (24) does not yield any compact representations for $S(q)$ and is thereby not very suited for actual reasoning. Here we can benefit from stream calculus as follows. Consider again a state q in a weighted automaton $(Q, \langle o, t\rangle)$, and let $\{q_1, \ldots, q_n\}$ be the set of all states u for which $t(q)(u) \neq 0$. Then the stream behaviour $S(q) \in \mathbb{R}^\omega$ of the state q can also be defined by the following

system of behavioural differential equations:

derivative	initial value
$S(q)' = t(q)(q_1) \times S(q_1) + \cdots + t(q)(q_n) \times S(q_n)$	$S(q)(0) = o(q)$

.

We saw in Section 2.6 how to solve such a system of behavioural differential equations. It follows from the Fundamental Theorem 2.6.1 that it is equivalent to the following system of equations:

$$\boxed{S(q) = o(q) + \big(t(q)(q_1) \times X \times S(q_1)\big) + \cdots + \big(t(q)(q_n) \times X \times S(q_n)\big)} \; .$$

If the automaton (that is, the state space Q) is finite, this is a finite system of equations, which can be solved in the familiar algebraic way.

It is not very difficult to prove the equivalence of the definition of $S(q)$ by formula (24), on the one hand, and the definition of $S(q)$ by means of behavioural differential equations, on the other.

THEOREM 2.8.1. *For a weighted automaton $(Q, \langle o, t\rangle)$ and $q \in Q$, let $S(q)$ be the stream defined by the above system of behavioural differential equations. Then $S(q)$ satisfies identity* (24) *above. That is, for all $k \geq 0$,*

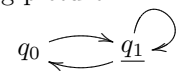
$$S(q)(k) = \Sigma\{l_1 \times \cdots \times l_k \times l \mid \exists q_0, \ldots, q_k \colon q = q_0 \xrightarrow{l_1} \cdots \xrightarrow{l_k} q_k \text{ and } o(q_k) = l\}.$$

PROOF. Using the differential equation for $S(q)$ and the observation that

$$\begin{aligned} S(q)(k+1) &= S(q)^{(k+1)}(0) \\ &= \big(S(q)'\big)^{(k)}(0) \\ &= \big(t(q)(q_1) \times S(q_1) + \cdots + t(q)(q_n) \times S(q_n)\big)^{(k)}(0) \\ &= t(q)(q_1) \times S(q_1)^{(k)}(0) + \cdots + t(q)(q_n) \times S(q_n)^{(k)}(0) \\ &= t(q)(q_1) \times S(q_1)(k) + \cdots + t(q)(q_n) \times S(q_n)(k) \end{aligned}$$

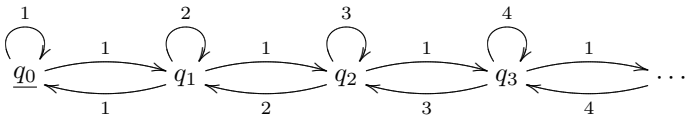
the proof follows by induction on k. □

As an example of the stream behaviour of a weighted automaton, consider the automaton defined by the following picture



(which we shall consider in the counting example of Section 4.1). The stream $S(q_0)$ represented by the state q_0 can be computed in two ways. According to formula (24), one finds $S(q_0)(k)$ by simply counting (since all weights are 1) the number of paths of length k from q_0 to q_1 (which is the only state with a non-zero output), yielding $S(q_0) = (0, 1, 1, 2, 3, 5, 8, \ldots)$. By the second definition, one has $S(q_0) = X \times S(q_1)$, and $S(q_1) = 1 + \big(X \times S(q_0)\big) + \big(X \times S(q_1)\big)$, yielding $S(q_0) = X/1 - X - X^2$.

In the present chapter, we shall only deal with finite weighted automata, but infinite weighted automata, such as



are interesting too. We shall see this and many similar examples of infinite weighted automata in Chapter 4.

2.9. Rational Streams

The streams that can be represented by finite weighted automata are precisely the *rational* streams, which are introduced and characterised in Theorem 2.9.1 below. First we present the following definition. A stream $\sigma \in \mathbb{R}^\omega$ is *polynomial* if there exist $n \geq 0$ and $s_0, \ldots, s_n \in \mathbb{R}$ such that

$$\sigma = s_0 + (s_1 \times X) + (s_2 \times X^2) + \cdots + (s_n \times X^n)$$

or, equivalently, $\sigma = (s_0, s_1, s_2, \ldots, s_n, 0, 0, 0, \ldots)$.

THEOREM 2.9.1. *The following three conditions are equivalent. Any stream σ satisfying them is called* rational:

(1) *There exist polynomial streams π and ρ with $\rho(0) \neq 0$ such that*

$$\sigma = \frac{\pi}{\rho}.$$

(2) *There exist $n \geq 1$, real numbers s_{ij}, for $1 \leq i \leq n$ and $1 \leq j \leq n$, real numbers $k_1, \ldots, k_n$, and streams $\sigma_1, \sigma_2, \ldots, \sigma_n$ with $\sigma = \sigma_1$, satisfying the following finite system of behavioural differential equations*:

derivative	initial value
$(\sigma_1)' = s_{11}\sigma_1 + s_{12}\sigma_2 + \cdots + s_{1n}\sigma_n$	$\sigma_1(0) = k_1$
$(\sigma_2)' = s_{21}\sigma_1 + s_{22}\sigma_2 + \cdots + s_{2n}\sigma_n$	$\sigma_2(0) = k_2$
$\cdots$	$\cdots$
$(\sigma_n)' = s_{n1}\sigma_1 + s_{n2}\sigma_2 + \cdots + s_{nn}\sigma_n$	$\sigma_n(0) = k_n$

(3) *σ has a finite representation (is recognizable): there exist a finite weighted automaton Q and a state $q \in Q$ with $\sigma = S(q)$.*

PROOF. A full proof can be found in [**37**], the main ingredients of which we mention next. For the proof that (1) implies (2), consider polynomials $\pi = p_0 + p_1 X + \cdots + p_n X^n$ and $\rho = r_0 + r_1 X + \cdots + r_m X^m$ and let $\sigma = \pi \times \rho^{-1}$. For notational convenience, we assume that $r_0 = 1$. We also assume that $0 < n < m$ (the case that $m \leq n$ can be dealt with similarly). Using the defining behavioural differential equations of sum, convolution product, and inverse, σ is readily seen to satisfy the following finite system of behavioural differential equations:

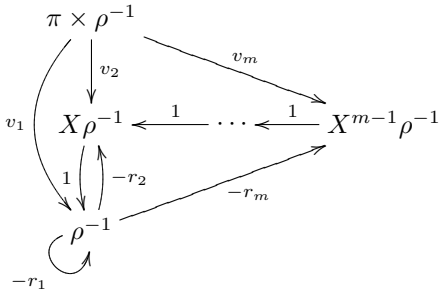
derivative	initial value
$(\pi \times \rho^{-1})' = v_1\rho^{-1} + v_2 X\rho^{-1} + \cdots + v_m X^{m-1}\rho^{-1}$	$(\pi \times \rho^{-1})(0) = p_0$
$(\rho^{-1})' = -r_1\rho^{-1} - r_2 X\rho^{-1} - \cdots - r_m X^{m-1}\rho^{-1}$	$(\rho^{-1})(0) = 1$
$(X\rho^{-1})' = \rho^{-1}$	$(X\rho^{-1})(0) = 0$
$\cdots$	$\cdots$
$(X^{m-1}\rho^{-1})' = X^{m-2}\rho^{-1}$	$(X^{m-1}\rho^{-1})(0) = 0$

where

$$v_i = \begin{cases} p_i - p_0 r_i & \text{if } 1 \leq i \leq n, \\ -p_0 r_i & \text{if } n < i \leq m. \end{cases}$$

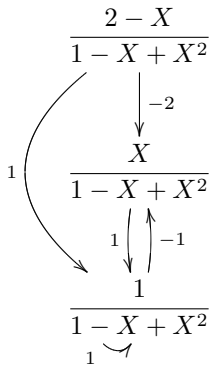
The equivalence between (2) and (3) follows from the correspondence between this finite system of behavioural differential equations and the following finite weighted

automaton for $\pi \times \rho^{-1}$:



with $o(\pi \times \rho^{-1}) = p_0$ and $o(\rho^{-1}) = 1$. □

Note that in the construction of the weighted automaton for $\pi \times \rho^{-1}$ above, we have used the streams themselves ($\pi \times \rho^{-1}$, $X\rho^{-1}$, etc.) as states. The transitions of any stream (state) in this automaton are obtained by "splitting the derivative" of this stream into its "+" components. Here is a concrete instance of the above automaton, taking $\pi = 2 - X$ and $\rho = 1 - X + X^2$:



In this automaton, the transitions of for instance the lower state are determined by the fact that its derivative is a sum of two streams, as follows:

$$\left(\frac{1}{1 - X + X^2}\right)' = -\frac{X}{1 - X + X^2} + \frac{1}{1 - X + X^2}.$$

2.10. Streams and Languages

Recall from Section 1.6 that stream automata are coalgebras of the functor $\mathbb{R} \times (-)$, which is a special instance of the more general functor $A \times (-)^B$. We also saw that $2 \times (-)^B$ is another special instance of the latter functor, with final coalgebra $2^{B^*} = \{L \mid L \subseteq B^*\}$, the set of all languages over B. Next, we briefly illustrate that much of the present chapter also applies to languages, by a straightforward generalisation as follows. First, we spell out the final coalgebra structure $(2^{B^*}, \langle o, t \rangle)$ on 2^{B^*}. It consists of functions $o\colon 2^{B^*} \to 2$ and $t\colon 2^{B^*} \to (2^{B^*})^B$, defined, for all $L \subseteq B^*$ and $b \in B$, as

$$o(L) = \begin{cases} 1 & \text{if } \varepsilon \in L, \\ 0 & \text{if } \varepsilon \notin L \end{cases}, \quad t(L)(b) = L_b.$$

Here ε denotes the empty word, and L_b denotes the *input derivative* (or b-derivative) of L, which we define as

$$L_b = \{w \in B^* \mid b \cdot w \in L\}$$

($b \cdot w$ denotes the word obtained by prefixing w with b). The value $o(L)$, for which we shall write $L(0)$, corresponds to the initial value $\sigma(0)$ of a stream σ. The input derivative L_b generalises the stream derivative σ'. Using these operations, one can define (operators on) languages by means of behavioural differential equations, just as we did with streams. For instance, the following system of equations, which is a minor variation of the corresponding system for streams, defines the operations of sum (union) and product (concatenation) of languages $K, L \subseteq B^*$:

derivative	initial value
$(K + L)_b = K_b + L_b$	$(K + L)(0) = \max\{K(0), L(0)\}$
$(K \times L)_b = (K_b \times L) + \big(L(0) \times L_b\big)$	$(K \times L)(0) = \min\{K(0), L(0)\}$

(Note that $\big(L(0) \times L_b\big) = 0$, the empty set, if $L(0) = 0$; if $L(0) = 1$ then $\big(L(0) \times L_b\big) = L_b\big)$. These equations define the familiar operations of union and concatenation since, for instance,

$$K \times L = \{v \cdot w \mid v \in K, w \in L\}.$$

As with streams, the advantage of the use of behavioural differential equations is that they allow us to reason about languages by coinduction.

More generally still, all of this applies also to formal power series in many non-commutative variables and with coefficients in any semi-ring, which we also mentioned in Section 1.6. Details and many examples of definitions and proofs by coinduction for languages and formal power series, can be found in [**37**].

2.11. Discussion

The material of the present chapter is taken from [**39**]. Motivating sources of examples of streams and stream operators have been the books [**22, 47**], and the papers by McIlroy [**28, 29**] and Karczmarczuk [**24, 25**]. The solution of difference equations by means of stream calculus is conceptually very simple, since the entire game is played within the world of streams. In contrast to the classical technique of generating functions (as in [**22, 47**]), functions (from $\mathbb{R}$ to $\mathbb{R}$) are just not needed in stream calculus, and convergence issues thereby simply do not enter the picture. As we have seen, streams can be viewed as formal power series (in one variable), which are often used as a formal alternative to generating functions, precisely to avoid convergence considerations. We see some advantages of stream calculus also over the use of formal power series. First, there is the rigorous use of coinduction, both in definitions and in proofs, which makes stream calculus possibly more formal than the use of "formal" power series usually is. Notably this applies to the use of the operation of inverse, which is not always treated strictly formally within theories of power series. Secondly, stream calculus is very expressive, as we shall see in Chapter 3, where additional operators (including shuffle product and shuffle inverse) will be introduced. The notion of input derivative of a language, which we saw in Section 2.10 goes back to [**14**]. It plays a role also in [**16**], where the chapter "The differential calculus of events" already suggests a connection with classical calculus. The notion of bisimulation-up-to, in Definition 2.3.1, is a variation on a similar notion by Milner [**32**] (see also [**41**]). In [**9**], variations and coalgebraic generalisations

of coinductive proof methods are given. The present notion of bisimulation-up-to (identity and sum), can be easily generalised along the lines of [**9**], to a version which would allow derivatives to be bisimilar up to arbitrary contexts (including product, inverse, and the other operators). For a general reference on (rational) formal power series, see [**11**]. Theorem 2.9.1 is a classical result. New about our use of weighted automata as representations for streams are: -the coinductive definition of their behaviour; -the way such automata are constructed by means of splitting derivatives (as in the proof of Theorem 2.9.1); -and our use of infinite weighted automata (in Chapter 4).

CHAPTER 3

Analytical Differential Equations

The power of stream calculus is further increased by the introduction of three new operators: shuffle product, shuffle inverse, and stream exponentiation. These operators and specifically their interplay with the ordinary (convolution) product and inverse, are interesting in themselves. They are also useful in applications, as we shall illustrate by the solution of analytical differential equations.

3.1. Shuffle Product and Shuffle Inverse

For streams σ and τ, let the *shuffle product* $\sigma \otimes \tau$ and the *shuffle inverse* $\sigma^{-\underline{1}}$ be the unique streams satisfying the following behavioural differential equations:

derivative	initial value
$(\sigma \otimes \tau)' = (\sigma' \otimes \tau) + (\sigma \otimes \tau')$	$(\sigma \otimes \tau)(0) = \sigma(0) \times \tau(0)$
$(\sigma^{-\underline{1}})' = -\sigma' \otimes (\sigma^{-\underline{1}} \otimes \sigma^{-\underline{1}})$	$\sigma^{-\underline{1}}(0) = \sigma(0)^{-1}$

.

(The shuffle inverse is defined only when $\sigma(0) \neq 0$.) We use an underlined symbol $\underline{1}$ to distinguish shuffle inverse from the inverse to the convolution product. As we observed earlier, stream differentiation of (convolution) product and inverse does not behave the way we are used to from analysis. A somewhat formalistic way of motivating the definition of shuffle product and inverse is that they constitute, in this respect, more familiarly behaved alternatives to product and inverse. Another motivation will follow in Section 3.4, where we shall see that the Taylor series of the (pointwise) product of two real-valued functions is equal to the shuffle product of their Taylor series. Finally, the *name* shuffle product is chosen because this operator is the stream variant of the shuffle (or merge) operation on languages.

Similar to formula (6) for the convolution product, there is the following formula for $(\sigma \otimes \tau)(n)$, for any $n \geq 0$,

$$(\sigma \otimes \tau)(n) = \sum_{k=0}^{n} \binom{n}{k} \times \sigma(n-k) \times \tau(k). \tag{25}$$

This formula could also have been taken as an alternative definition to the system of behavioural differential equations above. Reasoning in terms of (25), however, is again unnecessarily complicated, because of the use of the indices, and the occurrence of the binomial coefficient. And as with ordinary inverse, no similar such formula for shuffle inverse is known.

We shall use the following conventions, for all $n \geq 0$:

$$\sigma^{\underline{0}} \equiv 1, \quad \sigma^{\underline{n+1}} \equiv \sigma \otimes \sigma^{\underline{n}}, \quad \sigma^{-\underline{n}} \equiv (\sigma^{-\underline{1}})^{\underline{n}}.$$

The following theorem contains some basic properties of shuffle product and shuffle inverse.

THEOREM 3.1.1. *For all streams $\sigma = (s_0, s_1, s_2, \dots)$, τ and ρ, for all $r \in \mathbb{R}$, and for all $n \geq 0$,*

$$\sigma \otimes \tau = \tau \otimes \sigma, \tag{26}$$

$$r \otimes \sigma = r \times \sigma, \tag{27}$$

$$0 \otimes \sigma = 0, \tag{28}$$

$$1 \otimes \sigma = \sigma, \tag{29}$$

$$\sigma \otimes (\tau \otimes \rho) = (\sigma \otimes \tau) \otimes \rho, \tag{30}$$

$$\sigma \otimes (\tau + \rho) = (\sigma \otimes \tau) + (\sigma \otimes \rho), \tag{31}$$

$$(\sigma^{\underline{n+1}})' = (n+1) \otimes \sigma' \otimes \sigma^{\underline{n}}, \tag{32}$$

$$X^{\underline{n}} = n! \times X^n, \tag{33}$$

$$\sigma \otimes \sigma^{-\underline{1}} = 1, \tag{34}$$

$$(\sigma^{-\underline{1}})^{-\underline{1}} = \sigma, \tag{35}$$

$$(\sigma \otimes \tau)^{-\underline{1}} = \sigma^{-\underline{1}} \otimes \tau^{-\underline{1}}, \tag{36}$$

$$X \otimes \sigma = s_0 X^1 + 2s_1 X^2 + 3s_2 X^3 + \cdots = (0, s_0, 2s_1, 3s_2, \dots), \tag{37}$$

$$(X \otimes \sigma')' = s_1 + 2s_2 X^1 + 3s_3 X^2 + \cdots = (s_1, 2s_2, 3s_3, \dots), \tag{38}$$

$$(1 - X)^{-\underline{1}} = 1 + 1!X + 2!X^2 + 3!X^3 \cdots . \tag{39}$$

PROOF. We treat a few examples. For (26), define

$$R = \{\angle\sigma \otimes \tau, \tau \otimes \sigma\rangle \mid \sigma, \tau \in \mathbb{R}^\omega\}$$

and observe that

$$\begin{aligned}(\sigma \otimes \tau)' &= (\sigma' \otimes \tau) + (\sigma \otimes \tau') \\ &\Sigma R\ (\tau \otimes \sigma') + (\tau' \otimes \sigma) \\ &= (\tau' \otimes \sigma) + (\tau \otimes \sigma') \\ &= (\tau \otimes \sigma)'.\end{aligned}$$

Identity (26) now follows by coinduction-up-to (Theorem 2.3.2). Identity (32) can be proven by induction on n. For (34) define

$$Q = \{\langle \rho \otimes (\sigma \otimes \sigma^{-\underline{1}}), \rho\rangle \mid \rho, \sigma \in \mathbb{R}^\omega\}.$$

Computing derivatives gives

$$\begin{aligned}\big(\rho \otimes (\sigma \otimes \sigma^{-\underline{1}})\big)' &= \rho' \otimes (\sigma \otimes \sigma^{-\underline{1}}) + \rho \otimes \big((\sigma' \otimes \sigma^{-\underline{1}}) + \big(\sigma \otimes (-\sigma' \otimes \sigma^{-\underline{1}} \otimes \sigma^{-\underline{1}})\big)\big) \\ &= \rho' \otimes \big((\sigma \otimes \sigma^{-\underline{1}}) + \rho \otimes (\sigma' \otimes \sigma^{-\underline{1}}) - \big(\rho \otimes (\sigma' \otimes \sigma^{-\underline{1}})\big) \otimes (\sigma \otimes \sigma^{-\underline{1}})\big) \\ &\Sigma Q\ \rho' + \rho \otimes (\sigma' \otimes \sigma^{-\underline{1}}) - \rho \otimes (\sigma' \otimes \sigma^{-\underline{1}}) \\ &= \rho'\end{aligned}$$

which proves that Q is a bisimulation-up-to, allowing us again to apply coinduction-up-to. For identity (37), let

$$T = \{\langle X \otimes \sigma, s_0 X^1 + 2s_1 X^2 + 3s_2 X^3 + \cdots\rangle \mid \sigma = (s_0, s_1, s_2, \dots) \in \mathbb{R}^\omega\}$$

and note that

$$\begin{aligned}(X \otimes \sigma)' &= \sigma + (X \otimes \sigma') \\ &\quad \Sigma T\ \sigma + (s_1 X^1 + 2s_2 X^2 + 3s_3 X^3 + \cdots) \\ &\quad = (s_0 + s_1 X^1 + s_2 X^2 + \cdots) + (s_1 X^1 + 2s_2 X^2 + 3s_3 X^3 + \cdots) \\ &\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{[by Theorem 2.5.1]} \\ &\quad = s_0 + 2s_1 X^1 + 3s_2 X^2 + 4s_3 X^3 + \cdots \\ &\quad = (s_0 X + 2s_1 X^2 + 3s_2 X^3 + 4s_3 X^4 + \cdots)'\end{aligned}$$

showing that T is a bisimulation-up-to. Finally for (39), observe that

$$\{\langle n!(1-X)^{-\underline{(n+1)}}, n! + (n+1)!X + (n+2)!X^2 + \cdots \rangle \mid n \geq 0\}$$

is a bisimulation relation on $\mathbb{R}^\omega$. □

3.2. Stream Exponentiation

We next introduce the operation of *stream exponentiation*. For a stream σ let $exp(\sigma)$ be the unique stream satisfying the following behavioural differential equation:

derivative	initial value
$\exp(\sigma)' = \sigma' \otimes \exp(\sigma)$	$\exp(\sigma)(0) = e^{\sigma(0)}$

where $e^{\sigma(0)}$ is the analytical function e^x applied to the real number $\sigma(0)$. The theorem below contains a number of more and less familiar identities for stream exponentiation.

THEOREM 3.2.1. *For all $r, s \in \mathbb{R}$, $\sigma, \tau \in \mathbb{R}^\omega$, and $n \geq 0$,*

$$\exp(rX) = \frac{1}{1-rX}, \tag{40}$$

$$\exp(\sigma) = 1 + \frac{\sigma^{\underline{1}}}{1!} + \frac{\sigma^{\underline{2}}}{2!} + \cdots, \tag{41}$$

$$\exp(\sigma) \otimes \exp(\tau) = \exp(\sigma + \tau), \tag{42}$$

$$\exp(-\sigma) = \exp(\sigma)^{-\underline{1}}, \tag{43}$$

$$\frac{1}{1-rX} \otimes \frac{1}{1-sX} = \frac{1}{1-(r+s)X}. \tag{44}$$

PROOF. Again we treat only a few examples. For (40) consider $R = \{\langle s \times \exp(rX), s/1 - rX \rangle \mid r, s \in \mathbb{R}\}$. It is a bisimulation relation, since

$$\begin{aligned}\big(s \times \exp(rX)\big)' &= (r \times s) \times \exp(rX) \\ &\quad R\ \frac{r \times s}{1-rX} \\ &\quad = \left(\frac{s}{1-rX}\right)'.\end{aligned}$$

For identity (43) let $Q = \{\langle \tau \otimes \exp(-\sigma), \tau \otimes \exp(\sigma)^{-\underline{1}} \rangle \mid \sigma, \tau \in \mathbb{R}^\omega\}$ and compute as follows:

$$\begin{aligned}
\big(\tau \otimes \exp(-\sigma)\big)' &= \big(\tau' - (\tau \otimes \sigma')\big) \otimes \exp(-\sigma) \\
&\quad Q\ \big(\tau' - (\tau \otimes \sigma')\big) \otimes \exp(\sigma)^{-\underline{1}} \\
&= \big(\tau' \otimes \exp(\sigma)^{-\underline{1}}\big) - \big(\tau \otimes \sigma' \otimes \exp(\sigma)^{-\underline{1}}\big) \\
&= \big(\tau' \otimes \exp(\sigma)^{-\underline{1}}\big) - \big(\tau \otimes \sigma' \otimes \exp(\sigma) \otimes \exp(\sigma)^{-\underline{1}} \otimes \exp(\sigma)^{-\underline{1}}\big) \\
&= \big(\tau' \otimes \exp(\sigma)^{-\underline{1}}\big) + \big(\tau \otimes \big(-\sigma' \otimes \exp(\sigma) \otimes \exp(\sigma)^{-\underline{1}} \otimes \exp(\sigma)^{-\underline{1}}\big)\big) \\
&= \big(\tau' \otimes \exp(\sigma)^{-\underline{1}}\big) + \big(\tau \otimes \big(\exp(\sigma)^{-\underline{1}}\big)'\big) \\
&= \big(\tau \otimes \exp(\sigma)^{-\underline{1}}\big)'
\end{aligned}$$

which proves that Q is a bisimulation. Identity (44) follows from identities (40) and (42). □

Identity (40) is somewhat surprising. Also identity (44) is a bit unusual, but note that it involves (ordinary) inverse combined with shuffle product.

3.3. Comparing Convolution and Shuffle Product

We investigate in some detail the relation between the two types of product and inverse, for which we shall make use of a new type of stream derivation.

Let the *analytical* stream derivative of a stream σ be defined by

$$\frac{d}{dX}(\sigma) = (X \otimes \sigma')'. \tag{45}$$

As usual, we shall also write

$$\frac{d\sigma}{dX} \equiv \frac{d}{dX}(\sigma).$$

There is the following general formula for our new type of derivation:

$$\frac{d}{dX}(s_0 + s_1X^1 + s_2X^2 + \cdots) = s_1 + 2s_2X^1 + 3s_3X^2 + \cdots \tag{46}$$

which we recognise as identity (38), and which explains the name of analytical stream derivation. Analytical stream derivation behaves for (convolution) product and inverse in the, from analysis, familiar way.

THEOREM 3.3.1. *For all $\sigma, \tau \in \mathbb{R}^\omega$ and $n \geq 0$,*

$$\frac{d(\sigma \times \tau)}{dX} = \left(\frac{d\sigma}{dX} \times \tau\right) + \left(\sigma \times \frac{d\tau}{dX}\right), \tag{47}$$

$$\frac{d\sigma^{-1}}{dX} = -\frac{d\sigma}{dX} \times \sigma^{-1} \times \sigma^{-1}, \tag{48}$$

$$\frac{d\sigma^{n+1}}{dX} = (n+1) \times \frac{d\sigma}{dX} \times \sigma^n. \tag{49}$$

PROOF. The proof of identity (47) uses the following equalities (the proofs of these are left to the reader). For all summable families $\{\sigma_n\}_{n=0}^\infty$ and all τ in $\mathbb{R}^\omega$,

for all $k \geq 0$,

$$\frac{d(\sum_{n=0}^{\infty} \sigma_n)}{dX} = \sum_{n=0}^{\infty} \frac{d\sigma_n}{dX},$$
$$\frac{d(X^k \times \tau)}{dX} = \left(\frac{dX^k}{dX} \times \tau\right) + \left(X^k \times \frac{d\tau}{dX}\right),$$
$$\frac{dX^{k+1}}{dX} = (k+1) \times X^k.$$

Next consider $\sigma = (s_0, s_1, s_2, \dots)$ and τ in $\mathbb{R}^\omega$ and note that

$$\begin{aligned}
\frac{d(\sigma \times \tau)}{dX} &= \frac{d((\sum_{n=0}^{\infty} s_n \times X^n) \times \tau)}{dX} && \text{[Theorem 2.5.1]} \\
&= \frac{d(\sum_{n=0}^{\infty} s_n \times (X^n \times \tau))}{dX} \\
&= \sum_{n=0}^{\infty} \frac{d(s_n \times (X^n \times \tau))}{dX} \\
&= \sum_{n=0}^{\infty} s_n \times \frac{d(X^n \times \tau)}{dX} \\
&= \sum_{n=0}^{\infty} s_n \times \left(\frac{dX^n}{dX} \times \tau + X^n \times \frac{d\tau}{dX}\right) \\
&= \sum_{n=0}^{\infty} s_n \times \left(\frac{dX^n}{dX} \times \tau\right) + \sum_{n=0}^{\infty} s_n \times \left(X^n \times \frac{d\tau}{dX}\right) \\
&= \sum_{n=1}^{\infty} (n \times s_n \times X^{n-1} \times \tau) + \sum_{n=0}^{\infty} s_n \times \left(X^n \times \frac{d\tau}{dX}\right) \\
&= \left(\sum_{n=1}^{\infty} n \times s_n \times X^{n-1}\right) \times \tau + \left(\sum_{n=0}^{\infty} s_n \times X^n\right) \times \frac{d\tau}{dX} \\
&= \left(\frac{d\sigma}{dX} \times \tau\right) + \left(\sigma \times \frac{d\tau}{dX}\right) && \text{[identity (46) and Theorem 2.5.1].}
\end{aligned}$$

Identities (48) and (49) easily follow from (47). □

Note that properties (47)–(49) reflect precisely how ordinary stream derivation behaves on shuffle product and shuffle inverse, which satisfy

$$(\sigma \otimes \tau)' = (\sigma' \otimes \tau) + (\sigma \otimes \tau'),$$
$$(\sigma^{\underline{-1}})' = -\sigma' \otimes (\sigma^{\underline{-1}} \otimes \sigma^{\underline{-1}}),$$
$$(\sigma^{\underline{n+1}})' = (n+1) \otimes \sigma' \otimes \sigma^{\underline{n}}.$$

Analytical stream derivation is used in the following definition of a new operator on streams:

derivative	initial value
$\Lambda_C(\sigma)' = \Lambda_C\left(\frac{d\sigma}{dX}\right)$	$\Lambda_C(\sigma)(0) = \sigma(0)$

.

One can easily prove that Λ_C transforms a stream $(s_0, s_1, s_2, s_3, \dots)$ into $(0!s_0, 1!s_1, 2!s_2, 3!s_3, \dots)$, or, equivalently,

$$\Lambda_C\left(s_0 + \frac{s_1}{1!}X + \frac{s_2}{2!}X^2 + \frac{s_3}{3!}X^3 + \cdots\right) = s_0 + s_1X + s_2X^2 + s_3X^3 + \cdots .$$

In combinatorics, this is referred to as the *Laplace–Carson* transform (cf. [**10**, p. 350] and [**15**, p. 48]), hence our notation. The following theorem shows that with Λ_C, we can relate the two types of product and inverse.

THEOREM 3.3.2. *For all $r \in \mathbb{R}$, $\sigma, \tau \in \mathbb{R}^\omega$,*

$$\begin{aligned}
\Lambda_C\left(\frac{d\sigma}{dX}\right) &= \Lambda_C(\sigma)',\\
\Lambda_C(r) &= r,\\
\Lambda_C(X) &= X,\\
\Lambda_C(\sigma + \tau) &= \Lambda_C(\sigma) + \Lambda_C(\tau),\\
\Lambda_C(\sigma \times \tau) &= \Lambda_C(\sigma) \otimes \Lambda_C(\tau),\\
\Lambda_C(\sigma^{-1}) &= \Lambda_C(\sigma)^{-\underline{1}}.
\end{aligned}$$

PROOF. The first equality is by definition and the next two are trivial. For the latter three, let $R \subseteq \mathbb{R}^\omega \times \mathbb{R}^\omega$ be the smallest relation such that

(1) $\{\langle \sigma, \sigma\rangle \mid \sigma \in \mathbb{R}^\omega\} \subseteq R$,
(2) for all $\sigma_1, \sigma_2, \tau_1, \tau_2 \in \mathbb{R}^\omega$, if $\langle \Lambda_C(\sigma_1), \tau_1\rangle \in R$ and $\langle \Lambda_C(\sigma_2), \tau_2\rangle \in R$ then
    (a) $\langle \Lambda_C(\sigma_1 + \sigma_2), \tau_1 + \tau_2\rangle \in R$,
    (b) $\langle \Lambda_C(\sigma_1 \times \sigma_2), \tau_1 \otimes \tau_2\rangle \in R$,
    (c) $\langle \Lambda_C\big((\sigma_1)^{-1}\big), (\tau_1)^{-\underline{1}}\rangle \in R$.

Then R is a bisimulation on $\mathbb{R}^\omega$ and the result follows by coinduction. □

Thus the operator Λ_C allows one to switch between the two different ring structures on $\mathbb{R}^\omega$ that are determined by convolution product and shuffle product, each of which comes along with its own type of (partially defined) operation of inverse, and its own type of derivative:

$$\Lambda_C \colon \left\langle \mathbb{R}^\omega, +, \times, (-)^{-1}, \frac{d(-)}{dX} \right\rangle \to \langle \mathbb{R}^\omega, +, \otimes, (-)^{-\underline{1}}, (-)'\rangle.$$

Let us emphasize, however, that for the stream calculus we are developing, it is of crucial importance to have both structures present at the same time. Notably the interplay between the various operators from both worlds, turns out to constitute the most interesting part of the calculus. The following identities clearly illustrate this point, since they involve both the shuffle product and the (convolution) inverse. They all have in common that they provide a way of eliminating the occurrence of the shuffle product, a procedure we shall sometimes refer to as *shuffle elimination*.

THEOREM 3.3.3. *For all $r \in \mathbb{R}$ and $\sigma \in \mathbb{R}^\omega$:*

$$1 = \frac{1}{1+rX} \otimes \frac{1}{1-rX}, \tag{50}$$

$$X \otimes \sigma = \left(X^2 \times \frac{d\sigma}{dX}\right) + (X \times \sigma), \tag{51}$$

$$\left(\frac{1}{1-rX}\right) \otimes \sigma = \left(\frac{1}{1-rX}\right) \times \left(\sigma \circ \frac{X}{1-rX}\right) \tag{52}$$

$$= \frac{s_0}{1-rX} + \frac{s_1 X}{(1-rX)^2} + \frac{s_2 X^2}{(1-rX)^3} + \cdots. \tag{53}$$

PROOF. Identity (50) follows from (44). For (51), observe that

$$\begin{aligned}
\left(X^2 \times \frac{d\sigma}{dX}\right) + (X \times \sigma) &= \big(X^2 \times (X \otimes \sigma')'\big) + (X \times \sigma) \\
&= \big(X \times \big(X \times (X \otimes \sigma')'\big)\big) + (X \times \sigma) \\
&= \big(X \times (X \otimes \sigma')\big) + (X \times \sigma) \\
&\qquad\qquad \text{[Fundamental Theorem 2.6.1 } (X \otimes \sigma')(0) = 0\text{]} \\
&= X \times \big((X \otimes \sigma') + \sigma\big) \\
&= X \times \big((X \otimes \sigma)'\big) \\
&= X \otimes \sigma \quad \text{[Fundamental Theorem 2.6.1 } (X \otimes \sigma)(0) = 0\text{]}.
\end{aligned}$$

And for (52), define

$$R = \left\{ \left\langle \left(\frac{s}{1-rX}\right) \otimes \sigma, \left(\frac{s}{1-rX}\right) \times \left(\sigma \circ \frac{X}{1-rX}\right) \right\rangle \;\middle|\; r, s \in \mathbb{R}, \sigma \in \mathbb{R}^\omega \right\}$$

and compute as follows to see that R is a bisimulation-up-to:

$$\begin{aligned}
\left(\left(\frac{s}{1-rX}\right) \otimes \sigma\right)' &= \left(\frac{r \times s}{1-rX}\right) \otimes \sigma + \left(\frac{s}{1-rX}\right) \otimes \sigma' \\
&\quad \Sigma R \left(\frac{r \times s}{1-rX}\right) \times \left(\sigma \circ \frac{X}{1-rX}\right) + \left(\frac{s}{1-rX}\right) \times \left(\sigma' \circ \frac{X}{1-rX}\right) \\
&= \left(\frac{s}{1-rX}\right)' \times \left(\sigma \circ \frac{X}{1-rX}\right) + \left(s \times \left(\sigma \circ \frac{X}{1-rX}\right)'\right) \\
&= \left(\left(\frac{s}{1-rX}\right) \times \left(\sigma \circ \frac{X}{1-rX}\right)\right)'.
\end{aligned}$$

Identity (52) then follows by coinduction-up-to. Identity (53) follows from (52) and Theorem 2.5.3. □

To illustrate identity (51), we compute

$$\begin{aligned}
X \otimes \frac{1}{1+X^2} &= X^2 \times \frac{d}{dX}\left(\frac{1}{1+X^2}\right) + \left(X \times \frac{1}{1+X^2}\right) \\
&= X^2 \times \frac{-2X}{(1+X^2)^2} + \frac{X}{1+X^2} \qquad \text{[identity (48)]} \\
&= \frac{X - X^3}{(1+X^2)^2}.
\end{aligned}$$

Similar computations give

$$X \otimes \frac{X}{1+X^2} = \frac{2X^2}{(1+X^2)^2},$$
$$X^2 \otimes \frac{1}{1+X^2} = \frac{X^2 - 3X^4}{(1+X^2)^3},$$
$$X^2 \otimes \frac{X}{1+X^2} = \frac{-3X^3 + X^5}{(1+X^2)^3}.$$

The following two identities are useful special cases of (53):

$$X^n \otimes \frac{1}{1-rX} = \frac{X^n}{(1-rX)^{n+1}}, \tag{54}$$

$$(1+rX)^n \otimes \frac{1}{1-rX} = \frac{1}{(1-rX)^{n+1}}. \tag{55}$$

3.4. Solving Analytical Differential Equations

A classical technique in analysis for the solution of differential equations defining analytical functions, is the method of undetermined coefficients (cf. [**12**, p. 82]). The idea is quickly explained by means of an example. In order to solve the differential equation

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

one assumes the solution to be of the shape

$$f(x) = s_0 + \frac{s_1}{1!}x + \frac{s_2}{2!}x^2 + \cdots.$$

Computing f'' gives

$$f''(x) = s_2 + \frac{s_3}{1!}x + \frac{s_4}{2!}x^2 + \cdots.$$

Substituting the expressions for f and f'' in the differential equation, one obtains the following difference equation for the coefficients $(s_0, s_1, s_2, \dots)$ of f:

derivative	initial value
$s_{n+2} + s_n = 0$	$s_0 = 0,\ s_1 = 1$

.

Thus this method reduces the problem of solving a differential equation for f to the problem of solving a difference equation for the stream $(s_0, s_1, s_2, \dots)$ of Taylor coefficients of f. Though conceptually very simple, the method of undetermined coefficients has two major drawbacks. Firstly, more interesting differential equations quickly lead to very complicated difference equations. Secondly, there is no general technique for translating the solution of the difference equation (if found at all) back into a (closed) expression for f. Here we shall present a variant of the method above, which in many applications is free of these restrictions. Let

$$\mathcal{A} = \{ f \colon \mathbb{R} \to \mathbb{R} \mid \ f \text{ is analytic in (a neighbourhood of) } 0 \ \}.$$

The details of the definition of analytic function are not important here. It is for our present purposes sufficient to know that analytical functions are differentiable (in a neighbourhood of 0) and that their derivative is again analytic. These properties are used in the definition of the function $\mathcal{T} \colon \mathcal{A} \to \mathbb{R}^\omega$, which sends an analytic function f to its Taylor series $\mathcal{T}(f) = \big(f(0), f'(0), f''(0), \dots\big)$. The formal definition is given, as always, by means of behavioural differential equations.

DEFINITION 3.4.1. Let $\mathcal{T}(f)$ be defined by the following system of behavioural differential equations (one for each $f \in \mathcal{A}$):

derivative	initial value
$\mathcal{T}(f)' = \mathcal{T}(f')$	$\mathcal{T}(f)(0) = f(0)$

.

(Note that the first occurrence of $'$ in $\mathcal{T}(f)' = \mathcal{T}(f')$ stands for stream derivation, whereas the second denotes analytical function derivation.) We shall sometimes refer to the stream $\mathcal{T}(f)$ as the *Taylor transform* of f.

Our method is characterised by the following three steps:

(1) The function $\mathcal{T}$ is used to transform, in a systematic fashion, a differential equation for f into a behavioural differential equation for the stream $\mathcal{T}(f)$ of Taylor coefficients of f.
(2) The behavioural differential equation is solved in stream calculus by means of the techniques of Section 2.6.
(3) The resulting solution is translated back in a systematic manner into an expression for f.

First of all, the following theorem expresses that the function $\mathcal{T}$ transforms functions into their Taylor series in a systematic manner, indeed. For functions f and g, we use the following familiar definitions: for all $x \in \mathbb{R}$,

$$(f+g)(x) = f(x) + g(x), \quad (f \cdot g)(x) = f(x) \times g(x),$$
$$f^{-1}(x) = f(x)^{-1}, \quad e^f(x) = e^{f(x)}.$$

THEOREM 3.4.2. *For all analytic functions $f, g \in \mathcal{A}$,*

$$\mathcal{T}(f)(0) = f(0), \tag{56}$$
$$\mathcal{T}(f') = \mathcal{T}(f)', \tag{57}$$
$$\mathcal{T}\left(\int_0^x f(t)\,dt\right) = X \times \mathcal{T}(f), \tag{58}$$
$$\mathcal{T}(f+g) = \mathcal{T}(f) + \mathcal{T}(g), \tag{59}$$
$$\mathcal{T}(f \cdot g) = \mathcal{T}(f) \otimes \mathcal{T}(g), \tag{60}$$
$$\mathcal{T}(f^{-1}) = \mathcal{T}(f)^{-\underline{1}}, \tag{61}$$
$$\mathcal{T}(e^f) = \exp\big(\mathcal{T}(f)\big). \tag{62}$$

PROOF. Identities (56) and (57) are immediate by the definition of $\mathcal{T}$. Identity (58) follows by coinduction, using the fundamental theorem from analysis $\big((\int_0^x f)' = f\big)$. For the others, define $R \subseteq \mathbb{R}^\omega \times \mathbb{R}^\omega$ as the smallest relation on streams such that $\langle \mathcal{T}(f), \mathcal{T}(f)\rangle \in R$, for all $f \in \mathcal{A}$, and such that if $\langle \mathcal{T}(f_1), \mathcal{T}(f_2)\rangle \in R$ and $\langle \mathcal{T}(g_1), \mathcal{T}(g_2)\rangle \in R$ then

(1) $\langle \mathcal{T}(f_1 + g_1), \mathcal{T}(f_2) + \mathcal{T}(g_2)\rangle \in R$,
(2) $\langle \mathcal{T}(f_1 \cdot g_1), \mathcal{T}(f_2) \otimes \mathcal{T}(g_2)\rangle \in R$,
(3) $\langle \mathcal{T}(f_1^{-1}), \mathcal{T}(f_2)^{-\underline{1}}\rangle \in R$,
(4) $\big\langle \mathcal{T}(e^{f_1}), \exp\big(\mathcal{T}(f_2)\big)\big\rangle \in R$.

The relation R can be shown to be a bisimulation. For instance, consider $\langle \mathcal{T}(f \cdot g), \mathcal{T}(f) \otimes \mathcal{T}(g)\rangle \in R$. Both streams clearly have the same initial value.

And $\langle \mathcal{T}(f\cdot g)', \big(\mathcal{T}(f)\otimes\mathcal{T}(g)\big)'\rangle \in R$, since

$$\begin{aligned}\mathcal{T}(f\cdot g)' &= \mathcal{T}\big((f\cdot g)'\big)\\ &= \mathcal{T}(f'\cdot g + f\cdot g')\\ &R\ \mathcal{T}(f')\otimes\mathcal{T}(g) + \mathcal{T}(f)\otimes\mathcal{T}(g')\\ &= \mathcal{T}(f)'\otimes\mathcal{T}(g) + \mathcal{T}(f)\otimes\mathcal{T}(g)'\\ &= \big(\mathcal{T}(f)\otimes\mathcal{T}(g)\big)'.\end{aligned}$$

Similarly for the other elements of R. Now the theorem follows by coinduction. □

The following identities on the Taylor transforms of some well known functions will be useful when solving differential equations.

THEOREM 3.4.3. *For all $r\in\mathbb{R}$, $n\geq 0$, $f\in\mathcal{A}$,*

$$\mathcal{T}(r) = r, \tag{63}$$

$$\mathcal{T}(x^n) = n!X^n, \tag{64}$$

$$\mathcal{T}(e^{rx}) = \frac{1}{1-rX}, \tag{65}$$

$$\mathcal{T}\big(\sin(rx)\big) = \frac{rX}{1+r^2X^2}, \tag{66}$$

$$\mathcal{T}\big(\cos(rx)\big) = \frac{1}{1+r^2X^2}, \tag{67}$$

$$\mathcal{T}(f\cdot e^{rx}) = \left(\frac{1}{1-rX}\right)\times\left(\mathcal{T}(f)\circ\frac{X}{1-rX}\right), \tag{68}$$

$$\mathcal{T}(f\cdot x) = \left(X^2\times\frac{d\mathcal{T}(f)}{dX}\right) + \big(X\times\mathcal{T}(f)\big). \tag{69}$$

PROOF. The first two equalities are straightforward. For (65)–(67), define

$$R = \left\{\left\langle \mathcal{T}(s\cdot e^{rx}), \frac{s}{1-rX}\right\rangle \;\middle|\; r,s\in\mathbb{R}\right\} \cup \left\{\left\langle \mathcal{T}\big(s\cdot\sin(rx)\big), \frac{(s\times r)X}{1+r^2X^2}\right\rangle \;\middle|\; r,s\in\mathbb{R}\right\}$$
$$\cup \left\{\left\langle \mathcal{T}\big(s\cdot\cos(rx)\big), \frac{s}{1+r^2X^2}\right\rangle \;\middle|\; r,s\in\mathbb{R}\right\}$$

and use coinduction. For (68), note that

$$\begin{aligned}\mathcal{T}(f\cdot e^{rx}) &= \mathcal{T}(f)\otimes\mathcal{T}(e^{rx}) && \text{[identity (60)]}\\ &= \mathcal{T}(f)\otimes\left(\frac{1}{1-rX}\right) && \text{[identity (65)]}\\ &= \left(\frac{1}{1-rX}\right)\times\left(\mathcal{T}(f)\circ\frac{X}{1-rX}\right) && \text{[identity (52)].}\end{aligned}$$

For (69), finally, we have:

$$\begin{aligned}\mathcal{T}(f\cdot x) &= \mathcal{T}(f)\otimes\mathcal{T}(x) && \text{[identity (60)]}\\ &= \mathcal{T}(f)\otimes X && \text{[identity (64)]}\\ &= \left(X^2\times\frac{d\mathcal{T}(f)}{dX}\right) + \big(X\times\mathcal{T}(f)\big). && \text{[identity (51)].}\end{aligned}$$

□

By now, we are sufficiently prepared to tackle a variety of differential equations. As a warming up, consider the equation from the beginning of the present section:

differential equation	initial value
$f'' + f = 0$	$f(0) = 0,\ f'(0) = 1$

.

Applying $\mathcal{T}$ to both sides of the equation, and putting $\sigma = \mathcal{T}(f)$ gives

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

.

This constitutes step (1) of our method. In step (2), the resulting behavioural differential equation is solved according to the techniques of Section 2.6, yielding

$$\sigma = \frac{X}{1 + X^2}.$$

In order to translate this outcome back into the function f—step (3) of our method—we can apply identity (20) yielding

$$\sigma = X - X^3 + X^5 - X^7 + \cdots$$

which, in combination with identity (64) gives

$$f(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots .$$

Often, we can improve on the last step. In this particular example, it is sufficient to consult our set of basic identities on Taylor transforms, to find that identity (66) provides us immediately with an answer:

$$f(x) = sin(x).$$

For a second example, consider the following non-homogeneous equation:

differential equation	initial value
$f' - f = e^x$	$f(0) = 1$

.

Using identity (65) and writing $\sigma = \mathcal{T}(f)$, step (1) transforms this equation into

derivative	initial value
$\sigma' - \sigma = (1 - X)^{-1}$	$\sigma(0) = 1$

.

The solution of this equation, step (2), is identity (23). For step (3), we combine $\sigma = \mathcal{T}(f)$ with identities (18) and (64), yielding

$$f(x) = 1 + \frac{2}{1!}x + \frac{3}{2!}x^2 + \frac{4}{3!}x^3 \cdots .$$

Again, we can do better than this. Using some elementary stream calculus, we can rewrite σ as follows:

$$\begin{aligned} \sigma &= \frac{1}{(1 - X)^2} \\ &= (1 + X) \otimes \frac{1}{1 - X} && \text{[by identity (55)].} \end{aligned}$$

It follows that

$$f(x) = (1 + x) \cdot e^x.$$

Here is example number three:

differential equation	initial value
$f'' + r^2 f = 1$	$f(0) = 1,\ f'(0) = 1$

Putting $\sigma = \mathcal{T}(f)$, step (1) gives

derivative	initial value
$\sigma'' + r^2\sigma = 1$	$\sigma(0) = 1,\ \sigma'(0) = 1$

.

For step (2) we recall the solution of this equation from Section 2.6, and perform some elementary stream calculus on it:

$$\begin{aligned}\sigma &= \frac{1 + X + X^2}{1 + r^2X^2}\\ &= \frac{1}{1 + r^2X^2} + \frac{X}{1 + r^2X^2} + \frac{X^2}{1 + r^2X^2}\\ &= \frac{1}{1 + r^2X^2} + \frac{1}{r}\frac{rX}{1 + r^2X^2} + \frac{1}{r^2}\left(1 - \frac{1}{1 + r^2X^2}\right).\end{aligned}$$

The rewriting was done to make step (3) easy: applying identities (66) and (67) yields the final outcome:

$$f(x) = \cos(rx) + \frac{1}{r}\sin(rx) + \frac{1}{r^2}\big(1 - \cos(rx)\big).$$

As a last example, consider the following equation:

differential equation	initial value
$f'' - f' = 2 + 6x$	$f(0) = 1,\ f'(0) = 0$

.

Step (1) gives

derivative	initial value
$\sigma'' - \sigma' = 2 + 6X$	$\sigma(0) = 1,\ \sigma'(0) = 0$

.

As before, we recall the solution of this equation (step (2)) from Section 2.6, and perform some elementary stream calculus on it:

$$\begin{aligned}\sigma &= \frac{1 - X + 2X^2 + 6X^3}{1 - X}\\ &= -7 - 8X - 6X^2 + 8\frac{1}{1 - X}.\end{aligned}$$

For the last equality, we have used the following identity:

$$X^{n+1} \times \frac{1}{1 - X} = -1 - X - \cdots - X^n + \frac{1}{1 - X}.$$

Applying the by now familiar identities on the Taylor transforms, step (3) yields

$$f(x) = -7 - 8x - 3x^2 + 8e^x.$$

3.5. Discussion

Pavlović and Escardó's paper on calculus in coinductive form [**34**], emphasising the close connection between classical analysis and coinduction, has been an important source of inspiration for the work presented in this chapter. Definition 3.4.1 of $\mathcal{T}$ is a variation on a definition in [**34**], where $\mathcal{T}$ is characterised as a final coalgebra homomorphism, in order to give a coinductive characterization of the Laplace transform. Here $\mathcal{T}$ is studied in its own right, and serves rather as an alternative to Laplace transform.

Solving differential equations in stream calculus essentially amounts to the classical method of undetermined coefficients [**12**, p. 82]. The main difference is that the obtained difference equations for the Taylor coefficients of the analytical solution are

solved within the world of streams. This approach is technically closely related to the use of Laplace transforms (cf. [**30, 42**]), because the operation of assigning the Taylor series $\mathcal{T}(f)$ to an analytical function f implicitly uses the Laplace–Carson transform (introduced in Section 3.3). Formulae such as (63)–(69) are similar to but different from the formulae of the corresponding Laplace transforms (cf. [**42**, p. 519]). Conceptually, the use of stream calculus is different and, again, a bit simpler than these traditional approaches. In particular, analytical integration plays no role, since the difference equations are solved by stream integration (applying the Fundamental Theorem of stream calculus, Theorem 2.6.1).

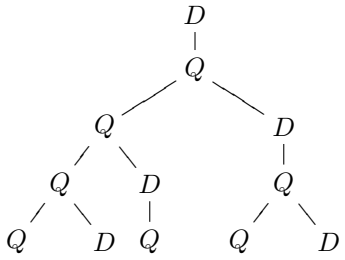
CHAPTER 4

Coinductive Counting

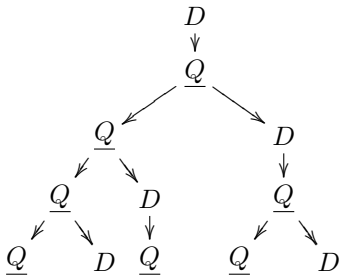
This chapter presents a further application of the stream calculus developed in Chapter 2, to counting problems from the world of enumerative combinatorics. The first section below introduces the kind of problems we shall be dealing with, and summarizes the approach.

4.1. Introduction

The following counting problem is taken from [**22**, p. 291]. Male bees are called drones and female bees are called queens. Drones are born out of a queen and have no father; a queen is born out of a father drone and a mother queen. The first few levels of the pedigree of a drone (drawn upside-down) look as follows:

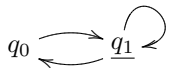


We see that each drone has one mother, one grandmother, two great-grandmothers, three great-great-grandmothers, and so on. What is, for any $k \geq 0$, the number s_k of female ancestors at level k? The key idea of coinductive counting is to use the very tree that enumerates all the (female) ancestors of a drone, as the basis for a representation of the infinite *stream* $\sigma = (s_0, s_1, s_2, \dots)$ containing all the answers. To this end, the tree is turned into an automaton, in which the arrows indicate transitions and in which all the queen states are output states:



Formally, it is a weighted automaton, a notion that was introduced in Section 2.8. There we saw that the stream behaviour of such automata can be defined both by means of behavioural differential equations, and in terms of transition sequences.

In the present automaton, the number s_k of female ancestors at level k is encoded as the number of paths of length k leading from the initial (topmost) drone state to an output queen state, since there are as many such sequences as there are queens at level k. Thus we have translated the original counting problem into a question about streams and their representation by automata. Using stream bisimulations, the above automaton can be simplified by identifying all equivalent states as follows. Every drone state is bisimulation equivalent to the state q_0 and every queen state is equivalent to the state q_1 of the following two state automaton:



Intuitively, any queen state and the state q_1 have the same transition behaviour: both can take two transitions to states that are again, respectively, equivalent. Similarly for drone states and q_0, which have only one transition. As a consequence, (the state q_0 of) this new automaton is an equivalent representation of our stream of answers σ: s_k corresponds again to the number of paths of length k leading from q_0 to the (in this case only) output state q_1. Using stream calculus, we can easily compute a closed expression for the infinite stream σ of answers represented by the state q_0, yielding

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

As with generating functions, this stream expression encodes the numbers s_k for all $k \geq 0$ (which turn out to be the Fibonacci numbers). We leave aside the computation of an explicit formula for s_k, and consider the above fraction, which is formulated in terms of stream constants and operators, as a satisfactory answer to our question.

Summarizing the above, we distinguish three phases in the procedure of coinductive counting:

(1) *Enumeration* of the objects to be counted in an infinite, tree-shaped weighted automaton.
(2) *Identification* of equivalent states using bisimulation.
(3) *Expression* of the resulting stream of counts in terms of stream constants and operators.

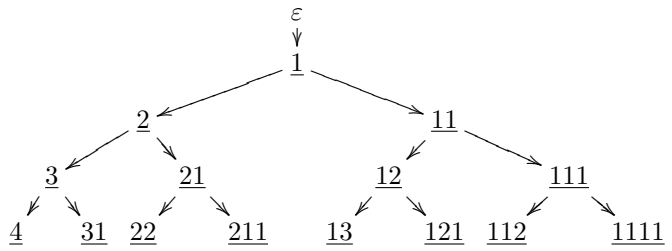
As we shall see shortly, the entire approach is essentially quantitative: both transitions and output states will generally be labelled with *weights* (here: real numbers), which are taken into account by the notion of stream bisimulation (and bisimulation-up-to).

4.2. Compositions of Natural Numbers

A *composition* of a natural number $k \geq 0$ is a sequence of natural numbers $n_1 \cdots n_l$ such that $k = n_1 + \cdots + n_l$. What is, for any $k \geq 0$, the number s_k of compositions of k? Or, equivalently, what is the stream of all counts $\sigma = (s_0, s_1, s_2, \dots)$? The problem will be solved coinductively by going through the three phases mentioned in the introduction.

Enumeration. The following automaton enumerates all compositions for all natural numbers (here and in what follows, pictures show only the first few levels

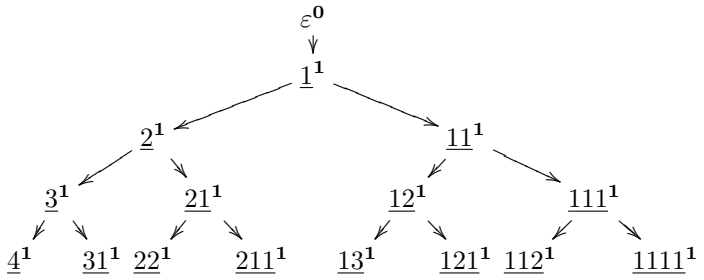
of what is understood to be an infinite automaton):



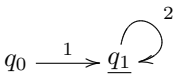
The automaton of this picture can be formally described by defining a state set $Q = \{w \mid w \in \mathbb{N}^*\}$; an output function $o\colon Q \to \mathbb{R}$ given by $o(\varepsilon) = 0$ and $o(w) = 1$ for all $w \neq \varepsilon$; and a transition function $t\colon Q \to (Q \to_f \mathbb{R})$, defined by $t(v)(w) = 1$ iff either $w = v \cdot 1$ or ($v = u \cdot k$ and $w = u \cdot (k+1)$, for some $u \in Q$ and $k \in \mathbb{N}$), and by $t(v)(w) = 0$ otherwise. (Here $u \cdot k$ denotes the concatenation of the word u and the one letter word k.) In what follows, however, we shall present automata usually by their pictures, becoming more formal only when strictly necessary.

Let $k \geq 0$ be any natural number. There is a one-to-one correspondence between paths of length k starting in the (topmost) initial state ε, on the one hand, and compositions of the natural number k (all of which are situated at the kth level of the automaton), on the other. As a consequence, the total number s_k of all compositions of k is equal to the total number of paths of length k starting in the initial state ε. We can now use identity (24), from Section 2.8, to conclude that the stream behaviour $S(\varepsilon)$ of the initial state ε equals the stream $\sigma = (s_0, s_1, s_2, \dots)$ of answers we are after: $S(\varepsilon) = \sigma$.

Identification. Next we identify as many states as possible in our infinite weighted automaton, as follows:



The superscripts that we have added to the states of our automaton, indicate to which state in the automaton below they are related:



More explicitly and precisely, the above pictures suggests the definition of a relation $R \subseteq \mathbb{R}^\omega \times \mathbb{R}^\omega$ as follows:

$$R = \{\langle S(\varepsilon), S(q_0)\rangle\} \cup \{\langle S(w), S(q_1)\rangle \mid w \in \mathbb{N}^*, w \neq \varepsilon\}$$

relating the streams represented by (that is, the stream behaviour of) the states in the first automaton above, with the streams represented by the states q_0 and q_1 in the second. In order to show that R is a bisimulation-up-to, first note that

the initial values of all related pairs match: $S(\varepsilon)(0) = 0 = S(q_0)(0)$ and, for all words $v \in \mathbb{N}^*$ with $v \neq \varepsilon$, $S(v)(0) = 1 = S(q_1)(0)$. Next we check derivatives: $S(\varepsilon)' = S(1)$, which is related to $S(q_1) = S(q_0)'$; and for all words $v \in \mathbb{N}^*$ and natural numbers n, we have $S(v \cdot n)' = S\big(v \cdot (n+1)\big) + S(v \cdot n \cdot 1)$, each component of which is related to $S(q_1)$, thus matching $S(q_1)' = 2 \times S(q_1) = S(q_1) + S(q_1)$. This proves that R is a bisimulation-up-to, indeed. Now it follows by coinduction-up-to that $S(\varepsilon) = S(q_0)$.

Expression. We can now easily compute a closed expression for $\sigma = S(q_0)$ by solving the system of equations, consisting of $S(q_0) = X \times S(q_1)$ and $S(q_1) = 1 + \big(2 \times X \times S(q_1)\big)$, yielding

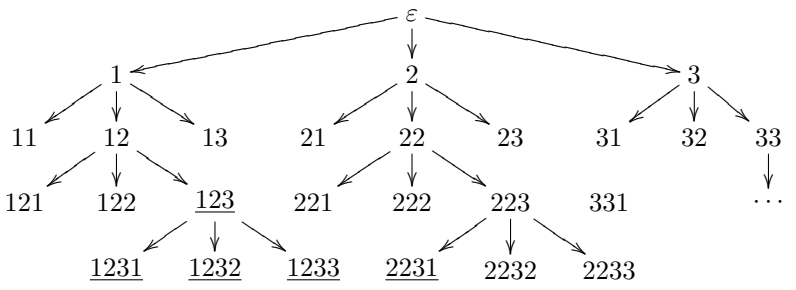
$$\sigma = S(q_0) = \frac{X}{1 - 2X}.$$

It is worthwhile emphasizing the quantitative aspect of the notion of bisimulation (up-to): the fact that any state of the original weighted automaton labelled by a non-empty word w can take *two* transitions to similar such states, is reflected by a 2-*labelled* transition from q_1 to itself.

4.3. Surjections

What is, for any natural number $k \geq 0$, the number s_k of surjections from the set $\{1, \ldots, k\}$ onto the set $\{1, 2, 3\}$ (defining s_0 to be 0)? Below we shall see how the answer can be generalized to surjections onto the set $\{1, \ldots, n\}$, for a fixed but arbitrary $n \geq 1$.

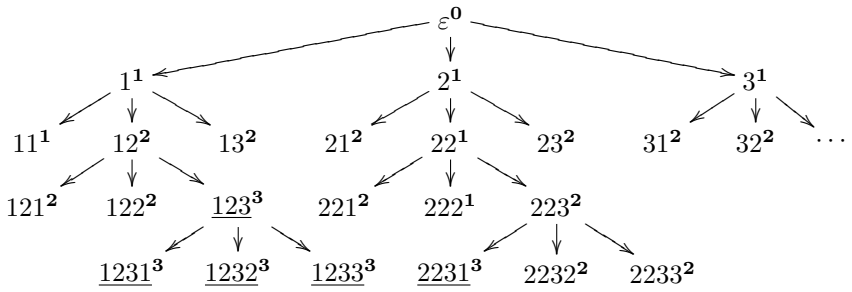
Enumeration. Let us denote a function $f \colon \{1, \ldots, k\} \to \{1, 2, 3\}$ by means of the word $f(1) \cdots f(k)$. The following automaton enumerates at each level k all such functions:



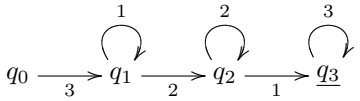
Note that all states labelled by a word representing a surjection (that is, containing at least one 1, one 2, and one 3), have been defined as output states. Also note that we have not only restricted the picture to the first five levels of the automaton, but that moreover not all transitions have been included, for lack of space. As before, it follows from (24) that the initial state ε represents the stream $\sigma = (s_0, s_1, s_2, \ldots)$ of answers we are interested in.

Identification. The automaton can be simplified by identifying all states (labelled by words) that have the same number of different symbols, as indicated by

the superscripts below:



More precisely, (the streams represented by) the i-superscripted states above can be related with (the stream represented by) the state q_i in the automaton below,



by means of a relation $R \subseteq \mathbb{R}^\omega \times \mathbb{R}^\omega$ that is defined as follows:

$$R = \{\langle S(w), S(q_i)\rangle \mid w \in \{1,2,3\}^*, i \in \{0,1,2,3\} : \sharp(w) = i\}$$

where, for this occasion, $\sharp(w)$ denotes the number of different symbols contained in w. It is straightforward to check that R is a bisimulation-up-to, from which $S(\varepsilon) = S(q_0)$ follows by coinduction-up-to.

Expression. The stream behaviour of our 4-state automaton is determined by the following system of equations:

$$\begin{aligned}
S(q_0) &= 3 \times X \times S(q_1),\\
S(q_1) &= \big(1 \times X \times S(q_1)\big) + \big(2 \times X \times S(q_2)\big),\\
S(q_2) &= \big(2 \times X \times S(q_2)\big) + \big(1 \times X \times S(q_3)\big),\\
S(q_3) &= 1 + \big(3 \times X \times S(q_3)\big).
\end{aligned}$$

Solving this system of equations, one finds:

$$\sigma = S(q_0) = \frac{3!X^3}{(1-X)(1-2X)(1-3X)}.$$

The above can be easily generalised. Let $n \geq 1$ be arbitrary but fixed, and let t_k, for any $k \geq 0$, be the number of all surjections from the set $\{1,\dots,k\}$ onto the set $\{1,\dots,n\}$. Then the stream of counts $\tau = (t_0, t_1, t_2, \dots)$ is given by

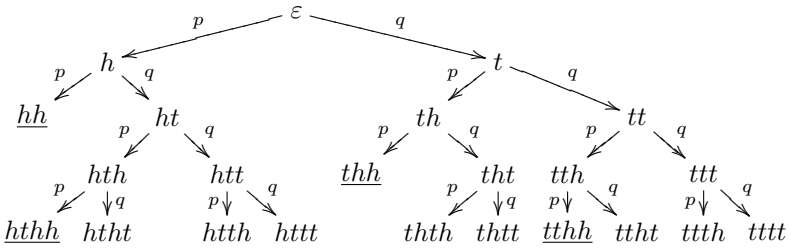
$$\begin{aligned}
\tau &= \frac{n!X^n}{(1-X)(1-2X)\cdots(1-nX)}\\
&= \left(\frac{1}{1-X} - 1\right)^{\underline{n}}
\end{aligned}$$

where the latter equality can be proved using some basic stream calculus.

4.4. Counting with Probabilities

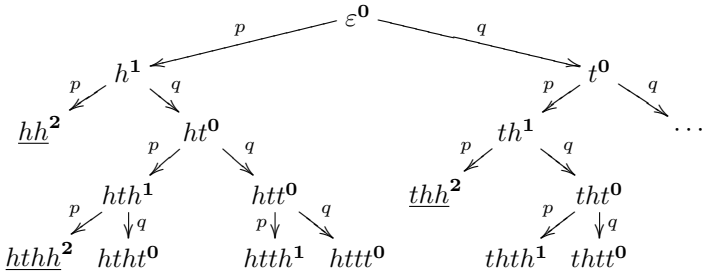
Consider a not necessarily fair coin with probability p of producing a head and probability $q = 1 - p$ of producing a tail. What is, for any $k \geq 0$, the probability s_k of getting, by flipping the coin k times, a sequence of heads and tails (of length k) without the occurrence of two consecutive heads but for the two very last outcomes, which are required to be heads? We hope the reader is by now already sufficiently experienced with coinductive counting, to be able to supply the details that have been left out in the succinct presentation of the solution below.

Enumeration. Here is a weighted automaton describing all possible scenarios:

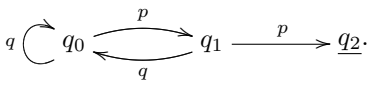


All states that are (labelled with a sequence) ending in two heads are output states, and have no further transitions. The stream $\sigma = (s_0, s_1, s_2, \dots)$ of all counts is represented by the initial state ε, that is, $\sigma = S(\varepsilon)$.

Identification. States can be identified according to the number of final heads:



yielding the following automaton:



More precisely, one can prove by coinduction-up-to that $S(\varepsilon) = S(q_0)$.

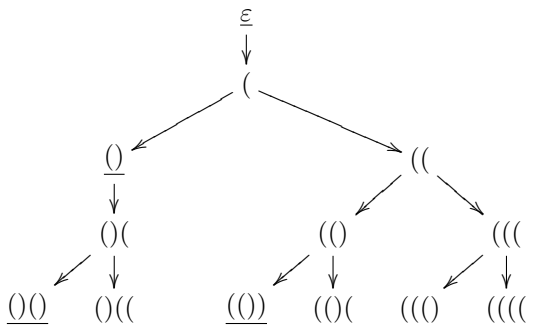
Expression. The stream behaviour of $S(q_0)$ can be easily computed:

$$\sigma = S(q_0) = \frac{p^2 X^2}{1 - qX - pqX^2}.$$

4.5. Well-Bracketed Words

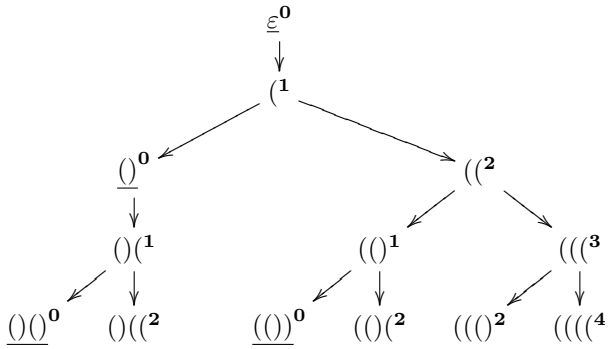
So far the identification phase of our coinductive counting exercises has always resulted in a finite automaton yielding a rational stream. Here is an example for which the stream of counts is no longer rational (but *algebraic*). Consider a two letter alphabet $\{(,)\}$ consisting of a left and a right bracket. What is, for any $k \geq 0$, the number s_k of well-bracketed words over this alphabet, of length k?

Enumeration. The output states at level k of the following automaton correspond precisely to all such words:

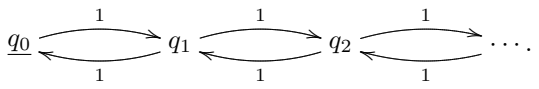


so the stream $\sigma = (s_0, s_1, s_2, \dots)$ of all counts is represented by the initial state: $\sigma = S(\varepsilon)$.

Identification. We identify states according to the number of left brackets they contain without a matching right bracket:



This yields the following well-structured, but still infinite automaton:



As before, one can prove $S(\varepsilon) = S(q_0)$ by coinduction-up-to.

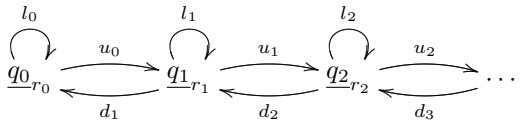
Expression. The stream behaviour of the above automaton is given by the following infinite system of equations: $S(q_0) = 1 + \big(X \times S(q_1)\big)$ and

$$\boxed{S(q_i) = \big(X \times S(q_{i-1})\big) + \big(X \times S(q_{i+1})\big) \quad | \quad i \geq 1}$$

but it is not immediately obvious how this infinite system of equations can be solved. Rather than trying to solve it, we shall in Section 4.6 develop some more general results directly on infinite automata of the type above. The discussion of the solution of the present problem is therefore postponed until Section 4.7.

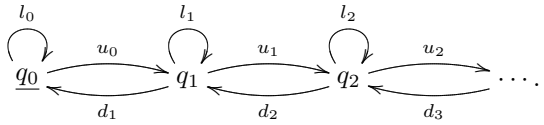
4.6. Streams and Continued Fractions

In this section, we study the stream behaviour of infinite weighted automata of the following type:



All of the labels and output values are arbitrary real numbers. The labels l_i and u_i, for $i \geq 0$, and d_i, for $i \geq 1$, could be pronounced as *level*, *up*, and *down*. The stream behaviour of this automaton will be expressed by means of a (generalised) continued fraction, but before we deal with it in its full generality, we first treat two special cases that are particularly relevant for many of the counting problems to come.

The first case concerns the situation that all output values r_i are 0 but for the output value r_0 of the first state q_0, which (may be arbitrary but for convenience) has been chosen to be 1:



THEOREM 4.6.1. *The stream behaviour $S(q_0)$ of the state q_0 in the automaton above is given by the following continued fraction*:

$$\cfrac{1}{1-(l_0\times X)-\cfrac{(u_0\times X)\times(d_1\times X)}{1-(l_1\times X)-\cfrac{(u_1\times X)\times(d_2\times X)}{1-(l_2\times X)-\cfrac{(u_2\times X)\times(d_3\times X)}{\ddots}}}}$$

PROOF. Let us first make sense of the continued fraction, which is denoted by what seems to be an infinite expression. Formally, it can be defined as follows. Let the streams $\sigma_0, \sigma_1, \ldots$ be the unique solutions of the following system of behavioural differential equations:

differential equation	initial value
$\sigma_i' = (l_i \times \sigma_i) + (u_i \times \sigma_{i+1} \times d_i \times X \times \sigma_i)$	$\sigma_i(0) = 1$

.

By Theorem 2.6.1, it follows that

$$\sigma_i = \frac{1}{1-(l_i\times X)-(u_i\times X)\times\sigma_{i+1}\times(d_{i+1}\times X)}$$

for all $i \geq 0$. The infinite continued fraction in the theorem can now simply be read as a suggestive notation for σ_0. Next we show that $S(q_0) = \sigma_0$. Consider the following relation on streams:

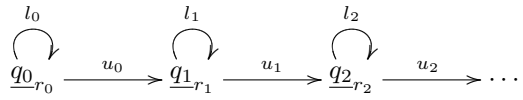
$$R = \{\langle r \times S(q_i), r \times \tau_i\rangle \mid r \in \mathbb{R}, i \geq 0\}$$

where the streams τ_i are defined by

$$\tau_i = \sigma_i \times (d_i \times X) \times \sigma_{i-1} \times (d_{i-1} \times X) \times \cdots \times \sigma_1 \times (d_1 \times X) \times \sigma_0.$$

Because R is a bisimulation relation on $\mathbb{R}^\omega$, $S(q_0) = \sigma_0$ follows by coinduction. □

For the second special case of the automaton presented at the beginning of this section, we assume that $d_i = 0$ for all $i \geq 1$, that is, there are no downward transitions. In contrast to the automaton of Theorem 4.6.1, however, all states may have again a non-trivial output value $r_i \in \mathbb{R}$:

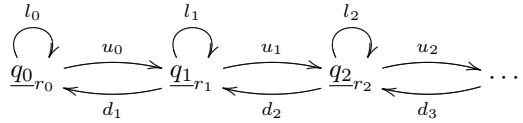


THEOREM 4.6.2. *The stream behaviour $S(q_0)$ of the state q_0 is given by the following "upward" continued fraction*:

$$r_0 + (u_0 \times X) \times \cfrac{r_1 + (u_1 \times X) \times \cfrac{r_2 + (u_2 \times X) \times \cfrac{\vdots}{1 - (l_3 \times X)}}{1 - (l_2 \times X)}}{1 - (l_1 \times X)} \Bigg/ 1 - (l_0 \times X)$$

The proof is omitted. It can be given along the same lines as the proof of Theorem 4.6.1.

Finally, we return to the automaton at the beginning of this section, which is the most general of all:



THEOREM 4.6.3. *The stream behaviour $S(q_0)$ of the state q_0 is given by the following crazy expression, which consists of (nested) continued fractions growing both upward and downward*:

$$\cfrac{r_0 + (u_0 \times X) \times \sigma}{1 - (l_0 \times X) - (u_0 \times X) \times \sigma \times (d_1 \times X)}$$

where

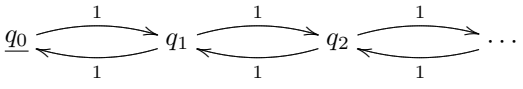
$$\sigma = \cfrac{r_1 + (u_1 \times X) \times (\cdots)}{1 - (l_1 \times X) - (u_1 \times X) \times (\cdots) \times (d_2 \times X)}.$$

PROOF. Also this theorem can be proved in the same manner as the previous two theorems. □

4.7. More on Well-Bracketed Words

We return to the problem of Section 4.5, and compute a closed expression for the stream $\sigma = (s_0, s_1, s_2, \dots)$ of numbers s_k of well-bracketed words of length k (over the alphabet $\{(,)\}$).

At the end of Section 4.5, the following automaton representation for the stream σ of counts was found:



An application of Theorem 4.6.1 yields the following continued fraction for $\sigma = S(q_0)$:

$$\sigma = S(q_0) = \cfrac{1}{1 - \cfrac{X^2}{1 - \cfrac{X^2}{\ddots}}}$$

Because of the regular structure of this fraction, it is possible to come up with a more succinct expression for σ. It is easy to see that

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

(cf. the proof of Theorem 4.6.1). This implies that σ is a solution of the following quadratic equation:

$$(X^2 \times \sigma^2) - \sigma + 1 = 0.$$

In order to solve it, we need a square root operator on streams. As usual in stream calculus, its definition can be given by means of a behavioural differential equation. Let $\sqrt{\sigma}$, for all streams σ with $\sigma(0) > 0$, be defined as the unique stream satisfying the following differential equation:

behavioural differential equation	initial value
$(\sqrt{\sigma})' = \frac{\sigma'}{\sqrt{\sigma(0)}+\sqrt{\sigma}}$	$\sqrt{\sigma}(0) = \sqrt{\sigma(0)}$

.

(Here $\sqrt{\sigma(0)}$ is the square root of the real number $\sigma(0)$.) With this operator, quadratic equations like the one above can be solved in much the same way we are used to in analysis (see [**39**] for details). This yields as the final outcome for our stream of counts of well-bracketed words:

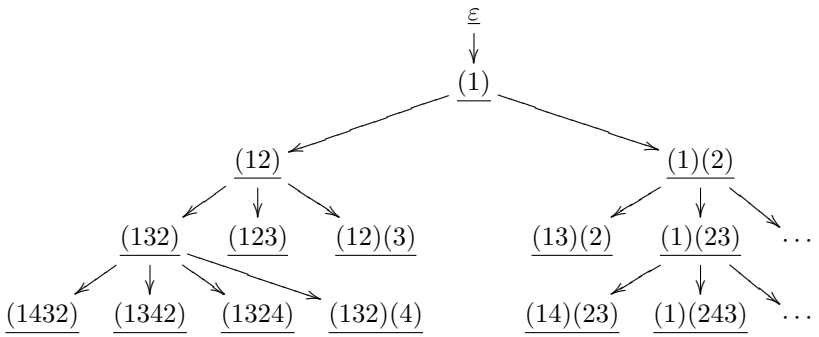
$$\sigma = \frac{2}{1 + \sqrt{1 - 4X^2}}$$

(which equals the stream $(1, 0, 1, 0, 2, 0, 5, 0, 14, 0, \dots)$ having the Catalan numbers at the even positions).

4.8. Permutations

A permutation (bijection) $p\colon \{1, \dots, k\} \to \{1, \dots, k\}$ can be represented by the corresponding sequence of images $p = \big(p(1) \cdots p(k)\big)$. Another very common and equivalent representation, which is the one that we shall be using, describes a permutation by the (unique) sequence of cycles of which the permutation is composed. For instance, the permutation $p = (532461)$, with $p(1) = 5$, $p(2) = 3$, $p(3) = 2$, $p(4) = 4$, $p(5) = 6$, and $p(6) = 1$, can also be represented by the following sequence: $p = (156)(23)(4)$. Here a cycle $c = (x_1 \cdots x_k)$ denotes a permutation of $\{x_1, \dots, x_k\}$ defined by: $c(x_i) = x_{i+1}$, for all $1 \leq i \leq k-1$, and $c(x_k) = x_1$. We start with a trivial question, for any $k \geq 0$: what is the number s_k of permutations of the set $\{1, \dots, k\}$ (defining $s_0 = 1$)? The following automaton enumerates all

permutations by listing all possible sequences of cycles:



Any state at level k represents a permutation of the set $\{1, \dots k\}$ and is therefore an output state. It can make a transition to a state at the next level, either by adding the number $k+1$ to one of the existing cycles or by adding the new cycle $(k+1)$. There are (for all states at level k) precisely k transitions of the first, and one transition of the second type, $k+1$ transitions in total. This explains the structure of the automaton above, and at the same time indicates that all states of every single level can be identified, yielding the following automaton:

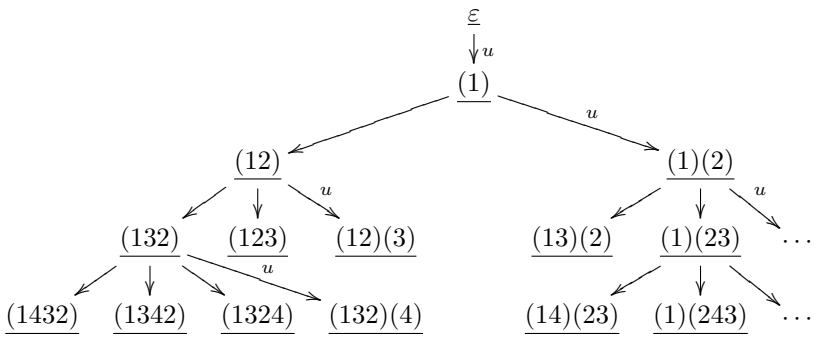
$$\underline{q_0} \xrightarrow{1} \underline{q_1} \xrightarrow{2} \underline{q_2} \xrightarrow{3} \cdots .$$

Applying Theorem 4.6.2, with $r_k = 1$, $l_k = 0$ and $u_k = k+1$ for all $k \geq 0$, we obtain

$$S(q_0) = 1 + 1!X + 2!X^2 + 3!X^3 + \cdots$$

for the stream $(s_0, s_1, s_2, \dots)$ of counts we are after. In other words, there are $k!$ different permutations of the set $\{1, \dots, k\}$, which comes as no surprise.

We have chosen to represent permutations by sequences of cycles in the automaton above, because it can be easily adapted to deal with various related counting problems. A first and straightforward variation is to keep track of the total number of cycles in each permutation. This we can do by fixing any real number (variable) u, which we use as a label for all transitions that represent the addition of a new cycle (recall that all other transitions have label 1, which is as usual omitted):



Identifying again all states of the same level gives the following equivalent automaton:

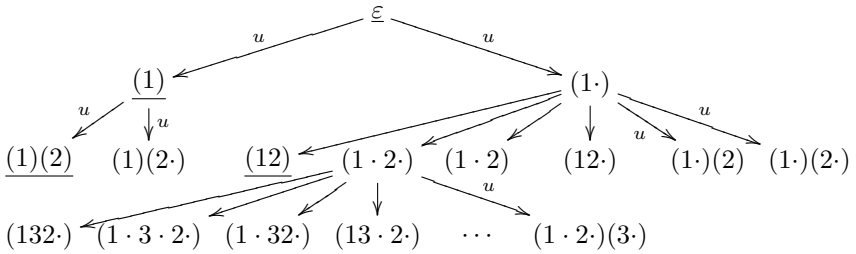
$$\underline{q_0} \xrightarrow{u} \underline{q_1} \xrightarrow{u+1} \underline{q_2} \xrightarrow{u+2} \cdots$$

to which, as before, Theorem 4.6.2 an be applied, now yielding

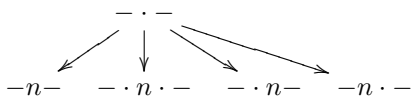
$$\begin{aligned} S(q_0) &= 1 + u \times X + u(u+1) \times X^2 + u(u+1)(u+2) \times X^3 + \cdots \\ &= (1-X)^{-\underline{u}} \end{aligned}$$

where the latter equality can be proved using some elementary stream calculus. (Note that this formula generalises the solution of our previous counting, which is obtained by taking $u = 1$.) The above stream can be considered as having u as a parameter. It encodes all numbers $s_{k,n}$, for $k, n \geq 0$, counting all permutations of $\{1, \ldots, k\}$ consisting of n cycles. These numbers are known as the Stirling numbers of the first kind, and can be computed from the stream above as the (Taylor) coefficients of the powers $u^n \times X^k$. (One can also treat both X and u as formal variables; this would bring us to multivariate stream calculus, which is omitted here.)

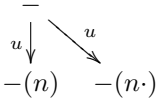
Next we present yet another way of counting permutations while keeping track of the number of cycles, yielding another representation of the Stirling numbers of the first kind. More specifically, permutations can, alternatively, be enumerated as follows:



The general pattern in which this tree-like weighted automaton is grown, is as follows. In any state (node) at level $n-1$, all of the numbers $\{1, \ldots, n-1\}$ occur, possibly together with a number of dots. All states without dots are output states, representing completed permutations. States with dots represent permutations that have not yet been completed, and the dots indicate the places where further growth is possible. More specifically, for each dot, there are four successor states:

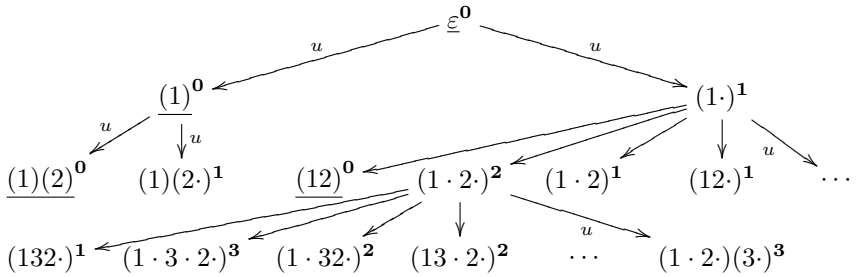


in which the dot has been replaced with n, $\cdot n\cdot$, $\cdot n$, and $n\cdot$, respectively. In addition, any state at level $n-1$ has two more successors, corresponding to the opening of a new cycle:

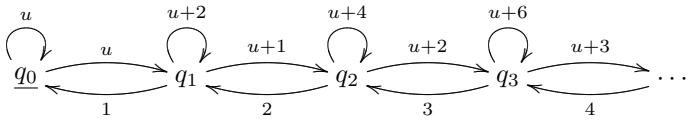


As before, the transitions in which a new cycle is opened, are labelled by u (instead of 1).

States can be identified according to the number of dots they contain, as is indicated by the superscripts below:



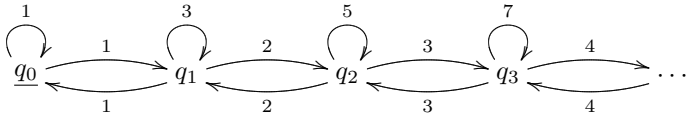
Identification of all equivalent states yields the following well-structured automaton:



to which Theorem 4.6.1 can be applied, resulting in the following continued fraction:

$$S(q_0) = \cfrac{1}{1 - uX - \cfrac{1uX^2}{1 - (u+2)X - \cfrac{2(u+1)X^2}{1 - (u+4)X - \cfrac{3(u+2)X^2}{\ddots}}}}$$
$$= (1-X)^{-\underline{u}}.$$

The second equality is obtained by combining our first and our second way of counting permutations and cycles. Thus we have obtained two different expressions for the Stirling numbers of the first kind. Forgetting about the number of cycles again, that is, taking $u = 1$, we obtain

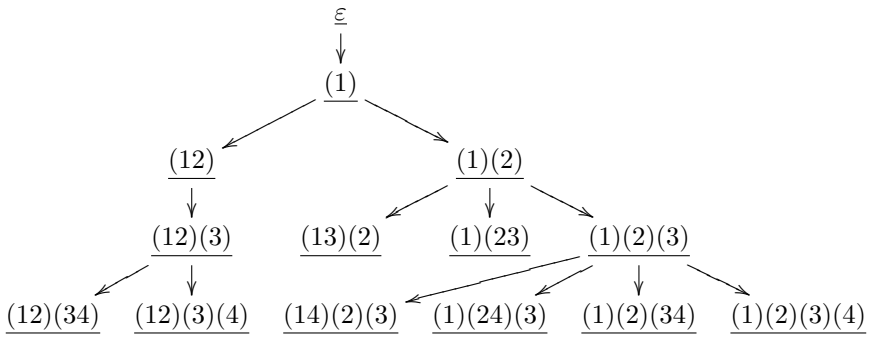


which leads to the following equalities:

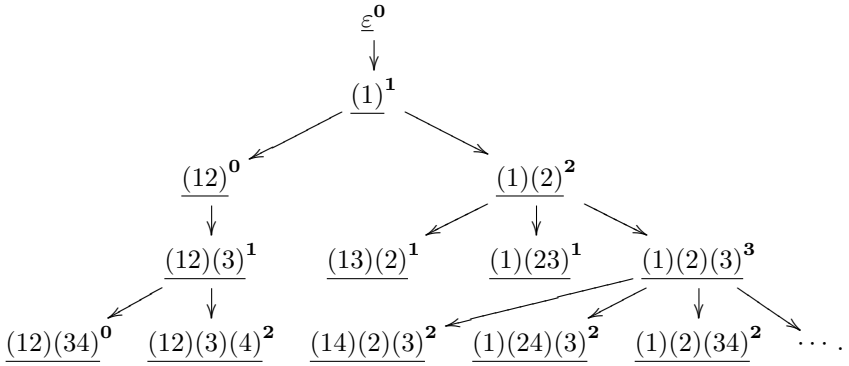
$$S(q_0) = \cfrac{1}{1 - X - \cfrac{1^2X^2}{1 - 3X - \cfrac{2^2X^2}{1 - 5X - \cfrac{3^2X^2}{\ddots}}}}$$
$$= (1-X)^{-\underline{1}}$$
$$= (0!, 1!, 2!, 3!, \dots).$$

What is, moving to a next question, for any $k \geq 0$ the number s_k of permutations p of $\{1, \dots, k\}$ such that $p \circ p = 1$ (the so-called involutions)? For this we

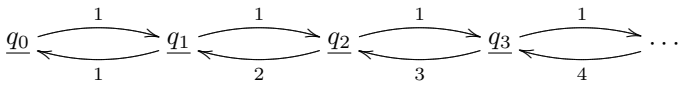
return to the first automaton of this section, from which we now remove all states but the ones consisting of 1- and 2-cycles only:



States can be identified according to the number of 1-cycles, which can still become 2-cycles, they contain, as indicated by the superscripts below:

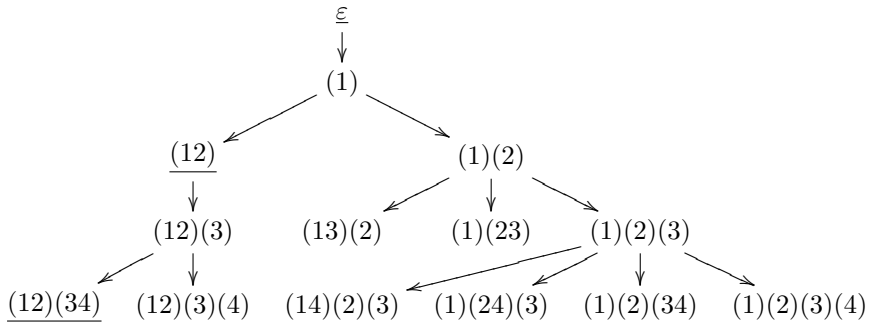


yielding the following automaton

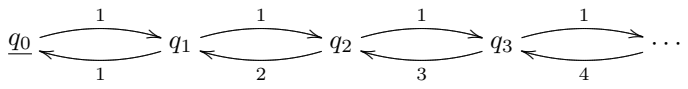


Applying Theorem 4.6.3 yields an expression for $S(q_0)$, but not a very pretty one (which is therefore omitted).

For the special case of permutations consisting of 2-cycles only, the same enumeration as for the involutions can be used, the only difference being that states that contain 1-cycles are no longer output states:



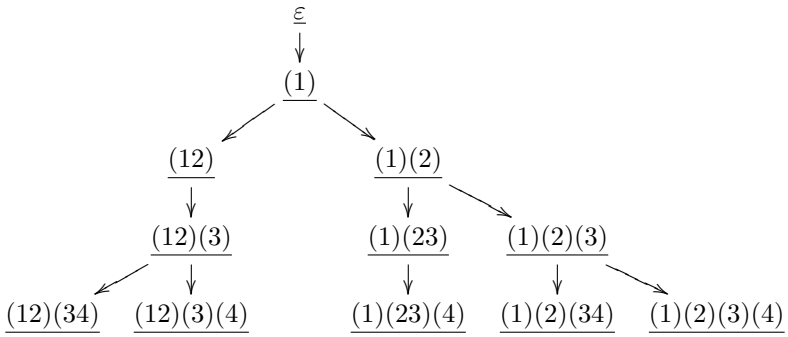
The corresponding reduced automaton now looks like



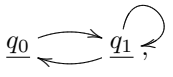
with, according to Theorem 4.6.1, a pleasant expression for the stream of answers:

$$S(q_0) = \cfrac{1}{1 - \cfrac{1 \cdot X^2}{1 - \cfrac{2 \cdot X^2}{1 - \cfrac{3 \cdot X^2}{\ddots}}}}$$

Yet another question: what is, for any $k \geq 0$, the number s_k of permutations $p\colon \{1, \ldots, k\} \to \{1, \ldots, k\}$ such that $|p(i) - i| \leq 1$, for all $1 \leq i \leq k$? These are precisely all permutations consisting of 1-cycles, and of 2-cycles (xy) with $y = x+1$. The following automaton, which happens to be a further pruning of the last tree-like automaton above, lists them all:



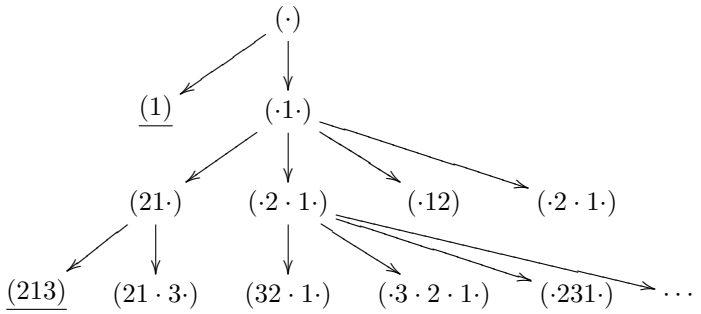
(Note that now all states are output states again.) All states ending with a 1-cycle and all states ending with a 2-cycle can be identified, respectively, leading to the following automaton and corresponding expression for the stream of answers:



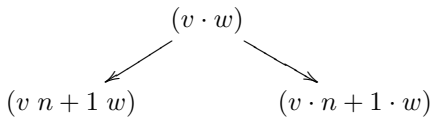
$$S(q_0) = \frac{1}{1 - X - X^2}.$$

For the last example of this section, we return to the representation of permutations mentioned at the beginning of this section. That is, a permutation (bijection) $p\colon \{1, \ldots, k\} \to \{1, \ldots, k\}$ will be represented below by the corresponding sequence of images $p = \big(p(1) \cdots p(k)\big)$. Referring to this representation, a permutation is called *alternating* whenever either $p(0) > p(1) < p(2) > \cdots p(n)$ or $p(0) < p(1) > p(2) < \cdots p(n)$. What is, we ask ourselves (because we already know that the question will have a pretty answer), for any *odd* natural number $k \geq 0$, the total number s_k of alternating permutations of the set $\{1, \ldots, k\}$ (putting $s_k = 0$

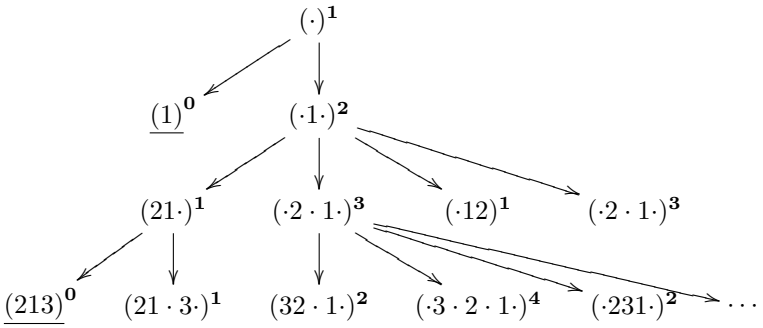
for all even k)? Here is an enumeration of all such odd alternating permutations:



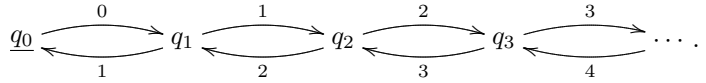
The general pattern according to which this tree is grown is as follows. As before, dots indicate places where further growth is possible. And any dot at level n has two children:



in which the dot has been replaced with $n+1$ or with $n+1$ with dots on both sides. Also as before, states can be identified according to the number of dots they contain:



After identification of all equivalent states, the following automaton is obtained:



Note that in this automaton, the stream of answers is represented by q_1 and not by q_0. Using (a minor variation of) Theorem 4.6.1, we get

$$S(q_1) = \cfrac{X}{1 - \cfrac{1 \cdot 2 \cdot X^2}{1 - \cfrac{2 \cdot 3 \cdot X^2}{1 - \cfrac{3 \cdot 4 \cdot X^2}{\ddots}}}}$$

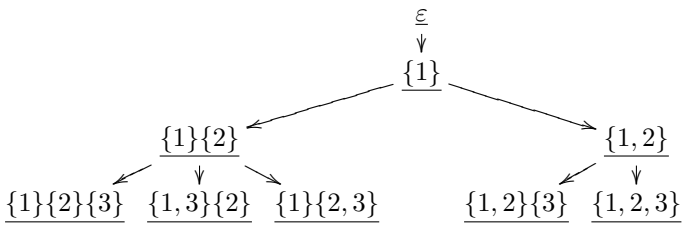
One can show that $S(q_1)$ is equal to $\mathrm{Taylor}\big(\tan(x)\big)$, the stream of Taylor coefficients of the function $\tan(x)$, because the automaton above can also be obtained

by applying the technique of "splitting derivatives" to Taylor$\big(\tan(x)\big)$ (see [**39**] for details).

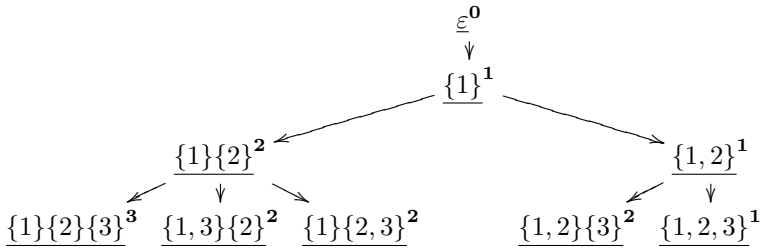
4.9. Set Partitions

Our last series of counting problems deals with set partitions, counted in various ways.

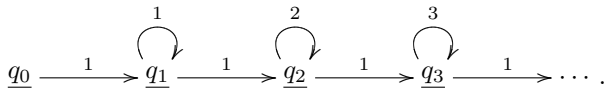
What is, for any $k \geq 0$, the number s_k of ways in which the set $\{1, \ldots, k\}$ can be partitioned into subsets (the so-called Bell numbers)? The following automaton enumerates at level k all such partitions:



States can be identified according to the number of subsets used in the corresponding partition, as indicated by the superscripts below:



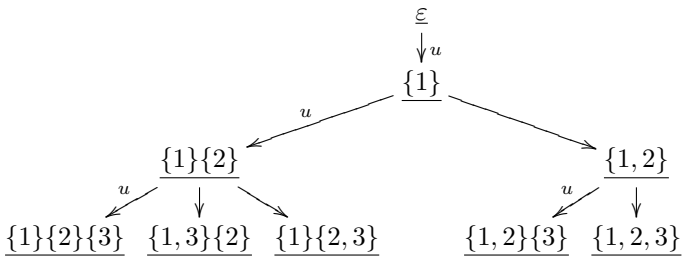
Identifying as usual all equivalent states gives the following reduced automaton:



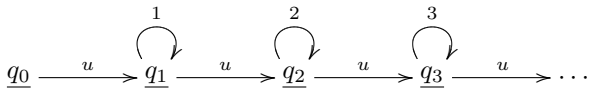
And by formula Theorem 4.6.2, the following expression is obtained for our stream $S(q_0)$ of answers:

$$1 + \frac{X}{1-X} + \frac{X^2}{(1-X)(1-2X)} + \frac{X^3}{(1-X)(1-2X)(1-3X)} + \cdots .$$

It is easy to keep track explicitly of the number of subsets used in any partition, by labelling all transitions in which a new subset is added to the partition by a fixed real number (variable) u:



Identification can be done as before, yielding the following reduced automaton

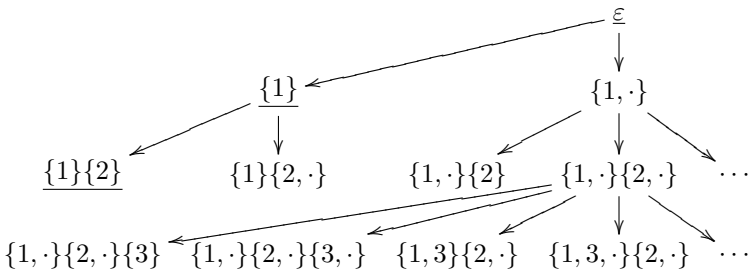


and corresponding expression for the stream $S(q_0)$ of answers:

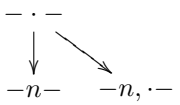
$$1 + \frac{uX}{1-X} + \frac{u^2X^2}{(1-X)(1-2X)} + \frac{u^3X^3}{(1-X)(1-2X)(1-3X)} + \cdots .$$

This stream encodes, for all $k, n \geq 0$, all numbers $s_{k,n}$ of partitioning the set $\{1, \ldots, k\}$ into n subsets. These numbers are called the Stirling numbers of the second kind. Note that by taking $u = 1$, the previous case is found back.

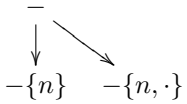
We have seen various examples where a different way of enumeration, using a different representation of the structures to be counted, leads to a different but equivalent expression for the solution of the counting problem. This phenomenon is here further illustrated by the following, alternative solution for the problem of counting set partitions. For any $k \geq 0$, the following automaton contains at level k all partitions of the set $\{1, \ldots, k\}$ as output states; in addition, each level contains various intermediate states containing dots that indicate possibilities for further growth:



The general pattern is as follows: For any dot occurring in a node at level $n-1$, there are two children, obtained by replacing the dot with either n or with n and a dot (allowing still further growth):

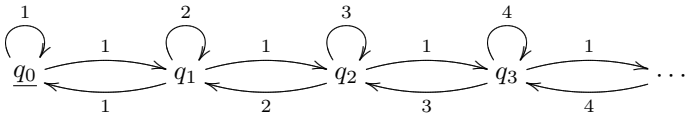


In addition, any node at level $n-1$ has two more children, obtained by opening a new subset containing n, again with or without the presence of a dot:



As in all the previous examples of representations involving dots, states can be identified according to the number of dots they contain, yielding the following

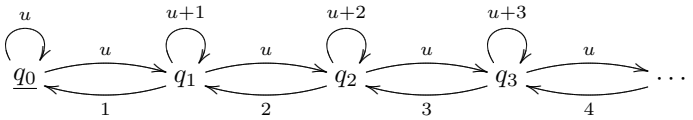
reduced automaton



and, by an application of Theorem 4.6.1, the following corresponding expression for the stream $S(q_0)$ of answers:

$$\cfrac{1}{1-X-\cfrac{X^2}{1-2X-\cfrac{2X^2}{1-3X-\cfrac{3X^2}{\ddots}}}} = 1+\frac{X}{1-X}+\frac{X^2}{(1-X)(1-2X)}+\frac{X^3}{(1-X)(1-2X)(1-3X)}+\cdots$$

where the latter equality is obtained by identifying the first and the second solution of this problem. Again, it is possible to keep track of the number of subsets used in each partition. Omitting the enumerating automaton, we only present the resulting reduced automaton:



and the corresponding expression for the stream $S(q_0)$ of answers:

$$\cfrac{1}{1-(u\times X)-\cfrac{uX^2}{1-\big((u+1)\times X\big)-\cfrac{2uX^2}{1-\big((u+2)\times X\big)-\cfrac{3uX^2}{\ddots}}}}$$

4.10. Special Numbers

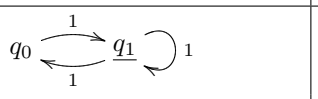
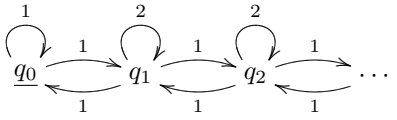
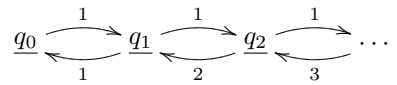
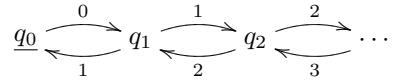
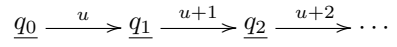
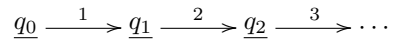
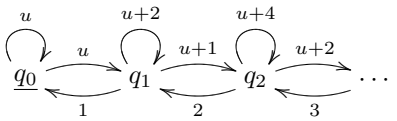
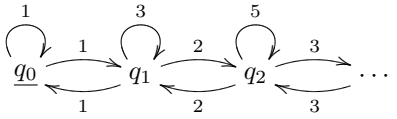
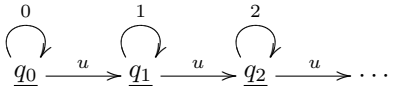
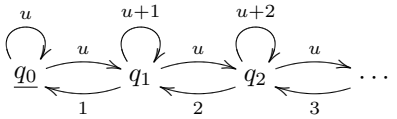
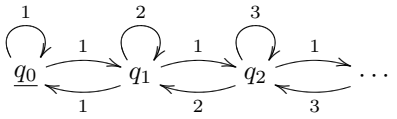
The theorem below summarizes the representations by means of a reduced weighted automaton, of the various sequences of so-called special numbers that we have encountered in the preceding sections. It clearly shows that these representations have a great structural similarity.

THEOREM 4.10.1. *In the Table* 4.1*, the stream of numbers mentioned on the right is represented by the state q_0 of the automaton on the left, except for the stream of the tangent numbers, which is represented by the state q_1.*

4.11. Discussion

For each and every single counting problem that we have dealt with in this chapter, there exist in the literature far more detailed and complete treatments. See for instance the standard works [**43, 44**]. The contribution of the method

TABLE 1. Representations of special numbers.

automaton	represents
	Fibonacci numbers
	Catalan numbers
	involutions
	tangent numbers
	Stirling numbers (1st)
	factorial numbers ($u = 1$)
	Stirling numbers (1st)
	factorial numbers ($u = 1$)
	Stirling numbers (2nd) Bell numbers ($u = 1$)
	Stirling numbers (2nd)
	Bell numbers ($u = 1$)

of coinductive counting lies in the fact that it solves all these different counting problems, in the very same way:

(1) The structures to be counted are always enumerated by infinite tree-shaped weighted automata.
(2) These automata are reduced by means of bisimulations.
(3) From the resulting reduced automata, an expression for the stream of answers is derived by means of elementary stream calculus. Traditionally, the situation with respect to these points is as follows:

(a) Many different mathematical structures are used as representations of combinatorial objects, including weighted graphs, sets of walks, various kinds of trees, formal languages (regular and context-free), automata, (transfer-) matrices, and so on.
(b) We know of no *systematic* way, in the sense of being based on one specific notion, of reducing such representations to smaller or better structured ones.
(c) Stream calculus can be viewed as a formalisation and generalisation of the use of both generating functions and formal power series (cf. [**39**] for a detailed comparison).

Technically, the novelty of the coinductive counting method is based on

(i) the systematic use of *infinite* weighted automata, which have received in the literature on automata theory so far no attention;
(ii) the use of the *quantitative* notion of *stream bisimulation*; and
(iii) the consequent use of a *coinductive* calculus of streams in the search for a closed expression for the outcome.

Inspiring references and closely related approaches include [**20–22, 43, 44**]. In fact, the presently proposed method of coinductive counting is far more restricted in scope than these references. Notably, no (classical) analytical methods are being used here. Such methods might be used, though, at the point where our methods stops: many of the obtained stream expressions are suited for further analytical treatment. The use of continued fractions has been inspired by [**19**], and most of the continued fractions that we have computed occur already there. The formal treatment of such continued fractions seems somewhat easier in the present setting of coinductive stream calculus. Also, some of the combinatorial interpretations of the various fractions discussed here, seem to be simpler and more uniform. Another approach to counting is the categorical theory of species [**10**]. It offers a framework that is far more general than the present calculus of streams, but there are many connections. It would be worthwhile, more generally speaking, to investigate the possible role of coinduction in the world of species in some detail.

CHAPTER 5

Component Connectors

Reo (from the Greek word $\rho\varepsilon\omega$ which means "[I] *flow*") is a recently introduced [**3, 4**] channel-based coordination model, wherein complex coordinators, called connectors, are compositionally built out of simpler ones. Reo is intended as a "glue language" for construction of connectors that orchestrate component instances in a component-based system. The emphasis in Reo is on connectors and their composition only, not on the components that are being connected. Using streams and coinduction, we present in this chapter a simple and transparent semantical model of connectors and connector composition, which can be used as a compositional *calculus*, in which properties such as connector equivalence, optimization, and realization can be expressed and proved. The material in this chapter builds only on Chapter 1.

5.1. Introduction

In Reo, the basic connectors are channels, each of which is a point-to-point communication medium with two distinct ends. Channels can be used as the only communication constructs in communication models for concurrent systems, because the primitives of other communication models (e.g., message passing or remote procedure calls) can be easily defined using channels. In contrast to other channel-based models, Reo uses a generalised concept of channel. In addition to the common channel types of synchronous and asynchronous, with bounded or unbounded buffers, and with fifo and other ordering schemes, Reo allows an open ended set of channels, each with its own, sometimes exotic, behaviour. For instance, a channel in Reo need not have both an input and an output end; it can have two input ends or two output ends instead. In addition to channels, Reo has one more basic connector, called the merge operator. More complex connectors can then be constructed from the basic connectors (channels and merge) through an operation of connector composition.

Because Reo is not concerned with the internal activity of the components that it connects, we represent components by their interfaces only. Therefore, we model the input ends and output ends of connectors as streams of abstract (uninterpreted) data items. Moreover, we associate with every such data stream an infinite sequence of (natural or non-negative real) numbers. These numbers stand for the respective moments in time at which their corresponding data items are being input or output. This allows us to describe and reason about the precise timing constraints of connectors (such as synchronous versus asynchronous, and bounded versus unbounded delay). Thus, we model the potential behaviour of connector ends as *timed data streams*, which are pairs consisting of a data stream and a time stream. Note that we use pairs of streams rather than streams of pairs

(of timed data elements), since this enables us to reason about time explicitly, which turns out to be particularly useful.

Having modelled connector ends as timed data streams, we then model connectors as *relations* on timed data streams, expressing which combinations of timed data streams are mutually consistent. This relational model is, in spite of its simplicity, already sufficiently expressive to study a number of notions and questions about component connectors, such as equivalence (when do two connectors have the same behavior?), expressiveness (which connectors can I build out of a given set of basic connectors?), optimization (given a connector, can I build an equivalent connector out of a smaller number of basic connectors?), verification (given the specification of a certain connector behavior and given a connector, does the connector meet the specification?), realization (given the specification of a certain connector behavior, can I actually build a connector with precisely that behavior out of a given set of basic connectors?), and the like. In this chapter, we shall mainly focus on connector equivalence and, to a lesser extent, the expressiveness of our connector calculus.

5.2. Timed Data Streams

We model connectors as relations on timed data streams, which we introduce in this section.

Let D be an (arbitrary) set, the elements of which will be called *data* elements. The set DS of *data streams* is defined as

$$\mathsf{DS} = \mathsf{D}^\omega$$

that is, the set of all streams $\alpha = \big(\alpha(0), \alpha(1), \alpha(2), \dots\big)$ over D. Let $\mathbb{R}_+$ be the set of non-negative real numbers, which play in the present context the role of *time* moments. Let $\mathbb{R}_+^\omega$ be the set of all streams $a = \big(a(0), a(1), a(2), \dots\big)$ over $\mathbb{R}_+$. Let $<$ and $\leq$ be the relations on $\mathbb{R}_+^\omega$ that are obtained as the pointwise extensions of the corresponding ("strictly smaller" and "smaller than") relations on $\mathbb{R}_+$. That is, for all $a = \big(a(0), a(1), a(2), \dots\big)$ and $b = \big(b(0), b(1), b(2), \dots\big)$ in $\mathbb{R}_+^\omega$,

$$a < b \equiv \forall n \geq 0, \quad a(n) < b(n), \quad a \leq b \equiv \forall n \geq 0, \quad a(n) \leq b(n).$$

The set TS of *time streams* is defined by the following subset of $\mathbb{R}_+^\omega$:

$$\mathsf{TS} = \{a \in \mathbb{R}_+^\omega \mid a < a'\}.$$

Note that time streams $a \in \mathsf{TS}$ satisfy, for all $n \geq 0$,

$$a(n) < a'(n) = a(n+1)$$

and thus consist of increasing time moments $a(0) < a(1) < a(2) < \cdots$.

Finally, the set TDS of *timed data streams* is defined by

$$\mathsf{TDS} = \mathsf{DS} \times \mathsf{TS}$$

and contains pairs $\langle \alpha, a \rangle$ consisting of a data stream

$$\alpha = \big(\alpha(0), \alpha(1), \alpha(2), \dots\big) \in \mathsf{DS}$$

and a time stream

$$a = \big(a(0), a(1), a(2), \dots\big) \in \mathsf{TS}\,.$$

As we shall see shortly, connectors will be modelled as relations on timed data streams. Each of the arguments of such a relation will be viewed as an input or as an output end of the connector that is modelled by the relation. Thus the following

operational interpretation of a timed data stream $\langle \alpha, a \rangle$ can be given: the time stream a specifies for each $n \geq 0$ the time moment $a(n)$ at which the nth data element $\alpha(n)$ is being input or output:

α:	$\alpha(0)$	$\alpha(1)$	$\alpha(2)$	$\cdots$	$\alpha(n)$	$\cdots$
a:	$a(0)$	$a(1)$	$a(2)$	$\cdots$	$a(n)$	$\cdots$

Connectors being relations, there are typically many timings a possible at a specific connector's end, which together with a given data stream α form admissible timed data streams $\langle \alpha, a \rangle$, that is, satisfying the connector's relation. A timed data stream $\langle \alpha, a \rangle$ could therefore also be viewed as a *scenario*, one out of many, for the behaviour of a connector end. Connectors, then, relate various such scenarios that together are mutually consistent.

As we already observed in the introduction, one could have, alternatively and equivalently, defined timed data streams as (a subset of) $(\mathsf{D} \times \mathbb{R}_+)^\omega$, because of the existence of an isomorphism

$$\mathsf{D}^\omega \times \mathbb{R}_+^\omega \cong (\mathsf{D} \times \mathbb{R}_+)^\omega,$$
$$\langle \alpha, a \rangle \mapsto \big(\langle \alpha(0), a(0) \rangle, \langle \alpha(1), a(1) \rangle, \langle \alpha(2), a(2) \rangle, \ldots\big).$$

We prefer to work with pairs of streams rather than streams of pairs, because this will allow us to reason about the data streams and time streams separately, which turns out to be of crucial importance for much of what follows.

We could also have used streams of *natural* numbers $0, 1, 2, \ldots$ for our timings, rather than (positive) real numbers. This difference would leave most of our model unaffected. Our model with "continuous time", however, is more abstract than the model with "discrete time" would be, in the sense that more connector equivalences can be proved. (In the world of temporal logic, this observation goes back to at least [**8**].) An example is the equivalence of a fifo_2 buffer (with capacity 2) with the composition of two fifo_1 buffers in Section 5.5.

Finally, it is often useful to require time streams a to be not only increasing: $a < a'$, but also *progressive*: for every $N \geq 0$ there exists $n \geq 0$ with $a(n) > N$. This assumption prevents "Zeno" paradoxes, where infinitely many actions take place in a bounded time interval. In most of what follows, the progressive time assumption is not used, but whenever it is, we shall mention it explicitly.

5.3. Basic Connectors: Channels

The most basic connectors are *channels*, which are formally defined as binary relations

$$R \subseteq \mathsf{TDS} \times \mathsf{TDS}$$

on timed data streams. For such relations, we distinguish between input and output argument positions, called *input ends* and *output ends*, respectively. This information will be relevant for the definition of connector composition in Section 5.5. In the pictures that we draw of channels and connectors, input and output ends are denoted by the following arrow shaft and head:

input: $\longmapsto\!\!\!\!\!-\!-\!-\cdots$ output: $\cdots\!-\!-\!-\!\!\!\longrightarrow$.

Here are the (for our purposes) most important examples of channels:

• The *synchronous channel* $\longmapsto$ is defined, for all timed data streams $\langle \alpha, a\rangle$ and $\langle \beta, b\rangle$, by

$$\langle \alpha, a\rangle \longmapsto \langle \beta, b\rangle \ \equiv \alpha = \beta \wedge a = b.$$

This channel inputs the data (elements in the) stream α at times a, and outputs the data stream β at times b. All data elements that come in, come out again (in the same order): $\alpha = \beta$. Moreover, each element enters and exits the channel at the very same time moment: $a = b$.

• The *synchronous drain* $\overset{\text{syndr}}{\longmapsto\!\!\!\!\mid}$ is defined, for all timed data streams $\langle \alpha, a\rangle$ and $\langle \beta, b\rangle$, by

$$\langle \alpha, a\rangle \overset{\text{syndr}}{\longmapsto\!\!\!\!\mid} \langle \beta, b\rangle \ \equiv a = b.$$

The corresponding data elements in the streams α and β enter the two input ends of this channel simultaneously: $a = b$. No relation on the data streams is specified (the data elements enter and "disappear").

• The *fifo buffer* $\overset{\text{fifo}}{\longmapsto}$ is defined, for all timed data streams $\langle \alpha, a\rangle$ and $\langle \beta, b\rangle$, by

$$\langle \alpha, a\rangle \overset{\text{fifo}}{\longmapsto} \langle \beta, b\rangle \ \equiv \alpha = \beta \wedge a < b.$$

This is an unbounded fifo (first-in-first-out) buffer. What comes in, comes out (in the same order): $\alpha = \beta$, but later: $a < b$ (which is equivalent to $a(n) < b(n)$, for all $n \geq 0$).

• The *fifo$_1$ buffer* $\overset{\text{fifo}_1}{\longmapsto}$ is defined, for all timed data streams $\langle \alpha, a\rangle$ and $\langle \beta, b\rangle$, by

$$\langle \alpha, a\rangle \overset{\text{fifo}_1}{\longmapsto} \langle \beta, b\rangle \ \equiv \alpha = \beta \wedge a < b < a'.$$

This models a 1-bounded fifo buffer. What comes in, comes out: $\alpha = \beta$, but later: $a < b$. Moreover, at any moment the next data item can be input only after the present data item has been output: $b < a'$, which is equivalent to $b(n) < a(n+1)$, for all $n \geq 0$.

• The *fifo$_k$ buffer* $\overset{\text{fifo}_k}{\longmapsto}$ for any $k \geq 1$, is defined, for all timed data streams $\langle \alpha, a\rangle$ and $\langle \beta, b\rangle$, by

$$\langle \alpha, a\rangle \overset{\text{fifo}_k}{\longmapsto} \langle \beta, b\rangle \ \equiv \alpha = \beta \wedge a < b < a^{(k)}.$$

(Recall from Section 1 that $a^{(k)}$ denotes the kth derivative of the stream a.) This models a k-bounded fifo buffer, generalizing the fifo$_1$ buffer above. What comes in, comes out: $\alpha = \beta$, but later: $a < b$. Moreover, at any moment the kth-next data item can be input only after the present data item has been output: $b < a^{(k)}$ (which is equivalent to $b(n) < a(n+k)$, for all $n \geq 0$).

• Let $x \in D$ be any fixed data element. The *fifo(x) buffer* $\overset{\text{fifo}(x)}{\longmapsto}$ is defined, for all timed data streams $\langle \alpha, a\rangle$ and $\langle \beta, b\rangle$, by

$$\langle \alpha, a\rangle \overset{\text{fifo}(x)}{\longmapsto} \langle \beta, b\rangle \ \equiv \beta(0) = x \wedge \alpha = \beta' \wedge a < b'.$$

This channel behaves precisely as the unbounded fifo buffer above, but for the fact that, initially, it contains the data element x, which is the first element to come out: $\beta(0) = x$ (at time $b(0)$).

5.4. More Channels and the Merge Connector

The definitions of the basic (channel) connectors so far have been, mathematically speaking, fairly straightforward. Next, we introduce some further basic connectors, including the merge operator, using greatest fixed point definitions. As we saw in Section 1.5, each of these definitions will come together with its own proof principle, similar to the coinduction principle in Section 1.

• The *asynchronous drain* $\overset{\text{asyndr}}{\longmapsto}$ inputs any two streams of data items at its two input ends, but never at the same time (in contrast to the synchronous drain of Section 5.3). It is defined, for all timed data streams $\langle \alpha, a\rangle$ and $\langle \beta, b\rangle$, by

$$\langle \alpha, a\rangle \overset{\text{asyndr}}{\longmapsto} \langle \beta, b\rangle \;\equiv a \bowtie b$$

where $\bowtie \subseteq \mathsf{TS} \times \mathsf{TS}$ is a relation on time streams, given by

$$a \bowtie b \equiv \big(a(0) < b(0) \wedge a' \bowtie b\big) \vee \big(b(0) < a(0) \wedge a \bowtie b'\big).$$

More precisely, $\bowtie$ is defined as the greatest fixed point of the following monotone operator, $\Phi_{\bowtie}\colon \mathcal{P}(\mathsf{TS} \times \mathsf{TS}) \to \mathcal{P}(\mathsf{TS} \times \mathsf{TS})$, defined for $R \subseteq \mathsf{TS} \times \mathsf{TS}$, by

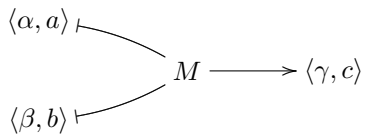
$$\Phi_{\bowtie}(R) = \big\{\langle a, b\rangle \mid \big(a(0) < b(0) \wedge \langle a', b\rangle \in R\big) \vee \big(b(0) < a(0) \wedge \langle a, b'\rangle \in R\big).$$

Thus $\bowtie = \mathrm{gfp}(\Phi_{\bowtie})$. A $\bowtie$-*bisimulation* is a relation $R \subseteq \mathsf{TS} \times \mathsf{TS}$ with $R \subseteq \Phi_{\bowtie}(R)$. There is, as an immediate consequence of (5) in Section 1.5, the following $\bowtie$-*coinduction* proof principle. For all time streams a and b:

(70) if $a\, R\, b$, for some $\bowtie$-bisimulation R, then $\alpha \bowtie \beta$.

For an example of a proof by $\bowtie$-coinduction, see the end of the present section.

• The connector *merge* is a ternary relation M with two input ends and one output end, and is defined, for timed data streams $\langle \alpha, a\rangle$, $\langle \beta, b\rangle$, and $\langle \gamma, c\rangle$, by



$$\equiv M(\langle \alpha, a\rangle, \langle \beta, b\rangle, \langle \gamma, c\rangle)$$

$$\equiv a(0) \neq b(0) \wedge \begin{cases} \alpha(0) = \gamma(0) \wedge a(0) = c(0) \wedge M(\langle \alpha', a'\rangle, \langle \beta, b\rangle, \langle \gamma', c'\rangle) \\ \qquad\qquad \text{if } a(0) < b(0), \\ \beta(0) = \gamma(0) \wedge b(0) = c(0) \wedge M(\langle \alpha, a\rangle, \langle \beta', b'\rangle, \langle \gamma', c'\rangle) \\ \qquad\qquad \text{if } b(0) < a(0). \end{cases}$$

This connector merges the two data streams α and β on its input ends into a stream γ on its output end, on a "first come first served" basis. It inputs one data element at a time: $a(0) \neq b(0)$. The data element that is handled first, say $\alpha(0)$ at time $a(0) < b(0)$, is the first element to come out: $\alpha(0) = \gamma(0)$, at exactly the same moment: $a(0) = c(0)$. After that, the connector handles the remainder of the streams in the same manner again: $M(\langle \alpha', a'\rangle, \langle \beta, b\rangle, \langle \gamma', c'\rangle)$. (Similarly for the case that $\beta(0)$ is handled first, at time $b(0) < a(0)$.) The relation M can be formally defined as the greatest fixed point of a monotone operator Φ_M defined, for

any $R \subseteq \mathsf{TDS} \times \mathsf{TDS} \times \mathsf{TDS}$, by

$$\Phi_M(R)(\langle\alpha, a\rangle, \langle\beta, b\rangle, \langle\gamma, c\rangle)$$
$$\equiv a(0) \neq b(0) \wedge \begin{cases} \alpha(0) = \gamma(0) \wedge a(0) = c(0) \wedge R(\langle\alpha', a'\rangle, \langle\beta, b\rangle, \langle\gamma', c'\rangle) \\ \qquad\qquad \text{if } a(0) < b(0), \\ \beta(0) = \gamma(0) \wedge b(0) = c(0) \wedge R(\langle\alpha, a\rangle, \langle\beta', b'\rangle, \langle\gamma', c'\rangle) \\ \qquad\qquad \text{if } b(0) < a(0). \end{cases}$$

An *M-bisimulation* is a relation R with $R \subseteq \Phi_M(R)$ and we have an *M-coinduction* proof principle: if $R(\langle\alpha, a\rangle, \langle\beta, b\rangle, \langle\gamma, c\rangle)$, for some M-bisimulation R, then $M(\langle\alpha, a\rangle, \langle\beta, b\rangle, \langle\gamma, c\rangle)$.

Under the assumption that our time streams are progressive (defined at the end of Section 5.2), the merge operator is *fair*: from both input ends, infinitely many data elements will be input.

Next, we illustrate the use of the coinduction proof principles that were introduced above. A look at the definition of M and one moment's thought suffice to see that, for all timed data streams $\langle\alpha, a\rangle, \langle\beta, b\rangle, \langle\gamma, c\rangle$,

$$\text{if } M(\langle\alpha, a\rangle, \langle\beta, b\rangle, \langle\gamma, c\rangle) \text{ then } a \bowtie b \tag{71}$$

since the merge connector never inputs two data items at its two input ends at the same time. But how to prove it formally? The answer is provided by what we have called $\bowtie$-coinduction above. Consider the following relation:

$$R = \{\langle k, l\rangle \mid \exists \kappa, , \lambda, \mu, m : M(\langle\kappa, k\rangle, \langle\lambda, l\rangle, \langle\mu, m\rangle)\}.$$

Using the definition of M, it is straightforward to prove that this is a $\bowtie$-bisimulation. As a consequence of (70), $R \subseteq \bowtie$, which implies (71). For a second example, consider the following equivalence, for all timed data streams $\langle\alpha, a\rangle, \langle\beta, b\rangle, \langle\gamma, c\rangle$:

$$M(\langle\alpha, a\rangle, \langle\beta, b\rangle, \langle\gamma, c\rangle) \equiv M(\langle\beta, b\rangle, \langle\alpha, a\rangle, \langle\gamma, c\rangle). \tag{72}$$

The implication from left to right (and thus the equivalence) follows by M-coinduction from the trivial observation that

$$S = \{(\langle\alpha, a\rangle, \langle\beta, b\rangle, \langle\gamma, c\rangle) \mid M(\langle\beta, b\rangle, \langle\alpha, a\rangle, \langle\gamma, c\rangle)\}$$

is an M-bisimulation, which implies $S \subseteq M$.

5.5. Composing Connectors

Connectors are relations and their composition can therefore be naturally modelled by relational composition. For instance, the composition of two copies of the synchronous channel yields the following binary relation, defined for all timed data streams $\langle\alpha, a\rangle$ and $\langle\beta, b\rangle$, by

$$\begin{aligned} \langle\alpha, a\rangle \longmapsto \circ \longmapsto \langle\beta, b\rangle \;&\equiv \exists\langle\gamma, c\rangle\colon\ \langle\alpha, a\rangle \longmapsto \langle\gamma, c\rangle \ \wedge\ \langle\gamma, c\rangle \longmapsto \langle\beta, b\rangle \\ &\equiv \exists\langle\gamma, c\rangle\colon (\alpha = \gamma \wedge a = c) \wedge (\gamma = \beta \wedge c = b) \end{aligned}$$

(which happens to be equivalent to $\langle\alpha, a\rangle \longmapsto \langle\beta, b\rangle \equiv \alpha = \beta \wedge a = b$). Composition essentially does two things at the same time: the output end (argument) of the first connector is identified with the input end of the second, and the resulting

"mixed" end is moreover hidden (encapsulated) by the existential quantification. We shall also use the following picture to denote connector composition:

$$\langle \alpha, a\rangle \longmapsto \exists\langle \gamma, c\rangle \longmapsto \langle \beta, b\rangle \ .$$

Here $\exists\langle \gamma, c\rangle$ is used to indicate that this is an internal end of the connector, which is no longer accessible (for further compositions) from outside. The two aspects of connector composition: identification and hiding, could, and for the full version of Reo actually should, be separated. But for the basic examples we shall be dealing with, this type of composition is sufficient.

Note that the identification of connector ends, which are timed data streams, includes the identification of the respective time streams, thus synchronising the timings of the two connectors.

The general definition of the composition of an arbitrary n-ary connector R and an m-ary connector T, is essentially the same. One has to select a number of (distinct) output ends and input ends from R, and equal numbers of input ends and output ends from T, which then are connected in pairs in precisely the same manner as in the example above. To describe this in full generality, one would have to be slightly more formal and explicit about the (input and output) types of argument positions. Although not very difficult, such a formalisation would not be very interesting. Moreover, it will not be necessary for the instances of connector composition that will be presented here. In all cases, the relevant typing information will be contained in the pictorial representations with which connector compositions will be introduced.

We shall also allow an output end to be connected to *several* input ends at the same time, to each of which the output is copied. Here is an example, in which the output end of a synchronous channel is connected to the input ends of two other synchronous channels:

$$\begin{array}{l} \langle \alpha, a\rangle \longmapsto \exists\langle \delta, d\rangle \longmapsto \langle \beta, b\rangle \\ \qquad\qquad\quad \big\downarrow \\ \qquad\qquad\quad \langle \gamma, c\rangle \\ \qquad\qquad\quad \equiv \exists\langle \delta, d\rangle : (\alpha = \delta \wedge a = d) \wedge (\delta = \beta \wedge d = b) \wedge (\delta = \gamma \wedge d = c) \\ \qquad\qquad\quad \equiv \alpha = \beta = \gamma \wedge a = b = c. \end{array}$$

The connection of several *output ends* to one and the same input end, can be modelled by means of the merge operator introduced in Section 5.4.

Finally, it is relevant to note that nothing in the above prevents us from connecting an output end to an input end of one and the same connector, simply by connecting them to the input end and the output end of a synchronous channel. In other words, we have in passing included the possibility of *feedback* loops into our calculus.

Here are various examples of composite connectors (including examples of feedback) built out of a number of basic connectors. As usual, $\langle \alpha, a\rangle, \langle \beta, b\rangle, \langle \gamma, c\rangle, \ldots$ are arbitrary timed data streams.

• The composition of two unbounded fifo buffers yields again an unbounded fifo buffer:

$$\overset{\text{fifo}}{\longmapsto} \circ \overset{\text{fifo}}{\longmapsto} = \overset{\text{fifo}}{\longmapsto}$$

because of the following equivalences:

$$\begin{aligned}\langle\alpha,a\rangle \overset{\text{fifo}}{\longmapsto} \circ \overset{\text{fifo}}{\longmapsto} \langle\beta,b\rangle \;&\equiv\; \langle\alpha,a\rangle \overset{\text{fifo}}{\longmapsto} \exists\langle\gamma,c\rangle \overset{\text{fifo}}{\longmapsto} \langle\beta,b\rangle\\ &\equiv \exists\langle\gamma,c\rangle : \alpha=\gamma\wedge a<c\wedge\gamma=\beta\wedge c<b\\ &\equiv \alpha=\beta\wedge a<b\\ &\equiv\; \langle\alpha,a\rangle \overset{\text{fifo}}{\longmapsto} \langle\beta,b\rangle\end{aligned}$$

- The composition of two fifo_1 buffers yields a fifo_2 buffer:

$$\overset{\text{fifo}_1}{\longmapsto} \circ \overset{\text{fifo}_1}{\longmapsto} = \overset{\text{fifo}_2}{\longmapsto}$$

because

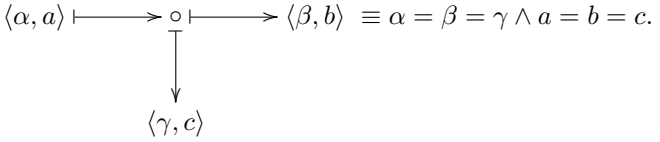
$$\begin{aligned}&\langle\alpha,a\rangle \overset{\text{fifo}_1}{\longmapsto} \circ \overset{\text{fifo}_1}{\longmapsto} \langle\beta,b\rangle\\ &\qquad\equiv\; \langle\alpha,a\rangle \overset{\text{fifo}_1}{\longmapsto} \exists\langle\gamma,c\rangle \overset{\text{fifo}_1}{\longmapsto} \langle\beta,b\rangle\\ &\qquad\equiv \exists\langle\gamma,c\rangle : \alpha=\gamma\wedge a<ca'\wedge\gamma=\beta\wedge c<b<c'\\ &\qquad\Rightarrow \alpha=\beta\wedge a<b<a''=a^{(2)} \text{since } c<a' \text{ implies } c'<a''\\ &\qquad\equiv\; \langle\alpha,a\rangle \overset{\text{fifo}_2}{\longmapsto} \langle\beta,b\rangle\ .\end{aligned}$$

Given $\alpha=\beta$ and $a<b<a''$, the converse of the above implication can be proved by defining $\gamma=\alpha$ and

$$c(n)=\tfrac{1}{2}\times\big(\max\{a(n),b(n-1)\}+\min\{a(n+1),b(n)\}\big)$$

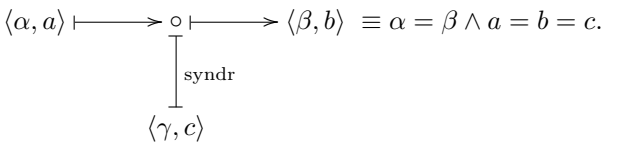
for all $n\geq 0$ (where $b(n-1)=0$ for $n=0$).

- Consider the following composition of three synchronous channels:



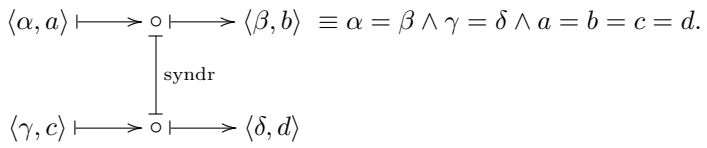
This connector can be viewed as a "take-cue" regulator: any time a data item is taken from γ (by some future context), that same data item flows from left to right. This constitutes one of the most basic examples of what could be called *exogenous coordination*, that is, coordination from outside.

- The following connector is a variation on the previous one, in that the lower channel now is a synchronous drain:



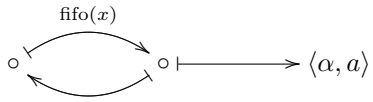
It is a "write-cue" regulator that regulates the flow of data items from left to right by inputs or writes on the lower channel end. Note that what is being input there is irrelevant. What matters is that such inputs are synchronised, through the synchronous drain, with both the channels above: $a=b=c$.

• With four synchronous channels and one synchronous drain, the following barrier synchroniser can be constructed:



The synchronous drain in the middle ensures that data items pass through the upper and lower channels simultaneously.

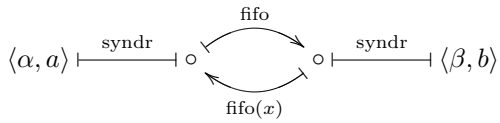
• Here is a simple example of a feedback loop, consisting of one (unbounded) fifo buffer containing an initial data element $x \in D$, and two synchronous channels:

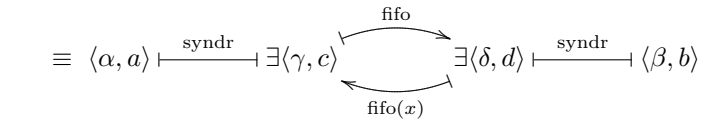


$$
\begin{aligned}
&\equiv \exists\langle \beta, b\rangle \exists\langle \gamma, c\rangle \text{ (fifo}(x)\text{)} \vdash \langle \alpha, a\rangle \\
&\equiv \exists\langle \beta, b\rangle\exists\langle \gamma, c\rangle : \big(\gamma(0) = x \wedge \beta = \gamma' \wedge b < c'\big) \wedge (\gamma = \beta \wedge c = b) \wedge (\gamma = \alpha \wedge c = a) \\
&\equiv \exists\langle \beta, b\rangle\exists\langle \gamma, c\rangle : \alpha = \beta = \gamma \wedge a = b = c \wedge \gamma(0) = x \wedge \gamma = \gamma' \wedge c < c' \\
&\equiv \exists\langle \beta, b\rangle\exists\langle \gamma, c\rangle : \alpha = \beta = \gamma \wedge a = b = c \wedge \gamma = (x, x, x, \dots) \\
&\equiv \alpha = (x, x, x, \dots).
\end{aligned}
$$

For the last but one equivalence, note that $\big(\gamma(0) = x \wedge \gamma = \gamma'\big)$ is equivalent to $\gamma = (x, x, x, \dots)$; moreover, the inequality $c < c'$ is redundant, because c is by assumption a time sequence and hence satisfies this inequality by definition. The behaviour of this connector is thus pretty much what it should be. It outputs perpetually the data element x. Note that there are no constraints on the time stream a $(= b = c)$. The only requirement is that it is indeed a time stream, that is, satisfies $a < a'$.

• Let $x \in D$ be again some fixed data element. The following connector acts as a *sequencer* on its two connector ends:





$$
\begin{aligned}
&\equiv \exists\langle \gamma, c\rangle\exists\langle \delta, d\rangle : a = c \wedge d = b \wedge (\gamma = \delta \wedge c < d) \wedge \big(\gamma(0) = x \wedge \gamma' = \delta \wedge d < c'\big) \\
&\equiv \exists\langle \gamma, c\rangle\exists\langle \delta, d\rangle\gamma = \delta = (x, x, x, \dots) \wedge c = a \wedge d = b \wedge (c < d < c') \\
&\equiv a < b < a'.
\end{aligned}
$$

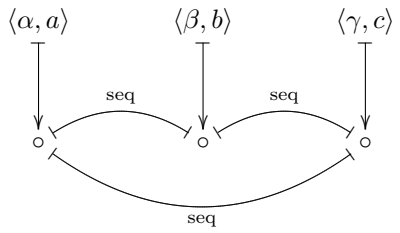
Thus, arbitrary data elements can be input alternately from the left and the right channel ends, at times

$$a(0) < b(0) < a(1) < b(1) < \cdots .$$

For future reference, we introduce the following notation for the sequencer connector:

$$\langle \alpha, a\rangle \overset{\text{seq}}{\longmapsto} \langle \beta, b\rangle \ \equiv a < b < a'. \tag{73}$$

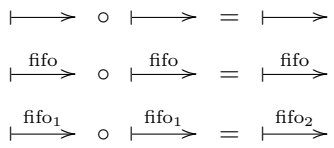
• One can construct a connector that serialises any number k of channel ends by combining $k+1$ sequencers. For instance,



$$\equiv (a < b < a') \wedge (a < c < a') \wedge (b < c < b')$$
$$\equiv a < b < c < a'.$$

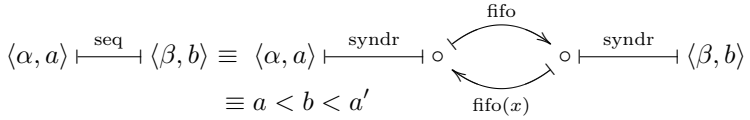
5.6. Connector Equivalence

In Section 5.5, we already saw some elementary examples of connector equivalence, such as:



Below we present some further, slightly less elementary examples.

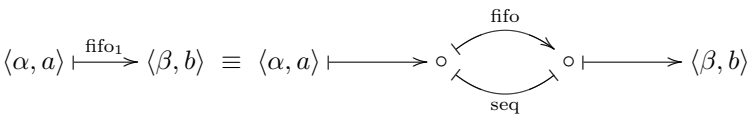
(1) Recall the definition of the sequencer in Section 5.5:



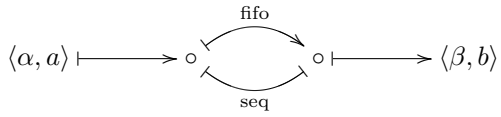
(where $x \in D$ is some fixed data item). Here is an alternative way of constructing the sequencer, now with a 1-bounded fifo buffer and a synchronous drain:

$$\begin{aligned}\langle \alpha, a\rangle \overset{\text{fifo}_1}{\longmapsto} \circ \overset{\text{syndr}}{\longmapsto} \langle \beta, b\rangle \ &\equiv \ \langle \alpha, a\rangle \overset{\text{fifo}_1}{\longmapsto} \exists \langle \gamma, c\rangle \overset{\text{syndr}}{\longmapsto} \langle \beta, b\rangle \\ &\equiv \exists \langle \gamma, c\rangle : (\alpha = \gamma \wedge a\langle c < a') \wedge c = b \\ &\equiv a < b < a' \\ &\equiv \ \langle \alpha, a\rangle \overset{\text{seq}}{\longmapsto} \langle \beta, b\rangle \ .\end{aligned}$$

(2) Conversely, a 1-bounded fifo buffer can be constructed using two synchronous channels, a sequencer, and an unbounded fifo buffer:



because we have the following equivalences:



$$\equiv \langle\alpha, a\rangle \longmapsto \exists\langle\gamma, c\rangle \overset{\text{fifo}}{\underset{\text{seq}}{\rightleftarrows}} \exists\langle\delta, d\rangle \longmapsto \langle\beta, b\rangle$$

$$\equiv \exists\langle\gamma, c\rangle, \exists\langle\delta, d\rangle : \langle\alpha, a\rangle = \langle\gamma, c\rangle \wedge \langle\delta, d\rangle = \langle\beta, b\rangle \wedge (\gamma = \delta \wedge c < d) \wedge (c < d < c')$$

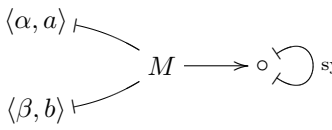
$$\equiv \alpha = \beta \wedge a < b < a'$$

$$\equiv \langle\alpha, a\rangle \overset{\text{fifo}_1}{\longmapsto} \langle\beta, b\rangle \ .$$

(3) Recall the definition of asynchronous drain in Section 5.4:

$$\langle\alpha, a\rangle \overset{\text{asyndr}}{\longmapsto\!\!\!\!\mid} \langle\beta, b\rangle \ \equiv a \bowtie b.$$

Here is another way of constructing the asynchronous drain, using the merge connector and a synchronous drain:



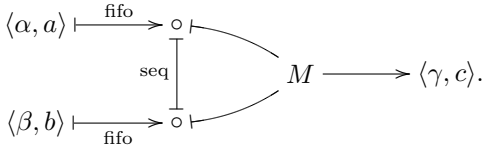
$$\equiv \exists\langle\gamma, c\rangle : M(\langle\alpha, a\rangle, \langle\beta, b\rangle, \langle\gamma, c\rangle) \wedge c = c$$

$$\equiv a \bowtie b$$

$$\equiv \langle\alpha, a\rangle \overset{\text{asyndr}}{\longmapsto\!\!\!\!\mid} \langle\beta, b\rangle \ .$$

For the middle equivalence, the implication from left to right follows from (71) in Section 5.4, which was proved by $\bowtie$-coinduction. The converse implication can be proved in a similar fashion, using M-coinduction.

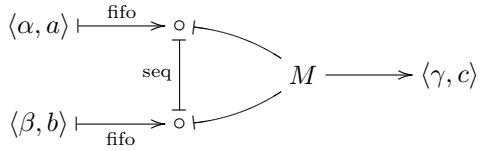
(4) Next we look at a connector that is built from two unbounded fifo buffers, the sequencer, and the merge operator:



Using the coinduction proof principle, we shall prove that this connector has the following behaviour (with the operations zip, even and odd as in Section 1):

$$\text{zip}(\alpha, \beta) = \gamma \wedge a < \text{even}(c) \wedge b < \text{odd}(c).$$

The proof consists of the following sequence of equivalences:



$$
\begin{aligned}
&\equiv \langle \alpha, a\rangle \xrightarrow{\text{fifo}} \exists\langle\delta, d\rangle \quad \text{(seq)} \quad M \longrightarrow \langle\gamma, c\rangle \quad \langle\beta, b\rangle \xrightarrow[\text{fifo}]{} \exists\langle\varepsilon, e\rangle \\
&\equiv \exists\langle\delta, d\rangle\exists\langle\varepsilon, e\rangle : (\alpha = \delta \wedge a < d) \wedge (\beta = \varepsilon \wedge b < e) \wedge (d < e < d') \\
&\qquad\qquad \wedge M(\langle\delta, d\rangle, \langle\varepsilon, e\rangle, \langle\gamma, c\rangle) \\
&\equiv \exists d, e : a < d \wedge b < e \wedge (d < e < d') \wedge M(\langle\alpha, d\rangle, \langle\beta, e\rangle, \langle\gamma, c\rangle) \\
&\equiv \exists d, e : a < d \wedge b < e \wedge \mathrm{zip}(\alpha, \beta) = \gamma \wedge \mathrm{zip}(d, e) = c \text{[using (74) below]} \\
&\equiv \exists d, e : a < d \wedge b < e \wedge \mathrm{zip}(\alpha, \beta) = \gamma \wedge d = \mathrm{even}(c) \wedge e = \mathrm{odd}(c) \\
&\equiv \mathrm{zip}(\alpha, \beta) = \gamma \wedge a < \mathrm{even}(c) \wedge b < \mathrm{odd}(c).
\end{aligned}
$$

We have used the following equivalence, which will be proved by coinduction:

$$(74) \qquad (d < e < d') \wedge M(\langle\alpha, d\rangle, \langle\beta, e\rangle, \langle\gamma, c\rangle) \equiv \big(\mathrm{zip}(\alpha, \beta) = \gamma \wedge \mathrm{zip}(d, e) = c\big).$$

For the implication from left to right, define the following relation on streams:

$$R = \langle \mathrm{zip}(\kappa, \lambda), \mu\rangle \mid \exists k, l, m : (k < l < k') \wedge M(\langle\kappa, k\rangle, \langle\lambda, l\rangle, \langle\mu, m\rangle).$$

We show that R is a bisimulation. Consider a pair $\langle \mathrm{zip}(\kappa, \lambda), \mu\rangle$ in R, with corresponding time streams k, l, m. Because $k < l$ it follows from $M(\langle\kappa, k\rangle, \langle\lambda, l\rangle, \langle\mu, m\rangle)$ that $\mu(0) = \kappa(0)$, and since $\kappa(0) = \mathrm{zip}(\kappa, \lambda)(0)$, this proves the first of the two bisimulation conditions. Next consider the pair of derivatives $\langle \mathrm{zip}(\kappa, \lambda)', \mu'\rangle = \langle \mathrm{zip}(\lambda, \kappa'), \mu'\rangle$. It follows from the definition of M that the latter pair is again in R, since

$$
\begin{aligned}
&(k < l < k') \wedge M(\langle\kappa, k\rangle, \langle\lambda, l\rangle, \langle\mu, m\rangle) \\
&\Rightarrow (l < k' < l') \wedge M(\langle\kappa', k'\rangle, \langle\lambda, l\rangle, \langle\mu', m'\rangle) \equiv (l < k' < l') \wedge M(\langle\lambda, l\rangle, \langle\kappa', k'\rangle, \langle\mu', m'\rangle) \\
&\qquad\qquad \text{[by equivalence (72)]}.
\end{aligned}
$$

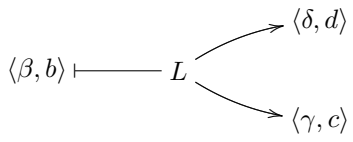
This proves that R is a bisimulation. Assuming now $(d < e < d')$ and $M(\langle\alpha, d\rangle, \langle\beta, e\rangle, \langle\gamma, c\rangle)$, $\mathrm{zip}(\alpha, \beta) = \gamma$ follows by coinduction. In the same manner, one shows $\mathrm{zip}(d, e) = c$. This proves the implication from left to right of equivalence (74). The implication from right to left can be proved along similar lines, using M-coinduction.

5.7. Protocol Verification

Our calculus of component connectors also allows the formulation and formal verification of communication protocols. We present a simple example, taken from [**13**, pp. 29–36]. It consists of an (unbounded fifo) lossy buffer composed with a driver that corrects the lossiness of the buffer. Below we specify both connectors,

and prove that their composition is equivalent to an ordinary (correct) unbounded fifo buffer.

Let $\langle \alpha, a\rangle$, $\langle \beta, b\rangle$, and $\langle \delta, d\rangle$ be timed data streams over an arbitrary data set D and let $\langle \gamma, c\rangle$ be a timed data stream with $\gamma \in \{0,1\}^\omega$. We define the lossy buffer as a ternary relation L on timed data streams with one input end and two output ends as follows:



$$\equiv L(\langle \beta, b\rangle, \langle \gamma, c\rangle, \langle \delta, d\rangle)$$

$$\equiv b < c \wedge b < d \wedge \begin{cases} \gamma(0) = 1 \wedge \delta(0) = \beta(0) \wedge L(\langle \beta', b'\rangle, \langle \gamma', c'\rangle, \langle \delta', d'\rangle) \\ \vee \gamma(0) = 0 \wedge L(\langle \beta', b'\rangle, \langle \gamma', c'\rangle, \langle \delta, d\rangle) \end{cases}$$

This connector inputs data items at the input end β. For every data item that is input, there are two possible scenarios:

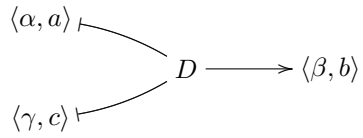
(1) the data item is stored successfully and is output (at some later moment) at the upper output end δ together (not necessarily simultaneously) with a success signal 1 along γ, after which the connector proceeds as before with the remainder of all streams involved;
(2) storage of the data item fails, no data item is output along the end β, a 0 signalling the failure is output along γ, and the connector proceeds as before, now with $\langle \beta', b'\rangle$ and $\langle \gamma', c'\rangle$, but with $\langle \delta, d\rangle$ unchanged.

It follows from the definition of the lossy buffer that eventually some data item gets successfully stored and output. In other words, there exists $n \geq 0$ with $\gamma(n) = 1$. In order to prove this, assume $L(\langle \beta, b\rangle, \langle \gamma, c\rangle, \langle \delta, d\rangle)$ and suppose that $\gamma(n) = 0$, for all $n \geq 0$. Then

$$L(\langle \beta^{(n)}, b^{(n)}\rangle, \langle \gamma^{(n)}, c^{(n)}\rangle, \langle \delta, d\rangle)$$

for all $n \geq 0$ (recall that the superscript (n) stands for the nth derivative). As a consequence, $b^{(n)}(0) = b(n) < d(0)$, for all n. Under the assumption that our time streams are progressive (cf. the end of Section 5.2): for any $N \geq 0$ there exists $n \geq 0$ with $b(n) > N$, this is a contradiction. Therefore, there exists $n \geq 0$ with $\gamma(n) = 1$. We see that, somewhat surprisingly, the fact that all our streams are infinite and the assumption that time streams are progressive, together imply here the right type of fairness (or liveness) behaviour.

Next we turn to the driver, which has two input ends and one output end, and is defined as the following ternary relation D:

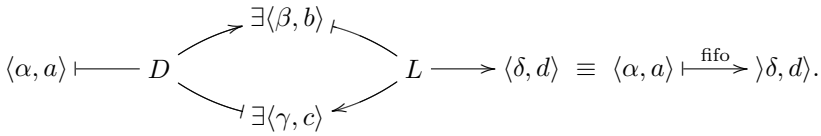


$$\equiv D(\langle \alpha, a\rangle, \langle \beta, b\rangle, \langle \gamma, c\rangle)$$

$$\equiv a = b \wedge a < c < a' \wedge \beta(0) = \alpha(0) \wedge \begin{cases} \gamma(0) = 1 \wedge D(\langle \alpha', a'\rangle, \langle \beta', b'\rangle, \langle \gamma', c'\rangle) \\ \vee \gamma(0) = 0 \wedge D(\langle \alpha, a'\rangle, \langle \beta', b'\rangle, \langle \gamma', c'\rangle) \end{cases}$$

The driver inputs data items at α and outputs them at β. Before proceeding with the next data item, it checks its input at γ. If $\gamma(0) = 1$ then the last data item that has been output is considered to have been handled correctly (by the lossy buffer in the composition below), and D proceeds as before with the remainder of all streams involved. If $\gamma(0) = 0$, however, something has gone wrong (the buffer has lost the data item), and D sends the data item again. This is modelled here by $D(\langle\alpha, a'\rangle, \langle\beta', b'\rangle, \langle\gamma', c'\rangle)$, in which all streams have progressed to their derivatives but for α, which remains unchanged. As a consequence, $\alpha(0)$ is (again) the next data item that D will output (but note that the time stream a *has* changed into a').

Composing the driver and the lossy buffer as below yields a connector that is equivalent with the (non-lossy) unbounded fifo buffer: for all timed data streams $\langle\alpha, a\rangle$ and $\langle\delta, d\rangle$,



For the implication from left to right, we have to show that $\alpha = \delta$ (and $a < d$). To this end, define the following relation on data streams:

$$R = \{\langle\alpha, \delta\rangle \mid \exists a, d, \langle\beta, b\rangle, \langle\gamma, c\rangle : D(\langle\alpha, a\rangle, \langle\beta, b\rangle, \langle\gamma, c\rangle) \wedge L(\langle\beta, b\rangle, \langle\gamma, c\rangle, \langle\delta, d\rangle)\}.$$

In order to prove that R is a bisimulation relation, consider a pair $\langle\alpha, \delta\rangle$ in R with "witnesses" $a, d, \langle\beta, b\rangle, \langle\gamma, c\rangle$ such that $D(\langle\alpha, a\rangle, \langle\beta, b\rangle, \langle\gamma, c\rangle)$ and $L(\langle\beta, b\rangle, \langle\gamma, c\rangle, \langle\delta, d\rangle)$. Let n be the smallest natural number such that $\gamma(n) = 1$ (which exists by the remark above). It follows that

$$D(\langle\alpha, a^{(n)}\rangle, \langle\beta^{(n)}, b^{(n)}\rangle, \langle\gamma^{(n)}, c^{(n)}\rangle) \wedge L(\langle\beta^{(n)}, b^{(n)}\rangle, \langle\gamma^{(n)}, c^{(n)}\rangle, \langle\delta, d\rangle).$$

Together with $\gamma^{(n)}(0) = \gamma(n) = 1$, this implies $\alpha(0) = \beta^{(n)}(0) = \delta(0)$ and, moreover,

$$D(\langle\alpha', a^{(n+1)}\rangle, \langle\beta^{(n+1)}, b^{(n+1)}\rangle, \langle\gamma^{(n+1)}, c^{(n+1)}\rangle) \\ \wedge L(\langle\beta^{(n+1)}, b^{(n+1)}\rangle, \langle\gamma^{(n+1)}, c^{(n+1)}\rangle, \langle\delta', d'\rangle).$$

Thus, $\langle\alpha', \delta'\rangle \in R$, which concludes the proof that R is a bisimulation. It now follows by coinduction that $\alpha = \delta$. (A minor variation on this argument proves that $a < d$.)

For the implication from right to left, choose $\langle\beta, b\rangle = \langle\alpha, a\rangle$, $\gamma = (1, 1, 1, \dots)$, and $c = \frac{1}{2} \times (a + a')$ (in a hopefully self explanatory notation). It is now a straightforward proof by (D- and L-)coinduction to show that $D(\langle\alpha, a\rangle, \langle\beta, b\rangle, \langle\gamma, c\rangle)$ and $L(\langle\beta, b\rangle, \langle\gamma, c\rangle, \langle\delta, d\rangle)$.

5.8. Discussion

Reo is more general than data-flow models, Kahn-networks, and Petri-nets, which can be viewed as specialised channel-based models that incorporate certain basic constructs for primitive coordination. While Reo is designed to deal with the flow of data, it, more specifically, differs fundamentally from classical data-flow models in four important aspects:

(1) Although not treated here, the topology of connections in "full" Reo is inherently dynamic and accommodates mobility.
(2) Reo supports a much more general notion of channels.
(3) The model of Reo is based on a clear separation of data and time.
(4) Coinduction is the main reasoning principle.

More details about Reo can be found in [**3, 4**]. The model presented in this chapter is described in [**5**]. Of related work, Broy and Stølen's book [**13**] deserves special mention, since it is also based on (timed) data streams. However, the points mentioned above distinguish our model also from theirs. In particular, the separation of data and time, in our model, in combination with the use of coinduction, leads to simpler specifications (definitions) and proofs. See Section 5.7 for a concrete example. Finally, coalgebra and coinduction have been used in models of component-based systems in [**7, 17**]. Also these models are distinguished from ours by the (first three) points above. Moreover, our model is far more concrete, and therefore allows actual equivalence proofs.

Our work on Reo and this model is on-going. One of the first questions to address is to decide what set(s) of basic channels and connectors to choose as the basis for a connector calculus (or calculi). Another plan is to look at more instances of connector protocol verification. On the basis of the example of Section 5.7 (and other examples not included in the present chapter, such as the alternating bit protocol), we expect that the present model will be competitive with both traditional data-flow networks and with process algebra, by combining the best of those two worlds. Like data flow and unlike process algebra, Reo is channel-based and models the (communication) topology of connectors explicitly. Like process algebra and unlike data flow, (our model of) Reo is a calculus in which complex connectors are compositionally built out of simpler ones. Moreover, unlike data flow, the model for Reo that we presented is both simple and formal enough to allow actual verification. And unlike process algebra, there is no need to use nondeterministic transition systems and computationally complicated notions such as weak or branching bisimulation. Instead, streams and coinduction are all that is needed.

APPENDIX A

Key Differential Equations

The following table contains the most important behavioural differential equations used in this book:

derivative	initial value	side conditions
$[r]' = [0]$	$[r](0) = r$	
$X' = [1]$	$X(0) = 0$	
$(\sigma + \tau)' = \sigma' + \tau'$	$(\sigma + \tau)(0) = \sigma(0) + \tau(0)$	
$(\sigma \times \tau)' = (\sigma' \times \tau) + (\sigma(0) \times \tau')$	$(\sigma \times \tau)(0) = \sigma(0) \times \tau(0)$	
$(\sigma^{-1})' = -\sigma(0)^{-1} \times (\sigma' \times \sigma^{-1})$	$(\sigma^{-1})(0) = \sigma(0)^{-1}$	$\sigma(0) \neq 0$
$(\sigma \circ \tau)' = \tau' \times (\sigma' \circ \tau)$	$(\sigma \circ \tau)(0) = \sigma(0)$	$\tau(0) = 0$
$\left(\sum_{n=0}^{\infty} \sigma_n\right)' = \sum_{n=0}^{\infty} (\sigma_n')$	$\left(\sum_{n=0}^{\infty} \sigma_n\right)(0) = \sum_{n=0}^{\infty} \sigma_n(0)$	$\{\sigma_n\}_{n=0}^{\infty}$ is summable
$(\sigma \otimes \tau)' = (\sigma' \otimes \tau) + (\sigma \otimes \tau')$	$(\sigma \otimes \tau)(0) = \sigma(0) \times \tau(0)$	
$(\sigma^{-\underline{1}})' = -\sigma' \otimes (\sigma^{-\underline{1}} \otimes \sigma^{-\underline{1}})$	$\sigma^{-\underline{1}}(0) = \sigma(0)^{-1}$	$\sigma(0) \neq 0$
$\exp(\sigma)' = \sigma' \otimes \exp(\sigma)$	$\exp(\sigma)(0) = e^{\sigma(0)}$	
$\Lambda_C(\sigma)' = \Lambda_C\left(\frac{d\sigma}{dX}\right)$	$\Lambda_C(\sigma)(0) = \sigma(0)$	
$\mathcal{T}(f)' = \mathcal{T}(f')$	$\mathcal{T}(f)(0) = f(0)$	f is analytical
$(\sqrt{\sigma})' = \frac{\sigma'}{\sqrt{\sigma(0)} + \sqrt{\sigma}}$	$\sqrt{\sigma}(0) = \sqrt{\sigma(0)}$	$\sigma(0) > 0$

Bibliography

1. P. Aczel, *Non-well-founded sets*, CSLI Lecture Notes, vol. 14, Center for the Study of Language and Information, Stanford University, Stanford, CA, 1988.
2. P. Aczel and N. Mendler, *A final coalgebra theorem*, Category Theory and Computer Science (Manchester, 1989), (D. H. Pitt, D. E. Ryeheard, P. Dybjer, A. M. Pitts, and A. Poigne, eds.), Lecture Notes in Comput. Sci., vol. 389, 1989, Springer, Berlin, pp. 357–365.
3. F. Arbab, *A channel-based coordination model for component composition*, Report SEN-R0203, CWI, 2002; `www.cwi.nl`.
4. F. Arbab and F. Mavaddat, *Coordination through channel composition*, 5th Internat. Conf. on Coordination Models and Languages (F. Arbab and C. Talcott, eds.), Lecture Notes in Comput. Sci., vol. 2315, Springer-Verlag, London, UK, 2002, pp. 22–39.
5. F. Arbab and J. J. M. M. Rutten, *A coinductive calculus of component connectors*, Proceedings of WADT 2002 (to appear); Report SEN-R0216, CWI, 2002 `www.cwi.nl`.
6. M. A. Arbib and E. G. Manes, *Machines in a category*, J. Pure Appl. Algebra **19** (1980), 9–20.
7. L. Barbosa, *Components as coalgebras*, Ph.D. thesis, Universidade do Minho, Braga, Portugal, 2001.
8. H. Barringer, R. Kuiper, and A. Pnueli, *A really abstract concurrent model and its temporal logic*, 13th Annual ACM Symposium on Principles of Programming Languages, ACM Press, New York, USA, 1986, pp. 173–183.
9. F. Bartels, *Generalised coinduction*, Report SEN-R0043, CWI, 2000; `www.cwi.nl`.
10. F. Bergeron, G. Labelle, and P. Leroux, *Combinatorial species and tree-like structures*, Encyclopedia Math. Appl., vol. 67, Cambridge University Press, Cambridge, 1998.
11. J. Berstel and C. Reutenauer, *Rational series and their languages*, Monogr. Theoret. Comput. Sci. EATCS Ser., vol. 12, Springer-Verlag, Berlin, 1988.
12. G. Birkhoff and G.-C. Rota, *Ordinary differential equations*, 3rd ed., John Wiley & Sons, New York–Chichester–Brisbane, 1978.
13. M. Broy and K. Stølen, *Specification and development of interactive systems*, Monogr. Comput. Sci., vol. 62, Springer-Verlag, New York, 2001.
14. J. A. Brzozowski, *Derivatives of regular expressions*, J. ACM **11** (1964), no. 4, 481–494.
15. L. Comtet, *Advanced combinatorics. The art of finite and infinite expansions*, D. Reidel Publishing Co., Dordrecht, 1974.
16. J. H. Conway, *Regular algebra and finite machines*, Chapman and Hall, 1971.
17. E.-E. Doberkat, *Pipes and filters*: *Modelling a software architecture through relations*, Report 123, Chair for software technology, University of Dortmund, 2002.
18. M. Forti and F. Honsell, *Set theory with free construction principles*, Ann. Scuola Norm. Sup. Pisa Cl. Sci. (4) **10** (1983), no. 3, 493–522.
19. P. Flajolet, *Combinatorial aspects of continued fractions*, Discrete Math. **32** (1980), 125–161.
20. P. Flajolet and R. Sedgewick, *The average case analysis of algorithms*: *Counting and generating functions*, Research Report 1888, INRIA Rocquencourt, 1993.
21. P. Flajolet and R. Sedgewick, *Analytical combinatorics*: *Functional equations, rational and algebraic functions*, Research Report 4103, INRIA Rocquencourt, 2001.
22. R. L. Graham, D. E. Knuth, and O. Patashnik. *Concrete mathematics. A foundation for computer science*, 2nd ed., Addison–Wesley Publishing Company, Reading, MA, 1994.
23. Bart Jacobs and J. J. M. M. Rutten, *A tutorial on (co)algebras and (co)induction*, Bull. Eur. Assoc. Theor. Comput. Sci. EATCS **62** (1997), 222–259; `www.cwi.nl/~janr`.

24. J. Karczmarczuk, *Generating power of lazy semantics*, Theoret. Comput. Sci. **187** (1997), 203–219.
25. ———, *Lazy processing and optimization of discrete sequences*, JFLA'2000, 2000 (French) (to appear); `http://www.info.unicaen.fr/ karczma`.
26. D. C. Kozen, *Automata and computability*, Undergrad. Texts in Comput. Sci., Springer-Verlag, New York, 1997.
27. E. G. Manes and M. A. Arbib, *Algebraic approaches to program semantics*, Texts Monogr. Comput. Sci., AKM Series in Theoretical Computer Science, Springer-Verlag, New York, 1986.
28. M. D. McIlroy, *Power series, power serious*, J. Funct. Programming **9** (1999), 323–335.
29. ———, *The music of streams*, Inform. Process. Lett. **77** (2001), 189–195.
30. J. Mikusinski, *Operational calculus.* I, 2nd ed., Int. Ser. Monogr. Pure Appl. Math., vol. 109, Pergamon Press, Oxford; PWN—Polish Scientific Publishers, Warsaw, 1983.
31. R. Milner, *A calculus of communicating systems*, Lecture Notes Comput. Sci., vol. 92, Springer-Verlag, Berlin-New York, 1980.
32. ———, *Communication and concurrency*, Prentice Hall, 1989.
33. D. M. R. Park, *Concurrency and automata on infinite sequences*, 5th GI Conference (Karlsruhe, Germany) (P. Deussen, ed.), Lecture Notes in Comput. Sci., vol. 104, Springer-Verlag, 1981, pp. 167–183.
34. D. Pavlović and M. Escardó, *Calculus in coinductive form*, 13th Annual IEEE Symposium on Logic in Computer Science (Indianapolis, 1998), IEEE Computer Soc., Los Alamitos, 1998, pp. 408–417.
35. J. J. M. M. Rutten, *Automata and coinduction* (*an exercise in coalgebra*), CONCUR'98: Concurrency Theory (Nice) (D. Sangiorgi and R. de Simone, eds.), Lecture Notes in Comput. Sci., vol. 1466, Springer, Berlin, 1998, pp. 194–218.
36. ———, *Automata, power series, and coinduction*: *Taking input derivatives seriously* (*extended abstract*), Automata, Languages and Programming (Prague, 1999) (J. Wiedermann, P. van Emde Boas, and M. Nielsen, eds.), Lecture Notes in Comput. Sci., vol. 1644, Springer, Berlin, 1999, pp. 645–654.
37. ———, *Behavioural differential equations*: *A coinductive calculus of streams, automata, and power series*, Theoret. Comput. Sci., vol. 308 (1–3), Elsevier Science Publishers, 2003, pp. 1–53.
38. ———, *Universal coalgebra*: *A theory of systems*, Theoret. Comput. Sci. **249** (2000), no. 1, 3–80.
39. ———, *Elements of stream calculus* (*an extensive exercise in coinduction*), Technical Report SEN-R0120, CWI, Amsterdam, 2001, pp. 1–54; Proceedings MFPS '01, ENTCS, vol. 45, Elsevier Science B.V., 2001; Math. Structures in Comput. Sci. (to appear).
40. ———, *Coinductive counting with weighted automata*, J. Automat. Lang. Comb. **8** (2003), no. 2, 319–352.
41. D. Sangiorgi, *On the bisimulation proof method*, Math. Structures Comput. Scie. **8** (1998), no. 5, 447–479.
42. I. N. Sneddon, *The use of integral transforms*, McGraw-Hill, 1972.
43. R. P. Stanley, *Enumerative combinatorics.* I, Cambridge Stud. Adv. Math., vol. 49, Cambridge University Press, Cambridge, 1997.
44. ———, *Enumerative Combinatorics.* II, Cambridge Stud. Adv. Math., vol. 62, Cambridge University Press, Cambridge, 1999.
45. A. Tarski, *A lattice-theoretical fixpoint theorem and its applications*, Pacific J. Math. **5** (1955), 285–309.
46. J. van Benthem, *Modal correspondence theory*, Ph.D. thesis, University of Amsterdam, Amsterdam, 1976.
47. H. S. Wilf, *Generatingfunctionology*, Academic Press, 1994.