Centrum voor Wiskunde en Informatica

Behavioural differential equations: a coinductive calculus of streams, automata, and power series

J.J.M.M. Rutten

Software Engineering (SEN)

SEN-R0023 September 30, 2000

Report SEN-R0023
ISSN 1386-369X

CWI
P.O. Box 94079
1090 GB Amsterdam
The Netherlands

CWI is the National Research Institute for Mathematics and Computer Science. CWI is part of the Stichting Mathematisch Centrum (SMC), the Dutch foundation for promotion of mathematics and computer science and their applications.
SMC is sponsored by the Netherlands Organization for Scientific Research (NWO). CWI is a member of ERCIM, the European Research Consortium for Informatics and Mathematics.

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

Behavioural Differential Equations: A Coinductive Calculus of Streams, Automata, and Power Series

J.J.M.M. Rutten
CWI, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands
Email: `janr@cwi.nl`, URL: `www.cwi.nl/~janr`

ABSTRACT
Streams, (automata and) languages, and formal power series are viewed coalgebraically. In summary, this amounts to supplying these sets with a deterministic automaton structure, which has the universal property of being final. Finality then forms the basis for both definitions and proofs by coinduction, the coalgebraic counterpart of induction. Coinductive definitions take the shape of what we have called behavioural differential equations, after Brzozowski's notion of input derivative. A calculus is developed for coinductive reasoning about all of the afore mentioned structures, closely resembling (and at times generalising) calculus from classical analysis.

2000 Mathematics Subject Classification: 68Q10, 68Q55, 68Q85
1998 ACM Computing Classification System: F.1, F.3
Keywords & Phrases: Coalgebra, automaton, finality, coinduction, stream, formal language, formal power series, differential equation, input derivative, behaviour, semiring, max-plus algebra

Contents

1 Introduction **3**

2 Streams and stream automata **5**

3 Behavioural differential equations **8**

4 Proofs by coinduction **14**

5 Stream calculus **17**

6 More stream calculus: shuffle product and shuffle inverse **20**

7 Rational streams **24**

8 Nondeterministic stream automata **25**

9 Formal power series **29**

10 Languages **33**

11 An example in the max-plus semiring **38**

12 Related work and discussion **40**

13 Appendix: the proof of Theorem 3.2 **41**

14 Appendix: automata are coalgebras **43**

" ··· in this case, as in many others, the process gives the minimal machine directly to anyone skilled in input differentiation. The skill is worth acquiring ··· "
— J.H. Conway [Con71, chap. 5]

1 Introduction

The classical theories of streams (infinite sequences), automata and languages, and formal power series, are presented in terms of the notions of homomorphism and bisimulation, which are the cornerstones of the theory of (universal) coalgebra. This coalgebraic perspective leads to a unified theory, in which the observation that the sets of streams, languages, and formal power series each carry a *final* automaton structure, plays a central role. In all cases, the transitions of the final automaton are determined by the notion of *input derivative* (and initial value). This notion, which already occurs in work of Brzozowski [Brz64] and Conway [Con71], can be understood, in a very precise sense, as an abstract generalisation of the analytical notion of function derivative. Finality gives rise to both a coinduction definition and a coinduction proof principle, formulated in terms of derivatives. It is exactly this use of derivatives in coinduction that makes our theory into a *calculus*, again in much the same way as one has calculus in analysis. In fact, the connection is so close that the theory of mathematical analysis can serve as an important source of inspiration.

Much emphasis will be put on coinductive definitions, which are formulated, as indicated above, in terms of input derivatives. Since the latter can be understood as describing the (dynamic) behaviour of streams, languages, and formal power series, we have called these coinductive definitions *behavioural differential equations*. To give an example, let us introduce the set of streams of real numbers, which are functions $\sigma : \{0, 1, 2, \ldots\} \to \mathbb{R}$. The initial value of σ is defined as its first element $\sigma(0)$, and the (input or) stream derivative, denoted by σ', is defined by $\sigma'(n) = \sigma(n+1)$, for $n \geq 0$. (In other words, initial value and derivative equal head and tail of σ, respectively.) Viewed as a state of the final automaton of all streams, the behaviour of a stream σ consists of two aspects: it (shyly) allows for the observation of (only) its initial value $\sigma(0)$; and it can make a transition to the new state σ', consisting of the original stream from which the first element has been removed. The initial value of σ', which is $\sigma'(0) = \sigma(1)$, can at its turn be observed, but note that we had to move from σ to σ' first in order to do so. Now a behavioural differential equation defines a stream by specifying its initial value together with a description of its derivative, which tells us how to continue. For instance, the sum $\sigma + \tau$ and product $\sigma \times \tau$ of two streams σ and τ can be defined by specifying their initial values and stream derivatives in terms of the initial values and (sums and products of) the derivatives of σ and τ, as follows:

differential equation	initial value	name
$(\sigma + \tau)' = \sigma' + \tau'$	$(\sigma + \tau)(0) = \sigma(0) + \tau(0)$	sum
$(\sigma \times \tau)' = (\sigma' \times \tau) + (\sigma(0) \times \tau')$	$(\sigma \times \tau)(0) = \sigma(0) \times \tau(0)$	product

The precise interpretation of such equations will become clear later. (For one thing, the overloading of the symbol $\times$ in the equation for the product needs to be explained.) For now, it is sufficient to know that they indeed uniquely define an operation of sum and of product on streams, a fact which is based on the finality of the automaton of all streams. The above definitions can be shown to be equivalent to the traditional definitions of sum and of convolution product, which are typically given in an 'elementwise' fashion: for all $n \geq 0$,

$$(\sigma + \tau)(n) = \sigma(n) + \tau(n), \quad (\sigma \times \tau)(n) = \sum_{k=0}^{n} \sigma(n-k) \times \tau(k)$$

As it turns out, coinductive definitions by means of behavioural differential equations have a number of advantages over this latter type of definition:

- Coinductive definitions seem to be 'at the right level of abstraction': One behavioural differential equation often defines seemingly different operators at the same time. An example is the definition of convolution product, which on languages will correspond to language

concatenation. We shall see more extreme examples of this phenomenon later (for instance, the definition of the shuffle product in Section 6).

- The use of indices (such as k and n in the above definition of product) makes reasoning about the operators unnecessarily complicated. In contrast, coinductive proofs present themselves as more transparent alternatives. Section 4 contains many examples.

- Coinductive definitions using behavioural differential equations seem to be more generally applicable. For instance, the inverse σ^{-1} of a stream σ, satisfying $\sigma \times \sigma^{-1} = 1$, will be defined, in Section 3, using a differential equation. It is by no means clear how an elementwise definition for inverse should look like. Another example is the coinductive definition of an operator $\sigma^{-\underline{1}}$, which acts as an inverse for the shuffle product. This operator seems to be new, not to be definable elementwise (or inductive, for that matter), and can be used to denote 'everywhere diverging' streams, as in:

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

  where X is the constant stream defined by $X = (0, 1, 0, 0, 0, \ldots)$.

- Behavioural differential equations have an operational reading, from which algorithms can be easily derived. For instance, we shall see that the differential equation for σ^{-1} can be read as an algorithm that produces the elements of this infinite stream one by one.

- The explicit occurrence of the sum operator $+$ in behavioural differential equations (such as the one for product above) will give rise, in Section 8, to rather efficient representations of power series in the form of (generalised versions of) nondeterministic automata. As an example, a very simple automaton will be constructed representing the so-called *tangent* numbers, the Taylor series of the function $tan(x)$. From this automaton, a closed (non-recurrent) formula for this series can be derived (where no such formula is known in the literature). Another example concerns finite nondeterministic representations of so-called rational streams.

The most general level at which we shall be working is that of formal power series (in many non-commutative variables), which are functions $\sigma : A^* \to k$ from the set of words over an alphabet A (of variables, also called input symbols) to some semiring k (such as the reals or the Booleans). But before dealing with formal power series in Section 9, the paper develops in all detail a coinductive calculus for the afore mentioned streams of real numbers, which can be obtained as one particular instance of formal power series by setting $A = \{X\}$ (a singleton set containing only one formal variable) and $k = \mathbb{R}$. Streams are for many readers probably somewhat more familiar than power series, and form an inexhaustible source of entertaining examples (including Taylor series of analytic functions on the reals). And because the calculus is ultimately based on the universal property of the finality of the automaton of all streams, its generalisation to the case of formal power series is straightforward, requiring no serious rethinking and hardly any reformulation. The theory is further illustrated with two more special instances: (classical deterministic automata and) formal languages, in Section 10, including a new coinductive algorithm for deciding the equality of regular expressions; and power series over the so-called max-plus semiring of the real numbers (with max and plus rather than plus and times), in Section 11.

Although the development of our theory has been entirely dictated by the coalgebraic perspective, no explicit reference to coalgebraic notions and results will be made. The paper is intended to be self-contained, without assuming any prior knowledge on coalgebra. In this way, it constitutes a study in and, we hope, an introduction to what could be called concrete coalgebra, as opposed to so-called universal coalgebra, which deals with properties that are common to all coalgebras at the same time. (This mirrors the situation in algebra, where one has the concrete theories of, for instance, groups and rings, as well as the general theory of universal algebra.) For the interested reader, the connection with coalgebra is explicitly described in Section 14.

Summarising the contributions of the present paper, it takes the coalgebraic perspective to give a unified treatment of streams, languages, and formal power series. The general scheme is to *specify* (the behaviour of) streams, languages, and formal power series alike, by means of behavioural differential equations; to prove the equality of streams, languages, and formal power series alike, by means of coinduction, by constructing suitable bisimulation relations; and to deduce efficient *implementations* from the defining equations, in terms of nondeterministic automata. Unfolding the theory, the paper provides many illustrations of the use of coinduction along the way. (Most examples in the literature sofar have been of a rather elementary nature; some of the examples presented here can be considered, we hope, as a little bit less trivial.) In addition to the unification and simplification of existing definitions and proofs, the paper also introduces a number of new operators by coinduction, such as the operation of shuffle inverse mentioned above. Examples of their use are given, suggesting that these new operators actually have some interest. Since for most of them no obvious alternative definitions without the use of coinduction can be found, they provide some evidence that the use of coinduction sometimes is essential.

Section 12 contains some concluding remarks and discusses related work. In summary, the present paper reworks and extends [Rut99a], on power series, which was a generalisation of [Rut98a], where automata and languages were treated coinductively. General references on the coalgebraic approach are [Rut96] and [JR97]. Our notion of input derivative for power series generalises Brzozowski's original definition for regular expressions [Brz64, Con71]. In addition to our own earlier work, the presentation of the calculus for streams has been influenced by [PE98], which gives a coinductive treatment of analytic functions in terms of their Taylor series, and by [McI99], which treats power series (in one variable) as lazy lists in the programming language Haskell.

2 Streams and stream automata

Some elementary coalgebra is developed for the set of streams of real numbers: the notion of stream automaton is introduced, along with the corresponding notions of homomorphism and bisimulation, and the set of streams is characterised as a final stream automaton, leading to both a coinductive proof and a coinductive definition principle.

A *stream automaton* is a pair $Q = (Q, \langle o, t\rangle)$ consisting of a set Q of *states*, and a pair of functions: an *output* or *observation* function $o : Q \to \mathbb{R}$, and a *transition* or *next state* function $t : Q \to Q$. We write $q{\Downarrow}r$ to denote that state $q \in Q$ has output value $r \in \mathbb{R}$: $o(q) = r$, and $q \to q'$ denotes that the next state after q is q': $t(q) = q'$. Often it is convenient to include in such a transition step the information about outputs as well: $q{\Downarrow}r \to q'{\Downarrow}r'$ denotes $t(q) = q'$, $o(q) = r$, and $o(q') = r'$. The name 'stream automaton' is motivated by the fact that the behaviour of a state q in an automaton Q can be described by the infinite sequence or *stream* of consecutively observed output values, obtained by repeatedly applying the transition function:

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

The set of all streams is formally defined by

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

Streams σ will be often informally denoted as $\sigma = (s_0, s_1, s_2, \ldots)$, where $s_n = \sigma(n)$ is called the n-th *element* of σ. Streams are what we are actually interested in, and stream automata are relevant to us only as an aid to represent (and define) streams.

Example 2.1 Consider an automaton $Q = (Q, \langle o, t\rangle)$ with $Q = \{q_0, q_1, q_2, q_3\}$, output values $o(q_0) = o(q_2) = 0$, $o(q_1) = 1$, and $o(q_3) = -1$, and transitions $t(q_0) = q_1$, $t(q_1) = q_2$, $t(q_2) = q_3$, and $t(q_3) = q_0$. Output and transition functions could have also been defined implicitly by simply

drawing the following picture, which summarises all the relevant information:

$$\begin{tikzcd}
(q_0 \Downarrow 0) \arrow[d] & (q_3 \Downarrow -1) \arrow[l] \\
(q_1 \Downarrow 1) \arrow[r] & (q_2 \Downarrow 0) \arrow[u]
\end{tikzcd}$$

The behaviour of q_0 is the stream $(0, 1, 0, -1, 0, 1, 0, -1, \ldots)$ and, in fact, the (finite) automaton Q can be taken as a definition of this (infinite) stream. Here is another automaton, this time infinite, representing the same stream. Let $T = \{t_0, t_1, t_2, \ldots\}$ with transitions $t(t_i) = t_{i+1}$, for all $i \geq 0$, and with outputs $o(t_{0+4k}) = o(t_{2+4k}) = 0$, $o(t_{1+4k}) = 1$, and $o(t_{3+4k}) = -1$, for all $k \geq 0$:

$$(t_0 \Downarrow 0) \longrightarrow (t_1 \Downarrow 1) \longrightarrow (t_2 \Downarrow 0) \longrightarrow (t_3 \Downarrow -1) \longrightarrow \cdots$$

Clearly, any of the states t_{0+4k}, for $k \geq 0$, represents the stream $(0, 1, 0, -1, 0, 1, 0, -1, \ldots)$. □

Example 2.2 (Pavlović and Escardó [PE98]) More generally, here is a particularly rich source of examples of representations of streams. Let

$$\mathcal{A} = \{f : \mathbb{R} \to \mathbb{R} \mid f \text{ is analytic in } 0 \ \}$$

Recall that such functions are arbitrarily often differentiable in (a neighbourhood around) 0 and that all the derivatives are analytic again. Therefore it is possible to turn $\mathcal{A}$ into a stream automaton by defining the following output and transition functions: for an analytic function f put $o(f) = f(0)$ and $t(f) = f'$, the derivative of f. The behaviour of an analytic function f in the automaton $(\mathcal{A}, \langle o, t \rangle)$ then consists of the stream $(f(0), f'(0), f''(0), \ldots)$, which we recognise as the *Taylor series* of f. For instance, the transitions for the function $sin(x)$ (together with the corresponding output values) look like

$$\begin{tikzcd}
(sin(x) \Downarrow 0) \arrow[d] & (-cos(x) \Downarrow -1) \arrow[l] \\
(cos(x) \Downarrow 1) \arrow[r] & (-sin(x) \Downarrow 0) \arrow[u]
\end{tikzcd}$$

□

The set $\mathbb{R}^\omega$ of all streams can itself be turned into a stream automaton as follows. Let the *initial value* of a stream $\sigma \in \mathbb{R}^\omega$ be defined by its first element: $\sigma(0)$, and let the *stream derivative* σ' of σ be given by $\sigma'(n) = \sigma(n+1)$, for all $n \geq 0$ or, informally, $(s_0, s_1, s_2, \ldots)' = (s_1, s_2, s_3, \ldots)$. (The main reason for preferring this notation and terminology above the more standard use of 'head' and 'tail' is the fact that we shall develop a calculus-like theory of streams.) Next define $o : \mathbb{R}^\omega \to \mathbb{R}$ by $o(\sigma) = \sigma(0)$ and $t : \mathbb{R}^\omega \to \mathbb{R}^\omega$ by $t(\sigma) = \sigma'$, and we obtain an automaton $(\mathbb{R}^\omega, \langle o, t \rangle)$.

We shall also use the following notation: $\sigma^{(n)} = t^n(\sigma)$, for all $n \geq 0$, which is obtained by taking n times the derivative of σ; as usual, $\sigma^{(0)} = \sigma$. (Also σ'' will be used as a shorthand for $(\sigma')'$.) With these definitions, the transitions of the stream σ viewed as a state of the automaton $\langle \mathbb{R}^\omega, (o, t) \rangle$ are

$$\sigma = \sigma^{(0)} \longrightarrow \sigma^{(1)} \longrightarrow \sigma^{(2)} \longrightarrow \cdots$$

and it is through these transitions and their corresponding output values, that we get to know the successive elements of the stream σ, one by one:

$$\sigma = (\sigma^{(0)}(0), \sigma^{(1)}(0), \sigma^{(2)}(0), \ldots)$$

Thus the n-th element of σ is given by $\sigma(n) = \sigma^{(n)}(0)$, which is easily proved by induction.

The automaton of streams has a number of universal properties, which can be nicely expressed in terms of the notions of homomorphism and bisimulation, which are introduced next. A *bisimulation* between stream automata $(Q, \langle o_Q, t_Q \rangle)$ and $(Q', \langle o_{Q'}, t_{Q'} \rangle)$ is a relation $R \subseteq Q \times Q'$ such that for all q in Q and q' in Q':

$$\text{if } \; q \, R \, q' \; \text{ then } \; \begin{cases} o_Q(q) = o_{Q'}(q') & \text{and} \\ t_Q(q) \; R \; t_{Q'}(q') \end{cases}$$

(Here qRq' denotes $\langle q, q' \rangle \in R$; both notations will be used.) A bisimulation between Q and itself is called a bisimulation *on* Q. Unions and (relational) compositions of bisimulations are bisimulations again. We write $q \sim q'$ whenever there exists a bisimulation R with $q \, R \, q'$. This relation $\sim$ is the union of all bisimulations and, therewith, the greatest bisimulation. The greatest bisimulation on one and the same automaton, again denoted by $\sim$, is called the *bisimilarity* relation. It is an equivalence relation.

Example 2.3 The states q_0 and t_0 in Example 2.1 are bisimilar: $q_0 \sim t_0$, since

$$\{\langle q_0, t_{0+4k} \rangle \mid k \geq 0\} \cup \{\langle q_1, t_{1+4k} \rangle \mid k \geq 0\} \cup \{\langle q_2, t_{2+4k} \rangle \mid k \geq 0\} \cup \{\langle q_3, t_{3+4k} \rangle \mid k \geq 0\}$$

can be readily seen to be a bisimulation relation between the automata Q and T. Of course, the same relation shows that $s_0 \sim t_{0+4k}$, for any $k \geq 0$. Also q_0 is bisimilar with the function $sin(x)$ from Example 2.2: $s_0 \sim sin(x)$, since

$$\{\langle q_0, sin(x) \rangle, \; \langle q_1, cos(x) \rangle, \; \langle q_2, -sin(x) \rangle, \; \langle q_3, -cos(x) \rangle\}$$

is a bisimulation between Q and $\mathcal{A}$. The same relation shows that $q_1 \sim cos(x)$. □

Bisimulation relations on the automaton $(\mathbb{R}^\omega, \langle o, t \rangle)$ of streams are particularly simple: they only contain pairs of identical elements.

Theorem 2.4 *For all streams σ and τ in $\mathbb{R}^\omega$, if $\sigma \sim \tau$ then $\sigma = \tau$.*

(Note that the converse trivially holds, since $\{\langle \sigma, \sigma \rangle \mid \sigma \in \mathbb{R}^\omega\}$ is a bisimulation relation on $\mathbb{R}^\omega$.) The fact that bisimulations on $\mathbb{R}^\omega$ relate only identical streams gives rise to the following, surprisingly powerful proof principle, called *coinduction*: in order to prove the equality of two streams σ and τ, it is sufficient show that they are bisimilar. And since bisimilarity is the union of all bisimulation relations, $\sigma \sim \tau$ can be proved by establishing the existence of a bisimulation relation on $R \subseteq \mathbb{R}^\omega \times \mathbb{R}^\omega$ with $\langle \sigma, \tau \rangle \in R$. We shall see many examples of proofs by coinduction.

Proof: Consider two streams σ and τ and let $R \subseteq \mathbb{R}^\omega \times \mathbb{R}^\omega$ be a bisimulation on the automaton $(\mathbb{R}^\omega, \langle o, t \rangle)$ 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 $n \geq 0$. This proves $\sigma = \tau$. □

A bisimulation relation that is actually a function is called *homomorphism*. Equivalently, a homomorphism between stream automata $(Q, \langle o_Q, t_Q \rangle)$ and $(Q', \langle o_{Q'}, t_{Q'} \rangle)$ is any function $f : Q \to Q'$ such that, for all s in Q, $o_Q(s) = o_{Q'}(f(s))$ and $f(t_Q(s)) = t_{Q'}(f(s))$. The set of all streams has the following universal property.

Theorem 2.5 *The automaton $(\mathbb{R}^\omega, \langle o, t \rangle)$ 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 $l : Q \to \mathbb{R}^\omega$.*

The existence part of this theorem can be used as a *(coinductive) definition principle*, as will be illustrated in many ways later on.

Proof: Let $(Q, \langle o_Q, t_Q \rangle)$ be an automaton and let the function $l : Q \to \mathbb{R}^\omega$ assign to a state q in Q the stream $(o_Q(q), \, o_Q(t_Q(q)), \, o_Q(t_Q(t_Q(q))), \ldots)$. It is straightforward to show that l is a homomorphism from $(Q, \langle o_Q, t_Q \rangle)$ to $(\mathbb{R}^\omega, \langle o, t \rangle)$. For uniqueness, suppose f and g are homomorphisms from Q to $\mathbb{R}^\omega$. 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 $\mathbb{R}^\omega$, which is proved next. Consider $\langle f(q), g(q)\rangle \in R$. Because f and g are homomorphisms, $o(f(q)) = o_Q(q) = o(g(q))$. Furthermore, $t(f(q)) = f(t_Q(q))$ and $t(g(q)) = g(t_Q(q))$. Because $\langle f(t_Q(q)), g(t_Q(q))\rangle \in R$, this shows that R is a bisimulation. Thus $f(q) \sim g(q)$, for any q in Q. Now $f = g$ follows by the coinduction proof principle Theorem 2.4. □

The stream $l(q)$ is (what we have called above) the *behaviour* of the state q of the automaton Q. Taking $Q = \mathbb{R}^\omega$ in Theorem 2.5, it follows that l equals the identity function on $\mathbb{R}^\omega$, since the latter trivially is a homomorphism. Thus the behaviour $l(\sigma)$ of a stream σ viewed as a state in $\mathbb{R}^\omega$ is equal to σ: $l(\sigma) = \sigma$. This yields the intriguing slogan that the states of the final automaton $\mathbb{R}^\omega$ 'do as they are'. More generally, the homomorphism l above is characterised by the following property.

Proposition 2.6 *Let Q be an automaton and let $l : Q \to \mathbb{R}^\omega$ be the unique homomorphism from Q to $\mathbb{R}^\omega$. For all q and q' in Q, $q \sim q'$ iff $l(q) = l(q')$.*

Since the homomorphism $l : Q \to \mathbb{R}^\omega$ assigns to each state in Q its behaviour, the proposition expresses that two states are related by a bisimulation relation iff their behaviour is the same.

Proof: Because the bisimilarity relation is itself a bisimulation relation and because l is a homomorphism, the relation $\{\langle l(q), l(q')\rangle \mid q \sim q'\}$ is easily seen to be a bisimulation relation on $\mathbb{R}^\omega$. The implication from left to right therefore follows by coinduction Theorem 2.4. The converse is a consequence of the fact that $\{\langle q, q'\rangle \in Q \mid l(q) = l(q')\}$ can be readily shown to be a bisimulation on Q, again using the fact that l is a homomorphism. □

Example 2.7 The unique homomorphism from the automaton $\mathcal{A}$ of analytic functions (Example 2.2) to the automaton of streams assigns to each analytic function $f : \mathbb{R} \to \mathbb{R}$ its Taylor series (and is therefore denoted by $\mathcal{T}$):

$$\mathcal{T} : \mathcal{A} \to \mathbb{R}^\omega, \quad \mathcal{T}(f) = (f(0), f'(0), f''(0), \ldots)$$

Because analytic functions f and g are entirely determined by their Taylor series, in the sense that $\mathcal{T}(f) = \mathcal{T}(g)$ implies $f = g$, an immediate consequence of Proposition 2.6 is the following coinduction proof principle for analytic functions: if $f \sim g$ then ($\mathcal{T}(f) = \mathcal{T}(g)$ by Proposition 2.6, and thus) $f = g$. As an example, this principle is used to prove the following familiar law. For any real number $a \in \mathbb{R}$,

$$sin(x + a) = cos(a)sin(x) + sin(a)cos(x)$$

Recalling that $sin(x + a)' = cos(x + a)$ and $cos(x + a)' = -sin(x + a)$, this equality follows by coinduction from the fact that the following 4-element set

$$\{\langle sin(x + a), cos(a)sin(x) + sin(a)cos(x)\rangle,\ \langle cos(x + a), cos(a)cos(x) - sin(a)sin(x)\rangle,$$

$$\langle -sin(x + a), -cos(a)sin(x) - sin(a)cos(x)\rangle,\ \langle -cos(x + a), -cos(a)cos(x) - sin(a)sin(x)\rangle\}$$

is easily seen to be a bisimulation relation on $\mathcal{A}$. □

3 Behavioural differential equations

The finality of the automaton $\mathbb{R}^\omega$ of all streams is used as a basis for the definition of a number of familiar and less familiar operators on streams, including sum, product, star, and division. Such definitions are called *coinductive* since the role of the finality of $\mathbb{R}^\omega$ is in a precise sense dual to the role of initiality of, for instance, the natural numbers, which underlies the principle of induction (see [Rut96, JR97] for more detailed explanations of this fact). Coinductive definitions will here be presented in terms of so-called behavioural differential equations.

Before defining the operators on streams we are interested in, the entire approach is illustrated by a coinductive definition of the stream

$$\sigma = (1, 1, 1, \ldots)$$

Although this expression makes perfectly clear which stream it is that we want to define, we do not want to take this expression itself as a formal definition, because of the presence of the dots, telling us 'how to continue'. Defining the stream above as the function $\sigma : \{0, 1, 2, \ldots\} \to \mathbb{R}$ with $\sigma(k) = 1$, for all $k \geq 0$, is formal enough but still, there are several reasons for not being satisfied with this type of definition, either. For one thing, it suggests that we are able to oversee in one go, as it were, all infinitely many elements of this stream. This is easy enough in this particular case, but we shall see many examples where this kind of global view is either very difficult or even impossible. Another objection to the definition is that the formula '$\sigma(k) = 1$ (for all $k \geq 0$)' does not do full justice to the stream's extreme regularity, consisting of the fact that removing the first element of σ yields a stream that is equal to σ again: $\sigma' = \sigma$. In fact, it is precisely this property which, together with the observation that the first element of σ equals 1, fully characterises this stream. Therefore, our proposal for a formal definition of σ is the following *behavioural differential equation*:

differential equation	initial value
$\sigma' = \sigma$	$\sigma(0) = 1$

Behavioural differential equations define streams by specifying their *behaviour*, that is, transitions and output values, in terms of derivatives and initial values. Now that we have motivated the above behavioural definition, it still has to be formally justified: we have to show that there exists a unique stream σ in $\mathbb{R}^\omega$ which satisfies the equation above. And this is precisely where the finality of the automaton $\mathbb{R}^\omega$ comes in. For this particular example, things are extremely simple of course. It suffices to consider an automaton $(S, \langle o_S, t_S \rangle)$ with only one state: $S = \{s\}$, and with transition $t_S(s) = s$ and output value $o_S(s) = 1$. By the finality of the automaton $(\mathbb{R}^\omega, \langle o, t \rangle)$ (Theorem 2.5), there exists a unique homomorphism $l : S \to \mathbb{R}^\omega$. We can now define $\sigma = l(s)$. Because l is a homomorphism, $\sigma' = t(\sigma) = t(l(s)) = l(t_S(s)) = l(s) = \sigma$, and $\sigma(0) = o(\sigma) = o(l(s)) = o_S(s) = 1$, indeed. Thus we have found a solution of our behavioural differential equation. If ρ is a stream satisfying $\rho' = \rho$ and $\rho(0) = 1$, then $\sigma = \rho$ follows, by the coinduction proof principle Theorem 2.4, from the fact that $\{\langle \sigma, \rho \rangle\}$ is a bisimulation relation of $\mathbb{R}^\omega$. Which shows that σ is the only solution of the differential equation.

The reader will have noticed that the behavioural differential equation above looks very familiar. When interpreted as an ordinary differential equation (over real-valued functions), it defines the function $exp(x) : \mathbb{R} \to \mathbb{R}$ from analysis. The fact that the Taylor series of $exp(x)$ equals our stream σ can hardly be a coincidence and, in fact, it is not. Recalling from Example 2.7 the (unique) homomorphism $\mathcal{T} : \mathcal{A} \to \mathbb{R}^\omega$ that assigns to an analytic function its Taylor series, we have $\sigma = \mathcal{T}(exp(x))$. This follows from the fact that the latter is also a solution to the behavioural differential equation for σ, which can be easily proved using $exp(0) = 1$ and $exp(x)' = exp(x)$, and the fact that $\mathcal{T}$ is a homomorphism: $\mathcal{T}(exp(x))(0) = exp(0) = 1$ and $\mathcal{T}(exp(x))' = \mathcal{T}(exp(x)') = \mathcal{T}(exp(x))$. Since we saw above that the equation has a unique solution, it follows that $\sigma = \mathcal{T}(exp(x))$.

Example 3.1 Here is another simple example, which introduces a useful convention on the inclusion of the reals into the set of streams. We want to view any real number r in $\mathbb{R}$ as a stream $[r] = (r, 0, 0, 0, \ldots)$. Using the formalism of behavioural differential equations that we have just learned, such streams $[r]$ can formally be defined by the following system of equations (one for each real number r):

differential equation	initial value
$[r]' = [0]$	$[r](0) = r$

We shall also need the following constant stream $X = (0, 1, 0, 0, 0, \ldots)$, which will play the role of a formal variable for stream calculus (see, for instance, Theorem 5.2). It is defined by the following equation:

differential equation	initial value
$X' = [1]$	$X(0) = 0$

Since the latter equation for X refers to the stream $[1]$ which is defined by the first system of equations, it will be necessary to justify all equations at the same time, again by finality of $\mathbb{R}^\omega$. To this end, let $(S, \langle o, t \rangle)$ be the automaton with as states the set $S = \{s_r \mid r \in \mathbb{R}\} \cup \{s_X\}$, containing one state for X and one state for each real number r. The output values and transitions are defined by $o(s_r) = r$ and $o(s_X) = 0$, and $t(s_r) = s_0$ and $t(s_X) = s_1$. (Note how these definitions precisely follow the equations above.) Now define $[r] = l(s_r)$ and $X = l(s_X)$, where $l : \mathbb{R} \to \mathbb{R}^\omega$ is the unique homomorphism into $\mathbb{R}^\omega$ given by finality. It is easily checked that the streams $[r]$ and X are the unique solutions of the equations. $\square$

Convention: we shall usually simply write r for $[r]$.

Next we turn to the definition of operators on streams, which will be specified by a system of mutually dependent differential equations.

Theorem 3.2 *There are unique operators on streams satisfying the following behavioural differential equations: For all $\sigma, \tau \in \mathbb{R}^\omega$,*

differential equation	*initial value*	*name*
$(\sigma + \tau)' = \sigma' + \tau'$	$(\sigma + \tau)(0) = \sigma(0) + \tau(0)$	*sum*
$(\sigma \times \tau)' = (\sigma' \times \tau) + (\sigma(0) \times \tau')$	$(\sigma \times \tau)(0) = \sigma(0) \times \tau(0)$	*product*
$(\sigma^*)' = \sigma' \times \sigma^*$	$(\sigma^*)(0) = 1$	*star*
$(\sigma^{-1})' = -(\sigma(0)^{-1} \times \sigma') \times \sigma^{-1}$	$(\sigma^{-1})(0) = \sigma(0)^{-1}$	*inverse*

In the formulation of the theorem, the following is to be noted:

- The same symbols are used for the sum of streams and the sum of real numbers, and similarly for the product. As usual, $\sigma \times \tau$ will often be denoted by $\sigma\tau$, and similarly for real numbers. And we shall use the following standard conventions: $\sigma^0 = 1$ and $\sigma^{n+1} = \sigma \times \sigma^n$, for all $n \geq 0$, not to be confused with our notation $\sigma^{(n)}$, which stands for the n-th derivative of σ (introduced in Section 2).
- In the definition of the product, $\sigma(0) \times \tau'$ is a shorthand for $[\sigma(0)] \times \tau'$, following the convention of Example 3.1.
- We write $-\sigma$ as a shorthand for $[-1] \times \sigma$.
- In the equation for σ^{-1}, the stream σ is supposed to have an invertible initial value: $\sigma(0) \neq 0$. Moreover, the expression $-(\sigma(0)^{-1} \times \sigma')$ is to be read as a shorthand for $[-1] \times ([\sigma(0)^{-1}] \times \sigma')$.
- Once we have the operation of inverse, an operation of *division* can be defined as usual by putting $\tau : \sigma = \tau \times \sigma^{-1}$.
- The product of streams defined by the equation above does not correspond to the pointwise product of functions, for which one has: $(f \cdot g)' = f' \cdot g + f \cdot g'$, but is the so-called *convolution* product. Later we shall see a different type of product (shuffle product, in Section 5) for which stream derivation behaves as it does for function product.

Before presenting the proof of this theorem, which will again rely on the finality of $\mathbb{R}^\omega$, we first want to motivate the particular shape of the above differential equations. This can be done best using the following easy but important result from Section 5: for every stream σ in $\mathbb{R}^\omega$,

$$\sigma = \sigma(0) + (X \times \sigma')$$

(Here X is the constant stream $(0,1,0,0,0,\ldots)$ introduced in Example 3.1, and $\sigma(0)$ is to be read as the stream $[\sigma(0)]$.) This formula makes it possible to calculate with streams as so-called *formal power series* (cf. Theorem 5.2). In the calculations below, we are assuming that our operators have the usual properties of commutativity, associativity, distributivity, and the like, all of which are formally proved in Theorem 4.1. Consider now two streams σ and τ and write $\sigma = \sigma(0) + X\sigma'$ and $\tau = \tau(0) + X\tau'$, omitting as usual the $\times$ symbol. Computing the *sum* of σ and τ,

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

and comparing this with $\sigma + \tau = (\sigma + \tau)(0) + X(\sigma + \tau)'$, we find the differential equation for sum:

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

Computing the *product* of σ and τ yields

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

Comparing this with $\sigma \times \tau = (\sigma \times \tau)(0) \times X(\sigma \times \tau)'$ leads to the equation for product:

differential equation	initial value
$(\sigma \times \tau)' = (\sigma' \times \tau) + (\sigma(0) \times \tau')$	$(\sigma \times \tau)(0) = \sigma(0) \times \tau(0)$

The equation for σ^* can be deduced from the the following requirement, which we would like to hold at least for all σ with $\sigma(0) = 0$:

$$\sigma^* = 1 + \sigma + \sigma^2 + \sigma^3 + \cdots$$

(For the well-definedness of such an infinite sum, see the end of the present section.) Since in the presence of inverse, star is a definable operator (cf. Theorem 4.1(13)), we shall concentrate for the time being on the inverse operator. (Later we shall see different structures without inverse, such as the set of languages over a given alphabet, where the star operator does play a prominent role.) The equation for inverse can be deduced from the property $1 = \sigma^{-1} \times \sigma$ (Theorem 4.1) as follows:

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

This implies $1 = \sigma^{-1}(0) \times \sigma(0)$ and $0 = (\sigma^{-1}(0) \times \sigma') + ((\sigma^{-1})' \times \sigma)$, from which the differential equation for inverse is derived:

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

Proof of Theorem 3.2: The proof consists of the construction of what could be called a *syntactic* stream automaton, whose states are given by *expressions* including all the possible shapes that occur on the right side of the behavioural differential equations. 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:

$$E ::= \underline{\sigma} \mid E + F \mid E \times F \mid E^* \mid E^{-1}$$

The set $\mathcal{E}$ contains for every stream σ in $\mathbb{R}^\omega$ a constant *symbol* $\underline{\sigma}$. Each of the operators that we are in the process of defining, is represented in $\mathcal{E}$ by a syntactic counterpart, denoted by the same symbols $+$, $\times$, and so on. (It were more precise to distinguish also here between semantics and syntax, for instance by writing $\underline{\times}$ for the syntactic version of product. But life is already exacting enough as it is, without having to write all these _'s. And we promise not to confuse the reader.) The set $\mathcal{E}$ is next supplied with an automaton structure $(\mathcal{E}, \langle o_\mathcal{E}, t_\mathcal{E}\rangle)$ by defining functions $o_\mathcal{E} : \mathcal{E} \to \mathbb{R}$ and $t_\mathcal{E} : \mathcal{E} \to \mathcal{E}$. It will be convenient to write $E(0)$ for $o_\mathcal{E}(E)$ and E' for $t_\mathcal{E}(E)$, for any expression E in $\mathcal{E}$. For the definition of $o_\mathcal{E}$ and $t_\mathcal{E}$ on the constant symbols $\underline{\sigma}$, the automaton structure of $\mathbb{R}^\omega$ is used:

definition of $t_\mathcal{E}$	definition of $o_\mathcal{E}$
$(\underline{\sigma})' = \underline{\sigma'}$	$\underline{\sigma}(0) = \sigma(0)$

Here σ' is the stream derivative of σ. 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:

definition of $t_\mathcal{E}$	definition of $o_\mathcal{E}$
$(E + F)' = E' + F'$	$(E + F)(0) = E(0) + F(0)$
$(E \times F)' = (E' \times F) + (E(0) \times F')$	$(E \times F)(0) = E(0) \times F(0)$
$(E^*)' = E' \times E^*$	$(E^*)(0) = 1$
$(E^{-1})' = -(E(0)^{-1} \times E') \times E^{-1}$	$(E^{-1})(0) = E(0)^{-1}$

An important step in the present proof is the observation that the above two systems of equations together establish a well-formed definition of the functions $o_\mathcal{E}$ and $t_\mathcal{E}$, by induction on the structure of the expressions. (Also note that in the definition of the expression $(E \times F)'$, we should strictly speaking have written $\underline{[E(0)]}$ instead of $E(0)$: it is a real number interpreted as a stream, which is included in $\mathcal{E}$ as a constant symbol; a similar remark applies to the definition of $(E^{-1})'$. Further note that, since we prefer our functions $o_\mathcal{E}$ and $t_\mathcal{E}$ to be total, we put $(E^{-1})' = 0$ and $E^{-1}(0) = 0$ in the case that $E(0) = 0$.)

Since $\mathcal{E}$ now has been turned into an automaton $(\mathcal{E}, \langle o_\mathcal{E}, t_\mathcal{E}\rangle)$, and because $\mathbb{R}^\omega$ is a final automaton, there exists, by Theorem 2.5, a unique homomorphism $l : \mathcal{E} \to \mathbb{R}^\omega$, which assigns to each expression E the stream $l(E)$ it represents. It can be used to define the operators on $\mathbb{R}^\omega$ that we are looking for, as follows:

$$\sigma + \tau = l(\underline{\sigma} + \underline{\tau}), \quad \sigma \times \tau = l(\underline{\sigma} \times \underline{\tau}), \quad \sigma^* = l(\underline{\sigma}^*), \quad \sigma^{-1} = l(\underline{\sigma}^{-1}) \tag{1}$$

Next one can show that the operators that we have just defined satisfy the behavioural differential equations, and that they are the only operators with this property. Both proofs use the coinduction proof principle of Theorem 2.4. Since they are neither difficult nor particularly instructive, their details are given in an appendix (Section 13). □

The reason why the above proof works is that it is possible to use the differential equations of the theorem as a basis for the definition of an automaton structure on the set $\mathcal{E}$ of expressions, by induction on their structure. This suggests that the proof can be adapted straightforwardly for other, similar such systems of equations. In fact, we shall later see other definitions of operators using differential equations, for the well-definedness of which we shall simply refer to (a variation on) the proof above. An obvious question to raise here is precisely which behavioural differential equations have unique solutions. This question will not be formally addressed here, but intuitively, these will include at least all systems of equations of which the right hand sides are given by expressions that are built from: the operators that the system is supposed to define; the arguments

to which they are applied; and the initial values and the (first) derivatives of these arguments. At the same time, the initial values of the operators should be defined in terms of the initial values of their arguments. A typical example is the defining equation for $(\sigma \times \tau)'$, whose right hand side uses $+$ and $\times$, (σ and) τ, as well as the derivatives σ' and τ' and initial value $\sigma(0)$; and whose initial value is defined as the product of the initial values of σ and τ. The reader is referred to [Bar00] (and the definitions mentioned there) for a general treatment of coinductive definition formats, which includes the above case as a particular example.

Coinductive definitions of streams and stream operators in terms of behavioural differential equations have an obvious algorithmic reading, by viewing them as executable recipes for the construction, step by step, of the streams they define. In order to illustrate this with some easy examples, we first introduce the following definition. A stream π is *polynomial* if there exist $n \geq 0$ and real numbers $p_0, p_1, p_2, \ldots, p_n \in \mathbb{R}$ such that

$$\pi = p_0 + p_1 X + p_2 X^2 + \cdots + p_n X^n$$

(Once again, note that $p_i X^i$ is a shorthand for $[p_i] \times X^i$.) As usual, if $p_n \neq 0$ then n is called the *degree* of π. For actual computations with polynomials, there is the following easy lemma. (Here and below, we are using some basic properties of our operators, such as $0 + \sigma = \sigma$, that will be proved later in Theorem 4.1.)

Lemma 3.3 *For all $r \in \mathbb{R}$, $\sigma \in \mathbb{R}^\omega$, $i \geq 1$, $n \geq 0$, and $p_0, p_1, p_2, \ldots, p_n \in \mathbb{R}$:*

1. $r' = 0$
2. $(X\sigma)' = \sigma$
3. $(X^i)' = X^{i-1}$
4. $(r\sigma)' = r\sigma'$
5. $(p_0 + p_1 X + p_2 X^2 + \cdots + p_n X^n)' = p_1 + p_2 X + p_3 X^2 + \cdots + p_n X^{n-1}$

Proof: The first fact is by definition: $r' = [r]' = [0] = 0$. For the second, one has $(X\sigma)' = (X \times \sigma)' = (1 \times \sigma) + (0 \times \sigma') = \sigma$. Since $X^i = X \times X^{i-1}$, (2) implies (3). Also $(r\sigma)' = ([r] \times \sigma)' = ([0] \times \sigma) + (r[0] \times \sigma') = (0 \times \sigma) + (r \times \sigma') = r\sigma'$. The last fact is an immediate consequence of the previous ones. □

It follows that π is polynomial iff there exist $n \geq 0$ and $p_0, p_1, p_2, \ldots, p_n \in \mathbb{R}$ such that $\pi = (p_0, p_1, p_2, \ldots, p_n, 0, 0, \ldots)$ iff there exist $n \geq 0$ such that $\pi^{(n)} = 0$.

Next consider the polynomials $1 + X$ and $2 + 7X^2$. By repeatedly computing derivatives, using Lemma 3.3 and the defining differential equation of sum, we find the following sequence (of transitions in the automaton $\mathbb{R}^\omega$ of streams):

$$(1 + X) + (2 + 7X^2) \longrightarrow (1 + 7X) \longrightarrow 7 \longrightarrow 0 \circlearrowleft$$

Computing for each of these terms the initial value, one obtains the stream $(3, 1, 7, 0, 0, \ldots)$, whence

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

Similarly, one has

$$(1 + X) \times (2 + 7X^2) \longrightarrow 2 + 7X + 7X^2 \longrightarrow 7 + 7X \longrightarrow 7 \longrightarrow 0 \circlearrowleft$$

(where the second term follows from $((1 + X) \times (2 + 7X^2))' = (1 \times (2 + 7X^2)) + (1 \times 7X) = 2 + 7X + 7X^2$). Computing initial values again yields the stream

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

For a slightly more exciting example, consider the polynomial $\sigma = 2 + 3X + 7X^2$, for which we want to compute the inverse σ^{-1}. Using the the defining differential equations for sum, product, and inverse, the first three derivatives are computed:

$$\sigma^{-1} \longrightarrow (-\tfrac{3}{2} - \tfrac{7}{2}X)\sigma^{-1} \longrightarrow (-\tfrac{5}{4} + \tfrac{21}{4}X)\sigma^{-1} \longrightarrow (\tfrac{57}{8} + \tfrac{35}{8}X)\sigma^{-1} \longrightarrow \cdots$$

Computing as before the respective initial values, the four first coefficients of σ^{-1} are obtained:

$$(2 + 3X + 7X^2)^{-1} = (\frac{1}{2}, -\frac{3}{4}, -\frac{5}{8}, \frac{57}{16}, \ldots) = \frac{1}{2} - \frac{3}{4}X - \frac{5}{8}X^2 + \frac{57}{16}X^3 + \cdots$$

Note that here the result is no longer a polynomial stream. The equality, which will be formally justified in Theorem 5.2, uses the operator of infinite sum, which is introduced next

Let I be a possibly infinite index set and let $\{\sigma_i \mid i \in I\}$ be a family of streams indexed by I. Such a family is *locally finite* if for every $k \geq 0$, the set

$$\{i \in I \mid \sigma_i(k) \neq 0\}$$

is finite. This is a sufficient condition for the existence of the *indexed sum* of the entire family, which is defined by the following behavioural differential equation:

differential equation	initial value
$(\sum_I \sigma_i)' = \sum_I \sigma_i'$	$(\sum_I \sigma_i)(0) = \sum_I \sigma_i(0)$

Note that in the definition of the initial value, the sum symbol on the right refers to ordinary summation of real numbers, which is finite by the condition of local finiteness. For streams $\sigma, \tau \in \mathbb{R}^\omega$ with $\tau(0) = 0$, the operation of *stream composition* $\sigma(\tau)$ (read: σ after τ), is defined as follows:

differential equation	initial value
$\sigma(\tau)' = \tau' \times \sigma'(\tau)$	$\sigma(\tau)(0) = \sigma(0)$

This operator behaves as one would expect. For instance, using coinduction one easily shows that

$$(2 + 3X + 7X^2)(\tau) = 2 + 3\tau + 7\tau^2$$

In conclusion of this section, let us note that the definition of (sum and) product $\sigma \times \tau$ could also have been given in the following, more traditional fashion, by specifying, for all $n \geq 0$, the value of the n-th element of $\sigma \times \tau$, in terms of the elements of σ and τ:

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

The reader is invited to use the coinduction proof principle to prove this equality formally, possibly after first having gained some experience with coinductive proofs in Section 4. There are several reasons, however, for preferring the coinductive definition by means of a behavioural differential equation above this type of pointwise definition. The most important ones have been listed in the introduction.

4 Proofs by coinduction

Here we convince ourselves that our operators have the usual properties and, at the same time, gain some experience in the technique of proofs by coinduction.

In order to prove the equality $\sigma = \tau$ of two streams σ and τ, it is sufficient according to Theorem 2.4, to show that there exists a bisimulation relation $R \subseteq \mathbb{R}^\omega \times \mathbb{R}^\omega$ with $\langle \sigma, \tau \rangle \in R$. Such a relation R can be constructed in stages, by computing the respective derivatives of both σ and τ step by step. The first pair to be included in R is $\langle \sigma, \tau \rangle$. Next the following step is repeated until it does not yield any new pairs: for a pair $\langle \phi, \psi \rangle$ in R, one computes $\langle \phi', \psi' \rangle$ and adds it

to R if it was not yet present. When adding a pair $\langle \phi, \psi \rangle$ to R, at any stage of its construction, one should check whether $\phi(0) = \psi(0)$. If this condition is not fulfilled, the procedure aborts and we conclude that $\sigma \neq \tau$. If the procedure never aborts then the relation R is, by construction, a bisimulation and $\sigma = \tau$ follows, by Theorem 2.4.

Coinduction can be used to prove the equality of 'concrete' equalities, such as $\mathcal{T}(exp(x)) = (1, 1, 1, \ldots)$ at the beginning of Section 3. But it is also possible to use coinduction to prove *laws* for streams, such as $\sigma + \tau = \tau + \sigma$, in which the streams are universally quantified. The following theorem and its proof contains many examples.

Theorem 4.1 *For all streams σ, τ, and ρ in $\mathbb{R}^\omega$, and real numbers $r \in \mathbb{R}$:*

$$\begin{aligned}
\sigma + 0 &= \sigma && (2)\\
\sigma + \tau &= \tau + \sigma && (3)\\
\sigma + (\tau + \rho) &= (\sigma + \tau) + \rho && (4)\\
\sigma \times (\tau + \rho) &= (\sigma \times \tau) + (\sigma \times \rho) && (5)\\
(\tau + \rho) \times \sigma &= (\tau \times \sigma) + (\rho \times \sigma) && (6)\\
1 \times \sigma &= \sigma && (7)\\
\sigma \times 1 &= \sigma && (8)\\
0 \times \sigma &= 0 && (9)\\
\sigma \times 0 &= 0 && (10)\\
\sigma \times (\tau \times \rho) &= (\sigma \times \tau) \times \rho && (11)\\
r \times \sigma &= \sigma \times r && (12)\\
\sigma^* &= (1 + \sigma(0) - \sigma)^{-1} && (13)
\end{aligned}$$

If σ is such that $\sigma(0) = 0$ then also

$$\begin{aligned}
\sigma^* &= 1 + \sigma + \sigma^2 + \sigma^3 + \cdots && (14)\\
1 + \sigma\sigma^* &= \sigma^* && (15)\\
\tau = (\sigma \times \tau) + \rho &\Rightarrow \tau = \sigma^* \times \rho && (16)\\
(\sigma + \tau)^* &= \tau^* \times (\sigma \times \tau^*)^* && (17)\\
(\sigma + \tau)^* &= (\tau^* \times \sigma)^* \times \tau^* && (18)
\end{aligned}$$

Moreover, for $\sigma \in \mathbb{R}^\omega$ with $\sigma(0) \neq 0$ and $r \in \mathbb{R}$ with $r \neq 0$,

$$\begin{aligned}
\sigma^{-1} &= \sigma(0)^{-1}(1 - \sigma(0)^{-1}\sigma)^* && (19)\\
(r \times \sigma)^{-1} &= r^{-1} \times \sigma^{-1} && (20)\\
\sigma \times \sigma^{-1} &= 1 && (21)\\
\sigma^{-1} \times \sigma &= 1 && (22)
\end{aligned}$$

Proof: As usual, we shall be writing $\sigma\tau$ for $\sigma \times \tau$. We shall prove a few of the above identities, leaving the others as exercises for the reader. For (2), note that

$$\{\langle \sigma + 0, \sigma \rangle \mid \sigma \in \mathbb{R}^\omega\}$$

is a bisimulation relation on $\mathbb{R}^\omega$, because $(\sigma+0)(0) = \sigma(0)+0 = \sigma(0)$ and $\langle (\sigma+0)', \sigma' \rangle = \langle \sigma'+0, \sigma' \rangle$, which is an element of the relation again. The identity now follows by coinduction. Similarly, (3) follows by coinduction from the fact that

$$\{\langle \sigma + \tau, \tau + \sigma \rangle \mid \sigma, \tau \in \mathbb{R}^\omega\}$$

is a bisimulation relation. The identity (4) is proved in the same way. In the construction of a bisimulation relation for (5), one starts with the set

$$R_1 = \{\langle \sigma(\tau + \rho), \sigma\tau + \sigma\rho \rangle \mid \sigma, \tau, \rho \in \mathbb{R}^\omega\}$$

Computing the (elementwise) derivative of such pairs yields

$$\begin{aligned}
&\langle (\sigma(\tau+\rho))',\, (\sigma\tau+\sigma\rho)'\rangle \\
&\quad = \langle \sigma'(\tau+\rho)+\sigma(0)(\tau'+\rho'),\, (\sigma'\tau+\sigma(0)\tau')+(\sigma'\rho+\sigma(0)\rho'\rangle \\
&\quad = \langle \sigma'(\tau+\rho)+\sigma(0)(\tau'+\rho'),\, (\sigma'\tau+\sigma'\rho)+(\sigma(0)\tau'+\sigma(0)\rho')\rangle \quad \text{[using (4) and (3)]}
\end{aligned}$$

which is not in R_1, even though each of the pairs

$$\langle \sigma'(\tau+\rho),\, \sigma'\tau+\sigma'\rho\rangle \quad \text{and} \quad \langle \sigma(0)(\tau'+\rho'),\, \sigma(0)\tau'+\sigma(0)\rho'\rangle$$

is. As described above, the way to turn R_1 into a bisimulation is by simply adding all new pairs. One moment's thought tells us that all pairs that we shall encounter while computing derivatives of old pairs are included in the set

$$R_2 = \{\langle \sigma_1(\tau_1+\rho_1)+\cdots+\sigma_n(\tau_n+\rho_n),\, (\sigma_1\tau_1+\sigma_1\rho_1)+(\sigma_n\tau_n+\sigma_n\rho_n)\rangle \mid \sigma_i,\tau_i,\rho_i \in \mathbb{R}^\omega\}$$

which is readily verified to be a bisimulation. Now (5) follows by coinduction. Using (3), (4), and (5), it is similarly easy to show that

$$\{\langle \sigma_1(\tau_1\rho_1)+\cdots+\sigma_n(\tau_n\rho_n),\, (\sigma_1\tau_1)\rho_1+\cdots+(\sigma_n\tau_n)\rho_n\rangle \mid \sigma_i,\tau_i,\rho_i \in \mathbb{R}^\omega\}$$

is a bisimulation relation on $\mathbb{R}^\omega$, which proves (11). For (21), we compute

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

and, using some of the earlier identities,

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

This shows that the set

$$\{\langle \sigma\sigma^{-1},1\rangle \mid \sigma \in \mathbb{R}^\omega,\ \sigma(0)\neq 0\} \cup \{\langle 0,0\rangle\}$$

is a bisimulation relation on $\mathbb{R}^\omega$, from which (21) follows. Identity (17) follows from the fact that

$$\{\langle \rho(\sigma+\tau)^*,\, \rho\tau^*(\sigma\tau^*)^*\rangle \mid \sigma,\tau,\rho \in \mathbb{R}^\omega,\ \sigma(0)=0\}$$

is a bisimulation relation, and identity (13) follows from the fact that

$$\{\langle \tau_1\sigma^*+\cdots+\tau_n\sigma^*,\, \tau_1(1-\sigma)^{-1}+\cdots+\tau_n(1-\sigma)^{-1}\rangle \mid \sigma \in \mathbb{R}^\omega\}$$

is a bisimulation relation. The reader is invited to prove the remaining cases him or herself. Notably (16) is an entertaining exercise in coinduction, at least, if one is willing to prove it without making use of the presence of the inverse operator. □

Some of the above equalities are more interesting than others. Most of them are very familiar, but not all. For instance, the definition of the inverse operator by means of a behavioural differential

equation, contains a description of an algorithm for its stepwise computation (closely resembling so-called long division). From this perspective, the identity

$$\sigma \times \sigma^{-1} = 1$$

with its almost trivial proof above, shows that this algorithm is correct. In fact, it was this equality which was taken as a 'specification', from which the behavioural differential equation for inverse was deduced.

5 Stream calculus

We have seen that it is possible to define streams by means of behavioural differential equations, in very much the same way as one uses differential equations in mathematical analysis to define functions on the reals. Some further basic 'stream calculus' will be developed next, including a formalisation of the view of streams as formal power series, which was used in the motivations of Theorem 3.2. In Section 6, a second product operator will be defined, called shuffle product, which is better behaved with respect to stream derivation, and with which many more streams can be defined.

The main ingredient of stream calculus is the following theorem.

Theorem 5.1 [Fundamental Theorem of stream calculus] *For all streams $\sigma \in \mathbb{R}^\omega$:*

$$\sigma = \sigma(0) + (X \times \sigma')$$

Proof: Left multiplication with X amounts to prefixing with 0. Informally, therefore, the equality of the theorem simply asserts that $(s_0, s_1, s_2, \ldots) = (s_0, 0, 0, \ldots) + (0, s_1, s_2, \ldots)$. More formally, the theorem follows by coinduction from the fact that

$$\{\langle \sigma, \sigma(0) + X\sigma' \rangle \mid \sigma \in \mathbb{R}^\omega\} \cup \{\langle \sigma, \sigma \rangle \mid \sigma \in \mathbb{R}^\omega\}$$

is a bisimulation relation on $\mathbb{R}^\omega$, which is immediate by the equality $(X\sigma)' = \sigma$ of Lemma 3.3. $\square$

If σ is the Taylor series of an analytical function, that is, $\sigma = \mathcal{T}(f)$ for a function $f \in \mathcal{A}$ (recall the homomorphism $\mathcal{T} : \mathcal{A} \to \mathbb{R}^\omega$ from Example 2.7), then the theorem expresses what the fundamental theorem of analysis asserts for functions: $f = f(0) + \int_0^x f'$. In order to prove this correspondence, we apply $\mathcal{T}$ to the latter equation:

$$\begin{array}{rcl}
\sigma & = & \mathcal{T}(f) \\
& = & \mathcal{T}(f(0) + \int_0^x f') \\
& = & \mathcal{T}(f(0)) + \mathcal{T}(\int_0^x f') \\
& = & \mathcal{T}(f)(0) + X\mathcal{T}(f') \\
& = & \mathcal{T}(f)(0) + X\mathcal{T}(f)' \\
& = & \sigma(0) + X\sigma'
\end{array}$$

(Here the fact is used that $\mathcal{T}(\int_0^x g) = X\mathcal{T}(g)$, for any $g \in \mathcal{A}$, which can be easily proved by coinduction, since $\{\langle \mathcal{T}(\int_0^x g), X\mathcal{T}(g) \rangle \mid g \in \mathcal{A}\} \cup \{\langle \sigma, \sigma \rangle \mid \sigma \in \mathbb{R}^\omega\}$ is a bisimulation on $\mathbb{R}^\omega$.) This correspondence thus shows that left multiplication with X can be interpreted as stream integration. As a consequence, stream calculus is rather pleasant in that every stream is both differentiable and integrable: σ' and $X\sigma$ are defined for every stream σ.

Next a formal power series expansion theorem for streams is formulated. Intuitively, it is obtained by applying Theorem 5.1 to $\sigma' = \sigma^{(1)}$ again, and substituting the result in $\sigma = \sigma(0) + X\sigma'$. This yields

$$\begin{array}{rcl}
\sigma & = & \sigma(0) + X\sigma^{(1)} \\
& = & \sigma(0) + X(\sigma^{(1)}(0) + X\sigma^{(2)}) \\
& = & \sigma(0) + \sigma^{(1)}(0)X + X^2\sigma^{(2)}
\end{array}$$

where the latter equality holds by the distributivity law (5) and the commutativity law for scalar multiplication (12). Continuing this way, one finds an expansion theorem for streams.

Theorem 5.2 *For all streams* $\sigma \in \mathbb{R}^\omega$,

$$\begin{aligned}\sigma &= \sum_{n=0}^{\infty} \sigma^{(n)}(0) \times X^n \\ &= \sigma(0) + \sigma^{(1)}(0)X + \sigma^{(2)}(0)X^2 + \sigma^{(3)}(0)X^3 + \cdots \\ &= \sigma(0) + \sigma(1)X + \sigma(2)X^2 + \sigma(3)X^3 + \cdots\end{aligned}$$

Note that the expression on the right denotes a stream as well, which is built from constants ($\sigma^{(n)}(0)$ and X), product, and infinite sum. The theorem asserts that σ is equal to a *formal power series* (in the single formal variable X). The theorem is similar but different from Taylor's expansion theorem from analysis, since the coefficients of X^n are $\sigma^{(n)}(0)$ rather than $\sigma^{(n)}(0)/n!$. A true Taylor expansion theorem for streams will be formulated later (Theorem 6.3) using a new (shuffle) product operator.

Proof of Theorem 5.2: First note that the family $\{\sigma^{(n)}(0)X^n \mid n \geq 0\}$ is locally finite since $(X^k)' = X^{k-1}$, for $k \geq 1$ (Lemma 3.3), and $X' = 1$ imply that, for all $k \geq 0$,

$$\sigma^{(n)}(0)(X^n)^{(k)}(0) \neq 0 \iff k = n$$

Next note that the set $R = \{\langle \sigma, \sum_{n=0}^{\infty} \sigma^{(n)}(0)X^n \rangle \mid \sigma \in \mathbb{R}^\omega\}$ is a bisimulation relation on $\mathbb{R}^\omega$, since all pairs in R have the same initial value $\sigma(0)$ and, moreover,

$$\begin{aligned}&(\sum_{n=0}^{\infty} \sigma^{(n)}(0)X^n)' \\ &= \sum_{n=0}^{\infty} (\sigma^{(n)}(0)X^n)' \\ &= \sum_{n=1}^{\infty} \sigma^{(n)}(0)X^{n-1} \quad [\text{using } (r\tau)' = r\tau' \text{ for } r \in \mathbb{R} \text{ and } \tau \in \mathbb{R}^\omega] \\ &= \sum_{n=0}^{\infty} \sigma^{(n+1)}(0)X^n \\ &= \sum_{n=0}^{\infty} (\sigma')^{(n)}(0)X^n\end{aligned}$$

and $\langle \sigma', \sum_{n=0}^{\infty} (\sigma')^{(n)}(0)X^n \rangle$ is in R. The theorem now follows by coinduction Theorem 2.4. □

The next example shows how the fundamental theorem for streams can be used to construct so-called closed forms for a large family of differential equations.

Example 5.3 Recall our first behavioural differential equation from Section 3:

differential equation	initial value
$\sigma' = \sigma$	$\sigma(0) = 1$

There we saw that this equation has a unique solution (the stream $(1, 1, 1, \ldots)$), using the finality of $\mathbb{R}^\omega$. Alternatively, a solution can be quickly computed using the fundamental stream theorem, as follows. Substituting $\sigma(0) + X\sigma' = 1 + X\sigma'$ for σ, we find

$$\sigma' = 1 + X\sigma'$$

which is equivalent to $\sigma'(1-X) = 1$, implying $\sigma' = (1-X)^{-1}$. Since $\sigma' = \sigma$, we find $\sigma = (1-X)^{-1}$. This gives us a description of the solution of the differential equation in terms of constants and

operators. We shall call such a term a *closed form* for σ. Note that in the computation above, the fact that we already knew a solution to the equation, was not used. In other words, we have obtained a new method for solving such equations. Because no properties of σ were used, it follows that the obtained solution is unique.

Here is another example. The following behavioural differential equation defines the constant stream of the Fibonacci numbers $(0, 1, 1, 2, 3, 5, 8, 13, \ldots)$:

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

Note that the equation avoids the use of indices of the usual definition of the Fibonacci numbers $(F_n)_n$ in terms of the recurrence $F_0 = 0$, $F_1 = 1$ and, for $n \geq 2$,

$$F_n = F_{n-1} + F_{n-2}$$

The fact that our differential equation uses a second derivative need not bother us: putting $\tau = \sigma'$, the following system of two (mutually dependent) ordinary equations also defines σ:

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

Returning to the original higher-order equation, we have, according to Theorem 5.1, $\sigma' = 1 + X\sigma''$ and $\sigma = X\sigma' = X + X^2\sigma''$. Substituting this in $\sigma'' = \sigma' + \sigma$ gives an equation with σ'' as the only unknown:

$$\sigma'' = (1 + X\sigma'') + (X + X^2\sigma'')$$

which is equivalent to

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

yielding $\sigma'' = (1 + X)(1 - X - X^2)^{-1}$. Together with $\sigma = X + X^2\sigma''$, we find

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

which gives us a closed expression for the Fibonacci numbers. The following table summarises the above and similar such examples:

differential equation	initial value	solution	closed form
$\sigma' = 0$	$\sigma(0) = r$	$(r, 0, 0, \ldots)$	$[r]$
$\sigma' = [1]$	$\sigma(0) = 0$	$(0, 1, 0, 0, \ldots)$	X
$\sigma' = \sigma$	$\sigma(0) = 1$	$(1, 1, 1, \ldots)$	$(1 - X)^{-1}$
$\sigma' = -\sigma$	$\sigma(0) = 1$	$(1, -1, 1, -1, \ldots)$	$(1 + X)^{-1}$
$\sigma' = X\sigma$	$\sigma(0) = 1$	$(1, 0, 1, 0, \ldots)$	$(1 - X^2)^{-1}$
$\sigma' = \sigma + (1 - X)^{-1}$	$\sigma(0) = 0$	$(0, 1, 2, 3, \ldots)$	$X(1 - X)^{-2}$
$\sigma' = 2\sigma$	$\sigma(0) = 1$	$(1, 2, 4, 8, 16, \ldots)$	$(1 - 2X)^{-1}$
$\sigma' = r\sigma$	$\sigma(0) = 1$	$(1, r, r^2, r^3, \ldots)$	$(1 - rX)^{-1}$
$\sigma'' = -\sigma$	$\sigma(0) = 0,\ \sigma'(0) = 1$	$(0, 1, 0, -1, 0, 1, 0, -1, \ldots)$	$X(1 + X^2)^{-1}$
$\sigma'' = \sigma' + \sigma$	$\sigma(0) = 0,\ \sigma'(0) = 1$	$(0, 1, 1, 2, 3, 5, 8, 13, \ldots)$	$X(1 - X - X^2)^{-1}$

In all but the first two cases, which are by definition, the closed form is obtained from the defining differential equation using the fundamental theorem of stream calculus. □

The closed forms of most of the above streams are well-known from elementary analysis: if $(s_0, s_1, s_2, \ldots)$ is a stream of real numbers such that the power series $\sum s_n x^n$ has a positive radius of convergence, then the function

$$S(x) = \sum s_n x^n$$

is called a *generating function* for the stream $(s_0, s_1, s_2, \ldots)$. For instance, the function $S(x) = \sum x^n$ converges for all x between -1 and 1, and thus is a generating function for the stream $(1, 1, 1, \ldots)$. For this function, there is the following closed form

$$S(x) = (1 - x)^{-1}$$

which corresponds to the closed form $(1 - X)^{-1}$ that we found for the stream $(1, 1, 1, \ldots)$ above. There are a few differences between classical calculus and stream calculus to be noted, however:

- The expression $(1 - X)^{-1}$ is not a function, but is itself a stream: $(1 - X)^{-1} = (1, 1, 1, \ldots)$.
- A generating function in analysis generates a stream by means of classical differentiation, using the following formula: $S^{(n)}(0) = n!s_n$. For instance, the n-th derivative of the function $S(x) = (1 - x)^{-1}$ is $S^{(n)} = n!(1 - x)^{-(n+1)}$, whence $S^{(n)}(0) = n!$, which implies $s_n = 1$, indeed. In stream calculus, streams are generated by means of stream differentiation: if $\sigma = (s_0, s_1, s_2, \ldots)$ then $s_n = \sigma^{(n)}(0)$. But stream differentiation is rather different from analytic differentiation: for our closed form $(1 - X)^{-1}$ we have $((1 - X)^{-1})^{(n)} = (1 - X)^{-1}$, for any n, which implies $s_n = ((1 - X)^{-1})^{(n)}(0) = 1$.
- In stream calculus, *any* stream σ generates its elements by means of stream differentiation. Convergence of the power series corresponding to σ is simply not an issue. We shall later see examples of streams for which, in analysis, no generating functions exist, but which nevertheless have closed forms in stream calculus (cf. Example 6.6).

6 More stream calculus: shuffle product and shuffle inverse

There is somehow a 'mismatch' between the computation of the stream derivative of the product of two streams σ and τ:

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

and the the familiar rule from calculus for the derivative of function product:

$$(f \cdot g)' = f' \cdot g + f \cdot g'$$

(where $(f \cdot g)(x) = f(x) \cdot g(x)$). Amongst other things, this mismatch is responsible for the fact that Theorem 5.2 is only 'Taylor-like' and does not correspond precisely to the usual Taylor expansion theorem from analysis. This can be overcome using a different product operator on streams, called *shuffle product*, for which derivation behaves as we are used to. At the same time, the use of this operator will further increase the expressiveness of stream calculus.

The new product operator is defined by means of a behavioural differential equation, which simply asserts the property that we wish it to satisfy, namely, that the derivative of the product behaves as it does in analysis:

differential equation	initial value	name
$(\sigma \otimes \tau)' = (\sigma' \otimes \tau) + (\sigma \otimes \tau')$	$(\sigma \otimes \tau)(0) = \sigma(0) \times \tau(0)$	shuffle product

The name of this operator is explained by the fact that when σ and τ are formal languages, the formula above defines the set of all possible *shuffles* (interleavings) of words in σ and τ (see Section 10).

It will be convenient to have also an operator which acts as the inverse to shuffle product. Classical analysis is again our source of inspiration, where for the inverse of a function we have

$$(f^{-1})' = -f' \cdot (f^{-1} \cdot f^{-1})$$

(with $f^{-1}(x) = f(x)^{-1}$ for x such that $f(x) \neq 0$). This shows us the way how to define an operation of *shuffle inverse* on streams σ with $\sigma(0) \neq 0$:

differential equation	initial value	name
$(\sigma^{-\underline{1}})' = -\sigma' \otimes (\sigma^{-\underline{1}} \otimes \sigma^{-\underline{1}})$	$\sigma^{-\underline{1}}(0) = \sigma(0)^{-1}$	shuffle inverse

Note that the symbol $\sigma^{-\underline{1}}$ is used in order to distinguish this operator from the previous inverse operator σ^{-1}. The shuffle product can also be defined by the following more traditional formula:

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

For the same reasons as in the case of ordinary product, all computations involving shuffle product will be based on its coinductive definition by means of a differential equation (cf. the remarks at the end of Section 3). As with σ^{-1}, we have no idea how to define the shuffle inverse of a stream by means of a formula for its n-th element: '$\sigma^{-\underline{1}}(n) =$?'.

Remark 6.1 The relation between the (pointwise) operators on functions and the corresponding operators on streams, can be precisely expressed using again the homomorphism $\mathcal{T} : \mathcal{A} \to \mathbb{R}^\omega$ from Example 2.7. For all analytic functions f and g in $\mathcal{A}$,

$$\begin{aligned} \mathcal{T}(f+g) &= \mathcal{T}(f) + \mathcal{T}(g) \\ \mathcal{T}(f \cdot g) &= \mathcal{T}(f) \otimes \mathcal{T}(g) \\ \mathcal{T}(f^{-1}) &= \mathcal{T}(f)^{-\underline{1}} \end{aligned}$$

In order to prove this, let $R \subseteq \mathbb{R}^\omega \times \mathbb{R}^\omega$ be the smallest relation such that $\langle \mathcal{T}(f), \mathcal{T}(f) \rangle \in R$, for all $f \in \mathcal{A}$ and such that if $\langle \mathcal{T}(f), \sigma \rangle \in R$ and $\langle \mathcal{T}(g), \tau \rangle \in R$ then $\langle \mathcal{T}(f+g), \sigma + \tau \rangle \in R$, $\langle \mathcal{T}(f \cdot g), \sigma \otimes \tau \rangle \in R$, and $\langle \mathcal{T}(f^{-1}), \sigma^{-\underline{1}} \rangle \in R$. Then R is a bisimulation and the equations follow by coinduction. □

Next a number of properties of shuffle product and inverse is proved. We shall be using the following conventions: for all $\sigma, \tau \in \mathbb{R}^\omega$, $r \in \mathbb{R}$, $n \geq 0$,

$$\sigma^0 = 1, \quad \sigma^{n+1} = \sigma \times \sigma^n, \quad \sigma^{-n} = (\sigma^{-1})^n$$

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

$$r\sigma = r \times \sigma = r \otimes \sigma$$

(For the latter equality, see (23) below. Whenever σ is not a real number, $\sigma\tau$ will always mean $\sigma \times \tau$ and never $\sigma \otimes \tau$.)

Theorem 6.2 *For all $\sigma, \tau, \rho \in \mathbb{R}^\omega$, $r \in \mathbb{R}$, $n \geq 0$,*

$$r \times \sigma = r \otimes \sigma \tag{23}$$

$$\sigma \otimes (\tau \otimes \rho) = (\sigma \otimes \tau) \otimes \rho \tag{24}$$

$$\sigma \otimes \tau = \tau \otimes \sigma \tag{25}$$

$$\sigma \otimes (\tau + \rho) = (\sigma \otimes \tau) + (\sigma \otimes \rho) \tag{26}$$

$$\sigma \otimes \sigma^{-\underline{1}} = 1 \tag{27}$$

$$(\sigma^{\underline{n+1}})' = (n+1)\sigma' \otimes \sigma^{\underline{n}} \tag{28}$$

$$X^{\underline{n}} = n! X^n \tag{29}$$

Proof: Good exercise in coinduction and induction. □

Theorem 6.3 [Taylor's theorem for streams] *For all streams $\sigma \in \mathbb{R}^\omega$:*

$$\sigma = \sum_{n=0}^{\infty} \frac{\sigma^{(n)}(0)}{n!} X^{\underline{n}} = \sum_{n=0}^{\infty} \frac{\sigma(n)}{n!} X^{\underline{n}}$$

Proof: The theorem is a corollary of Theorem 5.2 and equation (29) above. □

The following formula is the basis for a good understanding of the shuffle product:

$$X \otimes (s_0, s_1, s_2, s_3, \ldots) = (0, 1s_0, 2s_1, 3s_2, 4s_3, \ldots)$$

At the same time, it explains the relevance of the following definition:

$$\Theta(\sigma) = X \otimes (\sigma')$$

since $\Theta(\sigma) = (0s_0, 1s_1, 2s_2, 3s_3, \ldots)$. It is used in the definition of the following operator on streams:

differential equation	initial value
$\Delta(\sigma)' = \Delta(\Theta(\sigma)')$	$(\Delta(\sigma))(0) = \sigma(0)$

One can easily prove that for $\sigma = (s_0, s_1, s_2, s_3, \ldots)$,

$$\Delta(\sigma) = (0!s_0, 1!s_1, 2!s_2, 3!s_3, \ldots)$$

The operator Δ transforms ordinary product and inverse into shuffle product and inverse, which will be proved using the following lemma.

Lemma 6.4 *For all streams σ and τ in $\mathbb{R}^\omega$,*

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

Proof: The proof is a stimulating exercise in coinduction. □

Theorem 6.5 *For all $r \in \mathbb{R}$, $\sigma, \tau \in \mathbb{R}^\omega$,*

$$\begin{aligned}
\Delta(r) &= r \\
\Delta(X) &= X \\
\Delta(\sigma + \tau) &= \Delta(\sigma) + \Delta(\tau) \\
\Delta(\sigma \times \tau) &= \Delta(\sigma) \otimes \Delta(\tau) \\
\Delta(\sigma^{-1}) &= \Delta(\sigma)^{-\underline{1}}
\end{aligned}$$

Proof: The first three equalities are straightforward. For the fourth, use Lemma 6.4 and coinduction. The last equality follows from the previous ones, since for all $\sigma \in \mathbb{R}^\omega$ with $\sigma(0) \neq 0$,

$$\begin{aligned}
&\Delta(\sigma^{-1}) \\
&= 1 \otimes \Delta(\sigma^{-1}) \\
&= (\Delta(\sigma)^{-\underline{1}} \otimes \Delta(\sigma)) \otimes \Delta(\sigma^{-1}) \quad \text{[equations (25) and (27)]} \\
&= \Delta(\sigma)^{-\underline{1}} \otimes (\Delta(\sigma) \otimes \Delta(\sigma^{-1})) \quad \text{[equations (24)]} \\
&= \Delta(\sigma)^{-\underline{1}} \otimes \Delta(\sigma \times \sigma^{-1}) \\
&= \Delta(\sigma)^{-\underline{1}} \otimes \Delta(1) \quad \text{[equation (21)]} \\
&= \Delta(\sigma)^{-\underline{1}} \otimes 1 \\
&= \Delta(\sigma)^{-\underline{1}}
\end{aligned}$$

This concludes the proof of the theorem. □

Next a few examples are presented of the use of shuffle product and shuffle inverse in various definitions of streams.

Example 6.6 Recall from Example 5.3 that $(1-X)^{-1} = (1,1,1,\ldots)$. By Theorem 6.5, one has

$$(1-X)^{-\underline{1}} = \Delta((1-X)^{-1}) = \Delta((1,1,1,\ldots)) = (0!,1!,2!,\ldots)$$

For this stream, there exists in traditional calculus no generating function, since the infinite sum

$$\sum_{n=0}^{\infty} n!x^n$$

diverges for all x with $x \neq 0$. This shows that in stream calculus, more streams can be represented.

The two differential equations below are another illustration of the difference between ordinary and shuffle product:

differential equation	initial value	solution
$\sigma' = X \times \sigma$	$\sigma(0) = 1$	$(1,0,1,0,\ldots)$
$\tau' = X \otimes \tau$	$\tau(0) = 1$	$(1,0,1\times 3,0,1\times 3\times 5,\ldots)$

For the former, we have a closed form $\sigma = (1-X^2)^{-1}$. (What is a closed form for τ, we wonder?)

Here is another example involving the operator Δ, in two differential equations defining streams of fractions:

differential equation	initial value	solution
$\Delta(\sigma)' = \Delta(\sigma)$	$\Delta(\sigma)(0) = 1$	$(1/0!,1/1!,1/2!,1/3!,\ldots)$
$\Delta(\tau)' = (1-X)^{-\underline{1}}$	$\Delta(\tau)(0) = 0$	$(0,1/1,1/2,1/3,\ldots)$

The first solution is obtained by first solving the differential equation, considering $\Delta(\sigma)$ as the variable. This yields $\Delta(\sigma) = (1-X)^{-1}$. Next the obtained equality is turned into an (infinite) system of equations, by unfolding the left and right sides:

$$(0!s_0, 1!s_1, 2!s_2, 3!s_3, \ldots) = (1,1,1,\ldots)$$

The second solution is found in a similar fashion.

We have seen examples of differential equations on functions, such as the one for the exponential function at the beginning of Section 3, that could be also interpreted as behavioural differential equations on streams. The correspondence between function multiplication and shuffle product allows us to interpret also equations involving products. For instance, the following analytical differential equation

analytical differential equation	initial value
$f' = 1 + f^2$	$f(0) = 0$

(where $f^2 = f \cdot f$) is equivalent, on the basis of the correspondence between function product and shuffle product, with the following behavioural differential equation:

behavioural differential equation	initial value
$\tau' = 1 + \tau^{\underline{2}}$	$\tau(0) = 0$

(where, recall, $\tau^{\underline{2}} = \tau \otimes \tau$). The equivalence consists of the fact that an analytic function f is a solution of the first equation if and only if its Taylor series $\mathcal{T}(f)$ is a solution of the second one. Since we know (recall, can look up) from analysis that the tangent function $tan(x)$ is the unique solution of the first equation, it follows that the second equation uniquely defines the Taylor series τ of $tan(x)$. The advantage of this translation, which allows us to reason about τ directly inside the world of stream calculus, will become apparent when we shall deal with nondeterministic representations of streams, which is the subject of Section 8. □

In conclusion of this part on stream calculus, two further examples of operators on streams are defined. The *square root* of a stream σ with $\sigma(0) \neq 0$ is defined by the following differential equation:

differential equation	initial value	name
$(\sqrt{\sigma})' = 1/2(\sqrt{\sigma})^{-\underline{1}} \otimes \sigma'$	$(\sqrt{\sigma})(0) = \sqrt{\sigma(0)}$	square root

Variations on this type of definition can be easily constructed. Proving the expected property that $\sqrt{\sigma} \otimes \sqrt{\sigma} = \sigma$ is yet another exercise in, as always, coinduction. The next equation defines for any stream σ the so-called *shuffle star* $\sigma^{\underline{*}}$ (also called shuffle closure):

differential equation	initial value	name
$(\sigma^{\underline{*}})' = \sigma' \otimes \sigma^{\underline{*}} \otimes \sigma^{\underline{*}}$	$(\sigma^{\underline{*}})(0) = 1$	shuffle star

The notation, name, and equation for this operator are best explained by the equalities below, which show that shuffle star is for shuffle product what star is for ordinary product. For all σ with $\sigma(0) = 0$,

$$\begin{aligned} \sigma^{\underline{*}} &= 1 + \sigma + (\sigma \otimes \sigma) + (\sigma \otimes \sigma \otimes \sigma) + \cdots \\ &= 1 + \sigma + \sigma^{\underline{2}} + \sigma^{\underline{3}} + \cdots \\ &= (1 - \sigma)^{-\underline{1}} \end{aligned}$$

So also shuffle star is a definable operator in the presence of shuffle inverse but, again, there will be situations without the presence of shuffle inverse, where the operator of shuffle star is still useful.

7 Rational streams

The set $\mathcal{R}$ of *rational* streams is the smallest collection such that

- $r \in \mathcal{R}$, for all $r \in \mathbb{R}$;
- $X \in \mathcal{R}$;
- and if $\sigma \in \mathcal{R}$ and $\tau \in \mathcal{R}$ then $\sigma + \tau \in \mathcal{R}$, $\sigma \times \tau \in \mathcal{R}$, and $\sigma^* \in \mathcal{R}$.

Expressions denoting rational streams are called *regular* and are generated by the following syntax:

$$E ::= r \ \ (\in \mathbb{R}) \mid X \mid E + F \mid EF \mid E^*$$

The following proposition shows that we might just as well have taken inverse rather than star in the definition above. (The reason for taking star is that we shall encounter situations where star is present but inverse is not; for instance, on the set of languages over a given alphabet.) Let us first reveal a long kept secret. For all streams σ and τ,

$$\sigma \times \tau = \tau \times \sigma \tag{30}$$

$$(\sigma \times \tau)^{-1} = \sigma^{-1} \times \tau^{-1} \tag{31}$$

The first equality is yet another nice exercise in coinduction, and the second is an easy consequence of the second (using identity (21)). Next recall that a stream π is polynomial if it is of the form $\pi = p_0 + p_1 X + p_2 X^2 + \cdots + p_n X^n$, for $n \geq 0$ and $p_0, \ldots, p_n \in \mathbb{R}$.

Proposition 7.1 *A stream σ is rational iff there exist polynomial streams π and ρ with $\rho(0) = 1$ such that*

$$\sigma = \pi \times \rho^{-1}$$

Proof: Let V be the collection of all streams of the form $\pi \times \rho^{-1}$ with π and ρ polynomial and $\rho(0) = 1$. Any polynomial π is clearly in $\mathcal{R}$ and so is the inverse of a polynomial ρ with $\rho(0) = 1$, because $\rho^{-1} = (1 - \rho)^*$, by identity (19). Thus $\pi \times \rho^{-1} \in \mathcal{R}$, which shows $V \subseteq \mathcal{R}$. Conversely, $r \in V$, for all $r \in \mathbb{R}$, and $X \in V$. If $\sigma = \pi\rho^{-1}$ and $\tau = \phi\psi^{-1}$ are in V, then so are

$$\begin{aligned} \sigma + \tau &= (\pi\psi + \phi\rho) \times (\psi\rho)^{-1} \\ \sigma \times \tau &= (\pi\phi) \times (\psi\rho)^{-1} \\ \sigma^* &= \rho \times (1 + \pi(0)\rho - \pi)^{-1} \end{aligned}$$

using (30), (31), and for the latter equality also (13). □

8 Nondeterministic stream automata

A polynomial $\pi = p_0 + p_1 X + \cdots + p_n X^n$ with $p_n \neq 0$ generates a finite subautomaton $\langle \pi \rangle \subseteq \mathbb{R}^\omega$ of size $n+2$, since

$$\begin{tikzcd} \pi \arrow[r] & (p_1 + p_2 X + \cdots + p_n X^{n-1}) \arrow[r] & \cdots \arrow[r] & (p_{n-1} + p_n X) \arrow[r] & p_n \arrow[r] & 0 \arrow[loop above] \end{tikzcd}$$

However, the subautomaton generated by the inverse of a polynomial or, more generally, by a rational stream, is usually infinite. A simple and typical example is the subautomaton of $\mathbb{R}^\omega$ generated by the stream $(1 - rX)^{-1} = (rX)^* = (1, r, r^2, r^3, \ldots)$:

$$(1 - rX)^{-1} \longrightarrow r(1 - rX)^{-1} \longrightarrow r^2(1 - rX)^{-1} \longrightarrow r^3(1 - rX)^{-1} \longrightarrow \cdots$$

which is infinite for all r with $r \neq 0, 1, -1$. In this section, a new type of automaton is introduced, with which finite representations for rational streams can be given. By allowing transitions with multiplicities in $\mathbb{R}$, the above transition sequence can then be captured by a one state automaton with one single transition (see Example 8.3 below).

An $\mathbb{R}$-*nondeterministic stream automaton*, or *nd-automaton* for short, is a pair $Q = (Q, \langle o, t \rangle)$ consisting of a set Q of states, and a pair of functions: As before, an output function $o : Q \to \mathbb{R}$; and a transition function which is now nondeterministic: $t : Q \to \mathbb{R}(Q)$ with

$$\mathbb{R}(Q) = \{\phi : Q \to \mathbb{R} \mid sp\,(\phi) \text{ is finite } \}$$

where $sp\,(\phi) = \{q \in Q \mid \phi(q) \neq 0\}$ is the *support* of ϕ. 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 $t(q) \in \mathbb{R}(Q)$. Such a function can be viewed as a kind of nondeterministic or distributed state, and specifies for any state q' in Q a real number $t(q)(q')$ in $\mathbb{R}$. This number can be thought of as the multiplicity with which the transition from q to q' occurs. (Another word for $t(q)(q')$ would be the *weight* of the transition. Our nd-automata could also be called *weighted* automata.) The following notation will be used:

$$q \xrightarrow{r} q' \iff t(q)(q') = r, \quad \text{and} \quad q \xrightarrow{r} \iff o(q) = r$$

which will allow us to present nd-automata by pictures. In such pictures, only those arrows will be drawn that have a non-zero label. If we put for a state q in an nd-automaton $(Q, \langle o, t \rangle)$ $sp(t(q)) = \{q_1, \ldots, q_n\}$ and let $r_i = t(q)(q_i)$, for $1 \leq i \leq n$, then the following diagram contains all the relevant information about q:

$$\begin{tikzcd} & q \arrow[r, "o(q)"] \arrow[dl, "r_1"'] \arrow[dr, "r_n"] & \\ q_1 & \cdots & q_n \end{tikzcd}$$

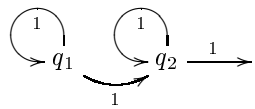
Note that the requirement of finite support implies that the automaton Q is *finitely branching*, in the sense that from q, there are only finitely many (non-zero) arrows.

The *behaviour* of a state $q \in Q$ with support $\{q_1, \ldots, q_n\}$ is a stream $S(q) \in \mathbb{R}^\omega$, defined, coinductively, by the following system of behavioural differential equations:

differential equation	initial value
$S(q)' = r_1 S(q_1) + \cdots + r_n S(q_n)$	$S(q)(0) = o(q)$

(where as before, $r_i = t(q)(q_i)$, for $1 \leq i \leq n$). The pair (Q, q) is called a *representation* of the stream $S(q)$. A stream $\sigma \in \mathbb{R}^\omega$ is called *finitely representable* if there exists a *finite* nd-automaton Q and $q \in Q$ with $\sigma = S(q)$.

Example 8.1 Consider the following two state nd-automaton:



We have the following equations for the behaviour of q_1 and q_2:

differential equation	initial value
$S(q_1)' = S(q_1) + S(q_2)$	$S(q_1)(0) = 0$
$S(q_2)' = S(q_2)$	$S(q_2)(0) = 1$

Calculating the solutions of these equations as we did in Section 5, we find:

$$S(q_1) = X(1-X)^{-2} = (0,1,2,3,\ldots), \quad S(q_2) = (1-X)^{-1} = (1,1,1,\ldots)$$

□

The following proposition gives a characterisation of the behaviour of a state of an nd-automaton in terms of its transition sequences.

Proposition 8.2 *For an nd-automaton Q, for all $q \in Q$ and $k \geq 0$,*

$$S(q)(k) = \sum \{l_0 l_1 \cdots l_{k-1} l \mid q = q_0 \xrightarrow{l_0} q_1 \xrightarrow{l_1} \cdots \xrightarrow{l_{k-1}} q_k \xrightarrow{l}\}$$

As an example, the reader may wish to check for q_1 of Example 8.1 above that this proposition implies that $S(q_1)(k) = k$, indeed.

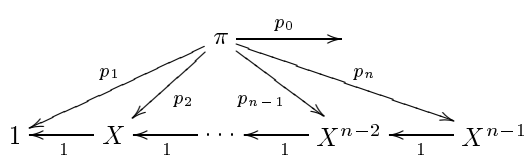
Proof: Using the differential equation for $S(q)$ and the observation that

$$\begin{aligned} S(q)(k+1) &= S(q)^{(k+1)}(0) \\ &= (S(q)')^{(k)}(0) \\ &= (r_1 S(q_1) + \cdots + r_n S(q_n))^{(k)}(0) \\ &= r_1 S(q_1)^{(k)}(0) + \cdots + r_n S(q_n)^{(k)}(0) \\ &= r_1 S(q_1)(k) + \cdots + r_n S(q_n)(k) \end{aligned}$$

the proof follows by induction on k. □

The reverse game is also interesting: given a stream σ, find it a representation; that is, construct an nd-automaton Q containing a state $q \in Q$ with $\sigma = S(q)$. The following examples give a rather general procedure for the construction of such an automaton, the essence of which is the 'splitting of derivatives'.

Example 8.3 For a typical example, consider a polynomial $\pi = p_0 + p_1 X + \cdots + p_{n-1} X^{n-1} + p_n X^n$, with $n \neq 0$. We are going to construct an nd-automaton with streams as states. The first state to be included is π itself. Computing its derivative yields $\pi' = p_1 + p_2 X + \cdots + p_{n-1} X^{n-2} + p_n X^{n-1}$. Now that we have written π' as a sum, we are going to 'split' it into its summands, each of which is included as a state of the automaton under construction. In principle, one then continues this process for each of these new states but in this particular example, we are already done: we set $Q = \{\pi, 1, X, X^2, \ldots, X^{n-1}\}$, and define outputs and transitions as specified by the following diagram:



The state π has transitions into each of its summands, all with the appropriate coefficient. The other transitions are obtained by computing derivatives again: $(X^i)' = X^{i-1}$, for $1 \leq i \leq n-1$, each of which is 'unsplittable'. (One might have included also the stream 0 as an additional state, with transitions $1 \xrightarrow{1} 0$ and $0 \xrightarrow{1} 0$, but there is no need for doing so, really.) The outputs are obtained by computing the respective initial values: all states have output 0 but for π, whose output is its initial value $\pi(0)$, and the state 1, which has output 1. (Here and below, the output of the constant 1 is *not* included in the picture.) It is now an easy exercise in coinduction to prove that the state $\pi \in Q$ represents the stream $\pi \in \mathbb{R}^\omega$: Let $R \subseteq \mathbb{R}^\omega \times \mathbb{R}^\omega$ be the smallest relation such that

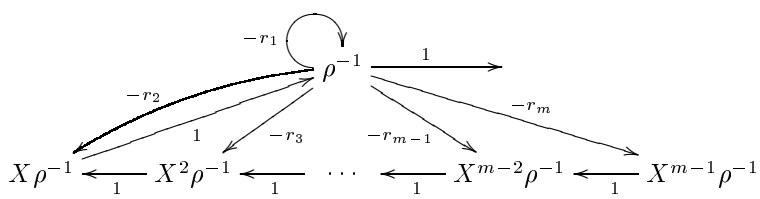
1. $\langle S(\pi), \pi \rangle \in R$ and $\langle S(X^i), X^i \rangle \in R$, for all $0 \leq i \leq n-1$;
2. if $\langle \tau, \rho \rangle \in R$ then $\langle r\tau, r\rho \rangle \in R$, for all $r \in \mathbb{R}$;
3. if $\langle \tau, \rho \rangle \in R$ and $\langle \tau', \rho' \rangle \in R$ then $\langle \tau + \tau', \rho + \rho' \rangle \in R$.

Then R is a bisimulation relation and it follows by coinduction that $S(\pi) = \pi$, indeed.

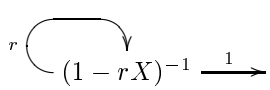
Another typical example concerns the construction of an automaton for the inverse of a polynomial. Let $\rho = r_0 + r_1 X + \cdots + r_{m-1} X^{m-1} + r_m X^m$ with $r_m \neq 0$. It will be convenient to assume that $r_0 = 1$ (but the construction below works for any $r_0 \neq 0$). In order to construct an nd-automaton for ρ^{-1}, we compute and split its derivative as follows:

$$
\begin{aligned}
&(\rho^{-1})' \\
&\quad = \quad -r_0^{-1} \rho' \times \rho^{-1} \\
&\quad = \quad -(r_1 + r_2 X + \cdots + r_{m-1} X^{m-2} + r_m X^{m-1}) \times \rho^{-1} \\
&\quad = \quad -r_1 \rho^{-1} - r_2 X \rho^{-1} - \cdots - r_{m-1} X^{m-2} \rho^{-1} - r_m X^{m-1} \rho^{-1}
\end{aligned}
$$

From this, the following picture can be deduced:



(Again, $S(\rho^{-1}) = \rho^{-1}$ follows easily by coinduction.) Applying this to the example at the beginning of the present section yields



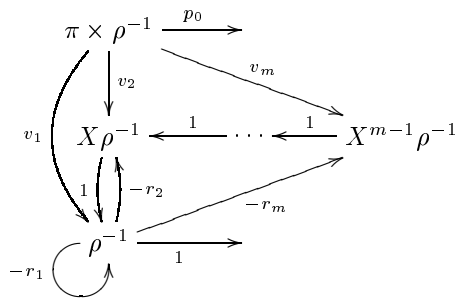
Thus we have obtained a *finite* nondeterministic representation for the stream $(1 - rX)^{-1}$ which deterministically, as we have seen above, could only be represented by an *infinite* automaton. □

The two examples above yielded finite representations. This is not a coincidence.

Theorem 8.4 *A stream σ is rational iff it has a finite representation: there exist an nd-automaton Q and a state $q \in Q$ with $\sigma = S(q)$.*

Proof: Consider polynomials $\pi = p_0 + p_1 X + \cdots + p_n X^n$ and $\rho = r_0 + r_1 X + \cdots + r_m X^m$ with $0 < n < m$ and $r_0 = 1$, and let $\sigma = \pi \times \rho^{-1}$. (The case that $m \leq n$ can be dealt with

similarly.) A computation similar to the ones in Example 8.3 gives rise to the following picture of an nd-automaton for σ, which is presented here without further ado:



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

This proves that every rational stream has a finite representation. For the converse, Example 8.1 can simply be generalised to arbitrary finite nd-automata Q. If $Q = \{q_1, \ldots, q_n\}$ then the streams $S(q_i)$ are defined by a system of n differential equations, containing $2n$ unknowns: $S(q_i)$ and $S(q_i)'$. Applying the fundamental theorem of stream calculus (Theorem 5.1) to each of $S(q_i)$ gives n more equations: $S(q_i) = S(q_i)(0) + XS(q_i)'$. By solving the $2n$ equations that we have now obtained, we find regular expressions for each of ($S(q_i)'$ and) $S(q_i)$, which shows that these streams are rational. □

One can show that both the nd-automata in Example 8.3 are minimal, and that the nd-automaton for $\sigma = \pi \times \rho^{-1}$ in Theorem 8.4 is minimal if π and ρ have no common factor. We leave the subject of minimality and minimisation to be discussed at another occasion.

One of the advantages of nondeterministic automata is that they form finite representations for rational streams whereas, generally, these cannot be finitely represented by deterministic stream automata (as was illustrated at the end of Example 8.3). Nevertheless, it may be worthwhile to study also infinite nd-automata representing non-rational streams. We hope the next example convinces the reader hereof.

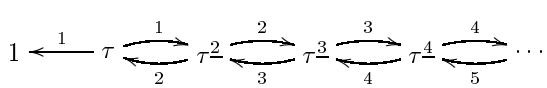
Example 8.5 Recall from Example 6.6 the behavioural differential equation for the Taylor series of the function $tan(x)$:

behavioural differential equation	initial value
$\tau' = 1 + (\tau \otimes \tau)$	$\tau(0) = 0$

This series is notoriously difficult in that no closed formula for its elements, the so-called *tangent* numbers, exists. Here an nd-automaton for τ is constructed, from which such a formula can be derived. Following again the 'splitting of derivatives' procedure for the construction of an nd-automaton Q for τ, the first states to be included are τ, 1, and $\tau \otimes \tau = \tau^{\underline{2}}$. Computing the derivative of the latter, we find, using the differential equation for τ again,

$$\begin{aligned} (\tau^{\underline{2}})' &= 2\tau' \otimes \tau \\ &= 2(1 + \tau^{\underline{2}}) \otimes \tau \\ &= 2\tau + 2\tau^{\underline{3}} \end{aligned}$$

Continuing this way, one obtains $Q = \{1, \tau, \tau^{\underline{2}}, \tau^{\underline{3}}, \tau^{\underline{4}}, \ldots\}$ with transitions as in the following diagram:



Thus we have obtained an, albeit infinite but extremely regular and simple nd-automaton, in which the state τ represents the Taylor series of $tan(x)$. Applying Proposition 8.2 now yields a formula for the n-th tangent number. □

There is also the following algebraic characterisation of the behaviour of a *finite* nd-automaton $Q = (Q, \langle o, t\rangle)$. It will play no role in the remainder of this paper. Let $Q = \{q_1, \ldots, q_n\}$ and let μ be the $n \times n$ matrix with entries $\mu_{ij} = t(q_i)(q_j)$. Furthermore write $o : Q \to \mathbb{R}$ as a column vector $o^t = (o(q_1), \ldots, o(q_n))^t$.

Proposition 8.6 *For any sequence of real numbers $a = (a_1, \ldots, a_n)$ (viewed as a row vector), and for all $k \geq 0$,*

$$a_1 S(q_1)(k) + \cdots + a_n S(q_n)(k) = a \times \mu^k \times o^t$$

where on the right, matrix multiplication is used.

Proof: The proof is an immediate consequence of Proposition 8.2. □

9 Formal power series

Time has come to generalise our theory of streams to a much wider setting. Recall that streams are formal power series in one single variable X with coefficients in $\mathbb{R}$, as is stated by Theorem 5.2. Two aspects will be generalised next: we shall deal with formal power series in *many* variables, and with coefficients in an *arbitrary semiring*. As we shall see, all results on streams that depend only on the semiring structure of the real numbers or, in other words, those parts of the theory that did not use the operations *minus* and *inverse*, will turn out to hold for this much larger family of formal power series, too, with definitions and proofs that are almost literally the same. The outcome of all this will be a calculus of formal power series, in which we shall be reasoning about streams, formal languages, so-called (max,+)-automata and many other structures, all at the same time. In addition to the results on streams of the previous sections, the theory of power series will be further illustrated in Section 10 on formal languages and in Section 11 on max-plus automata.

A *semiring* is something like a ring without subtraction. More precisely, it consists of a set k together with two binary operations $+$ and $\times$ (sum and product) and two constants 0 and 1:

$$k = \langle k, +, \times, 0, 1\rangle$$

such that, for all x, y, and z in k,

1. $(k, +, 0)$ is a commutative monoid with 0 as identity;
2. $(k, \times, 1)$ is a monoid with 1 as identity;
3. $x \times (y + z) = (x \times y) + (x \times z)$ and $(y + z) \times x = (y \times x) + (z \times x)$;
4. $0 \times x = x \times 0 = 0$

(Usually we shall write xy for $x \times y$.) Here are the semirings that interest us most:

k	k	$+$	$\times$	0	1
$\mathbb{R}$	$\mathbb{R}$	$+$	$\times$	0	1
$\mathbb{B}$	$\{0, 1\}$	$\vee$	$\wedge$	0	1
$\mathbb{R}_{\max}$	$[-\infty, \infty)$	max	$+$	$-\infty$	0

Sofar we have only been dealing with the first of these, the real numbers. The second semiring is that of the Booleans, and the third consists again of the real numbers, now extended with minus infinity and with different operators and constants. Note that both $\mathbb{B}$ and $\mathbb{R}_{\max}$ are *idempotent* semirings in that they satisfy $x + x = x$.

Next let A be an arbitrary set, the elements of which will be called *letters* or *variables* or *input symbols*, depending on where we are in our story. The letters a, b, ... denote typical elements of A, but occasionally X and Y will be used as well. Let A^* be the set of all finite words over A,

that is, finite sequences of elements of A. Prefixing a word w in A^* with a letter a in A is denoted by aw; concatenation of words v and w is denoted by vw; and 0 denotes the empty word (empty sequence). The context will always make clear which 'zero' is meant: $0 \in A^*$ or $0 \in k$.

We now come to the main definition of the present section: the set $k\langle\langle A\rangle\rangle$ of *formal power series* with variables in A and coefficients in k is given by

$$k\langle\langle A\rangle\rangle = k^{A^*} = \{\sigma \mid \sigma : A^* \to k\}$$

As mentioned above, formal power series generalise streams, which are obtained as a special case by taking $k = \langle \mathbb{R}, +, \times, 0, 1\rangle$ and $A = \{X\}$:

$$\mathbb{R}\langle\langle\{X\}\rangle\rangle = \mathbb{R}^{\{X\}^*} \cong \mathbb{R}^\omega$$

since $\{X\}^* \cong \{0, 1, 2, \ldots\} = \omega$. Another example that will be dealt with extensively, is obtained by taking A arbitrary and $k = \mathbb{B}$:

$$\mathbb{B}\langle\langle A\rangle\rangle = \{0,1\}^{A^*} \cong \{L \mid L \subseteq A^*\}$$

yielding the set $\mathcal{L} = \{L \mid L \subseteq A^*\}$ of formal languages over A.

Next we generalise the definition of deterministic stream automata, used for the representation of streams, to formal power series. A *deterministic automaton* with inputs in A and outputs in k or, simply, *automaton* is a pair $Q = (Q, \langle o, t\rangle)$ consisting of a set Q of *states*, and a pair of functions: an *output function* $o : Q \to k$, and a *transition function* $t : Q \to Q^A$. (Here Q^A is the set of all functions from A to Q.) The output function o assigns to a state $q \in Q$ an output value $o(q)$ in k. The transition function t assigns to a state $q \in Q$ a function $t(q) : A \to Q$, which specifies the state $t(q)(a)$ that is reached after the input symbol $a \in A$ has been consumed. We shall sometimes write $q \xrightarrow{x}$ for $o(q) = x$ and $q \xrightarrow{a} q'$ for $t(q)(a) = q'$. (This notation is not to be confused with the transitions of type $q \xrightarrow{r} q'$, with $r \in \mathbb{R}$, which we encountered in the nd-automata for streams.)

Taking in this definition $A = \{X\}$ and $k = \mathbb{R}$ yields our earlier definition of stream automaton indeed (modulo the isomorphism $Q^{\{X\}} \cong Q$). In the case of our second canonical choice: arbitrary A and $k = \mathbb{B}$, one obtains the classical definition of deterministic automaton of formal language theory. The latter is usually defined as a triple $(Q, F \subseteq Q, \delta : Q \times A \to Q)$ consisting of a set Q of states, a subset $F \subseteq Q$ of so-called final or accepting states, and a transition function δ (often an initial state is included as well). The equivalence with our definition of deterministic automaton above is an immediate consequence of the following two bijective correspondences:

$$\{F \mid F \subseteq Q\} \cong \{o \mid o : Q \to \{0,1\}\}, \qquad \{\delta \mid \delta : Q \times A \to Q\} \cong \{t \mid t : Q \to Q^A\}$$

Similar to the way in which the set $\mathbb{R}^\omega$ of streams was turned into a stream automaton, using the operation of stream derivative for the definition of the transition function, the set $k\langle\langle A\rangle\rangle$ of formal power series can be provided with an automaton structure too. However, we shall need more derivative operations, one for each variable in A, to be precise: For a variable or input symbol a in A, the *input derivative* or *a-derivative* σ_a of a series $\sigma \in k\langle\langle A\rangle\rangle$ is defined by

$$\sigma_a : A^* \to k, \;\; w \mapsto \sigma(aw)$$

The *initial value* (or output) of a series σ is defined by $\sigma(0)$. Now $k\langle\langle A\rangle\rangle$ can be turned into an automaton $(k\langle\langle A\rangle\rangle, \langle o, t\rangle)$ by defining, for $\sigma \in k\langle\langle A\rangle\rangle$ and $a \in A$,

$$o(\sigma) = \sigma(0), \qquad t(\sigma)(a) = \sigma_a$$

(The reader may wish to check that these definitions specialise to the ones for streams.) The following notation will sometimes be used: $\sigma_0 = \sigma$ (here 0 is the empty word) and $\sigma_{aw} = (\sigma_w)_a$, for any word $w \in A^*$. Note that with this notation, we have $\sigma_w(0) = \sigma(w)$.

All the results on the automaton of streams apply to the automaton of power series as well and are summarised below. We need to introduce a generalised version of the notion of bisimulation

first. A *bisimulation* between deterministic automata $(Q, \langle o_Q, t_Q \rangle)$ and $(Q', \langle o_{Q'}, t_{Q'} \rangle)$ is a relation $R \subseteq Q \times Q'$ such that for all q in Q and q' in Q':

$$\text{if } \; q \, R \, q' \; \text{ then } \; \begin{cases} o_Q(q) = o_{Q'}(q') & \text{and} \\ \forall a \in A, \; t_Q(q)(a) \; R \; t_{Q'}(q')(a) \end{cases}$$

As before, $q \sim q'$ denotes bisimilarity, and a *homomorphism* is defined as a functional bisimulation. The proof of the following theorem is an easy variation on the earlier results on streams, and is therefore omitted.

Theorem 9.1

1. Coinduction*: For all series σ and τ in $k\langle\langle A \rangle\rangle$, if $\sigma \sim \tau$ then $\sigma = \tau$.*
2. Finality*: For any automaton Q there exists a unique homomorphism $l : Q \to k\langle\langle A \rangle\rangle$. It satisfies, for all q and q' in Q, $q \sim q'$ iff $l(q) = l(q')$.*

□

For a state $q \in Q$, the formal power series $l(q)$ is again called the *behaviour* of q. Viewing an automaton Q as a machine that consumes sequences of input symbols (words in A^*), the power series $l(q)$ can be considered as a large table which gives us for any $w \in A^*$ the output value of the state that is reached after w has been consumed. More formally,

$$\text{if} \quad q \xrightarrow{a_0} q_1 \xrightarrow{a_1} \cdots \xrightarrow{a_n} q_{n+1} \xrightarrow{x} \quad \text{then} \quad l(q)(a_0 \cdots a_n) = x$$

Both definitions by coinduction, in terms of behavioural differential equations, and proofs by coinduction work the same way for series as they did for streams. We briefly summarise the main results (again without proof, as these are easy variations on the ones for streams). There are unique operators on series satisfying the following behavioural differential equations: For all $x \in k$, $a, b \in A$, $\sigma, \tau \in k\langle\langle A \rangle\rangle$,

differential equation	initial value	name
$x_a = 0$	$x(0) = x$	constant
$b_b = 1$, $b_a = 0$ $(b \neq a)$	$\mathrm{b}(0) = 0$	variable
$(\sigma + \tau)_a = \sigma_a + \tau_a$	$(\sigma + \tau)(0) = \sigma(0) + \tau(0)$	sum
$(\sigma \times \tau)_a = (\sigma_a \times \tau) + (\sigma(0) \times \tau_a)$	$(\sigma \times \tau)(0) = \sigma(0) \times \tau(0)$	product
$(\sigma^*)_a = \sigma_a \times \sigma^*$	$(\sigma^*)(0) = 1$	star
$(\sigma^{-1})_a = -(\sigma(0)^{-1} \times \sigma_a) \times \sigma^{-1}$	$(\sigma^{-1})(0) = \sigma(0)^{-1}$	inverse
$(\sigma \otimes \tau)_a = (\sigma_a \otimes \tau) + (\sigma \otimes \tau_a)$	$(\sigma \otimes \tau)(0) = \sigma(0) \times \tau(0)$	shuffle product
$(\sigma^{\underline{*}})_a = \sigma_a \otimes \sigma^{\underline{*}} \otimes \sigma^{\underline{*}}$	$(\sigma^{\underline{*}})(0) = 1$	shuffle star
$(\sigma^{\underline{-1}})_a = -\sigma_a \otimes (\sigma^{\underline{-1}} \otimes \sigma^{\underline{-1}})$	$\sigma^{\underline{-1}}(0) = \sigma(0)^{-1}$	shuffle inverse
$(\sum_{i \in I} \sigma_i)_a = \sum_{i \in I} (\sigma_i)_a$	$(\sum_{i \in I} \sigma_i)(0) = \sum_{i \in I} \sigma_i(0)$	generalised sum

In the above, the following is to be noted:

- We identify constant x and variable b with the power series x and b defined by the first two equations. In this way, the semiring k and the set of variables A can be considered as subsets of $k\langle\langle A \rangle\rangle$. Identifying moreover a word $w = a_0 \cdots a_n$ with the product of (the power series corresponding to) its letters:

  $$w = a_0 \times \cdots \times a_n$$

  we also have an inclusion of A^* in $k\langle\langle A \rangle\rangle$.

- Inverse and shuffle inverse are defined only when k is a ring (we need subtraction) and $\sigma(0)$ is invertible in k. We do *not* have these operations on the set $\mathcal{L}$ of languages (where $k = \mathbb{B}$).

- Generalised sum is defined only for families of series that are locally finite, which in the present context means: for every word $w \in A^*$, the set $\{i \in I \mid \sigma_i(w) \neq 0\}$ is finite.

The above operators satisfy again the usual properties. Keeping in mind the restrictions just mentioned, *all* of the stream laws given in Theorem 4.1 are valid for series as well. The bisimulation relations to be used for the proofs of these laws can be taken identical to those in the proofs for streams, replacing streams by series everywhere. Although the ordinary product of streams is commutative (as was stated in Section 7), this is no longer true for power series since, for instance,

$$a \times b \neq b \times a$$

for $a, b \in A$ with $a \neq b$. (Shuffle product still is commutative and also satisfies the other identities formulated in Theorem 6.2.) There is also the following generalisation of Theorem 5.1, which is proved in the same way.

Theorem 9.2 [Fundamental Theorem of series calculus] *For all formal power series $\sigma \in k\langle\langle A\rangle\rangle$:*

$$\sigma = \sigma(0) + \sum_{a \in A} a \times \sigma_a$$

Proof: The theorem follows by coinduction from the fact that

$$\{\langle \sigma, \sigma(0) + \sum_{a \in A} a \times \sigma_a \rangle \mid \sigma \in k\langle\langle A\rangle\rangle\} \cup \ \{\langle \sigma, \sigma \rangle \mid \sigma \in k\langle\langle A\rangle\rangle\}$$

is a bisimulation relation on $k\langle\langle A\rangle\rangle$, which is immediate from the equalities $(a \times \sigma)_a = \sigma$ and $(a \times \sigma)_b = 0$, for $b \neq a$. □

There is also the following expansion theorem for formal power series, generalising Theorem 5.2. For all series $\sigma \in k\langle\langle A\rangle\rangle$,

$$\sigma = \sum_{w \in A^*} \sigma(w) \times w$$

where with the last occurrence of w, we are using the shorthand $w = a_0 \times \cdots \times a_n$. If we use shuffle product instead of ordinary product, as in

$$\underline{w} = a_0 \otimes \cdots \otimes a_n$$

we can consider formal power series with *commuting* variables: Every series $\sigma \in k\langle\langle A\rangle\rangle$ induces a series $c(\sigma) \in k\langle\langle A\rangle\rangle$ defined by

$$c(\sigma) = \sum_{w \in A^*} \sigma(w) \times \underline{w}$$

The variables in this series $c(\sigma)$ can indeed be considered to be commutative since $c(\sigma)(w) = c(\sigma)(w')$, for all words $w, w' \in A^*$ with $\underline{w} = \underline{w'}$. For instance, $2aba + aab$ induces the series

$$2(a \otimes b \otimes a) + (a \otimes a \otimes b) = 3(a \otimes a \otimes b) = 3\underline{a^2} \otimes b$$

For such commutative series, a Taylor theorem can be formulated, generalising the one for streams (Theorem 6.3).

Finally, the definitions of rationality and of nd-automaton are formulated for series as well. A formal power series is *rational* if it can be constructed from finitely many constants $x \in k$ and variables $a \in A$, by means of the operators of sum, product, and star. A *k-nondeterministic automaton*, or again nd-automaton for short, is a pair $Q = (Q, \langle o, t\rangle)$ consisting of a set Q of states, and a pair of functions: an output function $o : Q \to k$; and a transition function which is now k-nondeterministic: $t : Q \to k(Q)^A$. Here $k(Q)^A$ is the set of all functions from A to $k(Q)$, which is defined by

$$k(Q) = \{\phi : Q \to k \mid sp\,(\phi) \text{ is finite } \}$$

(recall $sp(\phi) = \{q \in Q \mid \phi(q) \neq 0\}$). The transition function t assigns to a state q in Q a function $t(q) \in k(Q)^A$, which at its turn assigns to each input symbol $a \in A$ a function $t(q)(a) : Q \to k$. As before, the latter can be viewed as a kind of nondeterministic or distributed state, and specifies for any state q' in Q a multiplicity $t(q)(a)(q')$ in k. The following notation will be used:

$$q \xrightarrow{a|x} q' \iff t(q)(a)(q') = x, \quad \text{and} \quad q \xrightarrow{x} \iff o(q) = x$$

The behaviour of a state q in an nd-automaton Q is now a formal power series $S(q) \in k\langle\langle A\rangle\rangle$, again defined in terms of differential equations: if the support of $t(q)(a)$, for $a \in A$, is $\{q_1, \ldots, q_n\}$, then $S(q)$ is defined by the following system of behavioural differential equations:

differential equation	initial value
$S(q)_a = x_1 S(q_1) + \cdots + x_n S(q_n)$	$S(q)(0) = o(q)$

where $x_i = t(q)(a)(q_i)$, for $1 \leq i \leq n$. Theorem 8.4 also applies to formal power series: A series $\sigma \in k\langle\langle A\rangle\rangle$ is rational iff it has a finite representation, that is, there exist an nd-automaton Q and a state $q \in Q$ with $\sigma = S(q)$. (The proof is slightly more complicated than for streams, for which rationality is equivalent to being the quotient of polynomials, which simplifies matters.) Propositions 8.6 and 8.2 are easily adapted to series. The latter proposition now reads: For an nd-automaton Q, for all $q \in Q$ and $w = a_0 \cdots a_{n-1}$,

$$S(q)(w) = \sum \{x_0 x_1 \cdots x_{n-1} x \mid q = q_0 \xrightarrow{a_0|x_0} q_1 \xrightarrow{a_1|x_1} \cdots \xrightarrow{a_{n-1}|x_{n-1}} q_n \xrightarrow{x}\}$$

10 Languages

Let A be arbitrary and let $k = \mathbb{B}$. We have already seen that in this case,

$$\mathbb{B}\langle\langle A\rangle\rangle \cong \mathcal{L} = \{L \mid L \subseteq A^*\}$$

the set of *formal languages* over A. We shall first show how the various definitions of Section 9 look like for languages, and next discuss in detail the coinduction proof principle for rational languages.

As we have already seen, deterministic automata are the usual ones: sets of states Q with a transition function $t : Q \to Q^A$, giving for each state $q \in Q$ and input symbol $a \in A$ the next state $t(q)(a)$, and output function $o : Q \to \{0, 1\}$, telling whether q is *accepting*: $o(q) = 1$, or not: $o(q) = 0$. In pictures, we shall write $q\Downarrow$ to denote that $o(q) = 1$. The automaton structure on $\mathcal{L}$ itself looks as follows: $o(L) = L(0)$, where $L(0) = 1$ if $0 \in L$, and $L(0) = 0$ otherwise. Thus a language is accepting iff it contains the empty word 0. Transitions are given by

$$L \xrightarrow{a} L_a$$

where the a-derivative L_a of L is defined by $L_a = \{w \in A^* \mid aw \in L\}$. The finality of $\mathcal{L}$ asserts that for a deterministic automaton Q, there exists a unique homomorphism $l : Q \to \mathcal{L}$. It assigns to a state $q \in Q$ the language $l(q)$ it accepts:

$$l(q) = \{a_0 \cdots a_n \mid q \xrightarrow{a_0} q_1 \xrightarrow{a_1} \cdots \xrightarrow{a_n} q_{n+1}\Downarrow\}$$

Bisimulation relations on $\mathcal{L}$ are relations $R \subseteq \mathcal{L} \times \mathcal{L}$ such that, for all $K, L \in \mathcal{L}$, $\langle K, L\rangle \in R$ implies $o(K) = o(L)$ and $\langle K_a, L_a\rangle \in R$, for all $a \in A$. The coinductive definitions of the various operators in terms of the behavioural differential equations presented in Section 9, are easily proved (by coinduction) to be equivalent to the usual definitions:

$$\begin{aligned} 0 &= \emptyset \\ 1 &= \{0\} \quad \text{(here 0 denotes the empty word)} \\ a &= \{a\} \\ K + L &= K \cup L \end{aligned}$$

$$
\begin{aligned}
K \times L &= \{vw \mid v \in K,\ w \in L\} \\
K^* &= \sum_{n=0}^{\infty} K^n \\
K \otimes L &= \bigcup \{v \otimes w \mid v \in K,\ w \in L\}
\end{aligned}
$$

where $v \otimes w$ is defined, by induction on the length of words, as follows:

$$v \otimes w = v \mathbin{\|\!\llcorner} w + w \mathbin{\|\!\llcorner} v, \quad 0 \mathbin{\|\!\llcorner} v = \{v\}, \quad (av) \mathbin{\|\!\llcorner} w = \{au \mid u \in v \otimes w\}$$

(Note that the equality for the shuffle product at last explains the terminology: the shuffle product of two languages consists of the union of all the shuffles of their elements. For instance, $\{ab\} \otimes \{c\} = \{abc, acb, cab\}$.) The above identities are given merely to show that the coinductive definitions are equivalent to the traditional ones. However, all reasoning about languages and their operators will be in terms of their coinductive definitions, that is, differential equations.

Next we shall study in some detail the subautomaton $\langle L \rangle \subseteq \mathcal{L}$ generated by a language $L \in \mathcal{L}$. It turns out to be minimal among all automata accepting L, and it is moreover finite iff L is rational. (Note that the latter fact does not hold for power series over arbitrary semirings; for instance, in order to obtain finite representations for rational streams, one has to resort to nd-automata.) As a consequence, the coinduction proof principle will be shown to be effective for rational languages.

So consider an arbitrary language $L \in \mathcal{L}$. The subautomaton $\langle L \rangle \subseteq \mathcal{L}$ generated by L consists of the following states:

$$\langle L \rangle = \{L_w \in \mathcal{L} \mid w \in A^*\}$$

where, recall, $L_0 = L$ and $L_{wa} = (L_w)_a$. Since the inclusion function $i : \langle L \rangle \subseteq \mathcal{L}$ is a homomorphism, the language accepted by the state L of the automaton $\langle L \rangle$ is $i(L) = L$. Next consider any deterministic automaton Q and state $q \in Q$ with $l(q) = L$, where $l : Q \to \mathcal{L}$ is the (by finality) unique homomorphism from Q into $\mathcal{L}$. Assume that all states in Q are reachable from q: $Q = \langle q \rangle$ (where $\langle q \rangle$ is the subautomaton generated by q); otherwise switch from Q to $\langle q \rangle$, and call it Q again. Because l is a homomorphism, one easily shows that $l(\langle q \rangle) = \langle l(q) \rangle$. As a consequence, $l(Q) = l(\langle q \rangle) = \langle l(q) \rangle = \langle L \rangle$, implying that the size of Q is greater than or equal to the size of $\langle L \rangle$. Since Q was arbitrary, this shows that $\langle L \rangle$ is a minimal automaton for L.

For rational languages, which are built from finitely many constants (0 and 1) and variables ($a \in A$) by means of the operations of sum, product, and star, we have the following.

Theorem 10.1 *A language $L \in \mathcal{L}$ is rational iff $\langle L \rangle$ is finite.*

Proof: For the implication from left to right, note that $\langle 0 \rangle = \{0\}$, $\langle 1 \rangle = \{1, 0\}$, $\langle a \rangle = \{a, 1, 0\}$ and, for all $K, L \in \mathcal{L}$, $w \in A^*$,

$$(K + L)_w = K_w + L_w \tag{32}$$

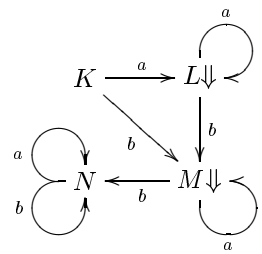
$$(K \times L)_w = K_w \times L + \sum_{uv=w} K_u(0) \times L_v \tag{33}$$

$$(K^*)_w = K_w \times K^* + \sum_{u_1 \cdots u_n v = w} K_{u_1}(0) \times \cdots \times K_{u_n}(0) \times K_v \times K^* \tag{34}$$

The latter equations can be proved by induction on the length of w, using the defining differential equations for the operators. Defining $w(L)$, for any language L, as the number of distinct w-derivatives of L, one has $w(0) = 1$, $w(1) = 2$, $w(a) = 3$, $w(K + L) \leq w(K) \times w(L)$, $w(K \times L) \leq w(K) \times 2^{w(L)}$, $w(K^*) \leq 2^{w(K)}$, using the fact that $K(0) \in \{0, 1\}$, for any language K. It follows by induction on the construction of a rational language L that $\langle L \rangle$ is finite.

For the converse, we shall treat an example, leaving the formulation of a proof for the general case to the diligent reader. Consider the following subautomaton $\langle K \rangle$ of $\mathcal{L}$:

$\langle K \rangle =$



We are going to use Theorem 4.1(16) and Theorem 9.2: for all $K, L, M \in \mathcal{L}$ with $L(0) = 0$,

$$K = (L \times K) + M \quad \Rightarrow \quad K = L^* M \tag{35}$$

and, for all $K \in \mathcal{L}$,

$$K \quad = \quad K(0) + \sum_{a \in A} a \times K_a \tag{36}$$

Applying (36) four times, we obtain

$$\begin{aligned} K &= a \times L + b \times M \\ L &= a \times L + b \times M + 1 \\ M &= a \times M + b \times N + 1 \\ N &= a \times N + b \times N \end{aligned}$$

Because $N = (a+b) \times N + 0$, (35) implies $N = (a+b)^* \times 0 = 0$. Thus $M = a \times M + 1$, which, again by (35), gives $M = a^*$. Similarly, one finds $L = a^* \times (b \times a^* + 1)$ and $K = a \times a^* \times (b \times a^* + 1) + (b \times a^*)$, which proves that K is rational, indeed. □

Remark 10.2 Formulas (32), (33), and (34) above are special instances of the following more general observations, due to Conway [Con71], on which the proof could also be based. Let for $K, L \in \mathcal{L}$ the K-derivative of L be defined by

$$L_K = \sum_{w \in K} L_w$$

(note that infinite sums of languages always exist). Then for all $K, L, M \in \mathcal{L}$,

$$\begin{aligned} (K + L)_M &= K_M + L_M \\ (K \times L)_M &= K_M \times L + L_{M_K} \\ (K^*)_M &= M(0) + K_{M_{K^*}} K^* \end{aligned}$$

□

Remark 10.3 Theorem 10.1 can also be used to prove that a language L is *not* rational, by showing that $\langle L \rangle$ is infinite. (This method can easily be seen to be equivalent to the traditional method for demonstrating that a language L is not rational by showing that the equivalence relation $R_L \subseteq A^* \times A^*$, defined for $v, w \in A^*$ by

$$v \, R_L \, w \quad \text{iff} \quad \forall u \in A^*, \; vu \in L \iff wu \in L.$$

is of infinite index, that is, has infinitely many equivalence classes.) Here are three classical examples, in which the following shorthand will be used. For a language K and $k \geq 0$, let the

language K_k be the resulting state after k a-steps: $K_k = K_{a^k}$. Let $L = \{a^n b^n \mid n \geq 0\}$, where as usual $a^0 = 1$ and $a^{n+1} = aa^n$. Clearly, $L_k = \{a^{n-k} b^n \mid n \geq k\}$ and thus L_k and $L_{k'}$ are different whenever k and k' are. This shows that $\langle L \rangle$ is infinite, hence L is not rational. For a second example, consider $M = \{w \in A^* \mid \sharp_a(w) = \sharp_b(w)\}$ consisting of all words with an equal number of a's and b's. All languages M_k are different because for any n and k, the word b^n is in M_k iff $k = n$. Thus $\langle M \rangle$ is infinite and M is not rational. Finally, let $N = \{a^{n^2} \mid n \geq 0\}$. Note that for any n the length of the shortest word in N_{n^2+1} is $|a^{(n+1)^2 - n^2 - 1}| = |a^{2n}| = 2n$. Therefore N_{n^2} and N_{m^2} are different whenever n and m are. Thus $\langle N \rangle$ is infinite and N is not rational. $\square$

Here is an example of the symbolic computation of the automaton generated by a rational language. (It should be clear how to deduce an algorithm from this example that works for any rational language; a proof of the correctness and termination of such an algorithm is contained in Theorem 10.1.) Let $K = (b^*a)^*ab^*$. (As usual, MN is written for $M \times N$, for languages $M, N \in \mathcal{L}$.) We first construct $\langle K \rangle$ and then $\langle K^* \rangle$. Following the differential equations for the operators, we compute:

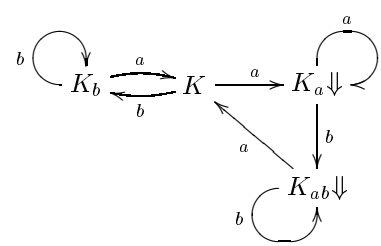
$$\begin{aligned}
K_a &= ((b^*a)^*ab^*)_a \\
&= ((b^*a)^*)_a ab^* + (b^*a)^*(0)(ab^*)_a \\
&= (b^*a)_a (b^*a)^*ab^* + 1(a_a b^* + a(0)(b^*)_a) \\
&= ((b^*)_a a + (b^*)(0)a_a)(b^*a)^*ab^* + 1(1b^* + 0(b^*)_a) \\
&= (b_a(b^*)a + 11)(b^*a)^*ab^* + 1(b^* + 0) \\
&= (0(b^*)a + 1)(b^*a)^*ab^* + 1b^* \\
&= (0 + 1)(b^*a)^*ab^* + b^* \\
&= (b^*a)^*ab^* + b^* \\
&= K + b^*
\end{aligned}$$

(Note that in the above calculations, the following identities have been used: $0L = 0$, $1L = L$, $0 + L = L$, $L + 0 = L$.) Similarly one computes $K_b = (b^*a)K$. Continuing with the computation of the derivatives of the new states K_a and K_b, one finds

$$K_{aa} = K_a, \quad K_{ab} = K_b + b^*, \quad K_{ba} = K, \quad K_{bb} = K_b$$

yielding only one new state, $K_{ab} = K_b + b^*$. Computing its derivatives gives $K_{aba} = K$ and $K_{abb} = K_{ab}$. No new states have been found and so the computation of the states in $\langle K \rangle$ is complete. Computing their initial values, we find $K_a(0) = 1 = K_{ab}(0)$, and 0 for all other states. The following picture of $\langle K \rangle$ has been obtained:

$\langle K \rangle =$



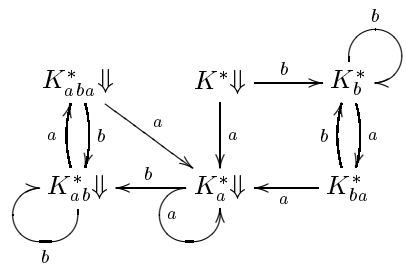
We continue with the construction of $\langle K^* \rangle$. It follows from formula (34) that each of the states in $\langle K^* \rangle$ is characterised by a finite subset of states in $\langle K \rangle$. Here are a few example calculations, in which we shall be writing K^*_w for $(K^*)_w$:

$$\begin{aligned}
K^*_a &= K_a \times K^* \\
K^*_{aa} &= (K_a)_a \times K^* + K_a(0) \times K^*_a \\
&= K_a \times K^* + 1 \times K_a \times K^* \\
&= K^*_a \\
K^*_{ab} &= (K_a)_b \times K^* + K_a(0) \times K^*_b
\end{aligned}$$

$$
\begin{aligned}
&= K_{ab} \times K^* + K_b \times K^* \\
&= (K_{ab} + K_b) \times K^* \\
K^*_{aba} &= (K_{ab} + K_b)_a \times K^* + (K_{ab} + K_b)(0) \times K^*_a \\
&= (K + K) \times K^* + K_a \times K^* \\
&= (K + K_a) \times K^* \\
K^*_b &= K_b \times K^* \\
K^*_{ba} &= K \times K^*
\end{aligned}
$$

Computing further derivatives and initial values, the following automaton for K^* is obtained:

$Q \;=$

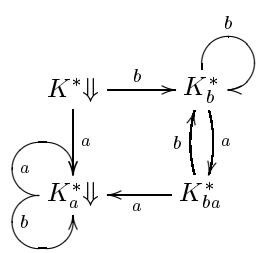


Is it the minimal automaton for K^*, that is, $Q = \langle K^* \rangle$? That depends on how we read the information provided by the picture. Consider the following three states in Q: K^*_a, K^*_{ab}, and K^*_{aba}. Reading these names purely symbolically, they are different, indeed. However, read as languages, they are identical:

$$K^*_a = K^*_{ab} = K^*_{aba}$$

This follows easily by coinduction from the fact that they are bisimilar as states in Q. In order to obtain the minimal automaton $\langle K^* \rangle$, these three states have to be identified, yielding:

$\langle K^* \rangle \;=$



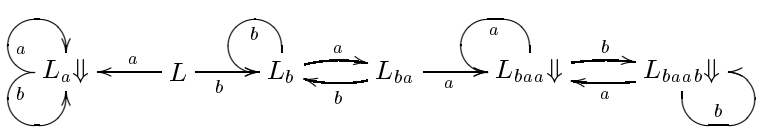
Thus some additional reasoning was needed in order to realise that our picture of Q in fact consists of 4 rather than 6 different states. More generally speaking, one has to identify all bisimilar states in order to obtain a minimal automaton. There exist various algorithms for this type of minimisation, but we shall not pursue the matter any further here. We simply do not need minimal automata for the algorithmic account of proofs by coinduction, which is presented next.

We illustrate the general idea by proving the following equality by coinduction:

$$((b^*a)^*ab^*)^* = 1 + a(a+b)^* + b(a+b)^*aa(a+b)^*$$

The language on the left is our friend K^* and let L denote the language on the right. We have to construct a bisimulation relation $R \subseteq \mathcal{L} \times \mathcal{L}$ with $\langle K^*, L \rangle \in R$. The first pair to be included in R is $\langle K^*, L \rangle$, and further pairs are determined by the pairwise derivatives $\langle K^*_w, L_w \rangle$. The derivatives for K^* were already analysed above during the construction of $\langle K^* \rangle$. A similar computation yields the following picture of $\langle L \rangle$, which contains all possible derivatives of L:

$\langle L \rangle =$



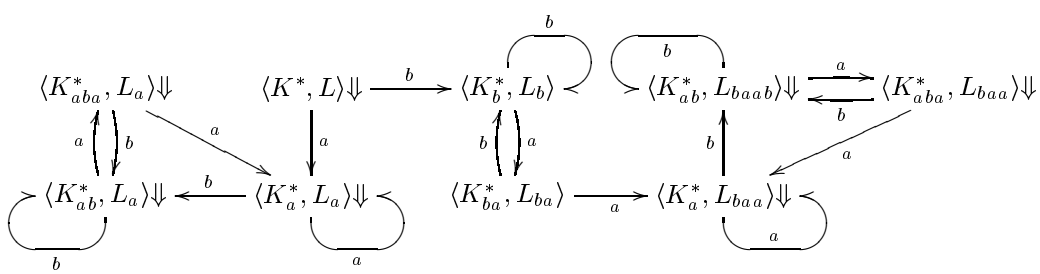
(Note that, as with $\langle K^* \rangle$, not all possible identifications have been made; the languages L_a, L_{baa}, and L_{baab} are bisimilar and hence equal.) Reading off the derivatives from the pictures of $\langle K^* \rangle$ and $\langle L \rangle$, the definition of R is now immediate:

$$\begin{aligned} R \;\; = \{ \;\; & \langle K^*, L \rangle,\ \langle K^*_a, L_a \rangle,\ \langle K^*_{ab}, L_a \rangle,\ \langle K^*_{aba}, L_a \rangle,\ \langle K^*_b, L_b \rangle, \\ & \langle K^*_{ba}, L_{ba} \rangle,\ \langle K^*_a, L_{baa} \rangle,\ \langle K^*_{ab}, L_{baab} \rangle,\ \langle K^*_{aba}, L_{baa} \rangle \;\; \} \end{aligned}$$

The relation R is a bisimulation: it is closed under taking derivatives, and $M(0) = N(0)$ for all $\langle M, N \rangle \in R$. The equality $K^* = L$ now follows by coinduction.

The above procedure works for arbitrary pairs $\langle M, N \rangle$ of rational languages: either one finds a bisimulation indeed, from which the equality $M = N$ can be concluded, or the attempt to construct a bisimulation relation fails, because one comes across a pair $\langle M_w, N_w \rangle$, for some $w \in A^*$, with $M_w(0) \neq N_w(0)$. In that case, the conclusion is $M \neq N$, and w is a witness to this fact: $w \notin M \cap N$. In view of the latter possibility, it is in general wise to compute the derivatives of M and N together and not separately. This will not make any difference if the languages in the end turn out to be bisimilar, as in our example of K^* and L above. But in case they are not, the computation of derivatives may be stopped as soon as a pair of derivatives with different initial values is found.

The classical way to prove the equality of two (regular expressions denoting) rational languages is to construct automata for each of them, minimising these automata, and then to see whether the resulting automata are isomorphic or not. This procedure can be related to the coinductive proof method explained above, by the observation that bisimulation relations carry themselves an automaton structure. (Cf. Section 14, where bisimulations are in fact *defined* as coalgebras, that is, generalised automata satisfying certain properties.) The outputs and transitions are determined component-wise, as is illustrated by the following picture, which gives the automaton for the relation R that was used in the proof of $K^* = L$ above:



The difference with the classical approach is twofold. Firstly, only one automaton is constructed for both languages at the same time. This has the advantage, as explained above, that in case the languages are different the whole construction can be aborted as soon as a witness is encountered. Secondly, the automaton need not be minimised. The automaton representing R above consists of 9 states, whereas its minimisation would only have 4 states. The conclusion that $K^* = L$ could nevertheless be based on the mere fact that R is a bisimulation relation or, equivalently, that R carries a (not necessarily minimal) automaton structure. No claims are made here as to whether this leads to more efficient algorithms for deciding language equality. As a proof method, coinduction seems to offer at least conceptually an interesting alternative.

11 An example in the max-plus semiring

Here we consider formal power series over an arbitrary alphabet A with coefficients in the so-called *max-plus* semiring:

$$\mathbb{R}_{\max} = \langle [-\infty, \infty), \max, +, -\infty, 0 \rangle$$

consisting of the reals extended with minus infinity, with the operations of max and $+$ for sum and product, respectively, with $-\infty$ and 0 as neutral elements. It is shown how the timed behaviour of

task-resource systems can be specified coinductively, that is, by means of a behavioural differential equation, and implemented by a finite nd-automaton, derived from this differential equation. The relevance of this example is not so much the obtained automaton (which is not new—see, e.g., [GM98]), but rather the way in which it illustrates once again our methodology of *deriving* finite representations from behavioural specifications.

A *(timed) task-resource system* is a four tuple (A, R, r, h) consisting of a finite set A of *tasks*, a finite set R of *resources*, a function $r : A \to \mathcal{P}(R)$ assigning to each task the finite non-empty set of resources it uses, and a function $h : A \to \mathbb{R}^+$ specifying for each task its execution time. A word $w = a_1 \cdots a_n \in A^*$ is interpreted as a *schedule*, that is, a sequence of tasks. The question to be addressed is how long it takes to perform all the tasks in a given schedule, while adhering to the following rules:

- all resources are available at time 0;
- the execution of a task a_i in w begins as soon as all resources $r(a_i)$ it requires, possibly used by earlier tasks, are available;
- a task a_i uses the resources in $r(a_i)$ for $h(a_i)$ time units, during which these resources are *not* available for other tasks.

To mention two extreme cases, the execution time of a word $a_1 \cdots a_n$ will be equal to the *maximum* of the durations $h(a_i)$ in case no tasks have any resources in common. If, in contrast, any two subsequent tasks do have a common resource, the execution time will be the *sum* of the durations $h(a_i)$.

The latter examples motivate the choice to work with the max-plus semiring. We take the finite set A of tasks as our input alphabet. As a solution for the task-resource scheduling problem, a formal power series $y : A^* \to \mathbb{R}_{\max}$ in $\mathbb{R}_{\max}\langle\langle A\rangle\rangle$ will be defined that assigns to each schedule w its execution time $y(w)$ according to the above informal specification. In order to give a formal definition of y, first a power series $\tilde{a}$ is associated with every task $a \in A$, which for a schedule $w \in A^*$ will give the execution time $\tilde{a}(w)$ of w counted *from the first time that the task a in w is executed*. Then y can be expressed as the maximum of all $\tilde{a}$. The following system of behavioural differential equations defines all these series formally:

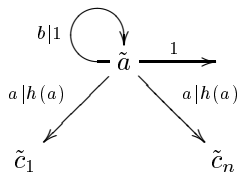
differential equation	initial value
$(\tilde{a})_b = \tilde{a}$ (for $b \neq a$) $(\tilde{a})_a = \max\{h(a) + \tilde{c} \mid r(a) \cap r(c) \neq \emptyset\}$ $y = \max\{\tilde{a} \mid a \in A\}$	$(\tilde{a})(0) = 0$

Here we silently adopt the convention that the maximum of an empty set is (the constant series) $-\infty$, which is the 0 of the semiring $\mathbb{R}_{\max}$. The existence of unique solutions for these equations is proved by a straightforward variation on the proof of Theorem 3.2.

Thus the scheduling problem has been formalised in terms of a system of behavioural differential equations, allowing for the use of coinductive techniques that we have seen many examples of in the previous sections. Here we shall briefly illustrate one aspect, namely the derivation of a finite representation for the series $\tilde{a}$ from its defining differential equation. The procedure is the same as in Section 8, where nd-automata are obtained by 'splitting' a derivative into its summands, according to its differential equation. Note that in the present setting, the operation max plays the role of sum and the operation + plays the role of multiplication. Denoting for a moment max, +, and 0 by +, $\times$, and 1, respectively, the differential equation for $\tilde{a}$ reads:

differential equation	initial value
$(\tilde{a})_b = \tilde{a}$ (for $b \neq a$) $(\tilde{a})_a = (h(a) \times \tilde{c}_1) + \cdots + (h(a) \times \tilde{c}_n)$	$(\tilde{a})(0) = 1$

with $\{c_1, \ldots, c_n\} = \{c \in A \mid r(a) \cap r(c) \neq \emptyset\}$. This gives rise to the following nd-automaton:



(where we have only indicated the transitions from $\tilde{a}$ and the b-transition from $\tilde{a}$ to itself occurs for all $b \in A$ with $b \neq a$). This type of automaton is sometimes called a *max-plus* automaton. By definition, the state $\tilde{a}$ in this automaton represents the series $\tilde{a}$, indeed: $S(\tilde{a}) = \tilde{a}$, which can be easily proved by—you guessed right—coinduction!!

12 Related work and discussion

The present paper subsumes and extends two of our earlier papers: The idea of coinductive definitions in terms of input derivatives stems from [Rut98a]; in [Rut99a], this was generalised to formal power series, using a notion of input derivative that generalises Brzozowski's original definition for regular expressions [Brz64, Con71]. Compared to our earlier papers, most of the material on stream calculus and nondeterministic stream automata is new. As explained in some detail in the second appendix, Section 14, the approach has been in essence coalgebraic. Notably the elementary results of Section 2 are instances of basic facts from universal algebra [Rut96, JR97]. Coalgebras and final coalgebras have been around in the literature already for quite some time now. But it was not until the formulation of a general notion of coalgebraic bisimulation, by Aczel and Mendler in [AM89], generalising Park's [Par81] and Milner's [Mil80] definition of strong bisimulation for concurrent processes, that coalgebra could be really 'put to work'. Notably, one needs the notion of coalgebraic bisimulation to formulate a general principle of coinduction, and it is coinduction which constitutes—all of this according to our taste, of course—the heart of the coalgebraic matter.

The important example of Taylor series of analytic functions, Example 2.2, is due to Pavlović and Escardó [PE98]. Their coinductive perspective on (certain aspects of) classical analysis has added to our motivation to take behavioural differential equations as a coinductive definition format, even more seriously. Also, the operator Δ in Section 6 occurs in [PE98] under the name g. Theorem 6.5 extends a similar result from that paper by including also the case of the inverse operator. Some of the coinductive definitions of streams can also be found in [McI99]. For a number of examples, it is shown there how this type of definition can be implemented in the functional programming language Haskell. A truly rich source of examples of streams and generating functions has been the book [GKP94]. Our main source on formal power series has been [BR88], which contains amongst many other results a proof that rational series and hence rational streams, which are series in one variable, have finite representations. The explicit construction of finite nd-automata for rational streams, in Section 8, which is directly based on the behavioural differential equations for product and inverse, seems to be new.

Our construction of automata for rational languages in Section 10, based on input derivatives, is in essence the same as the algorithm by Brzozowski in [Brz64]. The, closely related, coinductive proof method for language equality seems to be new. This in spite of the fact that in concurrency theory, it is well-known that Milner's strong bisimulation for transition systems amounts to trace equivalence if the systems at hand are deterministic: both the notion of bisimulation used here (which includes a condition on the outputs) and the use of input derivatives in the construction of bisimulations, are new. The relation with the more traditional methods for proving language equality, has already been discussed at the end of Section 10. Another well-known way of proving equality of (regular expressions denoting) rational languages is to use a complete axiom system, such as given by Salomaa in [Sal66], and apply purely algebraic reasoning. The reader is invited to consult [Gin68, pp.68-69], which contains a minor variation on the example $K^* = L$, at the end

of Section 10, and convince himself of the greater complexity of that approach. The connection between finality and minimality, in Section 10, can already be found in [Gog73].

There are a number of different directions in which the present work can be extended. Coinduction has been formulated here, as usual, in terms of bisimulations. In [Rut98a], a more general coinduction principle is discussed for language inclusion, in terms of *simulation* relations. This can be straightforwardly extended to formal power series over semirings that carry a partial order (such as the Booleans and the reals). The present setting seems also suitable for experiments with even more general notions of bisimulations, sometimes referred to as quantitative bisimulations. Related work includes [Rut98b, Bal00, Wor00]. Another way of generalising the current framework is to allow for *partially* defined transition functions. This is briefly worked out for deterministic automata in [Rut98a] (see also [Rut99b], where these partial automata are used in the context of supervisory control problems for discrete event systems). Again it would be interesting to study partiality also for streams and formal power series. Yet another direction for future work concerns the generalised nondeterministic automata of Section 8. They can be equipped with a notion of bisimulation of their own (as mentioned at the end of Section 14), which would generalise both Milner's strong bisimulation and Larsen and Skou's probabilistic bisimulation [LS91, dVR97]. Further study seems worthwhile. Finally there is the theory of combinatorial species, based on work by Joyal [Joy81] and explained in all detail in the book [BLL98], to which in particular the coinductive treatment of streams seems to be closely related. We would like to understand the exact nature of the relation between these two worlds.

Acknowledgements

Many many thanks are due to the following persons, for pointers to the literature, corrections, and discussions: Jaco de Bakker, Marcello Bonsangue, Falk Bartels, Alexandru Baltag, Franck van Breugel, Matteo Coccia, Dora Giammarresi, Bart Jacobs, Aart Middeldorp, Maurice Nivat, Andy Pitts, Fer-Jan de Vries.

13 Appendix: the proof of Theorem 3.2

The remaining steps in the proof of Theorem 3.2 are given here.
The operators satisfy the differential equations: In order to show that, for instance, the product satisfies its defining behavioural differential equation, we compute as follows:

$$
\begin{aligned}
&(\sigma \times \tau)(0) \\
&\quad = \; l(\underline{\sigma} \times \underline{\tau})(0) \\
&\quad = \; (\underline{\sigma} \times \underline{\tau})(0) \quad [l \text{ is a homomorphism}] \\
&\quad = \; \underline{\sigma}(0) \times \underline{\tau}(0) \quad [\text{by the definition of } o_{\mathcal{E}}] \\
&\quad = \; \sigma(0) \times \tau(0) \quad [\text{by the definition of } o_{\mathcal{E}}]
\end{aligned}
$$

This shows that $\sigma \times \tau$ has the correct initial value. Checking the equality for the derivative, we compute

$$
\begin{aligned}
&(\sigma \times \tau)' \\
&\quad = \; (l(\underline{\sigma} \times \underline{\tau}))' \\
&\quad = \; l((\underline{\sigma} \times \underline{\tau})') \quad [l \text{ is a homomorphism}] \\
&\quad = \; l(((\underline{\sigma})' \times \underline{\tau}) + (\underline{\sigma}(0) \times (\underline{\tau})')) \quad [\text{by the definition of } t_{\mathcal{E}}] \\
&\quad = \; l((\underline{\sigma'} \times \underline{\tau}) + (\sigma(0) \times \underline{\tau'})) \quad [\text{by the definitions of } o_{\mathcal{E}} \text{ and } t_{\mathcal{E}}] \\
&\quad = \; l(\underline{\sigma'} \times \underline{\tau}) + l(\sigma(0) \times \underline{\tau'}) \quad [\text{(ii): } l \text{ is compositional}] \\
&\quad = \; (l(\underline{\sigma'}) \times l(\underline{\tau})) + (l(\sigma(0)) \times l(\underline{\tau'})) \quad [\text{(ii): } l \text{ is compositional}] \\
&\quad = \; (\sigma' \times \tau) + (\sigma(0) \times \tau') \quad [\text{(i): } l \text{ is the identity on stream constants}]
\end{aligned}
$$

Once we have proved assumptions (i) and (ii), this finishes the proof that the product operator satisfies its defining behavioural differential equation (one shows in a similar way that the other operators satisfy their defining differential equation):

(i) For all streams σ in $\mathbb{R}^\omega$: $l(\underline{\sigma}) = \sigma$.

(ii) The homomorphism $l : \mathcal{E} \to \mathbb{R}^\omega$ is compositional. That is, for all expressions E and F in $\mathcal{E}$,

(a) $l(E + F) = l(E) + l(F)$
(b) $l(E \times F) = l(E) \times l(F)$
(c) $l(E^*) = l(E)^*$
(d) $l(E^{-1}) = l(E)^{-1}$

Note that the operators on the left are syntactic constructors of expressions, while on the right there are the operators on streams defined by the equalities in (1) above.

Fact (i) is an immediate consequence of our observation above that the behaviour of the constants $\underline{\sigma}$ in $\mathcal{E}$ is the same as that of σ in $\mathbb{R}^\omega$, which is formally expressed by the fact that

$$\{\langle \underline{\sigma}, \sigma \rangle \mid \sigma \in \mathbb{R}^\omega\}$$

is a bisimulation relation between the automaton $\mathcal{E}$ and the automaton $\mathbb{R}^\omega$. This proves $\underline{\sigma} \sim \sigma$. Because l is a homomorphism, $l(\underline{\sigma}) \sim \underline{\sigma}$, whence $l(\underline{\sigma}) \sim \sigma$. Now the equality $l(\underline{\sigma}) = \sigma$ follows by the coinduction proof principle Theorem 2.4.

For (ii), we again treat the case of the product; the other operators can be dealt with similarly. By the definition of the product operator, we have

$$l(E) \times l(F) = l(\underline{l(E)} \times \underline{l(F)})$$

so in order to obtain $l(E \times F) = l(E) \times l(F)$, we are left with proving

$$l(\underline{l(E)} \times \underline{l(F)}) = l(E \times F)$$

By Proposition 2.6, this is equivalent to

$$(\underline{l(E)} \times \underline{l(F)}) \sim (E \times F) \tag{37}$$

Note that $E \sim l(E)$ since l is a homomorphism, and that $l(E) \sim \underline{l(E)}$ by (the proof of) fact (i) above. Thus $\underline{l(E)} \sim E$ and similarly $\underline{l(F)} \sim F$. Now (37) follows the fact that bisimilarity is a congruence with respect to the operators, which we shall prove next: for all expressions E, F, G and H in $\mathcal{E}$, if $E \sim G$ and $F \sim H$ then

$$(E + F) \sim (G + H), \quad (E \times F) \sim (G \times H), \quad E^* \sim G^*, \quad E^{-1} \sim G^{-1} \tag{38}$$

For a proof, let $\sim^c \subseteq \mathbb{R}^\omega \times \mathbb{R}^\omega$ be the smallest congruence relation containing $\sim$. That is, let $\sim^c$ be the smallest relation on $\mathcal{E}$ with $\sim \subseteq \sim^c$ and such that, for all expressions E, F, G, and H in $\mathcal{E}$: if $E \sim^c G$ and $F \sim^c H$ then

$$(E + F) \sim^c (G + H), \quad (E \times F) \sim^c (G \times H), \quad E^* \sim^c G^*, \quad E^{-1} \sim^c G^{-1}$$

The relation $\sim^c$ is a bisimulation on $\mathcal{E}$, which can be shown by induction on its definition; this implies $\sim^c \subseteq \sim$, because bisimilarity is the greatest bisimulation relation and consequently, $\sim^c = \sim$, which proves (38). (To prove that $\sim^c$ is a bisimulation, consider for instance $(E \times F) \sim^c (G \times H)$, and assume, as an inductive hypothesis, that $E(0) = G(0)$, $F(0) = H(0)$, $E' \sim^c G'$, and $F' \sim^c H'$. Then $(E \times F)(0) = (G \times H)(0)$ is equivalent to $E(0) \times F(0) = G(0) \times H(0)$, which is immediate by the inductive hypothesis. To show $(E \times F)' \sim^c (G \times H)'$, first note that the inductive hypothesis together with the congruence property of $\sim^c$ implies $(E' \times F) \sim^c (G' \times H)$ and $(E(0) \times F') \sim^c (G(0) \times H')$. It follows that

$$(E \times F)' = (E' \times F) + (E(0) \times F') \sim^c (G' \times H) + (G(0) \times H') = (G \times H)'$$

The other cases are similar. This proves that $\sim^c$ is a bisimulation relation on $\mathcal{E}$.)

The operators are unique: Finally, it is shown that the solutions we have obtained are unique. To this end, let $\hat{+}$, $\hat{\times}$, $(-)^*$, and $(-)^{\hat{-1}}$ be operators satisfying the differential equations of the present theorem. Let R be defined as the smallest relation on $R \subseteq \mathbb{R}^\omega \times \mathbb{R}^\omega$ containing $\{\langle \sigma, \sigma \rangle \mid \sigma \in \mathbb{R}^\omega\}$ and such that, for all streams σ, τ, α, and β in $\mathbb{R}^\omega$: if $\sigma \, R \, \tau$ and $\alpha \, R \, \beta$ then

$$(\sigma \hat{+} \alpha) \, R \, (\tau + \beta), \quad (\sigma \hat{\times} \alpha) \, R \, (\tau \times \beta), \quad \sigma^{\hat{*}} \, R \, \tau^*, \quad \sigma^{\hat{-1}} \, R \, \tau^{-1}$$

The relation R is a bisimulation (which can be shown in precisely the same way as for $\sim^c$ above). It follows from the coinduction proof principle (Theorem 2.4) that $\sigma + \tau = \sigma \hat{+} \tau$, for all streams σ and τ, and similarly for the other operators. This shows that the operators are unique and concludes the proof of Theorem 3.2.

14 Appendix: automata are coalgebras

We explain precisely in what way our approach to streams is coalgebraic, by recalling a number of elementary definitions and results from universal coalgebra (as in [Rut96]). It is then straightforward to do the same for power series, which is left to the reader.

Let $F : \mathit{Set} \to \mathit{Set}$ be a functor on the category of sets and functions. An *F-coalgebra* is a pair (S, α) consisting of a set S and a function $\alpha : S \to F(S)$. If (S, α) and (T, β) are two F-coalgebras, then a function $f : S \to T$ is a *homomorphism of F-coalgebras*, or F-homomorphism, if $F(f) \circ \alpha = \beta \circ f$:

$$\begin{tikzcd} S \arrow[r, "f"] \arrow[d, "\alpha"'] & T \arrow[d, "\beta"] \\ F(S) \arrow[r, "F(f)"'] & F(T) \end{tikzcd}$$

A relation $R \subseteq S \times T$ is called an *F-bisimulation* if there exists an F-coalgebra structure $\gamma : R \to F(R)$ on R such that the projections $\pi_1 : R \to S$ and $\pi_2 : R \to T$ are F-homomorphisms:

$$\begin{tikzcd} S \arrow[d, "\alpha"'] & R \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] \arrow[d, dashed, "\exists \mid \gamma"] & T \arrow[d, "\beta"] \\ F(S) & F(R) \arrow[l, "F(\pi_1)"] \arrow[r, "F(\pi_2)"'] & F(T) \end{tikzcd}$$

An F-coalgebra (P, π) is *final* if for any F-coalgebra (S, α) there exists one and only one F-homomorphism from (S, α) to (P, π). The following results hold for all functors F satisfying some rather mild conditions (F should preserve weak pullbacks and should be bounded) which are explained in detail in [Rut96]:

1. The union of F-bisimulations is again an F-bisimulation. In particular, the union of all F-bisimulations relations, called F-bisimilarity and denoted by $\sim$, is itself an F-bisimulation.

2. An F-bisimulation relation which actually is a function is an F-homomorphism.

3. There exists a final F-coalgebra (P, π).

4. For any F-coalgebra (S, α), the unique F-homomorphism $l : (S, \alpha) \to (P, \pi)$ satisfies, for all $s, t \in S$: $s \sim t$ iff $l(s) = l(t)$.

5. For all $p, q \in P$: if $p \sim q$ then $p = q$.

The definitions and results in Section 2 can be obtained by considering the functor

$$\mathbb{R} \times (-) : \mathit{Set} \to \mathit{Set}$$

which maps a set S to the Cartesian product $\mathbb{R} \times S$, and a function $f : S \to T$ to the function $\mathbb{R} \times f : (\mathbb{R} \times S) \to (\mathbb{R} \times T)$, which sends $\langle r, s \rangle \in \mathbb{R} \times S$ to the pair $\langle r, f(s) \rangle \in \mathbb{R} \times T$. It is an easy exercise to verify that the definitions of stream automaton, homomorphism, and bisimulation, are equivalent to the above definitions of F-coalgebra, F-homomorphism, and F-bisimulation, for $F = \mathbb{R} \times (-)$. The final coalgebra for this functor is the set $\mathbb{R}^\omega$ (Theorem 2.5), which indeed satisfies the above mentioned properties, stated in Section 2 as Proposition 2.6 and Theorem 2.4.

Also the definition of nondeterministic stream automaton, in Section 8, can be obtained in a similar way. The functor involved is

$$\mathbb{R} \times \mathbb{R}(-) : \mathit{Set} \to \mathit{Set}$$

mapping a set S to the set $\mathbb{R} \times \mathbb{R}(S)$ with

$$\mathbb{R}(S) = \{\phi : S \to \mathbb{R} \mid sp\,(\phi) \text{ is finite } \}$$

Clearly, nd-automata are F-coalgebras for $F = \mathbb{R} \times \mathbb{R}(-)$. Note that we did not yet specify how this functor acts on functions. Although this is not needed for the reconstruction of the results on nd-automata (we did not introduce the notions of homomorphism and bisimulation *for nd-automata*), here is the definition, for completeness sake. A function $f : S \to T$ is mapped to

$$\mathbb{R} \times \mathbb{R}(f) : (\mathbb{R} \times \mathbb{R}(S)) \to (\mathbb{R} \times \mathbb{R}(T)), \quad \langle r, \phi \rangle \mapsto \langle r, \mathbb{R}(\phi) \rangle$$

where $\mathbb{R}(\phi) : T \to \mathbb{R}$ is defined, for $t \in T$, by

$$\mathbb{R}(\phi)(t) = \sum_{f(s)=t} \phi(s)$$

Note that this sum exists because of the requirement on ϕ to be of finite support. Although we have not dealt with the notions of homomorphism and bisimulation for this functor, it would actually be quite interesting to do so. In particular, the notion of $\mathbb{R} \times \mathbb{R}(-)$-bisimulation would generalise both Milner's classical strong bisimulation [Mil80] as well as the more recent notion of probabilistic bisimulation [LS91, dVR97]. But alas, this will have to wait for another occasion.

Before we conclude this section and therewith the paper, let us give a brief comment on the word 'coinduction'. This terminology suggest that we are dealing here with a principle that is somehow dual to that of induction. This is explained by the observation that induction principles apply to *initial algebras*. Somewhat more concretely, the duality can be understood as follows. It is not difficult to prove that coinduction on $\mathbb{R}^\omega$ is equivalent to the statement that $\mathbb{R}^\omega$ has no *proper quotients*, that is, if $f : \mathbb{R}^\omega \to Q$ is a surjective homomorphism then $\mathbb{R}^\omega \cong Q$. This property is dual to the principle of mathematical induction on the algebra of natural numbers, which essentially states that the algebra of natural numbers has no *proper subalgebras*. See [Rut96, Sec.13] for a more detailed explanation.

Summarising the above, we hope to have explained in what sense the treatment of automata in the preceding sections has been coalgebraic: the definitions of automaton, homomorphism, and bisimulation, as well as the focus on finality and coinduction, all have been derived from or motivated by very general definitions and observations from coalgebra. As such, this coalgebraic story of (stream) automata is just one out of many, in principle as many as there are functors (on *Set* but also on other categories). Many other examples have been studied in considerable detail already, including transition systems, data types (such as streams and trees), dynamical systems, probabilistic systems, object-based systems, and many more. And many more are still to follow.

References

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

[Bal00] Alexandru Baltag. A logic for coalgebraic simulation. In Horst Reichel, editor, *Electronic Notes in Theoretical Computer Science*, volume 33. Elsevier Science Publishers, 2000.

[Bar00] F. Bartels. Generalised coinduction. Report SEN-R00??, CWI, Amsterdam, 2000. In preparation.

[BLL98] F. Bergeron, G. Labelle, and P. Leroux. *Combinatorial species and tree-like structures*, volume 67 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, 1998.

[BR88] J. Berstel and C. Reutenauer. *Rational series and their languages*, volume 12 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, 1988.

[Brz64] J.A. Brzozowski. Derivatives of regular expressions. *Journal of the ACM*, 11(4):481–494, 1964.

[Con71] J.H. Conway. *Regular algebra and finite machines*. Chapman and Hall, 1971.

[dVR97] E.P. de Vink and J.J.M.M. Rutten. Bisimulation for probabilistic transition systems: a coalgebraic approach. In P. Degano, R. Gorrieri, and A. Marchetti-Spaccamela, editors, *Proceedings of ICALP'97*, volume 1256 of *Lecture Notes in Computer Science*, pages 460–470, 1997. To appear in Theoretical Computer Science.

[Gin68] A. Ginzburg. *Algebraic theory of automata*. ACM Monograph series. Academic Press, 1968.

[GKP94] R.L. Graham, D.E. Knuth, and O. Patashnik. *Concrete Mathematics, Second Edition*. Addison-Wesley, 1994.

[GM98] S. Gaubert and J. Mairesse. Task resource models and (max,+) automata. In *[Gun98]*, pages 133–144, 1998.

[Gog73] J. Goguen. Realization is universal. *Mathematical System Theory*, 6:359–374, 1973.

[Gun98] J. Gunawardena. *Idempotency*. Publications of the Newton Institute. Cambridge University Press, 1998.

[Joy81] A. Joyal. Une théorie combinatoire des séries formelles. *Advances in mathematics*, 42:1–82, 1981.

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

[LS91] K.G. Larsen and A. Skou. Bisimulation through probabilistic testing. *Information and Computation*, 94:1–28, 1991.

[McI99] M. D. McIlroy. Power series, power serious. *J. Functional Programming*, 9:323–335, 1999.

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

[Par81] D.M.R. Park. Concurrency and automata on infinite sequences. In P. Deussen, editor, *Proceedings 5th GI conference*, volume 104 of *Lecture Notes in Computer Science*, pages 15–32. Springer-Verlag, 1981.

[PE98] Dusko Pavlović and Martín Escardó. Calculus in coinductive form. In *Proceedings of the 13th Annual IEEE Symposium on Logic in Computer Science*, pages 408–417. IEEE Computer Society Press, 1998.

[Rut96] J.J.M.M. Rutten. Universal coalgebra: a theory of systems. Report CS-R9652, CWI, 1996. Available at URL: www.cwi.nl. To appear in Theoretical Computer Science.

[Rut98a] J.J.M.M. Rutten. Automata and coinduction (an exercise in coalgebra). Report SEN-R9803, CWI, 1998. Available at URL: www.cwi.nl. Also in the proceedings of CONCUR '98, LNCS 1466, 1998, pp. 194–218.

[Rut98b] J.J.M.M. Rutten. Relators and metric bisimulations. In Horst Reichel Bart Jacobs, Larry Moss and Jan Rutten, editors, *Electronic Notes in Theoretical Computer Science*, volume 11. Elsevier Science Publishers, 1998.

[Rut99a] J.J.M.M. Rutten. Automata, power series, and coinduction: taking input derivatives seriously (extended abstract). Report SEN-R9901, CWI, 1999. Available at URL: www.cwi.nl. Also in the proceedings of ICALP '99, LNCS 1644, 1999, pp. 645–654.

[Rut99b] J.J.M.M. Rutten. Coalgebra, concurrency, and control. Report SEN-R9921, CWI, 1999. Available at URL: www.cwi.nl. Extended abstract in: Discrete Event Systems, R. Boel and G. Stremersch (eds.), Kluwer, 2000.

[Sal66] A. Salomaa. Two complete axiom systems for the algebra of regular events. *Journal of the ACM*, 13(1):158–169, 1966.

[Wor00] James Worrel. Coinduction for recursive data types: partial orders, metric spaces and omega-categories. In Horst Reichel, editor, *Electronic Notes in Theoretical Computer Science*, volume 33. Elsevier Science Publishers, 2000.