The Method of Coalgebra: exercises in coinduction

Jan Rutten

February 2019
261 pages
ISBN 978-90-6196-568-8

Publisher: CWI, Amsterdam,
The Netherlands

The Method of Coalgebra: exercises in coinduction

Jan Rutten

CWI & RU

jjmmrutten@gmail.com

February 16, 2019

Contents

1 Introduction **7**
- 1.1 The method of coalgebra . . . 7
- 1.2 History, roughly and briefly . . . 7
- 1.3 Exercises in coinduction . . . 8
- 1.4 Enhanced coinduction: algebra plus coalgebra . . . 8
- 1.5 Universal coalgebra . . . 8
- 1.6 How to read this book . . . 8
- 1.7 Acknowledgements . . . 9

2 Categories – where coalgebra comes from **11**
- 2.1 The basic definitions . . . 11
- 2.2 Category theory in slogans . . . 12
- 2.3 Discussion . . . 16

3 Algebras and coalgebras **19**
- 3.1 Algebras . . . 19
- 3.2 Coalgebras . . . 21
- 3.3 Discussion . . . 23

4 Induction and coinduction **25**
- 4.1 Inductive and coinductive definitions . . . 25
- 4.2 Proofs by induction and coinduction . . . 28
- 4.3 Discussion . . . 32

5 The method of coalgebra **33**
- 5.1 Basic types of coalgebras . . . 34
- 5.2 Coalgebras, systems, automata . . . 34

6 Dynamical systems **37**
- 6.1 Homomorphisms of dynamical systems . . . 38
- 6.2 On the behaviour of dynamical systems . . . 41
- 6.3 Discussion . . . 44

7 Stream systems **45**
7.1 Homomorphisms and bisimulations of stream systems 46
7.2 The final system of streams 52
7.3 Defining streams by coinduction 55
7.4 Coinduction: the bisimulation proof method 60
7.5 Moessner's Theorem 67
7.6 The heart of the matter: circularity 73
7.7 Discussion 77

8 Deterministic automata **79**
8.1 Basic definitions 80
8.2 Homomorphisms and bisimulations of automata 82
8.3 The final automaton of languages 89
8.4 Regular languages 93
8.5 Proving language equality by coinduction 95
8.6 Minimal automata 100
8.7 Discussion 107

9 Partial automata **109**
9.1 Partiality 110
9.2 Partial systems with outputs 113
9.3 Partial automata 115
9.4 Discussion 118

10 Non-deterministic automata **119**
10.1 The powerset construction 121
10.2 Language equivalence of non-deterministic automata 122
10.3 Discussion 126

11 Stream differential equations **129**
11.1 Examples of stream differential equations 131
11.2 Introducing the syntactic solution method 134
11.3 GSOS: a general format for stream differential equations 140
11.4 Behavioural differential equations 147
11.5 Discussion 148

12 A calculus of streams **151**
12.1 Basic stream operations 151
12.2 Convolution inverse 158
12.3 Stream composition 162
12.4 Shuffle product 164
12.5 Laplace transform 168
12.6 Solving analytic differential equations 171

12.7 Newton transform 175
12.8 Rational streams and linear systems of SDEs 181
12.9 Discussion 188

13 Mealy automata 189
13.1 Causal stream functions 190
13.2 Constructing minimal Mealy automata 192
13.3 Moore automata 194
13.4 A calculus of bitstreams 195
13.5 Mealy machines for bitstream functions 199
13.6 Discussion 203

14 Weighted stream automata 205
14.1 Representing streams by weighted automata 205
14.2 Splitting stream derivatives 207
14.3 Weighted automata and linear stream systems 210
14.4 Rational streams and linear stream systems 216
14.5 Minimisation of linear stream systems 220
14.6 Weighted automata with inputs 224
14.7 Discussion 225

15 Universal coalgebra 227
15.1 Lambek's Lemma 227
15.2 Homomorphisms 229
15.3 Coproducts of coalgebras 229
15.4 Bisimulation and finality 231
15.5 Enhanced coinduction 234
15.6 Discussion 235

16 Notation and preliminaries 237
16.1 Numbers 237
16.2 Functions 237
16.3 Products and sums 238
16.4 Subsets 239
16.5 Relations 240
16.6 Words and languages 241
16.7 Monoids, (semi-)rings, integral domains, fields 241
16.8 Pre-orders and partial orders 243
16.9 Vector spaces and linear maps 243
16.10 Functors on sets 244
16.11 On-line Encyclopedia of Integer Sequences 245

Chapter 1

Introduction

So you heard about coalgebra, and maybe you want to know more. Or you know about coalgebra, and maybe you want to hear more still, or you just want to see how things are done here. This is what we will do.

1.1 The method of coalgebra

In Chapters 2 to 5, we will briefly sketch our view on the coalgebraic method. The notion of coalgebra arises as the dual, in the theory of categories, of the notion of algebra. So this is what we explain first, in Chapters 2 and 3. Then coalgebras come to life and become relevant, only after one dualises also the notions of congruence relation and induction, two of the essential ingredients of the theory of algebras. This yields the notions of bisimulation and coinduction, explained in Chapter 4. With this in place, we summarise, in Chapter 5, the method of coalgebra as a general formalism for the study of the behaviour – typically infinite, stemming from some form of circularity – of state-based, dynamical systems.

1.2 History, roughly and briefly

In the late 1970s, coalgebras occur as the dual of algebras in the work of Arbib and Manes, in their category theoretic approach to dynamical systems and automata [AM75, AM82]. Their textbook [MA86] already presents automata as coalgebras and contains an early proof that the set of formal languages is a final coalgebra (Theorem 153 in Chapter 8 of the present book). Still nothing much happens with coalgebra until (i) Park and Milner invent bisimulation as a notion of behavioural equivalence for concurrent processes [Mil80, Par81, Mil89]; and (ii) Aczel needs a notion of strong extensionality for his theory of non-well-founded sets [Acz88]. To this end, Aczel and Mendler generalise Park and Milner's notion of bisimulation to the level of arbitrary coalgebras [AM89]. With this new notion of coalgebraic bisimulation, it becomes obvious how to formulate a general notion of coinduction for arbitrary coalgebras. And as a consequence, coalgebra becomes a lively field of research from the early 1990s onwards.

It follows from the above that the foundations of coalgebra consist of a mix of ingredients from mathematics (dynamical systems, category theory), computer science (automata theory, concurrency theory), and logic (axiomatic set theory). The above suggests further that coinduction is the reason why coalgebra might be interesting.

1.3 Exercises in coinduction

Each of Chapters 6 - 14 deals with a concrete type of coalgebra. These coalgebras include many familiar mathematical structures, such as dynamical systems, streams (infinite sequences), and automata. These mathematical structures themselves are the main subject of these chapters, but methodologically, each chapter offers a playground for exercises in coinduction: the coalgebraic method for definitions and proofs.

1.4 Enhanced coinduction: algebra plus coalgebra

In the algebraic approach to the analysis of computational behaviour, one uses algebraic operators to represent the structure at hand as a composition of substructures. In this manner, behavioural complexity is reduced to the composition of the smaller complexities of the substructures, a phenomenon sometimes referred to as *divide and conquer*. In the coalgebraic approach, complexity is typically reduced by exploiting *circularity*, by observing that after a number of transition steps, essentially the same situation occurs again. As we shall see, the *combination of algebra and coalgebra* leads to powerful computational proof methods, typically referred to as *enhanced coinduction* or *coinduction up-to*, where one constructs bisimulation relations *up-to* the closure under a given set of algebraic operators. We will present various concrete instances of bisimulation and coinduction up-to. For more general accounts, see the references in Section 15.5.

1.5 Universal coalgebra

The classical field of universal algebra deals with general properties of algebras. In universal coalgebra, one studies general properties of coalgebras, and Chapter 15 gives the reader a quick impression of this subject. The present book is about exercises in coinduction with *concrete coalgebras*. Generality is not our aim here, and so we refer the reader to the literature for systematic treatments of universal coalgebra.

1.6 How to read this book

For most of this book, very little background in mathematics is required. The brief sketch of the coalgebraic method in Chapters 2 to 5 may be too technical for some readers but then again, these chapters can be safely skipped at first reading: nothing in the rest of the book depends on them. Still, we recommend to have at least a quick look at Chapters 2 to 5. Even without understanding much in detail, they do provide a sense of direction which all of the subsequent chapters follow.

Chapters 6, on dynamical systems; 7, on streams; and 8, on automata, together constitute an already substantial part of the message of this book: coalgebra is interesting because coinduction is. Having read these three chapters, it might make sense to return to Chapters 2 to 5, in case they had been skipped initially.

Chapters 11 and 12, on stream differential equations and stream calculus, need nothing from Chapters 8 – 13. As a consequence, Chapters 6, 7, 11, and 12 can be taken together as an introduction to the coalgebraic theory of streams.

Chapter 15 gives only a brief impression of the theory of universal coalgebra, by pointing out examples of implicit generalities that underlie the preceding chapters.

In Chapter 16, we have collected a few basic definitions and notations that are used throughout the book.

1.7 Acknowledgements

I am most grateful to my former PhD students for the joint study of coalgebra, on which much of the present book is based. In historical order, and referring to their PhD theses: Daniele Turi [Tur96], Falk Bartels [Bar04], Clemens Kupke [Kup06], Helle Hansen [Han09], David Costa [Cos10], Alexandra Silva [Sil10], Georgiana Caltais [Cal13], Joost Winter [Win14], Jurriaan Rot [Rot16], Henning Basold [Bas18], and Julian Salamanca [Sal18]. (More detailed references will appear throughout the book.)

Thank you very much, Marcello Bonsangue, for many years of inspiring and joyful collaboration, including the joint supervision of many of the above students.

I am much indebted to the communities of the conferences CMCS (Coalgebraic Methods in Computer Science) and CALCO (Conference on Algebra and Coalgebra) for many years of inspiration and instruction.

For discussions and joint work, thank you, Samson Abramsky, Jirí Adámek, Farhad Arbab, Eric Badouel, Adolfo Ballester-Bolinches, Luis Barbosa, Frank de Boer, Michele Boreale, Alexandru Baltag, Filippo Bonchi, Franck van Breugel, Andrea Corradini, Enric Cosme Llópez, the late Philippe Darondeau, Jörg Endrullis, Jan van Eijck, Herman Geuvers, Clemens Grabmayer, Dimitri Hendriks, Ralf Hinze, Furio Honsell, Bart Jacobs, Jan Willem Klop, Jan Komenda, Dexter Kozen, Alexander Kurz, Dorel Lucanu, Sun Meng, Wilfried Meyer-Viol, Stefan Milius, Milad Niqui, José Nuno Oliveira, Prakash Panangaden, Jean-Éric Pin, Damien Pous, Grigore Rosu, Davide Sangiorgi, Jan van Schuppen, Nico Temme, Tarmo Uustalu, Yde Venema, Erik de Vink, Fer-Jan de Vries.

I am grateful to Jan Willem Klop, Mike Mislove, Ugo Montanari, the late Maurice Nivat, and Prakash Panangaden for lessons and discussions, and for support during the early stages of the coalgebraic developments. And to Bart Jacobs, Larry Moss, and Horst Reichel, for jointly creating the workshop CMCS.

Many thanks also to the NII Shonan organisation, Japan, for hosting various workshops on coinduction.

For many years of friendly instruction in the ways of science and the world, I am much indebted to my former supervisor, the late Jaco de Bakker.

CWI has offered me an excellent home for all of my scientific life: thank you very much. I am grateful for having been a guest at the Free University Amsterdam and the Radboud University Nijmegen. Many thanks also to the Dutch Science Organisation NWO for many years of project support.

Thank you very much: Henning Basold, Helle Hansen, Jurriaan Rot, Erik de Vink, Jana Wagemaker and Joost Winter, for your corrections and constructive suggestions.

Tanti baci, Elena Marchiori, for all your patience, help and support

Chapter 2

Categories – where coalgebra comes from

We discuss some of the main principles of category theory, including duality. This will allow us to explain, in Chapters 3 and 4, how the notions of coalgebra and coinduction arise as the dual of the notions of algebra and induction. Since the theory of coalgebra is essentially categorical, the principles discussed in the present chapter form an integral part of the coalgebraic method, which will be discussed in Chapter 5.

2.1 The basic definitions

A *category* $\mathcal{C}$ consists of

- *objects* $A, B, C, \ldots$
- for each pair of objects A, B, a collection $\mathcal{C}(A, B)$ of *morphisms* (or *arrows*) $f\colon A \to B$ with domain A and codomain B.
- an operation of *composition* of morphisms: for any two morphisms $f\colon A \to B$ and $g\colon B \to C$, there is a morphism $g \circ f\colon A \to C$:

$$\begin{tikzcd} A \arrow[r, "f"'] \arrow[rr, bend left, "g \circ f"] & B \arrow[r, "g"'] & C \end{tikzcd}$$

- identities: for each object A, there is a morphism $1_A\colon A \to A$.
- axioms:

$$h \circ (g \circ f) \;=\; (h \circ g) \circ f \qquad\qquad f \circ 1_A \;=\; f \;=\; 1_B \circ f$$

Our guiding example will be the category Set of sets and functions.

Exercise 1. Show that the family of pre-ordered sets and monotone functions between them (see Section 16.8 for the definitions) is a category. The same for vector spaces and linear functions, as defined in Section 16.9. $\square$

Exercise 2. Show that a pre-ordered set $(P, \leqslant)$ can itself be viewed as a category: objects are elements p, q in P and we have an arrow $p \to q$ iff $p \leqslant q$. $\square$

Exercise 3. Show that every monoid $(M, \cdot, e)$ (see Section 16.7 for the definition) can be viewed as a category that has M as its only object and that has the elements of M as its arrows. $\square$

2.2 Category theory in slogans

Next we discuss some of the basic principles underlying the theory of categories.

Always ask: what are the types? All we have in category theory are objects and arrows, and for the latter, we always need to be precise about their domain and codomain. So we shall never say: *let f be a function defined for any x by $f(x) = \dots$*, but rather: *let $f\colon X \to Y$ be a function defined for any $x \in X$ by $f(x) = \dots$*, thus making the types of both x and $f(x)$ explicit.

Think in terms of arrows rather than elements. We shall explain this with an example. We all know the following definitions: A function $f\colon X \to Y$ (between sets) is *injective* if

$$\forall x, y \in X\colon \quad f(x) = f(y) \implies x = y$$

and f is *surjective* if

$$\forall y \in Y\ \exists x \in X\colon \quad f(x) = y$$

Now let us call a function $f\colon X \to Y$ *monic* if for all $g, h\colon Z \to X$,

$$f \circ g = f \circ h \implies g = h$$

and let us call f *epic* if for all $g, h\colon Y \to Z$,

$$g \circ f = h \circ f \implies g = h$$

Then f is injective iff f is monic, and f is surjective iff f is epic.

Exercise 4. Prove this. $\square$

The definitions of injective and surjective function are phrased in terms of elements; the two definitions are completely different and seem unrelated. The equivalent notions of monic and epic are defined using arrows; these two definitions are very similar: as we shall see below, they are *dual*. This precise and close relationship is not at all visible from the element-based definitions of injective and surjective.

Ask what mathematical structures do, not what they are. It is in general more relevant how an object behaves than how it is constructed. Here behaviour of an object should be understood as the way in which it interacts with other objects, through arrows. Let's look at an example. In set theory, the Cartesian product of two sets X and Y is typically defined as the set of all ordered pairs $\langle x, y\rangle$, for elements $x \in X$ and $y \in Y$:

$$X \times Y \;=\; \{\langle x, y\rangle \mid x \in X,\; y \in Y\}$$

Here an ordered pair $\langle x, y\rangle$ is defined by

$$\langle x, y\rangle \;=\; \{\{x, y\}, y\}$$

Such a definition is by no means canonical, nor does it express the essential properties of ordered pairs. Categorically, a product is defined by what it *does*, that is, by specifying by which arrows it is related to other objects. Formally: a *product* of two objects A and B is an object $A \times B$ together with two arrows

$$A \xleftarrow{\pi_1} A \times B \xrightarrow{\pi_2} B \tag{2.1}$$

(called projections) such that for all objects C and arrows $f\colon C \to A$ and $g\colon C \to B$ there exists a unique arrow (which we call) $\langle f, g\rangle\colon C \to A \times B$ satisfying

$$\begin{tikzcd} & \forall C \arrow[dl, bend right, "f"'] \arrow[dr, bend left, "g"] \arrow[d, dashed, "\exists! \mid \langle f,g\rangle"] & \\ A & A \times B \arrow[l, "\pi_1"] \arrow[r, "\pi_2"'] & B \end{tikzcd}$$

That is, $\pi_1 \circ \langle f, g\rangle = f$ and $\pi_2 \circ \langle f, g\rangle = g$ and, moreover, $\langle f, g\rangle$ is the unique arrow satisfying these equations.

Exercise 5. Show that if both C and D are products of A and B then C and D are *isomorphic*: there are arrows $f\colon C \to D$ and $g\colon D \to C$ such that $f \circ g = 1_D$ and $g \circ f = 1_C$. As a consequence, we say that products are *unique up to isomorphism*. □

Exercise 6. Show that $h = \langle \pi_1 \circ h, \pi_2 \circ h\rangle$, for any $h\colon C \to A \times B$. Show that $\langle f, g\rangle \circ k = \langle f \circ k, g \circ k\rangle$, for any $k\colon D \to C$. □

The categorical definition of product stays away from the "coding" that was used in the set-theoretic definition of ordered pairs. Instead, it emphasises the intrinsic behaviour of the product by describing its interaction with other objects.

Functoriality! Given that a category consists of objects and arrows, it is only natural to require operations on categories to act on both. A *functor*

$$F\colon \mathcal{C} \to \mathcal{D}$$

from a category $\mathcal{C}$ to a category $\mathcal{D}$ assigns

(i) to each object A in $\mathcal{C}$ an object $F(A)$ in $\mathcal{D}$, and

(ii) to each arrow $f\colon A \to B$ in $\mathcal{C}$ an arrow $F(f)\colon F(A) \to F(B)$ in $\mathcal{D}$, such that

(iii) F preserves composition and identities: $F(g \circ f) = F(g) \circ F(f)$ and $F(1_A) = 1_{F(A)}$.

Example 7. An example is the *powerset functor*

$$\begin{array}{lll} \mathcal{P}\colon \mathsf{Set} \to \mathsf{Set} & \mathcal{P}(X) \;=\; \{\, V \mid V \subseteq X \,\} & \\ \mathcal{P}(f)\colon \mathcal{P}(X) \to \mathcal{P}(Y) & \mathcal{P}(f)(V) \;=\; f(V) & (f\colon X \to Y,\, V \subseteq X) \end{array}$$

(The *image* $f(V)$ of V under f is defined in Section 16.2.) For another example, let $\mathbb{N}$ be the set of natural numbers. We define the functor

$$\mathsf{Str}\colon \mathsf{Set} \to \mathsf{Set} \qquad\qquad \mathsf{Str}(X) = \mathbb{N} \times X$$

where, for a set X, the set $\mathbb{N} \times X$ is the Cartesian product of the sets $\mathbb{N}$ and X. (The name Str of this functor stands for *streams*, see Chapter 7.) On functions $f\colon X \to Y$, we define

$$\mathsf{Str}(f) \;=\; (1_{\mathbb{N}} \times f)\colon (\mathbb{N} \times X) \to (\mathbb{N} \times Y) \qquad (1_{\mathbb{N}} \times f)(n, x) \;=\; (n, f(x))$$

for all $n \in \mathbb{N}$ and $x \in X$. □

Exercise 8. Show that $\mathcal{P}$ and Str are functors. Let $(P, \leqslant)$ and $(Q, \leqslant)$ be preorders. □

Exercise 9. Let $(P, \leqslant)$ and $(Q, \leqslant)$ be preorders, seen as categories. Show that a function $f\colon P \to Q$ is a functor $f\colon (P, \leqslant) \to (Q, \leqslant)$ if and only if f is monotone:

$$p \;\leqslant\; q \quad\Longrightarrow\quad f(p) \;\leqslant\; f(q)$$

(cf. Exercise 2.) □

Exercise 10. Let $\mathcal{C}$ be a category in which the product $A \times B$ of any pair of objects (A, B) exists. (The category Set is an example.) Show that the operation of assigning to (A, B) their product $A \times B$ can be extended to a functor $\times : \mathcal{C} \times \mathcal{C} \to \mathcal{C}$. (Note that one needs to explain what the category $\mathcal{C} \times \mathcal{C}$ looks like.) □

Universality! It is often useful to define notions as general as possible, looking beyond the irrelevant details of a given concrete setting. In category theory, an important class of examples are definitions that are phrased in terms of *universal properties*, which typically are formulated as: for all ... there exists a unique .... Here is an example: an object A in a category $\mathcal{C}$ is *initial* if for every object B in $\mathcal{C}$ there exists a unique arrow from A to B:

$$\begin{tikzcd} \forall\, B & A \arrow[l, dashed, "\exists!"'] \end{tikzcd}$$

Similarly, an object A is *final* if for any object B there exists a unique arrow from B to A:

$$\begin{tikzcd} \forall\, B \arrow[r, dashed, "\exists!"] & A \end{tikzcd}$$

Exercise 11. Show that initial objects in a category are unique up to isomorphism, that is, all initial objects are isomorphic (cf. Exercise 5). Similarly for final objects. Describe initial and final objects in Set. □

Exercise 12. Consider two objects A and B in a category $\mathcal{C}$. Let an (A, B)*-pairing* be a triple (P, p_1, p_2) consisting of an object P and two arrows $p_1 \colon P \to A$ and $p_2 \colon P \to B$ in $\mathcal{C}$. Think of a suitable notion of *morphism of* (A, B)*-pairings* and show that $(A \times B, \pi_1, \pi_2)$ (cf. diagram (2.1) above) is final in the resulting category. Note that as a consequence (cf. Exercise 11), products are unique up to isomorphism. □

Duality! Informally, duality refers to the elementary process of "reversing the arrows" in a diagram. If this diagram was used to give a definition or to express a property, then reversing the arrows leads to a new definition or a new property, which is called the *dual* of the original one. For instance, we saw that an arrow $f \colon B \to C$ is *monic* if, for all arrows $g, h \colon A \to B$, the following implication holds:

$$\begin{tikzcd} A \arrow[r, bend left, "g"] \arrow[r, bend right, "h"'] & B \arrow[r, "f"] & C \end{tikzcd} \quad \Longrightarrow \quad g = h$$

If we use a diagram such as the one above, on the left of the implication, then we shall always silently assume that it is *commuting*. In the present example, this means that the diagram expresses the assumption that $f \circ g = f \circ h$. Reversing the arrows in the diagram, we obtain

$$\begin{tikzcd} A & B \arrow[l, bend right, "g"'] \arrow[l, bend left, "h"] & C \arrow[l, "f"'] \end{tikzcd} \quad \Longrightarrow \quad g = h$$

This implication expresses the property of f being *epic*. As a consequence, we say that monic and epic are dual notions.

Here's a more formal way of expressing duality. The *opposite* of a category $\mathcal{C}$ is the category $\mathcal{C}^{op}$ that has the same objects as $\mathcal{C}$ and precisely one arrow $f\colon B \to A$ for every arrow $f\colon A \to B$ in $\mathcal{C}$. (As a consequence, the composition $g \circ f$ of two arrows $f\colon A \to B$ and $g\colon B \to C$ in $\mathcal{C}^{op}$ corresponds to the arrow $f \circ g$ in $\mathcal{C}$.) The principle of duality now says that we can dualise any statement about a category $\mathcal{C}$ by making the same statement about $\mathcal{C}^{op}$.

Exercise 13. Show that an object A is initial in $\mathcal{C}$ iff A is final in $\mathcal{C}^{op}$. □

Exercise 14. A *coproduct* of two objects A and B is an object $A + B$ together with two arrows $\kappa_1\colon A \to A + B$ and $\kappa_2\colon B \to A + B$ (called embeddings) such that for all objects C and arrows $f\colon A \to C$ and $g\colon B \to C$ there exists a unique arrow (which we call) $[f, g]\colon A + B \to C$ satisfying

$$\begin{tikzcd} & \forall C & \\ A \arrow[ur, bend left, "f"] \arrow[r, "\kappa_1"'] & A + B \arrow[u, dashed, "\exists!"', "{[f,g]}"] & B \arrow[ul, bend right, "g"'] \arrow[l, "\kappa_2"] \end{tikzcd}$$

that is, $[f, g] \circ \kappa_1 = f$ and $[f, g] \circ \kappa_2 = g$. Show that coproducts in Set are given by disjoint union. Show that C is a coproduct of A and B in $\mathcal{C}$ iff C is a product of A and B in $\mathcal{C}^{op}$. □

2.3 Discussion

The above formulation of category theory by means of slogans has been inspired by *Categories, why and how?*, a tutorial presented by Samson Abramsky at the Dagstuhl Seminar *Coalgebraic Semantics of Reflexive Economics*, early 2015. See also [AT11] for his lecture notes on an introduction to category theory.

In the present book, category theory is used explicitly only in Chapters 2 to 5 (and also in Chapter 15). As we already observed in the introduction, these chapters can be safely skipped at first reading; at the same time, they do provide a sense of direction for all the remaining chapters. More generally, we find category theory both interesting, relevant, and beautiful, and would recommend the reader to study some category theory at some moment in time. A classical reference is Mac Lane's [ML71], in which motivation and examples are mostly taken from mathematics. More recent texts, often requiring less background in mathematics, include [LS97] and [Awo10].

Chapter 3

Algebras and coalgebras

We present a categorical definition of the notion of algebra, and arrive at the notion of coalgebra by applying the categorical principle of duality, introduced in Chapter 2.

3.1 Algebras

Classically, algebras are presented as sets with operations. An example is the algebra

$$(\mathbb{N},\, 0,\, \mathsf{succ}) \qquad \mathbb{N} \;=\; \{0, 1, 2, \ldots\} \qquad 0 \in \mathbb{N} \qquad \mathsf{succ}\colon \mathbb{N} \to \mathbb{N}$$

consisting of the set $\mathbb{N}$ of natural numbers, the constant 0 and the successor operation defined by $\mathsf{succ}(n) = n + 1$, for $n \geqslant 0$. Equivalently, the algebra of natural numbers can be presented as one function

$$[\mathsf{zero}, \mathsf{succ}]\colon (\mathbf{1} + \mathbb{N}) \to \mathbb{N}$$

where $\mathbf{1} = \{*\}$ is the singleton set with element $*$; where $\mathbf{1} + \mathbb{N}$ is the coproduct, that is, the disjoint union, of $\mathbf{1}$ and $\mathbb{N}$; and where

$$\begin{array}{ll} \mathsf{zero}\colon \mathbf{1} \to \mathbb{N} & \qquad \mathsf{succ}\colon \mathbb{N} \to \mathbb{N} \\ \mathsf{zero}(*) \;=\; 0 & \qquad \mathsf{succ}(n) \;=\; n + 1 \end{array}$$

Presented in this manner, the algebra of natural numbers becomes an instance of the following categorical definition.

Definition 15 (**F-algebra).** Let $F\colon \mathcal{C} \to \mathcal{C}$ be a functor from a category $\mathcal{C}$ to itself. An *F-algebra* is a pair (A, α) consisting of an object A and an arrow

$$\alpha\colon F(A) \to A$$

We call F the *type*, A the *carrier*, and α the *structure map* of the algebra (A, α). □

Defining the functor $N\colon \mathsf{Set} \to \mathsf{Set}$, for every set X, by $N(X) = \mathbf{1} + X$, we observe that $(\mathbb{N}, [\mathsf{zero}, \mathsf{succ}])$ is an N-algebra.

Exercise 16. Consider the ring of integers $(\mathbb{Z}, 0, 1, +, \times, -)$, consisting of the set $\mathbb{Z}$ of integers, the constants 0 and 1, and the operations of addition $+\colon \mathbb{Z}\times\mathbb{Z} \to \mathbb{Z}$, multiplication $\times\colon \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}$ and minus $-\colon \mathbb{Z} \to \mathbb{Z}$. Show that $(\mathbb{Z}, 0, 1, +, \times, -)$ is an F-algebra, by defining a suitable functor $F\colon \mathsf{Set} \to \mathsf{Set}$. $\square$

The exercise above illustrates that any algebra, defined as a set with operations, can be presented as an F-algebra. Examples from mathematics are groups, rings, fields, etc.

The structure map α of an F-algebra (A, α) tells us how the elements of A are *constructed* from other elements in A. Or, equivalently, how by *composing* elements in A one obtains new elements. For instance, composing the integers 3 and 2 using the operation of multiplication, one obtains the number $3 \times 2 = 6$.

Representing algebras by *functors* allows us to introduce the following notion.

Definition 17 (algebra homomorphism). Let $F\colon \mathcal{C} \to \mathcal{C}$ be a functor. A *homomorphism of F-algebras* (A, α) and (B, β) is an arrow $f\colon A \to B$ such that

$$\begin{tikzcd}
F(A) \arrow[r, "F(f)"] \arrow[d, "\alpha"'] & F(B) \arrow[d, "\beta"] \\
A \arrow[r, "f"'] & B
\end{tikzcd}$$

that is, such that $f \circ \alpha = \beta \circ F(f)$. $\square$

Note that F needs to be a functor for this definition to make sense: F has to act not only on objects but also on arrows so that $F(f)$ is well-defined. Since functors preserve the composition of arrows, the composition of two F-algebra homomorphisms will again be a homomorphism (cf. Exercise 19).

One can understand mathematical objects by investigating how they interact with other objects. Sets interact with other sets through functions, and homomorphisms are for algebras what functions are for sets. As an example, we consider again the functor

$$N\colon \mathsf{Set} \to \mathsf{Set} \qquad\qquad N(X) = \mathbf{1} + X$$

with $\mathbf{1} = \{*\}$. On functions $f\colon X \to Y$, the functor N is defined by $N(f) = \mathbf{1} + f$, where

$$(\mathbf{1} + f)\colon (\mathbf{1} + X) \to (\mathbf{1} + Y) \qquad (\mathbf{1} + f)(*) \;=\; * \qquad (\mathbf{1} + f)(x) \;=\; f(x)$$

for all $x \in X$. We saw that $(\mathbb{N}, [\mathsf{zero}, \mathsf{succ}])$ is an N-algebra. If we define another N-algebra

$$(\{a, b\}, [i, \mathrm{r}]) \qquad\qquad i\colon 1 \to \{a, b\} \qquad\qquad \mathrm{r}\colon \{a, b\} \to \{a, b\}$$

by

$$i(*) = a \qquad \mathrm{r}(a) = b \qquad \mathrm{r}(b) = a$$

then the function $f\colon \mathbb{N} \to \{a, b\}$, defined by $f(2n) = a$ and $f(2n + 1) = b$, is an example of a homomorphism of N-algebras:

$$\begin{tikzcd}
\mathbf{1} + \mathbb{N} \arrow[r, "\mathbf{1} + f"] \arrow[d, "{[\mathsf{zero}, \mathsf{succ}]}"'] & \mathbf{1} + \{a, b\} \arrow[d, "{[i, \mathrm{r}]}"] \\
\mathbb{N} \arrow[r, "f"'] & \{a, b\}
\end{tikzcd}$$

Exercise 18. Let $\mathbb{Q}$ be the set of all rational numbers. Consider the N-algebra

$$(\mathbb{Q}, [\mathsf{one}, \mathsf{half}]) \qquad \begin{array}{rcl} \mathsf{one}\colon \mathbf{1} \to \mathbb{Q} & & \\ \mathsf{one}(*) & = & 1 \end{array} \qquad \begin{array}{rcl} \mathsf{half}\colon \mathbb{Q} \to \mathbb{Q} & & \\ \mathsf{half}(q) & = & q/2 \end{array}$$

Describe all homomorphisms between $(\mathbb{N}, [\mathsf{zero}, \mathsf{succ}])$ and $(\mathbb{Q}, [\mathsf{one}, \mathsf{half}])$, in either direction. □

Exercise 19. Show that F-algebras and F-homomorphisms form a category, for any functor $F\colon \mathcal{C} \to \mathcal{C}$. □

3.2 Coalgebras

The categorical principle of duality, which informally consists of reversing the arrows in a diagram, gives rise to new definitions and properties. Applying duality to the categorical notion of algebra leads to the notion of *coalgebra.*

Definition 20 (**F-coalgebra**)**.** Let $F\colon \mathcal{C} \to \mathcal{C}$ be a functor from a category $\mathcal{C}$ to itself. An *F-coalgebra* is a pair (A, α) consisting of an object A and an arrow

$$\alpha\colon A \to F(A)$$

We call F the *type*, A the *carrier*, and α the *structure map* of the coalgebra (A, α). □

Coalgebras are like algebras, but with the structure map reversed. Formally, (A, α) is an F-algebra in the category $\mathcal{C}$ iff it is an F-coalgebra in the category $\mathcal{C}^{op}$. For a first example of a coalgebra, we recall the functor

$$\mathsf{Str}\colon \mathsf{Set} \to \mathsf{Set} \qquad\qquad \mathsf{Str}(X) = \mathbb{N} \times X$$

from Example 7. Let $\mathbb{N}^\omega$ be the set of all *streams* – that is, infinite sequences – of natural numbers:

$$\mathbb{N}^\omega \;=\; \{\, \sigma \mid \sigma\colon \mathbb{N} \to \mathbb{N} \,\}$$

If we define functions

$$\begin{array}{ll} \mathsf{head}\colon \mathbb{N}^\omega \to \mathbb{N} & \mathsf{tail}\colon \mathbb{N}^\omega \to \mathbb{N}^\omega \\ \mathsf{head}(\sigma) \;=\; \sigma(0) & \mathsf{tail}(\sigma) \;=\; (\sigma(1), \sigma(2), \sigma(3), \ldots) \end{array}$$

for all $\sigma = (\sigma(0), \sigma(1), \sigma(2), \ldots) \in \mathbb{N}^\omega$, then pairing these functions together

$$\langle \mathsf{head}, \mathsf{tail} \rangle \colon \mathbb{N}^\omega \to (\mathbb{N} \times \mathbb{N}^\omega)$$

makes $(\mathbb{N}^\omega, \langle \mathsf{head}, \mathsf{tail} \rangle)$ an example of a Str-coalgebra.

The structure map of an *F-algebra* can intuitively be seen as a way of *composing* elements. Dually, the structure map α of an *F-coalgebra* (A, α) tells us how to *decompose* or *unfold* the elements of the coalgebra. For instance, the structure map of the coalgebra $(\mathbb{N}^\omega, \langle \mathsf{head}, \mathsf{tail} \rangle)$ above describes how a stream σ can be decomposed into (i) a natural number $\mathsf{head}(\sigma)$, which can be understood as an output of or (partial) observation on σ; and (ii) a (remainder) stream $\mathsf{tail}(\sigma)$. Thinking of coalgebras as dynamical systems and of their elements σ as states, we may view $\mathsf{tail}(\sigma)$ as the state that is reached from σ after one transition step.

We saw that homomorphisms are for algebras what functions are for sets. Similarly, there is the notion of homomorphism of coalgebras. We can understand the behaviour of a coalgebra by describing how it interacts with other coalgebras through homomorphisms.

Definition 21 (coalgebra homomorphism). Let $F\colon \mathcal{C} \to \mathcal{C}$ be a functor and let (A, α) and (B, β) be F-coalgebras. A *homomorphism* from (A, α) to (B, β) is an arrow $f\colon A \to B$ such that

$$\begin{tikzcd} A \arrow[r, "f"] \arrow[d, "\alpha"'] & B \arrow[d, "\beta"] \\ F(A) \arrow[r, "F(f)"'] & F(B) \end{tikzcd}$$

that is, $\beta \circ f = F(f) \circ \alpha$. □

For an example of a coalgebra homomorphism, consider the Str-coalgebra

$$\begin{array}{lll} (\mathbb{N}^\omega \times \mathbb{N}^\omega, \langle \mathsf{o}, \mathsf{tr} \rangle) & \mathsf{o}\colon (\mathbb{N}^\omega \times \mathbb{N}^\omega) \to \mathbb{N} & \mathsf{tr}\colon (\mathbb{N}^\omega \times \mathbb{N}^\omega) \to (\mathbb{N}^\omega \times \mathbb{N}^\omega) \\ & \mathsf{o}(\sigma, \tau) \;=\; \mathsf{head}(\sigma) & \mathsf{tr}(\sigma, \tau) \;=\; (\tau, \mathsf{tail}(\sigma)) \end{array}$$

The function $\mathsf{zip}\colon \mathbb{N}^\omega \times \mathbb{N}^\omega \to \mathbb{N}^\omega$, defined by

$$\mathsf{zip}(\sigma,\tau) \;=\; (\sigma(0),\tau(0),\sigma(1),\tau(1),\sigma(2),\tau(2),\ldots)$$

is an example of a homomorphism of Str-coalgebras:

$$\begin{tikzcd}
\mathbb{N}^\omega \times \mathbb{N}^\omega \arrow[r, "\mathsf{zip}"] \arrow[d, "\langle \mathsf{o}, \mathsf{tr}\rangle"'] & \mathbb{N}^\omega \arrow[d, "\langle \mathsf{head}, \mathsf{tail}\rangle"] \\
\mathbb{N} \times (\mathbb{N}^\omega \times \mathbb{N}^\omega) \arrow[r, "1 \times \mathsf{zip}"'] & \mathbb{N} \times \mathbb{N}^\omega
\end{tikzcd}$$

The commutativity of the diagram above is a consequence of the following equalities:

$$\begin{aligned}
\mathsf{head}(\mathsf{zip}(\sigma,\tau)) &= \mathsf{head}(\sigma) = \mathsf{o}(\sigma,\tau) \\
\mathsf{tail}(\mathsf{zip}(\sigma,\tau)) &= \mathsf{zip}(\tau,\mathsf{tail}(\sigma)) = \mathsf{zip}(\mathsf{tr}(\sigma,\tau))
\end{aligned}$$

Exercise 22. (i) Describe all homomorphisms from $(\mathbb{N}^\omega, \langle \mathsf{head}, \mathsf{tail}\rangle)$ to itself. Describe all homomorphisms from $(\mathbb{N}^\omega, \langle \mathsf{head}, \mathsf{tail}\rangle)$ to $(\mathbb{N}^\omega \times \mathbb{N}^\omega, \langle \mathsf{o}, \mathsf{tr}\rangle)$. (ii) Define a different Str-coalgebra structure on $\mathbb{N}^\omega \times \mathbb{N}^\omega$ such that the function $f\colon \mathbb{N}^\omega \times \mathbb{N}^\omega \to \mathbb{N}^\omega$, defined by $f(\sigma,\tau)(n) = \sigma(n) + \tau(n)$, is a Str-homomorphism. □

Exercise 23. Show that the family of all F-coalgebras and F-homomorphisms forms a category, for any functor $F\colon \mathcal{C} \to \mathcal{C}$. □

3.3 Discussion

As we mentioned in the introduction, coalgebras occur as the dual of algebras in the work of Arbib and Manes, in their category theoretic approach to dynamical systems and automata [AM75, AM82, MA86]. More about algebras and coalgebras can be found in [Rut00c, JR11]. See also the references at the end of Chapter 15 on universal coalgebra.

Chapter 4

Induction and coinduction

We explain how the well-known principle of *induction* can be formulated categorically, in terms of initial algebras and congruences. This will allow us to formulate, dually, a definition and proof principle for coalgebras called *coinduction*, in terms of final coalgebras and bisimulations.

4.1 Inductive and coinductive definitions

Remember the definition of an initial object, from Chapter 2.

Definition 24 (initial algebra). Let $F\colon \mathcal{C} \to \mathcal{C}$ be a functor. An *initial F-algebra* is an F-algebra that is an *initial object* in the category of all F-algebras and F-homomorphisms. In other words, an F-algebra (A, α) is initial if there exists for every F-algebra (B, β) a unique F-homomorphism $f\colon (A, \alpha) \to (B, \beta)$:

$$\begin{tikzcd}
F(A) \arrow[r, dashed, "F(f)"] \arrow[d, "\alpha"'] & F(B) \arrow[d, "\forall\beta"] \\
A \arrow[r, dashed, "\exists! f"'] & B
\end{tikzcd}$$

□

For an example of an initial algebra, we recall from Section 3.1 the functor $N\colon \mathsf{Set} \to \mathsf{Set}$ defined by $N(X) = \mathbf{1} + X$, with $\mathbf{1} = \{*\}$, and the N-algebra

$$(\mathbb{N}, [\mathsf{zero}, \mathsf{succ}]) \qquad \begin{aligned} \mathsf{zero}&\colon \mathbf{1} \to \mathbb{N} \\ \mathsf{zero}(*) &= 0 \end{aligned} \qquad \begin{aligned} \mathsf{succ}&\colon \mathbb{N} \to \mathbb{N} \\ \mathsf{succ}(n) &= n+1 \end{aligned}$$

As it turns out, the N-algebra $(\mathbb{N}, [\mathsf{zero}, \mathsf{succ}])$ is initial; that is, for every N-algebra

$$(T, [z_T, s_T]) \qquad z_T\colon \mathbf{1} \to T \qquad s_T\colon \mathbb{N} \to T$$

there exists a unique F-homomorphism $f\colon (\mathbb{N}, [\mathsf{zero}, \mathsf{succ}]) \to (T, [z_T, s_T])$:

$$\begin{tikzcd}
\mathbf{1} + \mathbb{N} \arrow[r, dashed, "1+f"] \arrow[d, "{[\mathsf{zero}, \mathsf{succ}]}"'] & \mathbf{1} + T \arrow[d, "{\forall[z_T, s_T]}"] \\
\mathbb{N} \arrow[r, dashed, "\exists! f"'] & T
\end{tikzcd}$$

The commutativity of the diagram above is equivalent to the equalities

$$f(0) \;=\; z_T(*) \qquad\qquad f(n+1) \;=\; s_T(f(n)) \tag{4.1}$$

which can be read as a definition of the function f *by induction*.

For an example of such an inductive definition by initiality, consider the N-algebra

$$(\mathbb{N}, [z_{\mathbb{N}}, s_{\mathbb{N}}]) \qquad \begin{array}{l} z_{\mathbb{N}}\colon \mathbf{1} \to \mathbb{N} \\ z_{\mathbb{N}}(*) = 1 \end{array} \qquad \begin{array}{l} s_{\mathbb{N}}\colon \mathbb{N} \to \mathbb{N} \\ s_{\mathbb{N}}(n) = 2 \times n \end{array}$$

By initiality of $(\mathbb{N}, [\mathsf{zero}, \mathsf{succ}])$, there exists a unique homomorphism

$$f\colon (\mathbb{N}, [\mathsf{zero}, \mathsf{succ}]) \to (\mathbb{N}, [z_{\mathbb{N}}, s_{\mathbb{N}}])$$

Because f is a homomorphism of algebras, it satisfies (4.1), yielding

$$f(0) \;=\; 1 \qquad\qquad f(n+1) \;=\; 2 \times f(n)$$

This can be understood as an inductive definition of the function $f\colon \mathbb{N} \to \mathbb{N}$ satisfying $f(n) = 2^n$, for all $n \geqslant 0$.

Exercise 25. Let $\mathbb{N}^{\mathbb{N}} = \{\, \phi \mid \phi\colon \mathbb{N} \to \mathbb{N} \,\}$. Consider the N-algebra

$$(\mathbb{N}^{\mathbb{N}}, [z_{\mathbb{N}^{\mathbb{N}}}, s_{\mathbb{N}^{\mathbb{N}}}]) \qquad \begin{array}{l} z_{\mathbb{N}^{\mathbb{N}}}\colon \mathbf{1} \to \mathbb{N}^{\mathbb{N}} \\ z_{\mathbb{N}^{\mathbb{N}}}(*)(n) \;=\; n \end{array} \qquad \begin{array}{l} s_{\mathbb{N}^{\mathbb{N}}}\colon \mathbb{N}^{\mathbb{N}} \to \mathbb{N}^{\mathbb{N}} \\ s_{\mathbb{N}^{\mathbb{N}}}(\phi)(n) \;=\; \phi(n+1) \end{array}$$

Describe the unique homomorphism from $(\mathbb{N}, [\mathsf{zero}, \mathsf{succ}])$ to $(\mathbb{N}^{\mathbb{N}}, [z_{\mathbb{N}^{\mathbb{N}}}, s_{\mathbb{N}^{\mathbb{N}}}])$. □

Next we recall, dually, the definition of a *final* object, from Chapter 2.

Definition 26 (final coalgebra). Let $F\colon \mathcal{C} \to \mathcal{C}$ be again a functor. A *final F-coalgebra* is an F-coalgebra that is a *final object* in the category of F-coalgebras and F-homomorphisms. In other words, an F-coalgebra (B, α) is final if for every F-coalgebra (A, α) there exists a unique F-homomorphism $f\colon (A, \alpha) \to (B, \beta)$:

$$\begin{tikzcd}
A \arrow[r, dashed, "\exists! f"] \arrow[d, "\forall\alpha"'] & B \arrow[d, "\beta"] \\
F(A) \arrow[r, dashed, "F(f)"'] & F(B)
\end{tikzcd}$$

□

For an example of a final coalgebra, we recall the functor $\mathsf{Str}\colon \mathsf{Set} \to \mathsf{Set}$ defined by $\mathsf{Str}(X) = \mathbb{N} \times X$ from Example 7. In Section 3.2, we saw that the set $\mathbb{N}^\omega = \{\sigma \mid \sigma\colon \mathbb{N} \to \mathbb{N}\}$ of all streams of natural numbers is a Str-coalgebra $(\mathbb{N}^\omega, \langle \mathsf{head}, \mathsf{tail}\rangle)$. As it turns out, this coalgebra is final; that is, for every Str-coalgebra

$$(S, \langle \mathsf{o}_S, \mathsf{tr}_S\rangle) \qquad \mathsf{o}_S\colon S \to \mathbb{N} \qquad \mathsf{tr}_S\colon S \to S$$

there exists a unique F-homomorphism $f\colon (S, \langle \mathsf{o}_S, \mathsf{tr}_S\rangle) \to (\mathbb{N}^\omega, \langle \mathsf{head}, \mathsf{tail}\rangle)$:

$$\begin{tikzcd}
S \arrow[r, dashed, "\exists ! f"] \arrow[d, "\forall\langle \mathsf{o}_S, \mathsf{tr}_S\rangle"'] & \mathbb{N}^\omega \arrow[d, "\langle \mathsf{head}, \mathsf{tail}\rangle"] \\
\mathbb{N} \times S \arrow[r, dashed, "1 \times f"'] & \mathbb{N} \times \mathbb{N}^\omega
\end{tikzcd}$$

The commutativity of the diagram above is equivalent to the equalities

$$\begin{aligned} \mathsf{head}(f(s)) &= \mathsf{o}_S(s) \\ \mathsf{tail}(f(s)) &= f(\mathsf{tr}_S(s)) \end{aligned}$$

which can be read as a *definition* of f by, well, ... , by what we will call *coinduction*! We observe that these equalities define f in terms of o_S and tr_S, which together constitute the coalgebra structure on S. If we put

$$f\colon S \to \mathbb{N}^\omega \qquad f(s)(n) = \mathsf{o}_S(\mathsf{tr}_S^n(s))$$

where $\mathsf{tr}_S^0(s) = s$ and $\mathsf{tr}_S^{n+1}(s) = \mathsf{tr}_S(\mathsf{tr}_S^n(s))$, then one can easily prove that f is the unique function satisfying the equations above.

One of the main goals of the present book is to make the reader familiar with such *coinductive* definitions. For now, it suffices to observe that final coalgebras give rise to the unique existence of a (coalgebra) homomorphism just as, dually, initial algebras give rise to the unique existence of an (algebra) homomorphism. And since in the latter we recognise definitions by induction, we call the former, dually, definitions by coinduction.

Exercise 27. (i) Consider the Str-coalgebra

$$(\mathbb{N}^\omega, \langle \mathsf{o}_e, \mathsf{tr}_e\rangle) \qquad \begin{aligned} &\mathsf{o}_e\colon \mathbb{N}^\omega \to \mathbb{N} \\ &\mathsf{o}_e(\sigma) = \mathsf{head}(\sigma) \end{aligned} \qquad \begin{aligned} &\mathsf{tr}_e\colon \mathbb{N}^\omega \to \mathbb{N}^\omega \\ &\mathsf{tr}_e(\sigma) = \mathsf{tail}(\mathsf{tail}(\sigma)) \end{aligned}$$

Show that the function

$$\mathsf{even}\colon \mathbb{N}^\omega \to \mathbb{N}^\omega \qquad \mathsf{even}(\sigma) = (\sigma(0), \sigma(2), \sigma(4), \ldots)$$

is the unique homomorphism from $(\mathbb{N}^\omega, \langle \mathsf{o}_e, \mathsf{tr}_e\rangle)$ to $(\mathbb{N}^\omega, \langle \mathsf{head}, \mathsf{tail}\rangle)$. (ii) Define another Str-coalgebra structure on $\mathbb{N}^\omega$ such that it induces, by coinduction, the function

$$\mathsf{odd}\colon \mathbb{N}^\omega \to \mathbb{N}^\omega \qquad \mathsf{odd}(\sigma) = (\sigma(1), \sigma(3), \sigma(5), \ldots) \qquad \square$$

Exercise 28. Consider the Str-coalgebra

$$(\mathbb{N}^\omega \times \mathbb{N}, \langle \mathsf{o}_s, \mathsf{tr}_s \rangle) \qquad \begin{array}{lll} \mathsf{o}_s \colon \mathbb{N}^\omega \times \mathbb{N} \to \mathbb{N} & & \mathsf{tr}_s \colon \mathbb{N}^\omega \times \mathbb{N} \to \mathbb{N}^\omega \times \mathbb{N} \\ \mathsf{o}_s(\sigma, n) \; = \; \mathsf{head}(\sigma) + n & & \mathsf{tr}_s(\sigma, n) \; = \; (\mathsf{tail}(\sigma), \mathsf{head}(\sigma) + n) \end{array}$$

Describe the unique homomorphism from $(\mathbb{N}^\omega \times \mathbb{N}, \langle \mathsf{o}_s, \mathsf{tr}_s \rangle)$ to $(\mathbb{N}^\omega, \langle \mathsf{head}, \mathsf{tail} \rangle)$. □

4.2 Proofs by induction and coinduction

We just saw that initial algebras and final coalgebras give rise to inductive and coinductive definitions. Next we formulate corresponding inductive and coinductive *proof* principles.

We all know the principle of *mathematical induction*: for every subset $P \subseteq \mathbb{N}$,

$$\text{if:} \quad P(0) \;\text{ and }\; (\forall n : \; P(n) \implies P(\mathsf{succ}(n))\,) \quad \text{then: } \forall n : \; P(n) \tag{4.2}$$

In order to arrive at a corresponding *coinduction* principle for coalgebras, we will first reformulate (and generalise) the above induction principle (4.2) in the language of category theory, using the notion of *congruence* relation on algebras. By duality, we will then obtain a coinductive proof principle for coalgebras, based on the dual notion of *bisimulation* relation on coalgebras.

Definition 29 (**F-congruence**)**.** Let $F \colon \mathsf{Set} \to \mathsf{Set}$ be a functor and let (S, α) and (T, β) be two F-algebras. A relation $R \subseteq S \times T$ is an F-*congruence* if there exists an F-algebra structure $\gamma \colon F(R) \to R$ such that the projections $\pi_1 \colon R \to S$ and $\pi_2 \colon R \to T$ are F-homomorphisms:

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

□

The restriction to sets in this definition is just for (notational) convenience. The definition and all that follows below can be formulated for arbitrary categories as well.

Theorem 30 (**induction proof principle**)**.** *Every congruence relation $R \subseteq A \times A$ on an initial F-algebra (A, α) contains the diagonal $\Delta = \{(a, a) \mid a \in A\}$ of A:*

$$\Delta \subseteq R$$

Proof: There exists a unique homomorphism $!\colon A \to R$, by initiality of (A, α):

$$\begin{tikzcd}
F(A) \arrow[r, dashed, bend left] \arrow[d, "\alpha"'] & F(R) \arrow[l, bend left] \arrow[r, bend right] \arrow[d, dashed, "\exists"', "\gamma"] & F(A) \arrow[l, dashed, bend right] \arrow[d, "\alpha"] \\
A \arrow[r, dashed, bend left, "!"] & R \arrow[l, bend left, "\pi_1"] \arrow[r, bend right, "\pi_2"'] & A \arrow[l, dashed, bend right, "!"']
\end{tikzcd}$$

Again by initiality, the identity function $1_A\colon A \to A$ is the only F-homomorphism from (A, α) to itself. It follows that $\pi_1 \circ ! = 1_A = \pi_2 \circ !$, which implies $\pi_1(!(a)) = a = \pi_2(!(a))$ and $!(a) = (a, a)$, for all $a \in A$. Thus $\Delta =!(A) \subseteq R$. $\square$

Recall that the natural numbers form an N-algebra $(\mathbb{N}, [\mathsf{zero}, \mathsf{succ}])$, for the functor $N\colon \mathsf{Set} \to \mathsf{Set}$ defined by $N(X) = \mathbf{1} + X$, with $\mathbf{1} = \{*\}$. For the case of the natural numbers, Theorem 30 is equivalent to the principle of mathematical induction, (4.2) above. This follows from the fact that a relation $R \subseteq \mathbb{N} \times \mathbb{N}$ is an N-congruence if and only if $(0, 0) \in R$ and, for all $(n, m) \in \mathbb{N} \times \mathbb{N}$,

$$(n, m) \in R \implies (\mathsf{succ}(n), \mathsf{succ}(m)) \in R$$

Exercise 31. (i) Prove the above characterisation of congruences on the algebra of natural numbers. (ii) Use this fact to show that for the natural numbers, Theorem 30 is equivalent to Equation (4.2). $\square$

The following exercise illustrates that the present notion of congruence does *not* require R to be an equivalence relation.

Exercise 32. Consider the following N-algebra:

$$(X, [f, 1_X]) \qquad X = \{x, y, z\} \qquad f\colon \mathbf{1} \to X \qquad f(*) = x$$

with 1_X the identity function on X. Let

$$R \subseteq X \times X \qquad R = \{ (x, x), (x, y), (y, z) \}$$

Prove that R is an N-congruence and that R is neither reflexive, nor symmetric, nor transitive. $\square$

Since the induction principle above is formulated categorically, in terms of congruences and initial F-algebras, all of the above can now be easily *dualised* for coalgebras.

Definition 33 (*F*-**bisimulation).** Let $F\colon \mathsf{Set} \to \mathsf{Set}$ be a functor and let (S, α) and (T, β) be two F-coalgebras. A relation $R \subseteq S \times T$ is an *F-bisimulation* if there exists an F-coalgebra structure $\gamma\colon R \to F(R)$ such that the projections $\pi_1\colon R \to S$ and $\pi_2\colon R \to T$ are F-homomorphisms:

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

□

The definition o af F-bisimulation is obtained from the definition of a congruence above, that is, by reversing all vertical arrows. Bisimulations are used in the formulation of the following *coinduction* proof principle for coalgebras, which is a dual version of the induction proof principle, Theorem 30 above.

Theorem 34 (coinduction proof principle). *Every bisimulation relation $R \subseteq B \times B$ on a final F-coalgebra (B, β)* is contained in *the diagonal $\Delta = \{(b, b) \mid b \in B\}$ of B:*

$$R \subseteq \Delta$$

Proof: Because both π_1 and π_2 are homomorphisms into the final coalgebra(B, β):

$$\begin{tikzcd}
B \arrow[d, "\beta"'] & R \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] \arrow[d, dashed, "\exists \mid \gamma"] & B \arrow[d, "\beta"] \\
F(B) & F(R) \arrow[l, "F(\pi_1)"] \arrow[r, "F(\pi_2)"'] & F(B)
\end{tikzcd}$$

they are equal: $\pi_1 = \pi_2$. This implies $R \subseteq \Delta$. □

We mentioned above that one of the main goals of the present book is to make the reader familiar with coinductive definitions. Another goal, equally important, is to instruct the reader in the art of *coinductive proofs*. As a first example – many others will follow in the subsequent chapters – we discuss next what bisimulations and coinduction look like for the final Str-coalgebra of streams.

It is a little exercise to prove that a relation $R \subseteq \mathbb{N}^\omega \times \mathbb{N}^\omega$ on $(\mathbb{N}^\omega, \langle \mathsf{head}, \mathsf{tail} \rangle)$ is a Str-bisimulation if and only if, for all $(\sigma, \tau) \in R$,

$$(i)\ \ \mathsf{head}(\sigma) = \mathsf{head}(\tau) \quad \text{and} \quad (ii)\ \ (\mathsf{tail}(\sigma), \mathsf{tail}(\tau)) \in R \tag{4.3}$$

For an example of a proof by coinduction, consider the functions

$$\begin{aligned} &\mathsf{zip}\colon \mathbb{N}^\omega \times \mathbb{N}^\omega \to \mathbb{N}^\omega && \mathsf{even}\colon \mathbb{N}^\omega \to \mathbb{N}^\omega \\ &\mathsf{zip}(\sigma,\tau) \;=\; (\sigma(0),\tau(0),\sigma(1),\tau(1),\ldots) && \mathsf{even}(\sigma) \;=\; (\sigma(0),\sigma(2),\sigma(4),\ldots) \end{aligned}$$

We will prove the following equality by coinduction: for all $\alpha, \beta \in \mathbb{N}^\omega$,

$$\mathsf{even}(\mathsf{zip}(\alpha,\beta)) \;=\; \alpha \tag{4.4}$$

In order to apply coinduction Theorem 34, we define

$$R \subseteq \mathbb{N}^\omega \times \mathbb{N}^\omega \qquad\qquad R \;=\; \{\, (\mathsf{even}(\mathsf{zip}(\alpha,\beta)),\, \alpha) \mid \alpha,\beta \in \mathbb{N}^\omega \,\}$$

The relation R contains for every α and β the pair of streams $\mathsf{even}(\mathsf{zip}(\alpha,\beta))$ and α that we want to prove equal. We show that R is a Str-bisimulation, by verifying (i) and (ii) of (4.3). Since

$$\mathsf{head}(\mathsf{even}(\sigma)) \;=\; \mathsf{head}(\sigma) \qquad\qquad \mathsf{head}(\mathsf{zip}(\sigma,\tau)) \;=\; \mathsf{head}(\sigma)$$

for all σ, τ, it follows that

$$\mathsf{head}(\mathsf{even}(\mathsf{zip}(\alpha,\beta))) \;=\; \mathsf{head}(\alpha)$$

for all α, β, which proves (i). For (ii), we use

$$\mathsf{tail}(\mathsf{even}(\sigma)) \;=\; \mathsf{even}(\mathsf{tail}(\mathsf{tail}(\sigma))) \qquad\qquad \mathsf{tail}(\mathsf{zip}(\sigma,\tau)) \;=\; \mathsf{zip}(\tau,\mathsf{tail}(\sigma))$$

for all σ, τ. Then

$$\begin{aligned} \mathsf{tail}(\mathsf{even}(\mathsf{zip}(\alpha,\beta))) &= \mathsf{even}(\mathsf{tail}(\mathsf{tail}(\mathsf{zip}(\alpha,\beta)))) \\ &= \mathsf{even}(\mathsf{tail}(\mathsf{zip}(\beta,\mathsf{tail}(\alpha)))) \\ &= \mathsf{even}(\mathsf{zip}(\mathsf{tail}(\alpha),\mathsf{tail}(\beta))) \end{aligned}$$

for all α, β, which implies that

$$(\mathsf{tail}(\mathsf{even}(\mathsf{zip}(\alpha,\beta))), \mathsf{tail}(\alpha)) \;=\; (\mathsf{even}(\mathsf{zip}(\mathsf{tail}(\alpha),\mathsf{tail}(\beta))), \mathsf{tail}(\alpha)) \in R$$

This proves that R satisfies (ii). Equality (4.4) now follows from Theorem 34.

Exercise 35. Prove by coinduction that $\mathsf{odd}(\mathsf{zip}(\alpha,\beta)) = \beta$, for all $\alpha, \beta \in \mathbb{N}^\omega$, where the function $\mathsf{odd}\colon \mathbb{N}^\omega \to \mathbb{N}^\omega$ is defined by $\mathsf{odd}(\sigma) = (\sigma(1), \sigma(3), \sigma(5), \ldots)$. □

Exercise 36 (fixed point (co)induction). The following classical forms of least fixed point induction and greatest fixed point coinduction turn out to be special instances of the algebraic and coalgebraic principles discussed above. Let $(P, \leqslant)$ be a pre-ordered set and let $f\colon P \to P$ be a monotone function. We have the following notions, for $p \in P$:

- p is a *pre-fixed point* of f if $f(p) \leqslant p$
- p is a *post-fixed point* of f if $p \leqslant f(p)$
- p is a *fixed point* of f if $p = f(p)$
- p is a *least fixed point* of f if $p = f(p)$ and $p \leqslant q$, for all $q \in P$ with $q = f(q)$
- p is a *greatest fixed point* of f if $p = f(p)$ and $q \leqslant p$, for all $q \in P$ with $q = f(q)$

We recall from Exercise 2 that a pre-ordered set $(P, \leqslant)$ can be viewed as a category, with as objects the elements p, q in P and with arrows $p \to q$ whenever $p \leqslant q$. In Exercise 9, we moreover saw that any function $f\colon P \to P$ that is *monotone*:

$$p \leqslant q \quad \Longrightarrow \quad f(p) \leqslant f(q)$$

is a *functor* $f\colon (P, \leqslant) \to (P, \leqslant)$ from $(P, \leqslant)$, viewed as category, to itself. Now observe the following: f-algebras are pre-fixed points of f, and f-coalgebras are post-fixed points of f.

Next let X be an arbitrary set and consider the following pre-ordered set:

$$(P, \leqslant) \qquad P = \mathcal{P}(X) \qquad V \leqslant W \iff V \subseteq W \qquad (V, W \in P)$$

(The pre-ordered set $(P, \leqslant)$ is in fact a so-called *complete lattice*.) Let $f\colon (P, \leqslant) \to (P, \leqslant)$ be a monotone function. We define:

$$\mu f = \bigcap \{ V \in P \mid f(V) \leq V \} \qquad \nu f = \bigcup \{ V \in P \mid V \leq f(V) \}$$

One can prove that μf is the unique least fixed point of f, and νf is the unique greatest fixed point of f. This is known as the Knaster - Tarski Theorem. Show that as a consequence, we have the following induction and coinduction proof principles for monotone functions:

$$f(V) \leqslant V \implies \mu f \leqslant V \qquad (\text{all } V \in P)$$
$$V \leqslant f(V) \implies V \leqslant \nu f \qquad (\text{all } V \in P)$$

Prove that these proof principles are equivalent to the statements that μf is an initial f-algebra and νf is a final f-coalgebra. □

4.3 Discussion

We recall from the introduction that the categorical notion of bisimulation relation for coalgebras goes back to Aczel and Mendler [AM89], as a generalisation of Park and Milner's notion of bisimulation [Mil80, Par81, Mil89] from the theory of parallel processes. Our presentation of the duality between induction and coinduction is based on [Rut00c]. For an introduction to bisimulation and the classical fixed point (co)induction discussed in Exercise 36, see [San12].

Chapter 5

The method of coalgebra

In a nutshell:

- The study of any class of coalgebras begins with the definition of its type, which is a functor $F\colon \mathsf{Set} \to \mathsf{Set}$ on the category of sets and functions. (However, other categories may be used as well.) An F-coalgebra is then a pair (S, α) consisting of a carrier set S and a structure map $\alpha\colon S \to F(S)$. We typically think of S as the set of internal states of the coalgebra. Throughout this book, we will often call coalgebras also by other names: coalgebra = system = automaton.

- We describe what coalgebras do rather than what they are. The basis of the behaviour of a coalgebra (S, α) is its structure map $\alpha\colon S \to F(S)$, which defines, for each state in S, the local dynamics and outputs or observations, possibly depending on inputs. The global behaviour of a coalgebra (S, α) is then given by its interaction with other coalgebras, that is, by homomorphisms between (S, α) and other coalgebras. The definition of homomorphism relies on the fact that the type F is a functor, acting not only on sets but also on functions.

- Many classes of F-coalgebras will contain a final F-coalgebra, which is characterised by the following universal property: for any coalgebra (S, α), there exists a unique homomorphism to the final one. This homomorphism assigns to every state a canonical representation of its global behaviour.

- The structure map of an initial algebra tells us how to compose or construct elements. Dually, the structure map of a final coalgebra specifies how elements can be decomposed or unfolded.

- Homomorphisms are structure-preserving functions. Similarly, bisimulations are structure-preserving relations. Bisimulations enlarge our vocabulary for describing the interaction between coalgebras. Notably, bisimulations are used in the formulation of the coinduction proof principle: if two states of a final coalgebra are related by some bisimulation relation, then they are equal.

5.1 Basic types of coalgebras

The type, that is, functor $F\colon \mathsf{Set} \to \mathsf{Set}$, of every class of coalgebras is constructed out of a small number of elementary functors. Here are four of the most relevant examples:

$$\begin{tikzcd} S \arrow[d, "\alpha"] & S \arrow[d, "\beta"] & S \arrow[d, "\gamma"] & S \arrow[d, "\delta"] \\ A & S & S^A & \mathcal{P}(S) \end{tikzcd}$$

corresponding to the following four possible choices for the functor F: $F(S) = A$, $F(S) = S$, $F(S) = S^A = \{f \mid f\colon A \to S\}$, and $F(S) = \mathcal{P}(S) = \{V \mid V \subseteq S\}$, respectively.

In the first example, A is an arbitrary but fixed set, and the structure map $\alpha\colon S \to A$ assigns to every state $s \in S$ an element $\alpha(s) \in A$, which we may think of as an *output* value produced by s or as a (partial) *observation* on s.

In the second example, the structure map $\beta\colon S \to S$ is a *transition* function, which maps a state $s \in S$ to a *successor* state $\beta(s) \in S$.

In the third example, we think of the elements of A as *inputs*. The structure map $\gamma\colon S \to S^A$ assigns to every state $s \in S$ a function $\gamma(s)\colon A \to S$, which in turn assigns to every input $a \in A$ a new state $\gamma(s)(a) \in S$, called the successor of s *on input* a.

Finally, in the fourth example, the structure map $\delta\colon S \to \mathcal{P}(S)$ assigns to every state $s \in S$ a *set of states* $\delta(s) \subseteq S$, which we may think of as the collection of possible next states out of which the system may choose one *non-deterministically*.

Coalgebras of more complex type are then obtained by combining (basic) types using operators such as (Cartesian) product $\times$ and coproduct (disjoint union) $+$.

5.2 Coalgebras, systems, automata ...

The types of coalgebra that we will study in this book include the following:

$\mathsf{Id}\colon \mathsf{Set} \to \mathsf{Set}$	$\mathsf{Id}(S) = S$	(dynamical systems)
$\mathsf{Str}\colon \mathsf{Set} \to \mathsf{Set}$	$\mathsf{Str}(S) = A \times S$	(stream systems)
$\mathsf{dA}\colon \mathsf{Set} \to \mathsf{Set}$	$\mathsf{dA}(S) = 2 \times S^A$	(deterministic automata)
$\mathsf{pA}\colon \mathsf{Set} \to \mathsf{Set}$	$\mathsf{pA}(S) = 2 \times (1+S)^A$	(partial automata)
$\mathsf{ndA}\colon \mathsf{Set} \to \mathsf{Set}$	$\mathsf{ndA}(S) = 2 \times \mathcal{P}(S)^A$	(non-deterministic automata)
$\mathsf{MA}\colon \mathsf{Set} \to \mathsf{Set}$	$\mathsf{MA}(S) = (B \times S)^A$	(Mealy automata)
$\mathsf{MoA}\colon \mathsf{Set} \to \mathsf{Set}$	$\mathsf{MoA}(S) = B \times S^A$	(Moore automata)
$\mathsf{wsA}\colon \mathsf{Set} \to \mathsf{Set}$	$\mathsf{wsA}(S) = \mathbb{R} \times \mathbb{R}^S_\omega$	(weighted stream automata)

(How these functors act on functions is explained in Section 16.10.) The examples above illustrate that coalgebras are state-based dynamical systems in which states can make transitions to successor states, possibly in reaction to a certain input. Furthermore, states may produce an output, which often means: states may be (partially) observed. It is the *type* F of a coalgebra that tells us precisely what this combination of dynamics, inputs, and outputs/observations looks like.

Chapter 6

Dynamical systems

Dynamical systems form an elementary class of coalgebras, which is well suited for a first introduction to the notion of a *homomorphism*. Formally, a dynamical system is a pair (S, α) consisting of a set S, of *states*, and a function

$$\alpha \colon S \to S$$

called the *transition function*.

Remark 37. For those who have already read Chapters 2 to 5: dynamical systems are Id-coalgebras of the identity functor $\mathsf{Id} \colon \mathsf{Set} \to \mathsf{Set}$, which is defined for sets S by $\mathsf{Id}(S) = S$ and for functions $f \colon S \to T$ by $\mathsf{Id}(f) = f \colon S \to T$. $\square$

We shall often present dynamical systems pictorially, using the following notation:

$$s \longrightarrow s' \quad \Longleftrightarrow \quad \alpha(s) \; = \; s'$$

We call s' the *successor* of s and s a *predecessor* of s', and we call the arrow a *transition* from s to s'. Here is an example of a finite dynamical system:

$$\begin{tikzcd}
 & s_2 \arrow[d] & & & & \\
s_1 \arrow[r] & s_3 \arrow[r] & s_4 \arrow[d] & s_6 \arrow[l] & s_7 \arrow[r] & s_8 \arrow[loop right] \\
 & & s_5 \arrow[ru] & & &
\end{tikzcd}$$

In this example, every state has exactly one successor; some but not all states have (one or more) predecessors; and all states eventually end in a loop or cycle. The latter does not always happen, as the following example of an infinite dynamical system shows:

$$(\mathbb{N}, \mathsf{succ}) \;=\; 0 \longrightarrow 1 \longrightarrow 2 \longrightarrow 3 \longrightarrow \cdots \tag{6.1}$$

Here the successor function $\mathsf{succ}\colon \mathbb{N} \to \mathbb{N}$ is given, for all $n \geqslant 0$, by

$$\mathsf{succ}(n) \ = \ n+1$$

The next example shows that transition sequences can also be "infinite to the left":

$$(\overline{\mathbb{N}}, \mathsf{pre}) \ = \ \begin{tikzcd} \omega \arrow[loop left] & \cdots \arrow[r] & 3 \arrow[r] & 2 \arrow[r] & 1 \arrow[r] & 0 \arrow[loop right] \end{tikzcd} \qquad (6.2)$$

In this example, $\overline{\mathbb{N}} = \mathbb{N} \cup \{\omega\}$, and the function $\mathsf{pre}\colon \overline{\mathbb{N}} \to \overline{\mathbb{N}}$ is defined by

$$\mathsf{pre}(\omega) \ = \ \omega \qquad\qquad \mathsf{pre}(n+1) \ = \ n \qquad\qquad \mathsf{pre}(0) \ = \ 0$$

We call a state $s \in S$ of a dynamical system (S, α) a *fixed point* whenever $s = \alpha(s)$. More generally, we say that s lies on an n-cycle, for $n \geqslant 1$, whenever $s = \alpha^n(s)$, where $\alpha^0 = 1_S$, the identity function on S, and $\alpha^{n+1} = \alpha \circ \alpha^n$. We also say that s is *periodic* with period n.

6.1 Homomorphisms of dynamical systems

Let us first briefly recall some elementary facts about sets. Any two sets S and T can be conveniently related by *functions* $f\colon S \to T$. We are all familiar with the following three situations: If f is *injective*:

$$\forall\, s, s' \in S : \ \ f(s) \ = \ f(s') \quad \Longrightarrow \quad s \ = \ s'$$

then f can be viewed as an *embedding* of S into T. If f is *surjective*:

$$\forall\, t \in T \ \exists\, s \in S : \ \ f(s) \ = \ t$$

then we call T a *quotient* of S. In that case, every $t \in T$ can be viewed as a representative for all the elements in the inverse image $f^{-1}(t)$, which is given by

$$f^{-1}(t) \ = \ \{s \in S \mid f(s) = t\}$$

And if f is *bijective*:

$$f \text{ is both injective and surjective}$$

then one can view f as a *renaming* of the elements in S with names from T. In that case, the names of the elements of S and T may be different but other than that, S and T can be regarded as essentially the same. If there exists a bijective function, or *bijection* between S and T, then one often writes $S \cong T$.

Now (coalgebra) *homomorphisms* are for dynamical systems what functions are for sets.

Definition 38 (homomorphism of dynamical systems).
A *homomorphism* $f\colon (S,\alpha) \to (T,\beta)$ of dynamical systems is a function $f\colon S \to T$ such that

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

that is, such that $\beta \circ f = f \circ \alpha$. □

Equivalently, f is a homomorphism if and only if

$$\forall\, s \in S: \quad s \longrightarrow s' \implies f(s) \longrightarrow f(s')$$

Thus a homomorphism between two dynamical systems is a *function* between the underlying sets of states that *preserves* transitions. In other words, homomorphisms are (transition) *structure preserving* functions.

Exercise 39. Let S be a set. Prove that the identity function $1_S\colon S \to S$ is a homomorphism $1_S\colon (S,\alpha) \to (S,\alpha)$ of dynamical systems, for any $\alpha\colon S \to S$. Show that if f and g are homomorphisms that can be composed then also $g \circ f$ is a homomorphism. For any homomorphism $f\colon (S,\alpha) \to (T,\beta)$, show that if $f\colon S \to T$ is bijective (as a function) then also $f^{-1}\colon T \to S$ is a homomorphism $f^{-1}\colon (T,\beta) \to (S,\alpha)$. In that case, we call f an *isomorphism* and we sometimes write $f\colon (S,\alpha) \cong (T,\beta)$. □

In Exercise 4, we saw the equivalence of the notions of injective function and monomorphism, and of surjective function and epimorpism. The next exercise contains the same statements, now for homomorphisms of dynamical systems instead of functions.

Exercise 40 (monomorphism, epimorphism). First do Exercise 4. Then let f be a homomorphism of dynamical systems. We call f a *monomorphism* of dynamical systems if for all (with f composable) homomorphisms g and h,

$$f \circ g \;=\; f \circ h \implies g \;=\; h$$

Prove that f is a monomorphism of dynamical systems if and only if f (as a function) is injective. Dually, we call f is an *epimorphism* of dynamical systems if for all (with f composable) homomorphisms g and h,

$$g \circ f \;=\; h \circ f \implies g \;=\; h$$

Prove that f is an epimorphism of dynamical systems if and only if f (as a function) is surjective. □

Exercise 41. Show that any homomorphism $f\colon (S,\alpha) \to (T,\beta)$ of dynamical systems can be written as the composition of an epimorphism and a monomorphism. □

One way of seeing that the notion of homomorphism of dynamical systems makes sense, is to note that isomorphic dynamical systems

$$(S,\alpha) \;\cong\; (T,\beta)$$

have essentially the same *dynamics*: they have not only the same number of states but also equal numbers of fixed points, the same number of cycles of length seven, equal numbers of states that move nine steps before entering a cycle of length four, etc. etc.

Exercise 42. Let $f\colon (S,\alpha) \to (T,\beta)$ be a homomorphism. Show that if $s \in S$ is part of a four cycle: $\alpha^4(s) = s$ then so is $f(s)$. Under what conditions does the converse hold? □

Functions between sets and homomorphisms between dynamical systems are different in various ways. For instance, if S and T are sets and T is nonempty then there always exists a function from S to T.

Exercise 43. Show that there exist no homomorphisms (in either direction) between the following two dynamical systems:

$$(S,\alpha) \;=\; \begin{tikzcd} s_0 \arrow[r] & s_1 \arrow[l, bend right] \end{tikzcd} \qquad\qquad (T,\beta) \;=\; \begin{tikzcd} t_0 \arrow[r] & t_1 \arrow[r] & t_2 \arrow[ll, bend right] \end{tikzcd}$$

Next, let (R,γ) be a dynamical system and suppose that there exist homomorphisms $f\colon (R,\gamma) \to (S,\alpha)$ and $g\colon (R,\gamma) \to (T,\beta)$. How many elements must R contain at least? And is there an upperbound to the size of R? Same questions, but now for

$$(S,\alpha) \;=\; \begin{tikzcd} s_0 \arrow[r] & s_1 \arrow[l, bend right] \end{tikzcd} \qquad\qquad (T,\beta) \;=\; \begin{tikzcd} t_0 \arrow[r] & t_1 \arrow[r] & t_2 \arrow[r] & t_3 \arrow[lll, bend right] \end{tikzcd}$$

□

Here is slightly more subtle example of a difference between functions and homomorphisms. If a function of sets $f\colon S \to T$ is injective and S is nonempty then there exists a function $g\colon T \to S$ with $g \circ f = 1_S$, where 1_S is the identity function on S. Such a function g is sometimes called a retraction or left-inverse of f.

In contrast, not every injective *homomorphism* $f\colon (S,\alpha) \to (T,\beta)$ has a retraction, that is, a *homomorphism* $g\colon (T,\beta) \to (S,\alpha)$ such that $g \circ f = 1_S$.

Exercise 44. Consider the following two dynamical systems:

$$(S,\alpha) \;=\; \begin{tikzcd} s_1 \arrow[r] & s_0 \arrow[loop right] \end{tikzcd} \qquad\qquad (T,\beta) \;=\; \begin{tikzcd} t_2 \arrow[r] & t_1 \arrow[r] & t_0 \arrow[loop right] \end{tikzcd}$$

and define $f\colon S \to T$ by $f(s_1) = t_1$ and $f(s_0) = t_0$. Show that f is a monomomorphism. Show that $f\colon S \to T$, as a function between sets, has two retractions. Show that $f\colon (S,\alpha) \to (T,\beta)$, as a homomorphism between dynamical systems, has none. □

6.2 On the behaviour of dynamical systems

An essential ingredient of the coalgebraic method is to consider systems as *black boxes*, with internal states that are not observable from the outside. Rather than asking what (the states of) coalgebras *are*, we will ask instead what coalgebras *do*. And what coalgebras do is determined by how they *interact* with other coalgebras or, more formally, by the *homomorphisms between* coalgebras. As a consequence, we will study the behaviour of coalgebras by investigating their homomorphisms.

Using homomorphisms, one can for instance express the fact that *all* dynamical systems are equivalent, in the following sense. Consider the following dynamical system:

$$(\mathbf{1}, id) \qquad \mathbf{1} = \{*\} \qquad id\colon \mathbf{1} \to \mathbf{1} \qquad id(*) \; = \; *$$

There is the following trivial fact: there exists a unique homomorphism from any dynamical system (S, α) to $(\mathbf{1}, id)$:

$$\begin{tikzcd} S \arrow[r, "\exists!\, f"] \arrow[d, "\alpha"'] & \mathbf{1} \arrow[d, "id"] \\ S \arrow[r, "f"'] & \mathbf{1} \end{tikzcd} \qquad f(s) \; = \; * \qquad (s \in S)$$

Because of this universal property, $(\mathbf{1}, id)$ is called a *final* dynamical system. We will see more interesting examples of final coalgebras shortly, in the next chapter. However, the fact that $(\mathbf{1}, id)$ is final *does* serve to illustrate a point, namely, that any state in any dynamical system essentially behaves the same as the single state $*$ in $(\mathbf{1}, id)$: all a state can do is to make a transition to a next state, which in turn can make a transition to a next state, and so on.

We mentioned above that in the coalgebraic method, systems are black boxes and states are considered to be not observable. What was meant, more precisely, is that the states of a coalgebra are not observable *unless the type of the coalgebra explicitly allows us to do so.* With the present definition of dynamical systems, we cannot make any observations on the states of the system. In the next chapter, we will study dynamical systems that have not only a transition function but also an *output* (or observation) function $\mathsf{o}\colon S \to A$, where A is a fixed set, indentical for all systems. For instance, if $A = \{0, 1\}$ then the function o can be used to signal for each state whether it satisfies a certain property or not. These observations then become part of the behaviour of the system, and as a consequence, systems will no longer all be equivalent. This will be reflected by the fact that the family of all such dynamical systems with observations has a non-trivial final coalgebra.

For now, we stay with our present definition of dynamical systems, without observations or inputs. In the remainder of this chapter, we will illustrate how one can learn more about (the internal structure of) a dynamical system (S, α) by probing it *from the outside* with homomorphisms.

For a first and simple example, there is a bijective correspondence between the set of all possible homomorphisms $(\mathbf{1}, id) \to (S, \alpha)$ and the set of all fixed points of (S, α).

Exercise 45. Let $n \geqslant 1$ and consider the following dynamical system:

$$(C_n, \gamma_n) \;=\; \begin{tikzcd} 0 \arrow[r] & 1 \arrow[r] & 2 \arrow[r] & \cdots \arrow[r] & n-1 \arrow[llll, bend right] \end{tikzcd} \tag{6.3}$$

Let (S, α) be an arbitrary dynamical system. For $s \in S$ show that $\alpha^n(s) = s$ iff there exists $f \in H$ with $f(0) = s$, where H is the following set of homomorphisms:

$$H \;=\; \{ f \mid f \colon (C_n, \gamma_n) \to (S, \alpha) \}$$

Show that H is isomorphic to the set of all points of (S, α) of period n. □

Exercise 46. Let (S, α) be a dynamical system such that for all $s \in S$ there exists a homomorphism

$$\begin{tikzcd} 1 \arrow[r] & 0 \arrow[loop right] & \arrow[r, "f"] & {} & (S, \alpha) \end{tikzcd}$$

with $s = f(1)$. What equation does α satisfy? □

Exercise 47. Let (S, α) be a dynamical system and consider the set of homomorphisms

$$H \;=\; \{ f \mid f \colon (\mathbb{N}, \mathsf{succ}) \to (S, \alpha) \}$$

(with $(\mathbb{N}, \mathsf{succ})$ as defined in (6.1)). Show that $S \cong H$. □

Definition 48 (subsystem of dynamical system). Let (S, α) be a dynamical system and let $T \subseteq S$. We say that T is *closed under transitions* if $\alpha(s) \in T$ for all $s \in T$. In that case, we call (T, β) a *subsystem* of (S, α), where β is the restriction of α to T. □

Exercise 49. Consider the following dynamical system:

$$\begin{tikzcd} & s_2 \arrow[d] & & & & \\ s_1 \arrow[r] & s_3 \arrow[r] & s_4 \arrow[d] & s_6 \arrow[l] & s_7 \arrow[r] & s_8 \arrow[loop right] \\ & & s_5 \arrow[ur] & & & \end{tikzcd}$$

How many different subsystems does this system have? □

Exercise 50. Consider two dynamical systems (S, α) and (T, β), and suppose T is a subset of S. Let $i\colon T \to S$ with $i(t) = t$, for all $t \in T$. Show that (T, β) is a subsystem of (S, α) iff i is a homomorphism. □

Let (T, β) be a subsystem of (S, α). For a state $s \in S$ there are three possibilities:

(i) s is a state of the subsystem (T, β);

(ii) s is not a state of the subsystem (T, β) but a finite number of transitions starting in s will lead to a state that *is* in (T, β);

(iii) s is not a state of the subsystem (T, β) and no finite number of transitions starting in s will ever lead to a state that is in (T, β)

With the above in mind, we can prove the following. There is a one-to-one correspondence between subsystems of (S, α), on the one hand, and homomorphisms

$$f\colon\ (S, \alpha) \to\ (\overline{\mathbb{N}}, \mathsf{pre})$$

on the other hand (with $(\overline{\mathbb{N}}, \mathsf{pre})$ as defined in (6.2)). For the proof in one direction, consider a subsystem (T, β) of (S, α). We define a mapping $f\colon S \to \overline{\mathbb{N}}$, for $s \in S$, as follows:

(i) if s satisfies (i) above then we put $f(s) = 0$;

(ii) if s satisfies (ii) above then we define $f(s)$ as the minimal number of transitions starting in s that are needed to reach T;

(iii) finally, if s satisfies (iii) above then we put $f(s) = \omega$.

Exercise 51. Show that f defined in this manner is a homomorphism. Also prove the other half of the statement: every homomorphism

$$f\colon\ (S, \alpha) \to\ (\overline{\mathbb{N}}, \mathsf{pre})$$

uniquely corresponds to a subsystem of (S, α). □

One can view the values in $\overline{\mathbb{N}}$ as a kind of *generalised truth values*, where 0 denotes truth; ω denotes (permanent) false; and any natural number denotes the *distance* to truth or, in terms of condition (ii) above, the number of steps it takes before truth is reached. The next exercise deals with yet another set of generalised truth values, sometimes called the *chaotic truth values*.

Exercise 52. Let $2 = \{0, 1\}$ and define the set of all *bitstreams*, that is, infinite sequences of 0's and 1's, by

$$2^\omega \;=\; \{(b_0, b_1, b_2, \ldots) \mid b_i \in 2\}$$

Consider the transition function

$$t\colon 2^\omega \to 2^\omega \qquad\qquad t((b_0, b_1, b_2, \ldots)) \;=\; (b_1, b_2, b_3, \ldots)$$

Let (S, α) be a dynamical system. Prove that there is a one-to-one correspondence between homomorphisms

$$f\colon (S, \alpha) \to (2^\omega, t)$$

on the one hand, and sub*sets* $V \subseteq S$, on the other. □

6.3 Discussion

Our explanation of the notion of homomorphisms of dynamical systems has been inspired by the discussion of *structure preserving maps* in [LS97]. Exercise 44 above is taken from [LS97, p.147]. Our discussion of truth values is based on [LS97, Session 32-3].

Chapter 7

Stream systems

Streams over a given set A are infinite sequences

$$(a_0, a_1, a_2, \ldots) \qquad (a_i \in A)$$

of elements in A. Streams will arise as the behaviour of *stream systems*, which are thus coalgebraic representations of streams, and which are formally defined as follows. A *stream system with outputs in a set* A is a pair

$$(S, \langle \mathsf{o}, \mathsf{tr} \rangle)$$

consisting of a set S, of *states*, and a *structure map*

$$\langle \mathsf{o}, \mathsf{tr} \rangle \colon S \to A \times S \qquad \mathsf{o} \colon S \to A \qquad \mathsf{tr} \colon S \to S$$

which is a pair consisting of an *output* function o and a *transition* function tr.

In other words, a stream system $(S, \langle \mathsf{o}, \mathsf{tr} \rangle)$ is a dynamical system (S, tr) together with an additional output function o. In many situations, the output value $\mathsf{o}(s) \in A$ of a state $s \in S$ contains information about s, thus making it partially *observable* to someone looking at the system from the outside.

The set A^ω of all streams will turn out to be a *final* stream system, giving rise to both definitions and proofs by the coalgebraic principle of *coinduction*. Because stream systems have both transitions and outputs, the notion of *bisimulation relation* becomes highly relevant, allowing us to compare and reduce stream systems. Furthermore, coinductive proofs will consist of the construction of suitable bisimulation relations.

All in all, streams have in recent years become for coalgebra and coinduction what the natural numbers are for algebra and induction: a canonical example.

Remark 53. For those who have already read Chapters 2 to 5: stream systems are coalgebras of the functor

$$\begin{array}{lll} \mathsf{Str} \colon \mathsf{Set} \to \mathsf{Set} & \mathsf{Str}(S) \ = \ A \times S & (S \in \mathsf{Set}) \\ \mathsf{Str}(f) \colon A \times S \to A \times T & \mathsf{Str}(f) \ = \ 1_A \times f & (f \colon S \to T) \end{array}$$

with

$$(1_A \times f)((a,s)) \;=\; (a, f(s))$$

for $a \in A$ and $s \in S$. □

7.1 Homomorphisms and bisimulations of stream systems

Homomorphisms are structure preserving functions, which here means functions that preserve both transitions and outputs.

Definition 54 (homomorphism of stream systems).
A *homomorphism* $f\colon (S, \langle \mathsf{o}_S, \mathsf{tr}_S\rangle) \to (T, \langle \mathsf{o}_T, \mathsf{tr}_T\rangle)$ of stream systems is a function $f\colon S \to T$ such that

$$\begin{tikzcd} S \arrow[r, "f"] \arrow[d, "\langle \mathsf{o}_S{,}\mathsf{tr}_S\rangle"'] & T \arrow[d, "\langle \mathsf{o}_T{,}\mathsf{tr}_T\rangle"] \\ A \times S \arrow[r, "1_A \times f"'] & A \times T \end{tikzcd}$$

that is, such that $\mathsf{o}_T \circ f = \mathsf{o}_S$ and $\mathsf{tr}_T \circ f = f \circ \mathsf{tr}_S$. □

Homomorphisms are functions that preserve transitions and outputs:

$$s|a \longrightarrow t|b \quad\Longrightarrow\quad f(s)|a \longrightarrow f(t)|b \tag{7.1}$$

where we are using the following notation:

$$s|a \longrightarrow t|b \quad\Longleftrightarrow\quad \mathsf{o}_S(s) = a \;\text{ and }\; \mathsf{tr}_S(s) = t \;\text{ and }\; \mathsf{o}_S(t) = b$$

Exercise 55. Prove implication (7.1). □

Example 56. Here is an example of a finite stream system:

$$\begin{tikzcd} s_0|a \arrow[r] & s_1|b \arrow[r] & s_2|c \arrow[r] & s_3|d \arrow[r] & s_4|e \arrow[r] & s_5|f \arrow[lll, bend right] \end{tikzcd}$$

where $A = \{a,b,c,d,e,f\}$. Similar to dynamical systems, all finite stream systems have this lasso shape: a finite number of transitions with at some point a transition looping back to one of the previous states. For an example of a stream system with infinitely many states, let $A = \{a, b\}$ and define

$$(\mathbb{Z},\ \langle \mathsf{o}_{\mathbb{Z}}, \mathsf{tr}_{\mathbb{Z}} \rangle) \qquad \mathsf{o}_{\mathbb{Z}} \colon \mathbb{Z} \to A \qquad \mathsf{tr}_{\mathbb{Z}} \colon \mathbb{Z} \to \mathbb{Z}$$

by

$$\mathsf{o}_{\mathbb{Z}}(z) \;=\; \begin{cases} a & \text{if } z \text{ is even} \\ b & \text{if } z \text{ is odd} \end{cases} \qquad \mathsf{tr}_{\mathbb{Z}}(z) \;=\; z + 1$$

where $\mathbb{Z}$ is the set of integers. Equivalently, with a picture:

$$(\mathbb{Z},\ \langle \mathsf{o}_{\mathbb{Z}}, \mathsf{tr}_{\mathbb{Z}} \rangle) \;=\; \cdots \longrightarrow -1|b \longrightarrow 0|a \longrightarrow 1|b \longrightarrow 2|a \longrightarrow \cdots$$

The output function $\mathsf{o}_{\mathbb{Z}} \colon \mathbb{Z} \to A$ provides us with partial information on the states of the system, by signalling whether they are even or odd.

If we define

$$\begin{tikzcd}
(S, \langle \mathsf{o}_S, \mathsf{tr}_S \rangle) \arrow[r, phantom, "="] & s_0|a \arrow[r] & s_1|b \arrow[r] & s_2|a \arrow[r] & s_3|b \arrow[r] & s_4|a \arrow[r] & s_5|b \arrow[lll, bend left] \\
(T, \langle \mathsf{o}_T, \mathsf{tr}_T \rangle) \arrow[r, phantom, "="] & t_0|a \arrow[r] & t_1|b \arrow[l, bend left]
\end{tikzcd}$$

then the function

$$f \colon S \to T \qquad f(s_0) = f(s_2) = f(s_4) \;=\; t_0 \qquad f(s_1) = f(s_3) = f(s_5) \;=\; t_1$$

is a homomorphism of stream systems. There exist no homomorphisms in the other direction. Also the following function

$$g \colon \mathbb{Z} \to T \qquad g(z) \;=\; \begin{cases} t_0 & \text{if } z \text{ is even} \\ t_1 & \text{if } z \text{ is odd} \end{cases}$$

is a homomorphism $g \colon (\mathbb{Z},\ \langle \mathsf{o}_{\mathbb{Z}}, \mathsf{tr}_{\mathbb{Z}} \rangle) \to (T, \langle \mathsf{o}_T, \mathsf{tr}_T \rangle)$. □

Exercise 57. Let $(\mathbb{Z},\ \langle \mathsf{o}, \mathsf{tr} \rangle)$ be defined by

$$\mathsf{o} \colon \mathbb{Z} \to \{a, b\} \qquad \mathsf{tr} \colon \mathbb{Z} \to \mathbb{Z}$$

$$\mathsf{o}(z) \;=\; \begin{cases} b & \text{if } z \geqslant 0 \\ a & \text{if } z < 0 \end{cases} \qquad \mathsf{tr}(z) \;=\; -2 \times z$$

Show that the function

$$h \colon \mathbb{Z} \to \mathbb{Z} \qquad h(z) = 2 \times z$$

is a homomorphism from $(\mathbb{Z},\ \langle \mathsf{o}, \mathsf{tr} \rangle)$ to itself. □

Exercise 58 (homomorphic images are subsystems). We call a subset $V \subseteq S$ of a stream system $(S, \langle \mathsf{o}_S, \mathsf{tr}_S \rangle)$ a *subsystem* if it is closed under transitions: if $s \in V$ then $\mathsf{tr}(s) \in V$. Show that the image $\mathsf{im}(f)$ of a homomorphism $f \colon (S, \langle \mathsf{o}_S, \mathsf{tr}_S \rangle) \to (T, \langle \mathsf{o}_T, \mathsf{tr}_T \rangle)$ is a subsystem of T. □

For the stream systems

$$\begin{tikzcd} s_0|a \arrow[r, bend right] & s_1|b \arrow[l, bend right] \end{tikzcd} \qquad \begin{tikzcd} t_0|a \arrow[r] & t_1|b \arrow[r] & t_2|a \arrow[r] & t_3|b \arrow[lll, bend right] \end{tikzcd}$$

there exists a single stream homomorphism from right to left, mapping t_0 to s_0. There exist no homomorphisms in the other direction. Still one could say that s_0 and t_0 are equivalent with respect to their *observational behaviour*. The infinite transition sequences starting in s_0 and t_0 give rise to identical streams of observations:

$$\begin{tikzcd} s_0|a \arrow[r] & s_1|b \arrow[r] & s_0|a \arrow[r] & s_1|b \arrow[r] & \cdots \\ t_0|a \arrow[r] & t_1|b \arrow[r] & t_2|a \arrow[r] & t_3|b \arrow[r] & \cdots \end{tikzcd}$$

The following notion of *bisimulation* captures exactly this type of *observational* or *behavioural* equivalence of stream systems.

Definition 59 (bisimulation of stream systems). A relation $R \subseteq S \times T$ is a *bisimulation* of stream systems $(S, \langle \mathsf{o}_S, \mathsf{tr}_S \rangle)$ and $(T, \langle \mathsf{o}_T, \mathsf{tr}_T \rangle)$ if there exists a structure map

$$\langle \mathsf{o}_R, \mathsf{tr}_R \rangle \colon R \to A \times R$$

such that the projection maps $\pi_1 \colon R \to S$ and $\pi_2 \colon R \to T$ are homomorphisms:

$$\begin{tikzcd} S \arrow[d, "\langle \mathsf{o}_S, \mathsf{tr}_S \rangle"'] & R \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] \arrow[d, dashed, "\exists\, \langle \mathsf{o}_R, \mathsf{tr}_R \rangle"] & T \arrow[d, "\langle \mathsf{o}_T, \mathsf{tr}_T \rangle"] \\ A \times S & A \times R \arrow[l, "1_A \times \pi_1"] \arrow[r, "1_A \times \pi_2"'] & A \times T \end{tikzcd} \tag{7.2}$$

□

Exercise 60. Prove that there exists at most one structure map $\langle \mathsf{o}_R, \mathsf{tr}_R \rangle$ making the diagram above commute. (For many other types of systems, such as non-deterministic automata, bisimulation relations can in general be equipped with more than one structure map.) □

Homomorphisms of stream systems are output and transition structure preserving *functions.* Similarly, bisimulations of stream systems are structure preserving *relations.* This is expressed by the following, equivalent, definition of bisimulation, which is easier to work with than Definition 59.

Definition 61 (bisimulation of stream systems, relational). A relation $R \subseteq S \times T$ is a bisimulation of stream systems if, for all $(s,t) \in R$,

$$(i)\ \ \mathsf{o}_S(s) \ = \ \mathsf{o}_T(t) \quad \text{and} \qquad (ii)\ \ (\mathsf{tr}_S(s), \mathsf{tr}_T(t)) \in R \tag{7.3}$$

□

Thus R is a bisimulation if R contains only pairs of states with equal outputs and is closed under (pair-wise) transitions.

Exercise 62. Prove the equivalence of Definitions 59 and 61. □

Exercise 63. Use Definition 61 to prove that the collection of all bisimulation relations between two given stream systems is closed under (i) arbitrary unions and intersections, (ii) inverse and (iii) composition (of composable relations). □

Definition 64 (bisimilarity of stream systems). The union of *all* bisimulation relations between two stream systems $(S, \langle \mathsf{o}_S, \mathsf{tr}_S \rangle)$ and $(T, \langle \mathsf{o}_T, \mathsf{tr}_T \rangle)$ is a bisimulation, called the *bisimilarity* relation:

$$\sim \, = \bigcup \{ R \subseteq S \times T \mid \ \ R \text{ is a bisimulation } \} \tag{7.4}$$

By definition, $\sim$ is the *greatest* bisimulation; we call $\sim$ the *bisimilarity* relation. If $s \sim t$, that is, if there exists a bisimulation relation R with $(s,t) \in R$, then we call s and t *bisimilar.* □

A special case of the above definition is the greatest bisimulation relation between a stream system S and itself.

Exercise 65. Use Exercise 63 to prove that the bisimilarity relation $\sim \subseteq S \times S$ between a stream system and itself is an equivalence relation. □

Exercise 66. Let $\mathsf{Rel}(S,T)$ be the set of all relations between S and T and define

$$\Phi\colon \mathsf{Rel}(S,T) \to \mathsf{Rel}(S,T)$$

$$\Phi(R) = \{ (s,t) \in S \times T \mid \mathsf{o}_S(s) = \mathsf{o}_T(t) \text{ and } (\mathsf{tr}_S(s), \mathsf{tr}_T(t)) \in R \}$$

Show that Φ is monotone, that R is a bisimulation if and only if $R \subseteq \Phi(R)$, and that $\sim$ is the greatest fixed point of Φ (with respect to the inclusion order). □

Example 67. Consider again the stream systems

$$\begin{tikzcd}
(S, \langle \mathsf{o}_S, \mathsf{tr}_S \rangle) = & s_0|a \arrow[r] & s_1|b \arrow[r] & s_2|a \arrow[r] & s_3|b \arrow[r] & s_4|a \arrow[r] & s_5|b \arrow[lll, bend left]
\end{tikzcd}$$

$$\begin{tikzcd}
(T, \langle \mathsf{o}_T, \mathsf{tr}_T \rangle) = & t_0|a \arrow[r] & t_1|b \arrow[l, bend left]
\end{tikzcd}$$

$$\begin{tikzcd}
(\mathbb{Z}, \langle \mathsf{o}_\mathbb{Z}, \mathsf{tr}_\mathbb{Z} \rangle) = & \cdots \arrow[r] & -1|b \arrow[r] & 0|a \arrow[r] & 1|b \arrow[r] & 2|a \arrow[r] & \cdots
\end{tikzcd}$$

The following relations are examples of bisimulations of stream systems:

$$\begin{aligned}
R &= \{(s_0,t_0),\ (s_2,t_0),\ (s_4,t_0),\ (s_1,t_1),\ (s_3,t_1),\ (s_5,t_1)\} \\
R^{-1} &= \{(t_0,s_0),\ (t_0,s_2),\ (t_0,s_4),\ (t_1,s_1),\ (t_1,s_3),\ (t_1,s_5)\} \\
U &= \{(z_1,z_2) \in \mathbb{Z} \times \mathbb{Z} \mid z_1 \text{ and } z_2 \text{ are both even or both odd}\}
\end{aligned}$$

Note that the latter relation is the greatest bisimulation on $(\mathbb{Z}, \langle \mathsf{o}_\mathbb{Z}, \mathsf{tr}_\mathbb{Z} \rangle)$. □

Exercise 68 (bisimilarity is product of stream systems). Let $(S, \langle \mathsf{o}_S, \mathsf{tr}_S \rangle)$ and $(T, \langle \mathsf{o}_T, \mathsf{tr}_T \rangle)$ be two stream systems, and consider the system $(\sim, \langle \mathsf{o}, \mathsf{tr} \rangle)$, consisting of the bisimilarity relation $\sim$ on S and T, with $\mathsf{o}((s,t)) = \mathsf{o}_S(s) = \mathsf{o}_T(t)$ and $\mathsf{tr}((s,t)) = (\mathsf{tr}_S(s), \mathsf{tr}_T(t))$. Show that $(\sim, \langle \mathsf{o}, \mathsf{tr} \rangle)$ is the (categorical) product of $(S, \langle \mathsf{o}_S, \mathsf{tr}_S \rangle)$ and $(T, \langle \mathsf{o}_T, \mathsf{tr}_T \rangle)$: for all stream systems $(Z, \langle \mathsf{o}_Z, \mathsf{tr}_Z \rangle)$ and homomorphisms $f\colon Z \to S$ and $g\colon Z \to T$, there exists a unique (mediating) homomorphism $h\colon Z \to \sim$ such that

$$\begin{tikzcd}
 & Z \arrow[ld, bend right, "\forall f"'] \arrow[d, dashed, "\exists!\, h"] \arrow[rd, bend left, "\forall g"] & \\
S & \sim \arrow[l, "\pi_1"] \arrow[r, "\pi_2"'] & T
\end{tikzcd}$$

That is, $\pi_1 \circ h = f$ and $\pi_2 \circ h = g$ and, moreover, h is the unique homomorphism satisfying these equations. □

The following proposition shows that bisimulations on stream systems generalise stream homomorphisms, in the same way as relations on sets generalise functions.

Proposition 69 (functional bisimulations). Let $(S, \langle \mathsf{o}_S, \mathsf{tr}_S\rangle)$ and $(T, \langle \mathsf{o}_T, \mathsf{tr}_T\rangle)$ be two stream systems. A function $f\colon S \to T$ is a homomorphism of stream systems if and only if its graph

$$\mathsf{graph}(f) \;=\; \{(s, f(s)) \in S \times T \mid s \in S\}$$

is a bisimulation. Such relations are sometimes called *functional bisimulations.*

Exercise 70. Prove Proposition 69. □

The kernel of a stream homomorphism is a *stream bisimulation equivalence.*

Proposition 71. Let $(S, \langle \mathsf{o}_S, \mathsf{tr}_S\rangle)$ and $(T, \langle \mathsf{o}_T, \mathsf{tr}_T\rangle)$ be two stream systems. If $f\colon S \to T$ is a homomorphism then $\mathsf{ker}(f) = \{(s, s') \in S \times S \mid f(s) = f(s')\}$ is both a bisimulation relation and an equivalence relation.

Proof: The kernel of any function is an equivalence relation. Because

$$\mathsf{ker}(f) \;=\; \mathsf{graph}(f) \circ \mathsf{graph}(f)^{-1}$$

it follows from Exercise 63 and Proposition 69 that $\mathsf{ker}(f)$ is a bisimulation relation. □

Exercise 72 (quotient of stream system). Prove that the quotient S/R of a stream system $(S, \langle \mathsf{o}_S, \mathsf{tr}_S\rangle)$ with respect to a bisimulation equivalence relation $R \subseteq S \times S$ has a unique stream system structure

$$(S/R, \langle \mathsf{o}_R, \mathsf{tr}_R\rangle) \qquad \mathsf{o}_R\colon S/R \to A \qquad \mathsf{tr}_R\colon S/R \to S/R$$

such that the quotient function

$$[-]_R\colon S \to S/R$$

which maps a state s to its equivalence class $[s]_R$, is a homomorphism. □

Proposition 73 (minimisation of stream systems). Let $(S/\!\sim, \langle \mathsf{o}_\sim, \mathsf{tr}_\sim\rangle)$ be the quotient of a stream system $(S, \langle \mathsf{o}_S, \mathsf{tr}_S\rangle)$ with respect to its bisimilarity relation $\sim\subseteq S \times S$.

(a) For every *surjective* homomorphism $f\colon (S, \langle \mathsf{o}_S, \mathsf{tr}_S\rangle) \to (T, \langle \mathsf{o}_T, \mathsf{tr}_T\rangle)$ there is a unique surjective homomorphism $g\colon (T, \langle \mathsf{o}_T, \mathsf{tr}_T\rangle) \to (S/\!\sim, \langle \mathsf{o}_\sim, \mathsf{tr}_\sim\rangle)$ such that

$$\begin{tikzcd} S \arrow[r, "\forall f"] \arrow[d, "{[-]_\sim}"'] & T \arrow[dl, dashed, "\exists ! g"] \\ S/\!\sim & \end{tikzcd} \qquad g \circ f \;=\; [-]_\sim$$

(b) Every stream homomorphism $f\colon (S/{\sim}, \langle \mathsf{o}_\sim, \mathsf{tr}_\sim\rangle) \to (T, \langle \mathsf{o}_T, \mathsf{tr}_T\rangle)$ is injective. □

According to Proposition 73, $(S/{\sim}, \langle \mathsf{o}_\sim, \mathsf{tr}_\sim\rangle)$ can be viewed as the minimisation of $(S,\ \langle \mathsf{o}_S, \mathsf{tr}_S\rangle)$: statement (a) says that among all possible quotients of S, the quotient $S/{\sim}$ makes the most identifications (of behaviourally equivalent states). Statement (b) says that no further identifications can be made.

Exercise 74. Prove (a) and (b) of Proposition 73. Compute the quotient of $(\mathbb{Z},\ \langle \mathsf{o}_\mathbb{Z}, \mathsf{tr}_\mathbb{Z}\rangle)$, from Example 67, with respect to its bisimilarity relation U. □

Exercise 75. Compute the minimisation of the stream system $(\mathbb{Z},\ \langle \mathsf{o}, \mathsf{tr}\rangle)$ from Exercise 57. □

7.2 The final system of streams

We define the set of all *streams* over A by

$$A^\omega \ = \ \{\sigma \mid \sigma\colon \mathbb{N} \to A\}$$

that is, the set of all infinite sequences of elements in A. For a stream $\sigma \in A^\omega$, we call $\sigma(0)$ the *initial value* of σ. We define the *stream derivative* of σ as the stream σ' given by

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

In computer science, the initial value $\sigma(0)$ and derivative σ' are often called $\mathsf{head}(\sigma)$ and $\mathsf{tail}(\sigma)$, but the present notation and terminology are both convenient and will help us, in Chapter 12, to develop a *calculus* of streams in close analogy to classical calculus in analysis.

We shall occasionally write

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

which makes it easier to refer to the individual elements of a stream. With this notation, the derivative of σ is given by

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

There is the following obvious and useful relation between the elements of a stream and its repeated derivatives.

Definition 76 (higher-order stream derivatives). For $\sigma \in A^\omega$, we define

$$\sigma^{(0)} \;=\; \sigma \qquad\qquad \sigma^{(n+1)} \;=\; (\sigma^{(n)})' \qquad \square$$

Lemma 77. *For $\sigma \in A^\omega$ and $n \geqslant 0$,*

$$\sigma(n) \;=\; \sigma^{(n)}(0) \qquad \square$$

Next we define the *functions* of initial value and derivative by

$$\mathsf{i}_{st}\colon A^\omega \to A \qquad\qquad \mathsf{i}_{st}(\sigma) \;=\; \sigma(0) \tag{7.5}$$

$$\mathsf{d}_{st}\colon A^\omega \to A^\omega \qquad\qquad \mathsf{d}_{st}(\sigma) \;=\; \sigma' \tag{7.6}$$

Pairing the functions of i_{st} and d_{st}, we have turned the set A^ω of all streams into a stream system $(A^\omega,\ \langle \mathsf{i}_{st}, \mathsf{d}_{st}\rangle)$ that is *final*.

Theorem 78 (finality – streams). *The stream system $(A^\omega,\ \langle \mathsf{i}_{st}, \mathsf{d}_{st}\rangle)$ is* final*: For every stream system $(S,\ \langle \mathsf{o}_S, \mathsf{tr}_S\rangle)$ there exists a* unique *homomorphism*

$$\begin{tikzcd}
S \arrow[r, dashed, "\exists!\, [\![-]\!]"] \arrow[d, "\forall\, \langle \mathsf{o}_S, \mathsf{tr}_S\rangle"'] & A^\omega \arrow[d, "\langle \mathsf{i}_{st}, \mathsf{d}_{st}\rangle"] \\
A \times S \arrow[r, dashed, "1_A \times [\![-]\!]"'] & A \times A^\omega
\end{tikzcd}$$

By abuse of language, this homomorphism $[\![-]\!]\colon S \to A^\omega$ into the final coalgebra *will sometimes be called* the final homomorphism.

Proof: For $s \in S$, let $s_0 = s$ and $s_{n+1} = \mathsf{tr}_S(s_n)$. We define $[\![-]\!]\colon S \to A^\omega$ by $[\![s]\!](n) = \mathsf{o}_S(s_n)$. The function $[\![-]\!]$ satisfies

$$[\![s]\!](0) \;=\; \mathsf{o}_S(s) \qquad\qquad [\![s]\!]' \;=\; [\![\mathsf{tr}_S(s)]\!] \qquad\qquad (\text{all } s \in S)$$

which says that it is a homomorphism of stream systems. One easily checks that it is unique. $\square$

We may think of the structure map $\langle \mathsf{o}, \mathsf{tr}\rangle$ as a specification of the *local* behaviour of the stream system, consisting for each state $s \in S$ of an output $\mathsf{o}(s)$ and a single transition step to a next state $\mathsf{tr}(s)$. The final homomorphism $[\![-]\!]$ assigns to every state s its *global* behaviour, which is the stream $[\![s]\!]$ of all successive outputs. One may view $[\![s]\!]$ as the *semantics* of s, hence our use of semantic brackets.

Example 79. If

$$\begin{tikzcd}(S, \langle \mathsf{o}, \mathsf{tr}\rangle) \quad = \quad s_0|a \arrow[r] & s_1|a \arrow[r] & s_2|b \arrow[r, bend left] & s_3|a \arrow[l, bend left]\end{tikzcd}$$

then

$$[\![s_0]\!] = aa(ba)^\omega \qquad [\![s_1]\!] = a(ba)^\omega \qquad [\![s_2]\!] = (ba)^\omega \qquad [\![s_3]\!] = (ab)^\omega$$

where $(ab)^\omega = (a, b, a, b, a, b, \ldots)$ and $(ba)^\omega = (b, a, b, a, b, a, \ldots)$. □

Exercise 80. Final coalgebras are unique up to isomorphism: prove that if two stream systems $(S, \langle \mathsf{o}_S, \mathsf{tr}_S\rangle)$ and $(T, \langle \mathsf{o}_T, \mathsf{tr}_T\rangle)$ both are final, then they are isomorphic. More challenging: prove that if $(S, \langle \mathsf{o}_S, \mathsf{tr}_S\rangle)$ is final then $\langle \mathsf{o}_S, \mathsf{tr}_S\rangle$ is bijective. (This will be proved more generally, for arbitrary final coalgebras, in Chapter 15, Theorem 406.) □

Exercise 81. Show that the stream system $(A^\omega \times A^\omega, \langle \mathsf{o}, \mathsf{tr}\rangle)$, defined by $\mathsf{o}((\sigma, \tau)) = \sigma(0)$ and $\mathsf{tr}((\sigma, \tau)) = (\tau, \sigma')$, for all $(\sigma, \tau) \in A^\omega \times A^\omega$, is final. □

Exercise 82. Consider the stream system $(\mathbb{Z}^\omega, \langle \mathsf{i}_{st}, \Delta\rangle)$ defined by

$$\begin{aligned} &\mathsf{i}_{st} \colon \mathbb{Z}^\omega \to \mathbb{Z} && \Delta \colon \mathbb{Z}^\omega \to \mathbb{Z}^\omega \\ &\mathsf{i}_{st}(\sigma) = \sigma(0) && \Delta(\sigma) = \sigma' - \sigma \end{aligned}$$

for all $\sigma \in \mathbb{Z}^\omega$. Thus $\Delta(\sigma)(n) = \sigma(n+1) - \sigma(n)$, for all $n \geq 0$. Prove that $(\mathbb{Z}^\omega, \langle \mathsf{i}_{st}, \Delta\rangle)$ is final among all stream systems with outputs in $\mathbb{Z}$. (The operation Δ will be studied in some detail in Section 12.7.) □

Exercise 83. Returning to the dynamical systems of Chapter 6, how would you define a bisimulation relation on dynamical systems? What does the greatest bisimulation relation on a dynamical system look like? Construct a dynamical system that is final. What is the categorical product – cf. Exercise 68 – of two dynamical systems? □

The final homomorphism $[\![-]\!] \colon S \to A^\omega$ from Theorem 78 maps every state to (a canonical representation of) its bisimulation equivalence class.

Theorem 84. *Let $(S, \langle \mathsf{o}, \mathsf{tr}\rangle)$ be a stream system. For all $s, t \in S$,*

$$s \sim t \iff [\![s]\!] = [\![t]\!]$$

Proof: We know that $\sim$ is a bisimulation on S, with projections $\pi_1, \pi_2 \colon \ \sim \to S$. By the definition of bisimulation, the projections are homomorphisms. Composition with $[\![-]\!]$ gives *two* homomorphisms from R to A^ω. By the finality of A^ω, they must be equal, which proves the implication from left to right. The converse follows from Proposition 71: since $[\![-]\!] \colon S \to A^\omega$ is a homomorphism, its kernel

$$\{\, (s, t) \mid [\![s]\!] = [\![t]\!] \,\}$$

is a bisimulation, hence $s \sim t$. □

Exercise 85 (minimisation and finality). Prove that it follows from Theorem 84 that $[\![S]\!]$ – the image of S under $[\![-]\!]$ – is isomorphic to the quotient $S/\!\sim$. (Note that $[\![S]\!]$ is a subsystem of A^ω, by Exercise 58, and that $S/\!\sim$ is a stream system, by Exercises 65 and 72.) By Proposition 73, it follows that $[\![S]\!]$ is the minimisation of $(S,\ \langle \mathsf{o}, \mathsf{tr} \rangle)$. □

7.3 Defining streams by coinduction

The *existence* of a unique homomorphism from any stream system into the final system of streams can be used to *define* streams and stream functions. Such definitions are in a precise sense – explained in Chapter 4 – dual to definitions by induction, and are therefore called *coinductive* or: *by coinduction*.

More precisely, we say that a stream $\sigma \in A^\omega$ is *defined by coinduction* if there exists a stream system $(S,\ \langle \mathsf{o}, \mathsf{tr} \rangle)$ and a state $s \in S$ such that

$$\sigma \;=\; [\![s]\!]$$

where

$$[\![-]\!] \colon (S,\ \langle \mathsf{o}, \mathsf{tr} \rangle) \to (A^\omega,\ \langle \mathsf{i}_{st}, \mathsf{d}_{st} \rangle)$$

is the unique homomorphism given by finality, Theorem 78. We may think of the system $(S,\ \langle \mathsf{o}, \mathsf{tr} \rangle)$ together with the state $s \in S$ as a specification of the behaviour of σ or, in short, as a *behavioural specification* of σ.

How interesting such a coinductive definition or behavioural specification is, will depend on how interesting the system $(S,\ \langle \mathsf{o}, \mathsf{tr} \rangle)$ is. For example, *any* stream $\sigma \in A^\omega$ can be defined by coinduction by considering

$$(S,\ \langle \mathsf{o}, \mathsf{tr} \rangle) \;=\; (A^\omega,\ \langle \mathsf{i}_{st}, \mathsf{d}_{st} \rangle)$$

with state $\sigma \in A^\omega$. Then $\sigma = [\![\sigma]\!]$, which clearly does not tell us much. Another extreme would be to require S to be *finite*. As we already saw in Example 56, finite stream systems are lasso shaped and give rise, as we shall explain below, to a well-defined but rather restricted class of streams. In practice, one will typically take S to be an infinite, well-structured set. Below, we limit ourselves to a few elementary examples of coinductive stream definitions, postponing a more systematic treatment until Chapter 11.

Example 86 (coinductive stream definitions). We present several elementary examples of coinductive definitions of streams and of functions on streams.

(1) Recall the system

$$\begin{tikzcd}(S, \langle \mathsf{o}, \mathsf{tr} \rangle) \arrow[r, phantom, "="] & s_0|a \arrow[r] & s_1|a \arrow[r] & s_2|b \arrow[r, bend left] & s_3|a \arrow[l, bend left]\end{tikzcd}$$

from Example 79. Since

$$[\![s_0]\!] \;=\; aa(ba)^\omega \qquad [\![s_1]\!] \;=\; a(ba)^\omega \qquad [\![s_2]\!] \;=\; (ba)^\omega \qquad [\![s_3]\!] \;=\; (ab)^\omega$$

we can say that the streams $aa(ba)^\omega$, $a(ba)^\omega$, $(ba)^\omega$ and $(ab)^\omega$ are defined by coinduction.

More generally, a stream σ can be defined by coinduction using a *finite* stream system if and only if σ is *eventually periodic*:

$$\sigma \;=\; v \cdot w^\omega \;=\; v \cdot w \cdot w \cdot w \cdot \cdots \tag{7.7}$$

for finite words $v, w \in A^*$. The proof is left as Exercise 87.

(2) Here is an example of a coinductive definition of a stream *function*. We consider the following stream system:

$$(A^\omega, \langle \mathsf{o}_e, \mathsf{tr}_e \rangle) \qquad \begin{array}{ll} \mathsf{o}_e \colon A^\omega \to A & \mathsf{tr}_e \colon A^\omega \to A^\omega \\ \mathsf{o}_e(\sigma) \;=\; \sigma(0) & \mathsf{tr}_e(\sigma) \;=\; \sigma'' \end{array}$$

By finality, there exists a unique homomorphism, which we call even:

$$\mathsf{even} \colon (A^\omega, \langle \mathsf{o}_e, \mathsf{tr}_e \rangle) \to (A^\omega,\, \langle \mathsf{i}_{st}, \mathsf{d}_{st} \rangle)$$

We call this a definition by coinduction of the stream function $\mathsf{even} \colon A^\omega \to A^\omega$. Because even is a homomorphism, by definition, we have

$$\mathsf{even}(\sigma)(0) \;=\; \sigma(0) \qquad\qquad \mathsf{even}(\sigma)' \;=\; \mathsf{even}(\sigma'')$$

One readily shows by induction that this implies

$$\mathsf{even}(\sigma) \;=\; (\sigma(0), \sigma(2), \sigma(4), \ldots)$$

For another example, we consider

$$(A^\omega, \langle \mathsf{o}_o, \mathsf{tr}_o \rangle) \qquad \begin{array}{ll} \mathsf{o}_o \colon A^\omega \to A & \mathsf{tr}_o \colon A^\omega \to A^\omega \\ \mathsf{o}_o(\sigma) \;=\; \sigma(1) & \mathsf{tr}_o(\sigma) \;=\; \sigma'' \end{array}$$

Then we obtain, in a similar fashion, a homomorphism

$$\mathsf{odd} \colon (A^\omega, \langle \mathsf{o}_o, \mathsf{tr}_0 \rangle) \to (A^\omega,\, \langle \mathsf{i}_{st}, \mathsf{d}_{st} \rangle)$$

satisfying

$$\mathsf{odd}(\sigma)(0) \;=\; \sigma(1) \qquad\qquad \mathsf{odd}(\sigma)' \;=\; \mathsf{odd}(\sigma'')$$

and

$$\mathsf{odd}(\sigma) \;=\; (\sigma(1), \sigma(3), \sigma(5), \ldots)$$

(3) For our next example, let

$$(\mathbb{N}^\omega, \langle \mathsf{o}_2, \mathsf{d}_{st}\rangle) \qquad \begin{array}{l} \mathsf{o}_2 \colon \mathbb{N}^\omega \to \mathbb{N} \\ \mathsf{o}_2(\sigma) \;=\; 2\cdot\sigma(0) \end{array} \qquad \begin{array}{l} \mathsf{d}_{st} \colon \mathbb{N}^\omega \to \mathbb{N}^\omega \\ \mathsf{d}_{st}(\sigma) \;=\; \sigma' \end{array}$$

By finality, there exists a unique homomorphism

$$2 \times (-) \colon (\mathbb{N}^\omega, \langle \mathsf{o}_2, \mathsf{d}_{st}\rangle \to (\mathbb{N}^\omega, \langle \mathsf{i}_{st}, \mathsf{d}_{st}\rangle$$

It follows that

$$(2\times\sigma)(0) \;=\; 2\cdot\sigma(0) \qquad\qquad (2\times\sigma)' \;=\; 2\times\sigma'$$

This implies

$$2\times\sigma \;=\; (2\cdot\sigma(0),\, 2\cdot\sigma(1),\, 2\cdot\sigma(3),\, \ldots)$$

More generally, one can define any *elementwise stream function* in this manner. Let $f\colon A \to A$ be a function and let

$$(A^\omega, \langle \mathsf{o}_f,\, \mathsf{d}_{st}\rangle) \qquad \begin{array}{l} \mathsf{o}_f \colon A^\omega \to A \\ \mathsf{o}_f(\sigma) \;=\; f(\sigma(0)) \end{array} \qquad \begin{array}{l} \mathsf{d}_{st} \colon A^\omega \to A^\omega \\ \mathsf{d}_{st}(\sigma) \;=\; \sigma' \end{array}$$

By finality, we obtain a homomorphism

$$\mathsf{map}_f \colon (A^\omega, \langle \mathsf{o}_f,\, \mathsf{d}_{st}\rangle) \to (A^\omega,\, \langle \mathsf{i}_{st}, \mathsf{d}_{st}\rangle)$$

satisfying

$$(\mathsf{map}_f(\sigma))(0) \;=\; f(\sigma(0)) \qquad\qquad (\mathsf{map}_f(\sigma))' \;=\; \mathsf{map}_f(\sigma')$$

which implies

$$\mathsf{map}_f(\sigma) \;=\; (f(\sigma(0)),\, f(\sigma(1)),\, f(\sigma(2)),\, \ldots)$$

(4) As a final example, consider the stream system

$$(A^\omega \times A^\omega, \langle \mathsf{o}_z, \mathsf{tr}_z\rangle) \qquad \begin{array}{l} \mathsf{o}_z \colon A^\omega \to A \\ \mathsf{o}_z((\sigma,\tau)) \;=\; \sigma(0) \end{array} \qquad \begin{array}{l} \mathsf{tr}_z \colon A^\omega \to A^\omega \\ \mathsf{tr}_z((\sigma,\tau)) \;=\; (\tau, \sigma') \end{array}$$

By finality, there exists a homomorphism

$$\mathsf{zip} \colon (A^\omega \times A^\omega, \langle \mathsf{o}_z, \mathsf{tr}_z\rangle) \to (A^\omega,\, \langle \mathsf{i}_{st}, \mathsf{d}_{st}\rangle)$$

satisfying

$$\mathsf{zip}(\sigma,\tau)(0) \;=\; \sigma(0) \qquad\qquad \mathsf{zip}(\sigma,\tau)' \;=\; \mathsf{zip}(\tau,\sigma')$$

This implies

$$\mathsf{zip}(\sigma,\tau) \;=\; (\sigma(0),\tau(0),\sigma(1),\tau(1),\ldots)$$

□

Exercise 87 (eventually periodic streams). Show that a stream σ can be defined by coinduction using a *finite* stream system if and only if σ is *eventually periodic* (defined in Equation (7.7)). □

Exercise 88 (sum). Define by coinduction the function $+\colon \mathbb{N}^\omega \times \mathbb{N}^\omega \to \mathbb{N}^\omega$ that maps two streams $\sigma,\tau \in \mathbb{N}^\omega$ to their elementwise *sum*:

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

Verify that it satisfies

$$(\sigma+\tau)(0) \;=\; \sigma(0)+\tau(0) \qquad\qquad (\sigma+\tau)' \;=\; \sigma'+\tau'$$

□

Exercise 89. Define the following function by coinduction:

$$\mathsf{zip}_3\colon (A^\omega)^3 \to A^\omega \qquad \mathsf{zip}_3(\sigma,\tau,\rho) \;=\; (\sigma(0),\tau(0),\rho(0),\sigma(1),\tau(1),\rho(1),\ldots)$$

Use the function zip_3 to define the function

$$\mathsf{zip}_{2,1}\colon (A^\omega)^2 \to A^\omega \qquad \mathsf{zip}_{2,1}(\sigma,\tau) \;=\; (\sigma(0),\sigma(1),\tau(0),\sigma(2),\sigma(3),\tau(1),\ldots)$$

Define the function

$$\mathsf{alt}\colon (A^\omega)^2 \to A^\omega \qquad \mathsf{alt}(\sigma,\tau) \;=\; (\sigma(0),\tau(1),\sigma(2),\tau(3),\sigma(4),\tau(5),\ldots)$$

by coinduction. □

Exercise 90. Consider the stream system

$$(\mathbb{N}^\omega,\ \langle \mathsf{o},\mathsf{tr}\rangle) \qquad \begin{array}{ll} \mathsf{o}\colon \mathbb{N}^\omega \to \mathbb{N} & \mathsf{tr}\colon \mathbb{N}^\omega \to \mathbb{N}^\omega \\ \mathsf{o}(\sigma) \;=\; \sigma(0) & \mathsf{tr}(\sigma) \;=\; (\sigma(0)+\sigma(1),\ \sigma(2),\ \sigma(3),\ \ldots) \end{array}$$

By finality, there exists a unique homomorphism $f\colon \mathbb{N}^\omega \to \mathbb{N}^\omega$. What is $f(\sigma)(n)$? □

For every stream and stream function that we have defined by coinduction so far, it is easy to come up with a formula describing its nth element. For instance, $\mathsf{even}(\sigma)(n) = \sigma(2n)$. The next exercise illustrates that this is not always the case.

Exercise 91 (ordered merge). Use coinduction to define a function $\parallel\colon \mathbb{N}^\omega \times \mathbb{N}^\omega \to \mathbb{N}^\omega$ satisfying

$$(\sigma \parallel \tau)(0) \;=\; \begin{cases} \sigma(0) & \text{if } \sigma(0) < \tau(0) \\ \tau(0) & \text{if } \sigma(0) \geqslant \tau(0) \end{cases} \qquad (\sigma \parallel \tau)' \;=\; \begin{cases} \sigma' \parallel \tau & \text{if } \sigma(0) < \tau(0) \\ \sigma' \parallel \tau' & \text{if } \sigma(0) \;=\; \tau(0) \\ \sigma \parallel \tau' & \text{if } \sigma(0) > \tau(0) \end{cases}$$

(Convince yourself by looking at some examples that $\sigma \parallel \tau$ is an ordered increasing stream of natural numbers whenever both σ and τ are.) Try to find a formula describing the nth element of $\sigma \parallel \tau$. □

Here is an example to illustrate that for the coinductive definition of streams and stream functions, we often need stream systems with some additional structure.

Example 92. Let $2 = \{0, 1\}$ and consider the following stream system:

$$(2 \times A^\omega, \langle \mathsf{o}_{ex}, \mathsf{tr}_{ex} \rangle) \qquad \mathsf{o}_{ex}\colon (2 \times A^\omega) \to A \qquad \mathsf{tr}_{ex}\colon (2 \times A^\omega) \to (2 \times A^\omega)$$

defined by

$$\mathsf{o}_{ex}(x, \sigma) \;=\; \begin{cases} \sigma(1) & \text{if } x \;=\; 1 \\ \sigma(0) & \text{if } x \;=\; 0 \end{cases} \qquad \mathsf{tr}_{ex}(x, \sigma) \;=\; \begin{cases} (0, \sigma) & \text{if } x \;=\; 1 \\ (1, \sigma'') & \text{if } x \;=\; 0 \end{cases}$$

By finality, there exists a unique homomorphism

$$f\colon (2 \times A^\omega, \langle \mathsf{o}_{ex}, \mathsf{tr}_{ex} \rangle) \to (A^\omega, \langle \mathsf{i}_{st}, \mathsf{d}_{st} \rangle)$$

We define a function

$$\mathsf{exchange}\colon A^\omega \to A^\omega \qquad\qquad \mathsf{exchange}(\sigma) \;=\; f((1, \sigma))$$

Because f is a homomorphism, it follows that

$$\mathsf{exchange}(\sigma)(0) = \sigma(1) \qquad \mathsf{exchange}(\sigma)(1) = \sigma(0) \qquad \mathsf{exchange}(\sigma)'' = \mathsf{exchange}(\sigma'')$$

As a consequence,

$$\mathsf{exchange}(\sigma) \;=\; (\sigma(1), \sigma(0), \sigma(3), \sigma(2), \ldots)$$

Intuitively, the role of the additional component 2 in the carrier set $2 \times A^\omega$ of the defining stream system is to store the information on which value to output and on when to take the double derivative. □

Exercise 93. Give a coinductive definition of the function

$$\mathsf{exchange}_3\colon A^\omega \to A^\omega \qquad \mathsf{exchange}_3(\sigma) \;=\; (\sigma(2), \sigma(1), \sigma(0), \sigma(5), \sigma(4), \sigma(3), \ldots)$$ □

Exercise 94 (shuffle product). We define the *shuffle product* $\otimes\colon \mathbb{N}^\omega \times \mathbb{N}^\omega \to \mathbb{N}^\omega$, for $\sigma, \tau \in \mathbb{N}^\omega$ and $n \geqslant 0$, by

$$(\sigma \otimes \tau)(n) \;=\; \sum_{k=0}^{n} \binom{n}{k} \cdot \sigma(k) \cdot \tau(n-k) \tag{7.8}$$

Prove that the shuffle product of σ and τ satisfies

$$(\sigma \otimes \tau)(0) \;=\; \sigma(0) \cdot \tau(0) \qquad\qquad (\sigma \otimes \tau)' \;=\; (\sigma' \otimes \tau) \;+\; (\sigma \otimes \tau') \tag{7.9}$$

where $+$ denotes the elementwise sum of streams (Exercise 88), by using *Pascal's* rule for binomial coefficients:

$$\binom{n}{k} + \binom{n}{k+1} \;=\; \binom{n+1}{k+1}$$

In Exercise 108, we shall see that $\otimes$ is the *only* function satisfying (7.9). In Section 12.4, the shuffle product will be studied in more detail. □

Exercise 95. Consider an arbitrary stream $\sigma \in A^\omega$. What is the smallest stream system $(S, \langle \mathsf{o}, \mathsf{tr}\rangle)$ with the property that $\sigma = [\![s]\!]$, for some $s \in S$? □

7.4 Coinduction: the bisimulation proof method

Finality of A^ω gives rise to the following proof principle.

Theorem 96 (coinduction proof principle for streams). *For all streams $\sigma, \tau \in A^\omega$,*

$$\sigma \;\sim\; \tau \quad\Longleftrightarrow\quad \sigma \;=\; \tau$$

Proof: Taking $(S,\, \langle \mathsf{o}, \mathsf{tr}\rangle) = (A^\omega,\, \langle \mathsf{i}_{st}, \mathsf{d}_{st}\rangle)$ in Theorem 84, this follows from the fact that the identity map is the only homomorphism from $(A^\omega,\, \langle \mathsf{i}_{st}, \mathsf{d}_{st}\rangle)$ to itself. □

By the implication from left to right in Theorem 96 above:

$$\sigma \;\sim\; \tau \quad\Longrightarrow\quad \sigma \;=\; \tau \tag{7.10}$$

one can *prove the equality* of two streams σ and τ by finding a bisimulation $R \subseteq A^\omega \times A^\omega$ with $(\sigma, \tau) \in R$. Using the characterisation of stream bisimulation relations in (7.3), this amounts to the construction of a relation $R \subseteq A^\omega \times A^\omega$ such that $(\sigma, \tau) \in R$, and such that for all $(\alpha, \beta) \in R$,

$$(i) \quad \alpha(0) \; = \; \beta(0) \qquad\qquad \text{and} \qquad\qquad (ii) \quad (\alpha', \beta') \in R \tag{7.11}$$

This method of proving equality by the construction of a bisimulation relation is often called *the bisimulation proof method*. Since such proofs are in a precise sense – explained in Chapter 4 – dual to proofs by induction, they are also called *coinductive* or: *by coinduction*.

The intuition behind proofs by coinduction is that in order to prove that two streams *are* the same, it suffices to show that they *behave* the same. And showing that two streams behave the same amounts precisely to finding a bisimulation relating them.

We present a number of examples of proofs by coinduction. All streams and stream functions used in these examples have been introduced in Example 86.

Example 97. (1) Let $A = \{a, b\}$. We shall prove

$$\mathsf{even}((ab)^\omega) \; = \; a^\omega$$

by coinduction. We always start the construction of a candidate bisimulation relation with the set containing the pair(s) of streams that we want to prove equal. So let R be the following singleton set:

$$R \; = \; \{ \, (\, \mathsf{even}((ab)^\omega), \; a^\omega \,) \, \}$$

In order to show that R is a stream bisimulation relation, we have to demonstrate (7.11) above. For (7.11)(i), we note that

$$\mathsf{even}((ab)^\omega)(0) \; = \; (ab)^\omega(0) \; = \; a \; = \; a^\omega(0)$$

and for (7.11)(ii) we have

$$\mathsf{even}((ab)^\omega)' \; = \; \mathsf{even}(((ab)^\omega)'') \; = \; \mathsf{even}(((ba)^\omega)') \; = \; \mathsf{even}((ab)^\omega)$$

Since $(a^\omega)' = a^\omega$, it follows that

$$(\, \mathsf{even}((ab)^\omega)', \; (a^\omega)' \,) \in R$$

Thus R is a stream bisimulation relation, which proves $\mathsf{even}((ab)^\omega) \sim a^\omega$. The equality $\mathsf{even}((ab)^\omega) = a^\omega$ now follows by coinduction (7.10).

(2) Next we prove by coinduction that

$$\mathsf{zip}(a^\omega, b^\omega) \; = \; (ab)^\omega$$

Again we start with the corresponding singleton relation:

$$R \; = \; \{ \, (\, \mathsf{zip}(a^\omega, b^\omega), \; (ab)^\omega \,) \, \}$$

for which we try to establish (7.11) above. Computing initial values gives

$$\mathsf{zip}(a^\omega, b^\omega)(0) \;=\; a^\omega(0) \;=\; a \;=\; (ab)^\omega(0)$$

which proves (7.11)(i). Computing derivatives

$$\mathsf{zip}(a^\omega, b^\omega)' \;=\; \mathsf{zip}(b^\omega, (a^\omega)'\,) \;=\; \mathsf{zip}(b^\omega, a^\omega) \qquad\qquad ((ab)^\omega)' = (ba)^\omega$$

we find that

$$(\,\mathsf{zip}(a^\omega, b^\omega)',\; ((ab)^\omega)'\,) \;=\; (\,\mathsf{zip}(b^\omega, a^\omega),\; (ba)^\omega\,) \;\notin\; R$$

which contradicts (ii) in (7.11) above. The way out is as simple as it is generally applicable: we extend the relation R with the pair of derivatives that we have just found, by defining

$$S \;=\; R \cup \{\,(\,\mathsf{zip}(b^\omega, a^\omega),\; (ba)^\omega\,)\,\}$$

Next we check whether the new candidate relation S is a bisimulation. We had already established (i) for the pair of streams in R. Note that by extending R to S we have trivially proved (ii) for this pair as well. Thus what remains to be proved are (i) and (ii) for the newly added pair. Computing initial values gives

$$\mathsf{zip}(b^\omega, a^\omega)(0) \;=\; b \;=\; (ba)^\omega(0)$$

which proves (i). For (ii), we compute the derivatives

$$(\,\mathsf{zip}(b^\omega, a^\omega)',\; ((ba)^\omega)'\,) \;=\; (\,\mathsf{zip}(a^\omega, b^\omega),\; (ab)^\omega\,) \in S$$

Thus S is a bisimulation and the equality we set out to prove follows by coinduction. We have in fact proved *two* equalities, one for each pair in S:

$$\mathsf{zip}(a^\omega, b^\omega) \;=\; (ab)^\omega \qquad\qquad \mathsf{zip}(b^\omega, a^\omega) \;=\; (ba)^\omega$$

(3) Extending a candidate bisimulation relation with missing pairs will most certainly *not* allow us to prove the equality of just *any* pair of streams: condition (i) of a bisimulation relation will prevent that. For example, let $A = \{a, b, c\}$ and consider the streams $(ab)^\omega$ and $(ac)^\omega$. With the aim of proving $(ab)^\omega = (ac)^\omega$ by coinduction, let

$$R \;=\; \{\,(\,(ab)^\omega,\; (ac)^\omega\,)\,\}$$

Both streams in this pair have the same initial value a. Computing the pair of derivatives gives $(\,(ba)^\omega,\; (ca)^\omega\,)$, which is not in R. As before, we define a new candidate bisimulation relation by

$$S \;=\; R \cup \{\,(\,(ba)^\omega,\; (ca)^\omega\,)\,\}$$

Trying to prove that S is a bisimulation will now fail: the initial values of the streams in the newly added pair are b and c, which are different. We have proved that the streams $(ab)^\omega$ and $(ac)^\omega$ are *not* equal and we have done so by giving a counter-example to their equality: the attempt of constructing a bisimulation relation led to the observation that these streams differ in their second element. Trivial as this may be for the present example, it illustrates that coinduction is also a very useful principle for *disproving* equalities.

(4) The next example deals with the equality of two stream *functions*:

$$\begin{array}{ll} \mathsf{even} \circ \mathsf{zip} \colon (A^\omega \times A^\omega) \to A^\omega & \pi_1 \colon (A^\omega \times A^\omega) \to A^\omega \\ (\mathsf{even} \circ \mathsf{zip})(\sigma, \tau) \;=\; \mathsf{even}(\mathsf{zip}(\sigma, \tau)) & \pi_1(\sigma, \tau) \;=\; \sigma \end{array}$$

For a proof by coinduction that

$$\mathsf{even}(\mathsf{zip}(\sigma, \tau)) \;=\; \sigma \qquad (\sigma, \tau \in A^\omega)$$

we consider the following infinite relation:

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

To prove that R is a stream bisimulation, let $(\, \mathsf{even}(\mathsf{zip}(\sigma, \tau)),\ \sigma\,) \in R$, for arbitrary but fixed $\sigma, \tau \in A^\omega$. Computing initial values gives

$$\mathsf{even}(\mathsf{zip}(\sigma, \tau))(0) \;=\; \mathsf{zip}(\sigma, \tau)(0) \;=\; \sigma(0)$$

which proves (i) in (7.11) above. For the derivatives, we have

$$\mathsf{even}(\mathsf{zip}(\sigma, \tau))' \;=\; \mathsf{even}(\mathsf{zip}(\sigma, \tau)'') \;=\; \mathsf{even}(\mathsf{zip}(\tau, \sigma')') \;=\; \mathsf{even}(\mathsf{zip}(\sigma', \tau'))$$

We see that

$$(\, \mathsf{even}(\mathsf{zip}(\sigma, \tau))',\ \sigma'\,) \;=\; (\, \mathsf{even}(\mathsf{zip}(\sigma', \tau')),\ \sigma'\,) \in R$$

which proves (7.11)(ii). Since the pair we started with, was arbitrary, we have proved that R is a bisimulation. Now the equality

$$\mathsf{even} \circ \mathsf{zip} \;=\; \pi_1$$

follows by coinduction. □

Exercise 98. Show that the function $\mathsf{zip} \circ \langle \mathsf{even}, \mathsf{odd} \rangle \colon A^\omega \to A^\omega$ with

$$\mathsf{zip} \circ \langle \mathsf{even}, \mathsf{odd} \rangle(\sigma) \;=\; \mathsf{zip}(\mathsf{even}(\sigma), \mathsf{odd}(\sigma))$$

is a stream homomorphism from $(A^\omega, \langle \mathsf{i}_{st}, \mathsf{d}_{st} \rangle$ to itself. Use this to prove

$$\mathsf{zip}(\mathsf{even}(\sigma), \mathsf{odd}(\sigma)) \;=\; \sigma$$

for all $\sigma \in A^\omega$. Alternatively, construct a suitable stream bisimulation on A^ω to prove the same fact by coinduction (7.10). □

Exercise 99. For all functions $f\colon A \to A$ and $g\colon A \to A$, prove that

$$\mathsf{apply}_g \circ \mathsf{apply}_f \;=\; \mathsf{apply}_{g\circ f}$$

by coinduction. □

Exercise 100. Prove by coinduction that the operation of sum $\sigma+\tau$, as defined in Exercise 88, is commutative and associative. □

Exercise 101 (2-step coinduction). Let us call a relation $R \subseteq A^\omega \times A^\omega$ a *2-step bisimulation* if, for all $(\sigma, \tau) \in R$,

$$(i)\;\; \sigma(0) = \tau(0),\;\; \sigma(1) = \tau(1) \quad \text{and} \quad (ii)\;\; (\sigma'', \tau'') \in R$$

Let $\sim_2$ be the union of all 2-step bisimulation relations. (a) Prove the following principle of what could be called 2-step coinduction: for all $\sigma, \tau \in A^\omega$,

$$\sigma \sim_2 \tau \quad \Longrightarrow \quad \sigma \;=\; \tau$$

using the principle of (ordinary) coinduction (7.10). (b) Use 2-step coinduction to prove

$$\mathsf{exchange}(\sigma) = \mathsf{zip}(\mathsf{odd}(\sigma),\, \mathsf{even}(\sigma))$$

for all $\sigma \in A^\omega$, with exchange as defined in Example 92. □

Exercise 102. Use coinduction to prove that the function $\parallel\colon \mathbb{N}^\omega \times \mathbb{N}^\omega \to \mathbb{N}^\omega$ introduced in Exercise 91 is idempotent: $\sigma \parallel \sigma = \sigma$, commutative: $\sigma \parallel \tau = \tau \parallel \sigma$, and associative: $(\sigma \parallel \tau) \parallel \rho = \sigma \parallel (\tau \parallel \rho)$, for all $\sigma, \tau, \rho \in \mathbb{N}^\omega$. □

In the following example, we illustrate a general technique that is often helpful in the construction of suitable bisimulation relation.

Example 103 (distributivity of shuffle product). We show by coinduction that shuffle product distributes over sum:

$$\sigma \otimes (\tau + \rho) \;=\; (\sigma \otimes \tau) + (\sigma \otimes \rho) \tag{7.12}$$

Sum and shuffle product were defined in Exercises 88 and 94, from which we repeat the following identities:

$$\begin{aligned} (\sigma+\tau)(0) &= \sigma(0) + \tau(0) & \qquad (\sigma+\tau)' &= \sigma' + \tau' \\ (\sigma\otimes\tau)(0) &= \sigma(0) \cdot \tau(0) & \qquad (\sigma\otimes\tau)' &= (\sigma' \otimes \tau) \;+\; (\sigma \otimes \tau') \end{aligned} \tag{7.9}$$

We define $R \subseteq \mathbb{N}^\omega \times \mathbb{N}^\omega$ by

$$R \;=\; \{\, (\sigma \otimes (\tau + \rho),\; (\sigma \otimes \tau) + (\sigma \otimes \rho)) \;\mid\; \sigma, \tau, \rho \in \mathbb{N}^\omega \,\}$$

and check whether R is a bisimulation. The initial values of the streams in every pair are equal (because the product of natural numbers distributes over sum). Computing derivatives gives

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

where in the last equality we have used the fact that $+$ is commutative and associative (Exercise 100). Clearly, the resulting pair of derivatives

$$(\, (\sigma \otimes (\tau + \rho))' \,,\; ((\sigma \otimes \tau) + (\sigma \otimes \rho))' \,) \;=$$

$$(\, (\sigma' \otimes (\tau + \rho)) + (\sigma \otimes (\tau' + \rho')) \,,\; ((\sigma' \otimes \tau) + (\sigma' \otimes \rho)) \;+\; ((\sigma \otimes \tau') + (\sigma \otimes \rho')) \,)$$

is not in R, and so R is not a bisimulation. Note, however, that the latter pair is of the shape

$$(\alpha_1 + \alpha_2,\; \beta_1 + \beta_2)$$

for streams $\alpha_1, \alpha_2, \beta_1, \beta_2$ with $(\alpha_1, \beta_1) \in R$ and $(\alpha_2, \beta_2) \in R$. This suggests the definition of the following extension of R. We define $R^+ \subseteq \mathbb{N}^\omega \times \mathbb{N}^\omega$ as the smallest relation satisfying

1. $R \subseteq R^+$
2. if $(\alpha_1, \beta_1) \in R^+$ and $(\alpha_2, \beta_2) \in R^+$ then $(\alpha_1 + \alpha_2,\; \beta_1 + \beta_2) \in R^+$

It is now straightforward to prove that R^+ is a bisimulation, which implies (7.12), by coinduction. □

Exercise 104. Prove the distributivity of $\otimes$ over $+$ *without* coinduction, using the defining Equation (7.8) from Exercise 94. □

The relation R in the coinductive proof in Example 103 above is not a bisimulation, but its closure under sum, R^+, is. For that reason, we call such a relation R a *bisimulation up-to-sum.*

Definition 105 (bisimulation up-to-sum). A relation $R \subseteq \mathbb{N}^\omega \times \mathbb{N}^\omega$ is called a *bisimulation up-to-sum* if, for all $(\sigma, \tau) \in R$,

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

where $R^+ \subseteq \mathbb{N}^\omega \times \mathbb{N}^\omega$ is the smallest relation such that

1. $R \subseteq R^+$
2. if $(\alpha_1, \beta_1) \in R^+$ and $(\alpha_2, \beta_2) \in R^+$ then $(\alpha_1 + \alpha_2,\ \beta_1 + \beta_2) \in R^+$ □

Proposition 106 (coinduction up-to-sum). If $R \subseteq \mathbb{N}^\omega \times \mathbb{N}^\omega$ is a bisimulation up-to-sum and $(\sigma, \tau) \in R$, then $\sigma = \tau$.

Proof: If R is a bisimulation up-to-sum, then one easily shows that R^+ is a bisimulation, by induction on its definition. Since $(\sigma, \tau) \in R \subseteq R^+$, the equality $\sigma = \tau$ follows by coinduction, Theorem 96. □

The notion of bisimulation up-to-sum and the corresponding proof principle of coinduction up-to-sum are particular instances of a far more general situation, in which a coalgebraic structure – here the coalgebra of streams – is combined with an algebraic structure – here streams together with the operation sum.

Exercise 107 (associativity of shuffle product). For all $\sigma, \tau, \rho \in \mathbb{N}^\omega$,

$$(\sigma \otimes \tau) \otimes \rho \ = \ \sigma \otimes (\tau \otimes \rho)$$

Prove this, first without coinduction, using the defining Equation (7.8) from Exercise 94, and then by coinduction up-to-sum, using the identities in (7.9). □

Exercise 108. Use coinduction up-to-sum to prove that if two functions $f, g \colon \mathbb{N}^\omega \times \mathbb{N}^\omega \to \mathbb{N}^\omega$ both satisfy the same equations as the shuffle product ((7.9) in Exercise 94):

$$\begin{array}{lcl} f(\sigma, \tau)(0) \ = \ \sigma(0) \cdot \tau(0) & \qquad & f(\sigma \otimes \tau)' \ = \ f(\sigma', \tau) \ + \ f(\sigma, \tau') \\ g(\sigma, \tau)(0) \ = \ \sigma(0) \cdot \tau(0) & \qquad & g(\sigma \otimes \tau)' \ = \ g(\sigma', \tau) \ + \ g(\sigma, \tau') \end{array}$$

then they must be equal: $f = g$. □

7.5 Moessner's Theorem

As a further illustration of the power of the coinduction proof method, we present a coinductive proof of an entertaining and non-trivial characterisation of the family of streams

$$(1^n, 2^n, 3^n, \ldots) \qquad (n \geqslant 2)$$

It is well-known that if, from the stream of positive natural numbers,

$$\mathsf{nat} = (1, 2, 3, \ldots)$$

one drops every second element, obtaining the stream of odd natural numbers,

$$(1, 3, 5, \ldots)$$

and finally one forms the corresponding stream of its partial sums,

$$(1,\ 1+3,\ 1+3+5, \ldots)$$

then one has obtained the stream of all the positive natural numbers squared:

$$(1^2, 2^2, 3^2, \ldots)$$

The cubes of the natural numbers can be obtained in a similar fashion, by dropping from the stream of positive natural numbers every *third* element, obtaining

$$(1, 2, 4, 5, 7, 8, \ldots)$$

then forming the corresponding stream of partial sums:

$$(1, 1+2, 1+2+4, \ldots)$$

and then, as above, dropping every second element:

$$(1, 7, 19, \ldots)$$

and finally forming the corresponding stream of partial sums again:

$$(1,\ 1+7,\ 1+7+19, \ldots)$$

One then has obtained the stream of all positive natural numbers cubed:

$$(1^3, 2^3, 3^3, , \ldots)$$

Moessner [Moe51] described how the above procedure of repeatedly alternating a drop and a partial sum operation can be generalised to obtain the stream

$$(1^n, 2^n, 3^n, \ldots)$$

for every $n \geqslant 1$: drop every nth element and form the subsequent stream of partial sums, then drop every $(n-1)$th element and form the subsequent stream of partial sums, etc.

Below we shall first formalise Moessner's procedure by introducing corresponding stream functions. Then we shall prove the theorem by coinduction up-to-sum (Proposition 106).

First, we define the following family of *drop operators*:

$$\begin{aligned} &D_n\colon \mathbb{N}^\omega \to \mathbb{N}^\omega && (n \geqslant 2) \\ &D_n(\sigma) \;=\; (\sigma(0), \ldots, \sigma(n-2), \sigma(n), \ldots, \sigma(2n-2), \sigma(2n), \ldots) && (7.13) \end{aligned}$$

which drop the elements

$$(\sigma(n-1),\ \sigma(2n-1),\ \sigma(3n-1),\ \ldots)$$

from the stream σ. Then we define the operator of *partial sums*:

$$\begin{aligned} &\Sigma\colon \mathbb{N}^\omega \to \mathbb{N}^\omega \\ &\Sigma(\sigma) \;=\; (\sigma(0),\ \sigma(0)+\sigma(1),\ \sigma(0)+\sigma(1)+\sigma(2),\ \ldots) \qquad (7.14) \end{aligned}$$

Finally, we define the *Hadamard product*:

$$\begin{aligned} &\odot\colon (\mathbb{N}^\omega \times \mathbb{N}^\omega) \to \mathbb{N}^\omega \\ &\sigma \odot \tau \;=\; (\sigma(0)\cdot\tau(0),\ \sigma(1)\cdot\tau(1),\ \sigma(2)\cdot\tau(2),\ \ldots) \qquad (7.15) \end{aligned}$$

We define

$$\overline{1} = (1, 1, 1, \ldots)$$

and note that $\overline{1} \odot \sigma = \sigma$.

Notation 109. We will write

$$\sigma^{\langle 0 \rangle} \;=\; \overline{1} \qquad\qquad \sigma^{\langle k+1 \rangle} \;=\; \sigma^{\langle k \rangle} \odot \sigma$$

Note the difference between $\sigma^{\langle n \rangle}$ and $\sigma^{(n)}$: the latter is the notation for higher-order stream derivatives, introduced in Definition 76. □

Moessner's Theorem can now be formally expressed as the following equality of streams:

$$\Sigma D_2 \Sigma D_3 \cdots \Sigma D_n\ \mathsf{nat} \;=\; \mathsf{nat}^{\langle n \rangle} \qquad (7.16)$$

for all $n \geqslant 2$. Note that on the left, we are using the following notationally convenient shorthand:

$$\Sigma D_2 \Sigma D_3 \cdots \Sigma D_n\ \mathsf{nat} \;=\; (\,\Sigma \circ D_2 \circ \Sigma \circ D_3 \circ \cdots \circ \Sigma \circ D_n\,)\,(\mathsf{nat})$$

It will furthermore be convenient to reformulate this expression as follows:

$$\begin{aligned} \Sigma D_2 \Sigma D_3 \cdots \Sigma D_n\ \mathsf{nat} \;&=\; \Sigma D_2 \Sigma D_3 \cdots \Sigma D_n \Sigma\ \overline{1} && (\text{since } \mathsf{nat} = \Sigma(\overline{1})) \\ &=\; \Sigma D_2 \Sigma D_3 \cdots \Sigma D_n \Sigma D_{n+1}\ \overline{1} && (\text{since } \overline{1} = D_{n+1}(\overline{1})) \end{aligned}$$

All in all, we have obtained the following formalisation of Moessner's theorem.

Theorem 110 (Moessner).

$$\Sigma D_2 \Sigma D_3 \cdots \Sigma D_{n+1}\ \overline{1} \ = \ \mathsf{nat}^{\langle n \rangle} \qquad (n \geqslant 1)$$

Exercise 111. Try and prove $\Sigma D_2 \Sigma D_3\ \overline{1} = \mathsf{nat} \odot \mathsf{nat}$ without coinduction. The same question for $\Sigma D_2 \Sigma D_3 \Sigma D_4\ \overline{1} = \mathsf{nat} \odot \mathsf{nat} \odot \mathsf{nat}$. Any thoughts about the general case? □

Before embarking on a coinductive proof of Theorem 110, we first collect a number of basic properties of the operators above. For the drop operators, we have

$$\begin{aligned} D_n(\sigma)(i) \ &= \ \sigma(i) && (0 \leqslant i \leqslant n-2) \\ D_n(\sigma)^{(n-1)} \ &= \ D_n\left(\sigma^{(n)}\right) && (7.17) \end{aligned}$$

(Higher-order stream derivatives were defined in Definition 76.) For the stream of partial sums, we have

$$\Sigma(\sigma)(0) \ = \ \sigma(0) \qquad\qquad \Sigma(\sigma)' \ = \ \overline{\sigma(0)} \ + \ \Sigma(\sigma') \qquad (7.18)$$

with

$$\overline{m} \ = \ (m, m, m, \ldots) \qquad (m \geqslant 0)$$

For the higher-order derivatives of $\Sigma(\sigma)$, we have, for all $k \geqslant 1$,

$$\Sigma(\sigma)^{(k)} \ = \ \overline{\sigma[k]} \ + \ \Sigma(\sigma^{(k)}) \qquad (7.19)$$

where we write

$$\sigma[k] \ = \ \sigma(0) + \sigma(1) + \cdots + \sigma(k-1) \qquad (7.20)$$

The proof of identity (7.19) is straightforward and uses, for all $l, m \geqslant 0$,

$$\overline{l} + \overline{m} \ = \ \overline{l+m}$$

If we define

$$k\sigma \ = \ (k \cdot \sigma(0), k \cdot \sigma(1), k \cdot \sigma(2), \ldots)$$

then

$$(k+l)\sigma \ = \ k\sigma + l\sigma \qquad\qquad \overline{k} = k\overline{1}$$

Exercise 112. Let $f\colon \mathbb{N}^\omega \to \mathbb{N}^\omega$ be a function on streams satisfying

$$f(\sigma)(0) \ = \ \sigma(0) \qquad\qquad f(\sigma)' \ = \ \sigma' + f(\sigma)$$

Show that $f(\sigma) = \Sigma(\sigma)$. (The above equation was suggested by Jurriaan Rot.) □

Both D_n and Σ preserve sums:

$$D_n(\sigma + \tau) \;=\; D_n(\sigma) \;+\; D_n(\tau) \qquad\qquad \Sigma(\sigma + \tau) \;=\; \Sigma(\sigma) \;+\; \Sigma(\tau) \tag{7.21}$$

Furthermore,

$$D_n(k\sigma) \;=\; k\, D_n(\sigma) \qquad\qquad \Sigma(k\sigma) \;=\; k\, \Sigma(\sigma) \tag{7.22}$$

The Hadamard product satisfies

$$(\sigma \odot \tau)(0) \;=\; \sigma(0) \cdot \tau(0) \qquad\qquad (\sigma \odot \tau)' \;=\; \sigma' \odot \tau'$$

Moreover, Hadamard product is commutative and associative, and distributes over sum:

$$\sigma \odot (\tau + \rho) \;=\; (\sigma \odot \tau) \;+\; (\sigma \odot \rho)$$

Finally, we observe that for the stream nat, we have

$$\begin{aligned} \mathsf{nat}' &= (1, 2, 3, \ldots)' \\ &= (2, 3, 4, \ldots) \\ &= (1, 1, 1, \ldots) \;+\; (1, 2, 3, \ldots) \\ &= \overline{1} \;+\; \mathsf{nat} \end{aligned} \tag{7.23}$$

Exercise 113. Warming up: prove $\Sigma D_2 \Sigma D_3 \, \overline{1} = \mathsf{nat} \odot \mathsf{nat}$ by coinduction. □

Now we are sufficiently prepared to prove Moessner's Theorem 110. We define

$$R \;\subseteq\; \mathbb{N}^\omega \times \mathbb{N}^\omega$$

by

$$R \;=\; \{(\overline{1}, \overline{1})\} \cup \{\big(\; \Sigma D_2 \Sigma D_3 \cdots \Sigma D_{n+1} \, \overline{1},\; \mathsf{nat}^{\langle n \rangle} \;\big) \mid n \geqslant 1\} \tag{7.24}$$

We will show that R is a bisimulation up-to-sum, satisfying conditions (i) and (ii) from Definition 105. For (i), we observe that the initial values of all streams in R are 1. For (ii), we compute the derivatives of all streams in R. Starting with the stream $\mathsf{nat}^{\langle n \rangle}$, we have

$$\begin{aligned} (\mathsf{nat}^{\langle n \rangle})' &= (\mathsf{nat}')^{\langle n \rangle} \\ &= (\overline{1} + \mathsf{nat})^{\langle n \rangle} && \text{(by (7.23))} \\ &= \binom{n}{0} \overline{1} + \binom{n}{1} \mathsf{nat} + \binom{n}{2} \mathsf{nat}^{\langle 2 \rangle} + \cdots + \binom{n}{n} \mathsf{nat}^{\langle n \rangle} \end{aligned} \tag{7.25}$$

We will be able to conclude that R satisfies (ii) from Definition 105 if we can prove that

$$\begin{aligned} &\left(\Sigma D_2 \Sigma D_3 \cdots \Sigma D_{n+1}\, \overline{1}\, \right)' \\ =\ &\binom{n}{0}\overline{1} \ +\ \binom{n}{1}\Sigma D_2\, \overline{1} \ +\ \binom{n}{2}\Sigma D_2 \Sigma D_3\, \overline{1} \ +\ \cdots \ +\ \binom{n}{n}\Sigma D_2 \Sigma D_3 \cdots \Sigma D_{n+1}\, \overline{1} \end{aligned} \tag{7.26}$$

To this end, we compute as follows:

$$\begin{aligned} &(\Sigma D_2 \Sigma D_3 \Sigma D_4 \cdots \Sigma D_{n+1} \overline{1})' \\ &= \overline{(D_2 \Sigma D_3 \Sigma D_4 \cdots \Sigma D_{n+1}\, \overline{1})[1]} \ +\ \Sigma (D_2 \Sigma D_3 \Sigma D_4 \cdots \Sigma D_{n+1}\, \overline{1})' && \text{(by (7.19))} \\ &= \overline{a_0} \ +\ \Sigma D_2 (\Sigma D_3 \Sigma D_4 \cdots \Sigma D_{n+1}\, \overline{1})^{(2)} && \text{(by (7.17))} \end{aligned}$$

where

$$a_0 \ =\ (D_2 \Sigma D_3 \Sigma D_4 \cdots \Sigma D_{n+1}\, \overline{1})[1]$$

the value of which we will worry about in a second. We continue with

$$\begin{aligned} &= \overline{a_0} \ +\ \Sigma D_2 \left(\overline{(D_3 \Sigma D_4 \cdots \Sigma D_{n+1}\, \overline{1})[2]} \ +\ \Sigma (D_3 \Sigma D_4 \cdots \Sigma D_{n+1}\, \overline{1})^{(2)} \right) && \text{(by (7.19))} \\ &= \overline{a_0} \ +\ \Sigma D_2 \left(\, \overline{a_1} \ +\ \Sigma D_3 (\Sigma D_4 \cdots \Sigma D_{n+1}\, \overline{1})^{(3)} \, \right) && \text{(by (7.17))} \\ &= \overline{a_0} \ +\ (\Sigma D_2 \overline{a_1}) \ +\ \left(\Sigma D_2 \Sigma D_3 (\Sigma D_4 \cdots \Sigma D_{n+1} \overline{1})^{(3)} \right) && \text{(by (7.21))} \end{aligned}$$

with

$$a_1 \ =\ (D_3 \Sigma D_4 \cdots \Sigma D_{n+1}\, \overline{1})[2]$$

Continuing this way, we ultimately arrive at

$$\begin{aligned} &= \cdots \\ &= \overline{a_0} \ +\ (\Sigma D_2 \overline{a_1}) \ +\ \cdots \ +\ (\Sigma D_2 \Sigma D_3 \cdots \Sigma D_n\, \overline{a_{n-1}}) \ +\ \left(\Sigma D_2 \Sigma D_3 \cdots \Sigma D_{n+1}\, (\overline{1})^{(n+1)} \right) \end{aligned}$$

where

$$a_i \ =\ (D_{i+2} \Sigma D_{i+3} \cdots \Sigma D_{n+1} \overline{1})[i+1] \qquad (0 \leqslant i \leqslant n-1)$$

Since

$$\overline{1}^{(n+1)} = \overline{1}$$

the $(n+1)$th derivative evaporates in the next step, yielding

$$= \overline{a_0} \ +\ (\Sigma D_2 \overline{a_1}) \ +\ \cdots \ +\ (\Sigma D_2 \Sigma D_3 \cdots \Sigma D_n \overline{a_{n-1}}) \ +\ \left(\Sigma D_2 \Sigma D_3 \cdots \Sigma D_{n+1} \overline{1} \right)$$

Finally, we use

$$\Sigma D_i\, \overline{m} \ =\ m \Sigma D_i\, \overline{1} \qquad (m \geqslant 0)$$

to obtain:

$$= a_0 \overline{1} \ +\ \left(a_1 \Sigma D_2\, \overline{1} \right) \ +\ \cdots \ +\ \left(a_{n-1} \Sigma D_2 \Sigma D_3 \cdots \Sigma D_n\, \overline{1} \right) \ +\ \left(\Sigma D_2 \Sigma D_3 \cdots \Sigma D_{n+1}\, \overline{1} \right)$$

We conclude the proof of (7.26) by showing that

$$a_i \;=\; \binom{n}{i} \qquad\qquad (0 \leqslant i \leqslant n-1)$$

By the definition of a_i above, this is equivalent to

$$\big(D_{k+1}\Sigma D_{k+2}\Sigma \cdots D_n \Sigma D_{n+1}\,\overline{1}\,\big)\,[k] \;=\; \binom{n}{k-1} \tag{7.27}$$

for all $n \geqslant 1$ and $1 \leqslant k \leqslant n$, which we prove next. We begin by observing that

$$\big(D_{k+1}\Sigma D_{k+2}\Sigma \cdots D_n \Sigma D_{n+1}\,\overline{1}\,\big)\,[k] \;=\; \Sigma^{n-k}\,\big(\overline{1}\big)\,[k] \tag{7.28}$$

where as usual,

$$\Sigma^0 \;=\; id \qquad\qquad \Sigma^{m+1} \;=\; \Sigma \circ \Sigma^m$$

Identity (7.28) follows from the observation that, for any stream σ, the value of

$$\sigma[k] = \sigma(0) + \sigma(1) + \cdots + \sigma(k-1)$$

depends on only the first k values of σ. The proof of (7.27) is then concluded with

$$\Sigma^{n-k}\,\big(\overline{1}\big)\,[k] \;=\; \Sigma^{n-k+1}\,\big(\overline{1}\big)\,(k-1) \;=\; \binom{n}{n-k+1} \;=\; \binom{n}{k-1}$$

where the second equality follows from Pascal's triangle: for all $m \geqslant 1$,

$$\Sigma^m\,\big(\overline{1}\big) \;=\; (\;\binom{m}{m},\;\binom{m+1}{m},\;\binom{m+2}{m},\;\ldots\;) \tag{7.29}$$

This concludes our proof that R, defined in (7.24), is a bisimulation up-to-sum. Moessner's Theorem 110 now follows by coinduction up-to-sum, Proposition 106.

Exercise 114. Prove identity (7.29). $\square$

Exercise 115. Prove the following generalisation of Moessner's Theorem, due to Long and Salié: for all $n, a, d \geqslant 1$,

$$\Sigma D_2 \Sigma D_3 \cdots \Sigma D_{n+1}\,(a, d, d, d, \ldots) \;=\; (\Sigma\;(a, d, d, d, \ldots)) \odot \mathsf{nat}^{\langle n-1\rangle} \tag{7.30}$$

$\square$

7.6 The heart of the matter: circularity

Moessner's Theorem is formulated as the equality of two streams: for all $n \geqslant 1$,

$$\Sigma D_2 \Sigma D_3 \cdots \Sigma D_{n+1}\ \overline{1} \ = \ \mathsf{nat}^{\langle n \rangle} \qquad \text{(Theorem 110)}$$

We proved, by coinduction up-to-sum, that these two streams *are* the same by showing that they *behave* the same. And we proved that they behave the same by constructing a bisimulation relation up-to-sum, which can be viewed as a witness of the behavioural equivalence of the two streams.

The behaviour of any stream σ consists of a transition to its derivative σ' in combination with an output, given by its initial value $\sigma(0)$. With a picture, we can represent the transitions of σ to its consecutive derivatives $\sigma', \sigma^{(2)}, \sigma^{(3)}, \ldots$ as

$$\sigma \longrightarrow \sigma' \longrightarrow \sigma^{(2)} \longrightarrow \sigma^{(3)} \longrightarrow \cdots$$

Here and below, we omit the respective outputs $\sigma(i)$ from this graphical representation of the behaviour of a stream σ.

For instance, for the stream $\overline{1} = (1, 1, 1, \ldots)$, which is the starting point of Moessner's construction, we have

$$\overline{1} \longrightarrow \overline{1} \longrightarrow \overline{1} \longrightarrow \cdots$$

or, equivalently:

$$\begin{tikzcd} \overline{1} \arrow[loop right] \end{tikzcd} \qquad (7.31)$$

We clearly prefer the second picture because it is finite and because of its graphical emphasis on the *circularity* of the behaviour of $\overline{1}$. For the stream $\mathsf{nat} = (1, 2, 3, \ldots)$, we have

$$\mathsf{nat} \longrightarrow \overline{1} + \mathsf{nat} \qquad (\mathsf{nat}' = \overline{1} + \mathsf{nat}, \text{ identity (7.23)})$$

There is implicit circularity also in this picture, since the stream nat occurs both at the left and at the right of the transition arrow. Using the *algebraic* operation of sum:

$$+ \colon (\mathbb{N}^\omega \times \mathbb{N}^\omega) \to \mathbb{N}^\omega$$

we can make this circularity explicit by *splitting* the transition from nat to $\overline{1} + \mathsf{nat}$ into two transitions, as follows:

$$\begin{tikzcd} \mathsf{nat} & \mathsf{nat} \arrow[l, "1"'] \arrow[r, "1"] & \overline{1} \end{tikzcd}$$

Writing the left transition as a self arrow, and combining this picture with that of (7.31) above, we obtain the following graphical representation of the behaviour of nat:

$$\begin{tikzcd} \mathsf{nat} \arrow[loop left, "1"] \arrow[r, "1"] & \overline{1} \arrow[loop right, "1"] \end{tikzcd}$$

More generally, if the derivative of a stream $\sigma \in \mathbb{N}^\omega$ can be split into a finite number of summands:

$$\sigma' \;=\; k_0\tau_0 \;+\; k_1\tau_1 \;+\; \cdots \;+\; k_m\tau_m$$

for $k_i \geqslant 0$ and $\tau_i \in \mathbb{N}^\omega$, then we will write

$$\begin{tikzcd}
 & \sigma \arrow[dl, bend right, "k_0"'] \arrow[d, "k_1"'] \arrow[drr, bend left, "k_m"] & & \\
\tau_0 & \tau_1 & \cdots & \tau_m
\end{tikzcd}$$

We shall see later, in Chapter 14, that this is actually a graphical notation for so-called *weighted stream automata*, but for now, we simply view it as a convenient representation of derivatives that can be written as finite sums.

Using this notation, we can represent the behaviour of function applications, such as $\Sigma(\mathsf{nat})$ and $D_2(\mathsf{nat})$, as well: since

$$\begin{aligned}
\Sigma(\mathsf{nat})' \;&=\; \overline{1} \;+\; \Sigma\left(\overline{1} + \mathsf{nat}\right) \;=\; \overline{1} \;+\; \mathsf{nat} \;+\; \Sigma(\mathsf{nat}) \\
D_2(\mathsf{nat})' \;&=\; D_2\left(\mathsf{nat}^{(2)}\right) \;=\; D_2\left(\overline{2} + \mathsf{nat}\right) \;=\; 2(\overline{1}) \;+\; D_2(\mathsf{nat})
\end{aligned}$$

we have

$$\begin{tikzcd}
\Sigma(\mathsf{nat}) \arrow[loop above, "1"] \arrow[r, "1"] \arrow[rr, bend right, "1"'] & \mathsf{nat} \arrow[loop above, "1"] \arrow[r, "1"] & \overline{1} \arrow[loop right, "1"] & & D_2(\mathsf{nat}) \arrow[loop above, "1"] \arrow[r, "2"] & \overline{1} \arrow[loop right, "1"]
\end{tikzcd}$$

Returning now to the proof of Moessner's Theorem, the characterisation of the derivative of $\mathsf{nat}^{\langle n \rangle}$, in (7.25), can be expressed by the following picture:

$$\begin{tikzcd}
\overline{1} \arrow[loop left, "\binom{0}{0}"] & \mathsf{nat} \arrow[l, "\binom{1}{0}"'] \arrow[loop below, "\binom{1}{1}"] & \mathsf{nat}^{\langle 2 \rangle} \arrow[l, "\binom{2}{1}"'] \arrow[ll, bend right, "\binom{2}{0}"'] \arrow[loop above, "\binom{2}{2}"] & \cdots \arrow[l, "\binom{3}{2}"'] & \mathsf{nat}^{\langle n \rangle} \arrow[l, "\binom{n}{n-1}"'] \arrow[ll, bend left, "\binom{n}{2}"] \arrow[lll, bend left, "\binom{n}{1}"] \arrow[llll, bend left, "\binom{n}{0}"] \arrow[loop above, "\binom{n}{n}"]
\end{tikzcd}$$

Based on (7.26), the derivative of $\Sigma D_2 \Sigma D_3 \cdots \Sigma D_{n+1}\ \overline{1}$ gives rise to exactly the same picture. Because of the structural similarity between the split derivatives of both streams in the equation of Moessner's Theorem 110, it was possible to define a bisimulation up-to-sum relation R, in (7.24), and thereby prove the theorem by coinduction up-to-sum.

Exercise 116 (factorial numbers). Observe that the stream $\phi = (0!, 1!, 2!, \ldots)$ satisfies

$$\phi' \;=\; \phi \odot \mathsf{nat} \tag{7.32}$$

Use Equation (7.32) to prove that

$$\begin{tikzcd}
\phi \arrow[r, "\binom{0}{0}"] & \phi_1 \arrow[loop above, "\binom{1}{0}"] \arrow[r, bend left, "\binom{1}{1}"] & \phi_2 \arrow[l, "\binom{2}{0}"] \arrow[loop above, "\binom{2}{1}"] \arrow[r, bend left, "\binom{2}{2}"] & \cdots & \phi_n \arrow[l, "\binom{n}{n-2}"'] \arrow[ll, bend left, "\binom{n}{1}"] \arrow[lll, bend left=40, "\binom{n}{0}"] \arrow[loop above, "\binom{n}{n-1}"] \arrow[r, "\binom{n}{n}"] & \phi_{n+1}
\end{tikzcd}$$

where $\phi_n = \phi \odot \mathsf{nat}^{\langle n \rangle}$. □

Example 117 (Paasche's Theorem). Moessner's construction uses drop operators

$$D_n \colon \mathbb{N}^\omega \to \mathbb{N}^\omega \qquad (n \geqslant 2)$$

satisfying

$$\begin{aligned} D_n(\sigma)(i) &= \sigma(i) && (0 \leqslant i \leqslant n-2) \\ D_n(\sigma)^{(n-1)} &= D_n\left(\sigma^{(n)}\right) && ((7.17)) \end{aligned}$$

The operators D_n drop elements at regular intervals of length n. The following operators

$$d_n \colon \mathbb{N}^\omega \to \mathbb{N}^\omega \qquad (n \geqslant 2)$$

are similar but drop elements at intervals of *increasing* length:

$$\begin{aligned} d_n(\sigma)(i) &= \sigma(i) && (0 \leqslant i \leqslant n-2) \\ d_n(\sigma)^{(n-1)} &= d_{n+1}\left(\sigma^{(n)}\right) \end{aligned}$$

On the right, we now have d_{n+1} instead of d_n, implying that a next element will be dropped after $n+1$ instead of n elements. For instance, we have

$$d_2(\mathsf{nat}) \;=\; (1, \cancel{2}, 3, 4, \cancel{5}, 6, 7, 8, \cancel{9}, \ldots)$$

The operators d_n are used in the following variant of Moessner's construction, called *Paasche's Theorem*. Starting again with the stream $\overline{1}$ of ones, we construct a new stream π by repeatedly carrying out the following procedure. We take the initial value of $\overline{1}$ and define

$$\pi(0) \;=\; \overline{1}(0) \;=\; 1 \;=\; 0!$$

Next we take the derivative of $\overline{1}$ and apply first d_2 and then Σ to it, yielding:

$$\Sigma\left(d_2\left(\overline{1}'\right)\right) \;=\; \Sigma\left(d_2\left(\overline{1}\right)\right) \;=\; \Sigma\left(\overline{1}\right) \;=\; \mathsf{nat}$$

We continue by applying the same procedure to the resulting stream nat. We define

$$\pi(1) \;=\; \mathsf{nat}(0) \;=\; 1 \;=\; 1!$$

and compute

$$\Sigma\left(d_2\left(\mathsf{nat}'\right)\right) \;=\; \Sigma\left(d_2\left((2,3,4,\ldots)\right)\right) \;=\; \Sigma(2,4,5,7,8,9,\ldots) \;=\; (2,6,11,18,26,\ldots)$$

Then we define

$$\pi(2) \;=\; (2,6,11,18,26,\ldots)(0) \;=\; 2 \;=\; 2!$$

and continue with

$$\Sigma\left(d_2\left((2,6,11,18,26,\ldots)'\right)\right) \;=\; \Sigma\left(d_2\left((6,11,18,26,\ldots)\right)\right) \;=\; (6,24,\ldots)$$

The next value of π is

$$\pi(3) \;=\; (6,24,\ldots)(0) \;=\; 6 \;=\; 3!$$

and so on. Paasche proved that the result of this procedure is $\pi = (0!, 1!, 2!, \ldots)$. □

Exercise 118 (proof of Paasche's Theorem). The construction in Example 117, of repeatedly taking the initial value of a stream and then continuing with applying Σ after d_2 to its derivative, can be formalised as follows. If we define

$$P\colon \mathbb{N}^\omega \to \mathbb{N}^\omega \qquad\qquad P(\sigma)(0) \;=\; \sigma(0) \qquad\qquad P(\sigma)' \;=\; P\left(\Sigma\left(d_2\left(\sigma'\right)\right)\right)$$

and

$$\pi \;=\; P\left(\overline{1}\right) \qquad\qquad\qquad \phi \;=\; (0!, 1!, 2!, \ldots)$$

then Paasche's Theorem is the following equality:

$$\pi \;=\; \phi \tag{7.33}$$

Towards a proof by coinduction, let

$$\pi_n \;=\; P\Sigma d_2 \Sigma d_3 \cdots \Sigma d_{n+1}\, \overline{1} \qquad (n \geqslant 2)$$

By definition of P, we have $\pi' = \pi_1$. Prove, for all $n \geqslant 2$,

$$\pi_n{}' \;=\; \binom{n}{0}\pi_1 \;+\; \binom{n}{1}\pi_2 \;+\; \binom{n}{2}\pi_3 \;+\; \cdots \;+\; \binom{n}{n-1}\pi_n \;+\; \binom{n}{n}\pi_{n+1} \tag{7.34}$$

Recall the definition of $\phi_n = \phi \odot \mathsf{nat}^{\langle n\rangle}$ from Exercise 116 and show that

$$\{(\pi,\phi)\} \;\cup\; \{(\;\pi_n,\, \phi_n\;) \mid n \geqslant 1\} \tag{7.35}$$

is a bisimulation up-to-sum. Paasche's Theorem follows by coinduction up-to-sum. □

7.7 Discussion

The present chapter is based on [Rut03a, Rut05a]. The first proof of Moessner's Theorem was given by Perron [Per51]. An alternative proof and further generalisations were provided by Paasche [Paa52] and Salié [Sal52]. All these proofs are based on a detailed bookkeeping of the elements of all the intermediate streams, and use nested inductions, involving binomial coefficients and falling factorial numbers. More details about the classical proofs of Moessner's theorem can be found in recent work [Hin08, Hin11] by Hinze, in which he has given a new proof of Moessner's theorem (and its generalisations), in a calculational style. In [KS13], Kozen and Silva give a short algebraic proof, using multivariate generating functions, of a general theorem that contains Moessner's and Paasche's theorems (and some further generalisations) as special cases. A first proof of Moessner's theorem by coinduction, rather more complicated than the one presented above, was given by Niqui en Rutten [NR11]; (a minor variation of) that proof was later formally verified by Krebbers, Parlant and Silva [KPS16], using the theorem prover COQ.

Chapter 8

Deterministic automata

Let A be a set and $2 = \{0,1\}$. A *deterministic automaton* with *outputs* in 2 and *inputs* from A is a pair

$$(S, \langle \mathsf{o}, \mathsf{tr} \rangle) \qquad \langle \mathsf{o}, \mathsf{tr} \rangle \colon S \to 2 \times S^A \qquad \mathsf{o} \colon S \to 2 \qquad \mathsf{tr} \colon S \to S^A$$

consisting of a set S of *states* and a *structure map* $\langle \mathsf{o}, \mathsf{tr} \rangle$. This structure map is a pair consisting of an *output* function o and a *transition* function tr. The output function o maps every state $s \in S$ to an output $\mathsf{o}(s) \in 2$. The transition function

$$\mathsf{tr} \colon S \to S^A \qquad S^A \;=\; \{\, f \mid f \colon A \to S \,\}$$

assigns to every state $s \in S$ a function $\mathsf{tr}(s) \colon A \to S$ that maps any *input* $a \in A$ to a next state $\mathsf{tr}(s)(a) \in S$. In the present chapter, we shall call a deterministic automaton with outputs in 2 and inputs from A simply by the name *automaton*.

The transition function of deterministic automata differs from that of stream systems in that it depends on inputs. As a consequence, the behaviour of automata will be characterised in terms of *languages* over the set of inputs A. The set $\mathcal{P}(A^*)$ of all such languages turns out to be a *final* automaton. The present chapter offers further possibilities to become familiar with both definitions and proofs by coinduction.

The present automata are called *deterministic* because applying the transition function tr to $s \in S$ and $a \in A$ yields a *unique* successor state $\mathsf{tr}(s)(a)$. In Chapter 10, we shall discuss automata with *non-deterministic* transitions, where on the same input, different states may be reached.

The output of a deterministic automaton is either 0 or 1, and depends only on the present state and not on the input. In Chapter 13, we shall discuss *Mealy automata*, which are more general in allowing arbitrary sets of outputs, and in which the output of a state may depend on the input.

Remark 119. For those who have already read Chapters 2 to 5: deterministic automata are coalgebras of the functor

$$\begin{array}{lll} \mathsf{dA} \colon \mathsf{Set} \to \mathsf{Set} & \mathsf{dA}(S) \;=\; 2 \times S^A & (S \in \mathsf{Set}) \\ \mathsf{dA}(f) \colon \big(2 \times S^A\big) \to \big(2 \times T^A\big) & \mathsf{dA}(f) \;=\; 1 \times f^A & (f \colon S \to T) \end{array}$$

with

$$\left(1 \times f^A\right)((a, g)) \;=\; (a,\, f \circ g)$$

for $a \in 2$ and $g\colon A \to S$. $\square$

8.1 Basic definitions

For an automaton $(S, \langle \mathsf{o}, \mathsf{tr}\rangle)$, the following notation is often convenient:

$$s_\varepsilon \;=\; s \qquad\qquad s_{w\cdot a} \;=\; \mathsf{tr}(s_w)(a) \qquad\qquad (s \in S,\, w \in A^*)$$

where ε denotes the empty word. We call s_w the state reached from the state s on input w. We denote transitions of automata by

$$s \xrightarrow{a} t \quad\Longleftrightarrow\quad t \;=\; s_a$$

which we read as: when the automaton is in the state s and receives input a, it makes a transition to the state t. We will use also transitions with more than one label, such as

$$s \xrightarrow{a,b} t \quad\Longleftrightarrow\quad s \xrightarrow{a} t \;\text{ and }\; s \xrightarrow{b} t$$

Since outputs of automata are binary, it suffices in diagrams of automata to mark only those states $s \in S$ with $o(s) = 1$, which we shall do by simply underlining them.

Example 120. Let $A = \{a, b\}$. Here is an example of an automaton:

$$\begin{tikzcd}(S, \langle \mathsf{o}, \mathsf{tr}\rangle) \;=\; & s_0 \arrow[loop left, "b"] \arrow[r, bend left, "a"] & s_1 \arrow[l, bend left, "b"] \arrow[r, "a"] & \underline{s_2} \arrow[loop right, "{a,b}"]\end{tikzcd}$$

The definitions of o and tr can be read from the diagram: $\mathsf{o}(s_0) = \mathsf{o}(s_1) = 0$ and $\mathsf{o}(s_2) = 1$ and, for instance, $(s_1)_b = \mathsf{tr}(s_1)(b) = s_0$ and $(s_2)_a = \mathsf{tr}(s_2)(a) = s_2$. $\square$

The behaviour of automata can be conveniently described in terms of the acceptance of *languages*, that is, sets of words of inputs. Automata are therefore also called language *acceptors*. As we shall see later, in Theorem 153, this definition of the behaviour of automata will be precisely captured by the final automaton of languages.

Definition 121 (acceptance of words, languages). We call a state s in an automaton $(S, \langle \mathsf{o}, \mathsf{tr}\rangle)$ *accepting* if $\mathsf{o}(s) = 1$ and non-accepting if $\mathsf{o}(s) = 0$. We say that s *accepts a word* $w \in A^*$ if $\mathsf{o}(s_w) = 1$. We denote the set of all words accepted by a state s by

$$l(s) \;=\; \{w \in A^* \mid \mathsf{o}(s_w) = 1\}$$

and we call $l(s)$ the *language accepted* by s. $\square$

In other words, a state s accepts a word $w = a_0 a_1 \cdots a_{n-1} \in A^*$ if starting in s and inputting the consecutive inputs in w leads to a sequence of transitions ending in an accepting state:

$$s \xrightarrow{a_0} s_{a_0} \xrightarrow{a_1} s_{a_0 a_1} \xrightarrow{a_2} \cdots \xrightarrow{a_{n-1}} \underline{s_w}$$

In this diagram, we have underlined only the accepting state s_w. Some of the other states are possibly accepting as well, but this is irrelevant for the acceptance of w by s. Also note that an accepting state always accepts the empty word ε.

For the automaton in Example 120, we observe that the state s_0 accepts all words that contain at least two consecutive a's; s_1 accepts all words that either start with a, or start with b and contain at least two consecutive a's; and state s_2 accepts all words:

$$\begin{aligned} l(s_0) &= \{vaaw \mid v, w \in \{a,b\}^* \} \\ l(s_1) &= \{aw \mid w \in \{a,b\}^* \} \cup \{bvaaw \mid v, w \in \{a,b\}^* \} \\ l(s_2) &= \{a,b\}^* \end{aligned}$$

Exercise 122. Consider the following automaton:

$$(T, \langle \mathsf{o}, \mathsf{tr} \rangle) \quad = \quad \begin{tikzcd} t_0 \arrow[loop left, "b"] \arrow[r, "a"] & t_1 \arrow[loop above, "b"] \arrow[r, "a"'] & \underline{t_2} \arrow[loop right, "{a,b}"] \end{tikzcd}$$

Describe the language accepted by the state t_0. □

Example 123. Consider the following automaton with inputs from $A = \{0, 1\}$:

$$\begin{tikzcd} \underline{s_0} \arrow[r, bend left, "1"] \arrow[d, bend left, "0"] & s_1 \arrow[l, bend left, "1"] \arrow[d, bend left, "0"] \\ s_2 \arrow[u, bend left, "0"] \arrow[r, bend left, "1"] & s_3 \arrow[l, bend left, "1"] \arrow[u, bend left, "0"] \end{tikzcd}$$

The state s_0 accepts the language consisting of all words $w \in \{0,1\}^*$ that contain both an even number of 0's and an even number of 1's. □

Exercise 124. Describe the languages accepted by the states s_1, s_2, s_3 in the automaton of Example 123. Design an automaton that accepts all words that have an even number of 0's and a number of 1's that is a multiple of 3. □

Example 125. Here is an example of an *infinite* automaton with inputs from $A = \{a, b\}$:

$$\begin{tikzcd}
\underline{s_0} \arrow[r, "a"'] \arrow[d, "b"'] & s_1 \arrow[r, "a"'] \arrow[d, "b"'] & s_2 \arrow[r, "a"'] \arrow[d, "b"'] & s_3 \arrow[r, "a"'] \arrow[d, "b"'] & \cdots \\
t_0 \arrow[loop below, "{a,b}"] & \underline{t_1} \arrow[l, "{a,b}"'] & t_2 \arrow[l, "b"'] \arrow[ll, bend left, "a"] & t_3 \arrow[l, "b"'] \arrow[lll, bend left, "a"] & \cdots \arrow[l, "b"']
\end{tikzcd}$$

The state s_0 of this automaton accepts the following language:

$$l(s_0) \;=\; \{a^n b^n \mid n \geqslant 0\}$$ □

Exercise 126. Describe the languages accepted by each of the states s_i and t_i in the automaton of Example 125. Design an automaton that accepts the language of all words that contain the same number of a's and b's. □

8.2 Homomorphisms and bisimulations of automata

The interaction between different automata can be described by means of homomorphisms and bisimulations.

Definition 127 (homomorphism of deterministic automata). A *homomorphism* $f\colon (S, \langle \mathsf{o}_S, \mathsf{tr}_S\rangle) \to (T, \langle \mathsf{o}_T, \mathsf{tr}_T\rangle)$ of deterministic automata is a function $f\colon S \to T$ such that

$$\begin{tikzcd}
S \arrow[r, "f"] \arrow[d, "{\langle \mathsf{o}_S, \mathsf{tr}_S\rangle}"'] & T \arrow[d, "{\langle \mathsf{o}_T, \mathsf{tr}_T\rangle}"] \\
2 \times S^A \arrow[r, "1 \times f^A"'] & 2 \times T^A
\end{tikzcd}$$

that is, such that $\mathsf{o}_T \circ f = \mathsf{o}_S$ and $\mathsf{tr}_T \circ f = f^A \circ \mathsf{tr}_S$, where the function $f^A\colon S^A \to T^A$ is defined, for $g\colon A \to S$, by $f^A(g) = f \circ g\colon A \to T$. □

If f is a homomorphism then s is accepting iff $f(s)$ is; and if $s \xrightarrow{a} t$ then $f(s) \xrightarrow{a} f(t)$.

Example 128. Let $A = \{0, 1\}$ and consider the following two automata:

$$\begin{tikzcd}
\underline{s_0} \arrow[loop above, "0"] \arrow[rr, bend left, "1"] & \underline{s_1} \arrow[l, "0"] \arrow[r, "1"] & s_2 \arrow[l, bend left, "0"] \arrow[loop above, "1"]
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
\underline{t_0} \arrow[loop above, "0"] \arrow[r, bend left, "1"] & t_1 \arrow[l, bend left, "0"] \arrow[loop above, "1"]
\end{tikzcd}$$

The function $f \colon \{s_0, s_1, s_2\} \to \{t_0, t_1\}$ defined by

$$f(s_0) \;=\; f(s_1) \;=\; t_0 \qquad\qquad f(s_2) \;=\; t_1$$

is a homomorphism of automata. Note hat there does not exist a homomorphism in the opposite direction. □

Definition 129 (mono-, epi-, and isomorphisms). A homomorphism of automata f is called a *monomorphism* if f is injective; an *epimorphism* if f is surjective; and an *isomorphism* if f has an inverse f^{-1} that is also a homomorphism. □

Exercise 130. Prove that, equivalently, a homomorphism of automata f is a monomorphism if for all (with f composable) homomorphisms g and h,

$$f \circ g \;=\; f \circ h \quad\Longrightarrow\quad g \;=\; h$$

Similarly, show that a homomorphism f is an epimorphism iff for all (with f composable) homomorphisms g and h,

$$g \circ f \;=\; h \circ f \quad\Longrightarrow\quad g \;=\; h$$

Finally, prove that if a homomorphism f is bijective then it is an isomorphism. □

Next we define the notion of bisimulation for automata, in the by now familiar way, using homomorphisms.

Definition 131 (bisimulation of deterministic automata). A relation $R \subseteq S \times T$ is a *bisimulation* of deterministic automata $(S, \langle \mathsf{o}_S, \mathsf{tr}_S \rangle)$ and $(T, \langle \mathsf{o}_T, \mathsf{tr}_T \rangle)$ if there exists a structure map

$$\langle \mathsf{o}_R, \mathsf{tr}_R \rangle \colon R \to 2 \times R^A$$

such that the projection maps $\pi_1 \colon R \to S$ and $\pi_2 \colon R \to T$ are homomorphisms of automata:

$$\begin{tikzcd}
S \arrow[d, "\langle \mathsf{o}_S, \mathsf{tr}_S \rangle"'] & R \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] \arrow[d, dashed, "\exists"', "\langle \mathsf{o}_R, \mathsf{tr}_R \rangle"] & T \arrow[d, "\langle \mathsf{o}_T, \mathsf{tr}_T \rangle"] \\
2 \times S^A & 2 \times R^A \arrow[l, "1 \times \pi_1^A"] \arrow[r, "1 \times \pi_2^A"'] & 2 \times T^A
\end{tikzcd}$$

□

As in the case of stream bisimulation, there is also a relational equivalent of the definition of bisimulation that is more convenient for actual computations.

Definition 132 (bisimulation of deterministic automata, relational). A relation $R \subseteq S \times T$ is a bisimulation of deterministic automata if, for all $(s, t) \in R$,

$$(i)\ \ \mathsf{o}_S(s) \ = \ \mathsf{o}_T(t) \quad \text{and} \qquad (ii)\ \ \forall\, a \in A : \quad (s_a, t_a) \in R \tag{8.1}$$

where we recall that $s_a = \mathsf{tr}_S(s)(a)$ and $t_a = \mathsf{tr}_T(t)(a)$. □

Exercise 133. Check that both definitions of bisimulation of automata are equivalent. Also check some of the other properties that we saw earlier, in Chapter 7, for stream bisimulations, such as: the family of bisimulations is closed under inverse, union and intersection; the graph of a homomorphism of automata is a bisimulation; etc. etc. □

We define the bisimilarity relation between two automata as the union of all bisimulation relations.

Definition 134 (bisimilarity of deterministic automata). The union of *all* bisimulation relations between two automata $(S, \langle \mathsf{o}_S, \mathsf{tr}_S \rangle)$ and $(T, \langle \mathsf{o}_T, \mathsf{tr}_T \rangle)$ is a bisimulation, called the *bisimilarity* relation:

$$\sim \ = \ \bigcup \{ R \subseteq S \times T \mid \ \ R \text{ is a bisimulation } \} \tag{8.2}$$

By definition, $\sim$ is the *greatest* bisimulation. If $s \sim t$, that is, if there exists a bisimulation relation R with $(s, t) \in R$, then we call s and t *bisimilar*. □

Bisimilarity captures exactly the notion of behavioural equivalence in terms of language acceptance: as we shall see shortly, in Corollary 157, two states are bisimilar if and only if they accept the same language of input words.

Example 135. In the first automaton of Example 128, we have $s_0 \sim s_1$, since

$$\{(s_0, s_1),\ (s_0, s_0),\ (s_1, s_1),\ (s_2, s_2)\}$$

is a bisimulation of automata. Note that both s_0 and s_1 accept the same language, consisting of the empty word together with all words ending with a 0. Also note that s_0 and s_2 are *not* bisimilar. □

Exercise 136. Compute the bisimilarity relation between the first and the second automaton of Example 128. □

Example 137. Here is an automaton with inputs from a one-element set $A = \{a\}$ and with no accepting states. The numbered, dashed lines represent six pairs of states that together form a bisimulation relation between the states s_1 and t_1:

$$\begin{tikzcd}
s_1 \arrow[rr, "a"] \arrow[dr, dashed, no head, "1"'] \arrow[drrr, dashed, no head, "4"] & & s_2 \arrow[rr, "a"] \arrow[dl, dashed, no head, "5"'] \arrow[dr, dashed, no head, "2"] & & s_3 \arrow[llll, bend right, "a"'] \arrow[dlll, dashed, no head, "3"'] \arrow[dl, dashed, no head, "6"] \\
& t_1 \arrow[rr, bend left, "a"] & & t_2 \arrow[ll, bend left, "a"] &
\end{tikzcd}$$

Note that the numbers indicate the order in which the pairs have been added during the construction of the bisimulation relation, which started with the pair (s_1, t_1). □

The following notion will help, in many cases such as the one above, to construct a much smaller relation that already suffices to conclude that two states are bisimilar.

Definition 138 (bisimulation up-to-equivalence). Let $(S, \langle \mathsf{o}, \mathsf{tr} \rangle)$ be a deterministic automaton. A relation $R \subseteq S \times S$ is a *bisimulation up-to-equivalence* if, for all $(s, t) \in R$,

$$(i)\ \mathsf{o}(s) = \mathsf{o}(t) \quad \text{and} \qquad (ii)\ \forall\, a \in A : \quad (s_a, t_a) \in R^e \tag{8.3}$$

where $R^e \subseteq S \times S$ – the *equivalence closure* of R – is the smallest relation such that

1. $R \subseteq R^e$
2. R^e is an equivalence relation, that is, such that

   (a) $(s, s) \in R^e$ (reflexivity)
   (b) if $(s, t) \in R^e$ then $(t, s) \in R^e$ (symmetry)
   (c) if $(s, t) \in R^e$ and $(t, u) \in R^e$ then $(s, u) \in R^e$ (transitivity)

   for all $s, t, u \in S$. □

There is the corresponding proof principle.

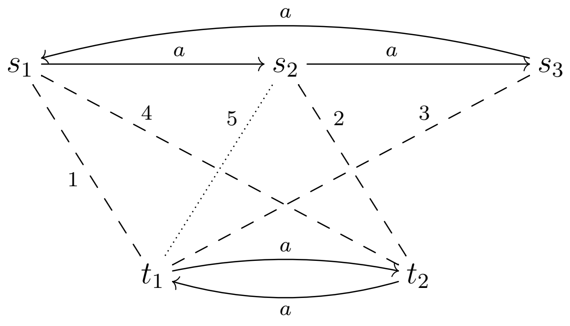
Proposition 139 (coinduction up-to-equivalence). If $R \subseteq S \times S$ is a bisimulation up-to-equivalence and $(s,t) \in R$, then $s \sim t$. □

Proof: Let R be a bisimulation up-to-equivalence. It suffices to show that its equivalence closure R^e is an ordinary bisimulation relation. Thus we have to show that conditions (i) and (ii) in (8.3) above hold, for all $(s,t) \in R^e$. We prove this by induction on the defining clauses of R^e. 1. For all pairs in R, (8.3) holds by the assumption that R is a bisimulation up-to-equivalence. 2(a). Any pair (s,s) trivially satisfies (8.3). 2(b). Assume that (s,t) satisfies (8.3). Then $\mathsf{o}(s) = \mathsf{o}(t)$ and $(s_a, t_a) \in R^e$, for all $a \in A$. This implies, by symmetry of R^e, that $(t_a, s_a) \in R^e$, for all $a \in A$. It follows that (t,s) satisfies (8.3). 2(c). Assuming that (s,t) and (t,u) satisfy (8.3), it follows, similarly, by the transitivity of R^e that (s,u) satisfies (8.3). □

Example 140. Returning to the automaton of Example 137, the construction of the bisimulation could have stopped after step 4, since the relation consisting of the pairs

$$\{1,2,3,4\} \;=\; \{(s_1,t_1),\,(s_2,t_2),\,(s_3,t_1),\,(s_1,t_2)\}$$

is a bisimulation up-to-equivalence:

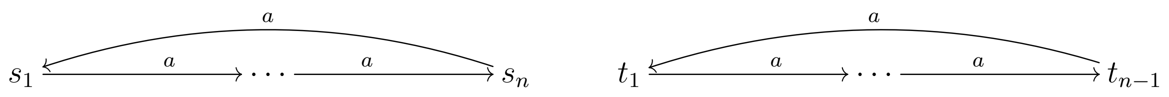

$$\begin{tikzcd}
s_1 \arrow[rr, "a"] \arrow[dr, dash, dashed, "1"'] \arrow[drrr, dash, dashed, "4"] & & s_2 \arrow[rr, "a"] \arrow[dl, dash, dotted, "5"'] \arrow[dr, dash, dashed, "2"] & & s_3 \arrow[llll, bend right, "a"'] \arrow[dlll, dash, dashed, "3"'] \\
& t_1 \arrow[rr, bend left, "a"] & & t_2 \arrow[ll, bend left, "a"] &
\end{tikzcd}$$

This follows from the fact that

$$(s_2,t_2),\,(t_2,s_1),\,(s_1,t_1) \in \{1,2,3,4\}^e$$

implies, by transitivity, that $(s_2,t_1) \in \{1,2,3,4\}^e$, which was the 5th pair to be added. Thus $\{1,2,3,4\}$ is a bisimulation up-to-equivalence and we conclude, by Proposition 139, that s_1 and t_1 are bisimilar. □

Exercise 141. Let $n \geqslant 3$. Consider the following automaton:


$$\begin{tikzcd}
s_1 \arrow[r, "a"] & \cdots \arrow[r, "a"] & s_n \arrow[ll, bend right, "a"'] & t_1 \arrow[r, "a"] & \cdots \arrow[r, "a"] & t_{n-1} \arrow[ll, bend right, "a"']
\end{tikzcd}$$

What are the smallest bisimulation and bisimulation up-to-equivalence relations containing the pair (s_1,t_1)? □

As for dynamical and stream systems, there are the following by now familiar connections between homomorphisms and bisimulations.

Proposition 142 (homomorphisms and bisimulations of deterministic automata). A function $f\colon S \to T$ is a homomorphism of automata if and only if

$$\mathsf{graph}(f) \;=\; \{(s, f(s)) \in S \times T \mid s \in S\}$$

is a bisimulation. If $f\colon S \to T$ is a homomorphism then

$$\mathsf{ker}(f) \;=\; \{(s, s') \in S \times S \mid f(s) = f(s')\}$$

is (both an equivalence relation and) a bisimulation. Conversely, if $R \subseteq S \times S$ is both a bisimulation and an equivalence relation, then there exists a unique automaton structure

$$(S/R, \langle \mathsf{o}_R, \mathsf{tr}_R\rangle) \qquad \mathsf{o}_R\colon S/R \to 2 \qquad \mathsf{tr}_R\colon S/R \to (S/R)^A$$

such that the quotient function

$$[-]_R\colon S \to S/R$$

which maps a state s to its equivalence class $[s]_R$, is a homomorphism of automata. $\square$

The notion of subautomaton is defined as follows.

Definition 143 (subautomata). We call an automaton $(V, \langle \mathsf{o}_V, \mathsf{tr}_V\rangle)$ a *subautomaton* of a deterministic automaton $(S, \langle \mathsf{o}_S, \mathsf{tr}_S\rangle)$ if $V \subseteq S$ and the inclusion map $i\colon V \subseteq S$ is a homomorphism of automata. $\square$

Exercise 144. Prove that $(V, \langle \mathsf{o}_V, \mathsf{tr}_V\rangle)$ is a subautomaton of $(S, \langle \mathsf{o}_S, \mathsf{tr}_S\rangle)$ iff V is closed under transitions and o_V and tr_V are the restrictions of o_S and tr_S to the set V. Show that $(V, \langle \mathsf{o}_V, \mathsf{tr}_V\rangle)$ is a subautomaton of $(S, \langle \mathsf{o}_S, \mathsf{tr}_S\rangle)$ iff the diagonal $\Delta_V = \{(v, v) \mid v \in V\}$ is a bisimulation relation of automata. $\square$

Exercise 145 (homomorphic (pre)images are subautomata). Let $f\colon S \to T$ be a homomorphism of automata. Show that $f(S)$ is a subautomaton of T and $f^{-1}(T)$ is a subautomaton of S. $\square$

Definition 146 (generated subautomaton). Let $(S, \langle \mathsf{o}_S, \mathsf{tr}_S \rangle)$ be an automaton. For every $V \subseteq S$, we define the subautomaton *generated by* V by

$$(\langle V \rangle, \langle \mathsf{o}, \mathsf{tr} \rangle) \qquad \mathsf{o}\colon \langle V \rangle \to 2 \qquad \mathsf{tr}\colon \langle V \rangle \to \langle V \rangle^A$$

where

$$\langle V \rangle \;=\; \{ s_w \mid s \in V \text{ and } w \in A^* \}$$

and where o and tr are the restrictions of $\mathsf{o}_S\colon S \to 2$ and $\mathsf{tr}_S\colon S \to S^A$ to the set $\langle V \rangle$. □

Exercise 147. Show that the collection of all subautomata of an automaton S is closed under intersections and unions. Show for every $V \subseteq S$ that $\langle V \rangle$ is the smallest subautomaton of S that contains V. □

Next we show that every homomorphism of automata factors through an epimorphism followed by a monomorphism. The proof will use the following lemma.

Lemma 148. *Let S, T, and U be three automata and consider three* functions

$$\begin{tikzcd} S \arrow[r, "g"'] \arrow[rr, bend left, "f"] & U \arrow[r, "h"'] & T \end{tikzcd}$$

with $f = h \circ g$.

1. *If g is surjective and f and g are homomorphisms, then h is a homomorphism.*
2. *If h is injective and f and h are homomorphisms, then g is a homomorphism.*

Exercise 149. Prove Lemma 148. □

Theorem 150 (first isomorphism theorem). *For every homomorphism of automata $f\colon S \to T$, there exists the following factorization:*

$$\begin{tikzcd} S \arrow[rr, "f"] \arrow[dr, bend right, "f"] \arrow[ddr, bend right, "e"'] & & T \\ & f(S) \arrow[ur, bend right, "\subseteq"] & \\ & S/\mathsf{ker}(f) \arrow[u, "\cong"'] \arrow[uur, bend right, "m"'] & \end{tikzcd}$$

Proof: For every *function* $f\colon S \to T$, we have such a factorisation, satisfying

$$e(s) \;=\; [s]_{\mathsf{ker}(f)} \qquad\qquad m([s]_{\mathsf{ker}(f)}) \;=\; f(s)$$

If f is moreover a homomorphism of automata, then $S/\mathsf{ker}(f)$ is a quotient automaton, by Proposition 142, and $f(S)$ is a subautomaton of T, by Exercise 145. This makes the quotient map e an epimorphism, and the inclusion map $\subseteq$ a monomorphism. By Lemma 148, it follows that $f\colon S \to f(S)$ is an epimorphism; that m is a monomorphism; and that the bijection between $S/\mathsf{ker}(f)$ and $f(S)$ is an isomorphism. $\square$

8.3 The final automaton of languages

We define the set of *all languages* over a set A by

$$\mathcal{P}(A^*) \;=\; \{\lambda \mid \lambda \subseteq A^*\}$$

The set $\mathcal{P}(A^*)$ can be turned into an automaton as follows.

Definition 151 (initial value, input derivative). We define the *initial value* $\lambda(\varepsilon)$ of a language $\lambda \in \mathcal{P}(A^*)$ by

$$\lambda(\varepsilon) \;=\; \begin{cases} 1 & \text{if } \;\varepsilon \in \lambda \\ 0 & \text{if } \;\varepsilon \notin \lambda \end{cases}$$

For $a \in A$, we define the *input derivative* or *a-derivative* of λ by

$$\lambda_a \;=\; \{w \in A^* \mid aw \in \lambda\}$$

For $v \in A^*$, we define, more generally, the *v-derivative* of λ by

$$\lambda_v \;=\; \{w \in A^* \mid vw \in \lambda\}$$

$\square$

Next we define two functions

$$\begin{array}{lll} \mathsf{i}_l\colon \mathcal{P}(A^*) \to 2 & \mathsf{d}_l\colon \mathcal{P}(A^*) \to \mathcal{P}(A^*)^A & \\ \mathsf{i}_l(\lambda) \;=\; \lambda(\varepsilon) & \mathsf{d}_l(\lambda)(a) \;=\; \lambda_a & (\lambda \in \mathcal{P}(A^*),\, a \in A) \end{array}$$

Pairing these two functions, we define the *automaton of languages* by

$$(\mathcal{P}(A^*),\, \langle \mathsf{i}_l, \mathsf{d}_l\rangle)$$

In this (infinite) automaton, a state λ is accepting if and only if $\lambda(\varepsilon) = 1$ or equivalently, $\varepsilon \in \lambda$. The transitions of λ are determined by its input derivatives. For instance, here is a little fragment (in fact a subautomaton) of the automaton $(\mathcal{P}(A^*),\, \langle \mathsf{i}_l, \mathsf{d}_l\rangle)$:

$$\begin{tikzcd}
\{a, ab, ac\} \arrow[r, "a"] \arrow[drr, bend right, "{b,c}"'] & \underline{\{\varepsilon, b, c\}} \arrow[r, "{b,c}"] \arrow[dr, "a"] & \underline{\{\varepsilon\}} \arrow[d, "{a,b,c}"] \\
& & \emptyset \arrow[loop below, "{a,b,c}"]
\end{tikzcd}$$

Exercise 152. Describe the languages accepted by each of the states of the above automaton. □

The automaton of languages is *final*.

Theorem 153 (finality – languages). *For every automaton* $(S, \langle \mathsf{o}, \mathsf{tr}\rangle)$ *there exists a* unique *homomorphism* $[\![-]\!]\colon (S, \langle \mathsf{o}, \mathsf{tr}\rangle) \to (\mathcal{P}(A^*), \langle \mathsf{i}_l, \mathsf{d}_l\rangle)$*:*

$$\begin{tikzcd}
S \arrow[r, dashed, "\exists!\,[\![-]\!]"] \arrow[d, "\forall\,\langle \mathsf{o}, \mathsf{tr}\rangle"'] & \mathcal{P}(A^*) \arrow[d, "\langle \mathsf{i}_l, \mathsf{d}_l\rangle"] \\
2 \times S^A \arrow[r, dashed, "1 \times [\![-]\!]^A"'] & 2 \times \mathcal{P}(A^*)^A
\end{tikzcd}$$

Proof: We define

$$[\![-]\!]\colon S \to \mathcal{P}(A^*) \qquad [\![s]\!] \;=\; l(s) \;=\; \{w \in A^* \mid \mathsf{o}(s_w) = 1\} \qquad (s \in S,\, w \in A^*)$$

One easily verifies that $[\![-]\!]$ is the only function making the diagram above commute. □

The final homomorphism $[\![-]\!]$ assigns to every state s its *global* behaviour, consisting of the language $l(s)$ of all words accepted by s.

Exercise 154. Show that in the automaton $(\mathcal{P}(A^*), \langle \mathsf{i}_l, \mathsf{d}_l\rangle)$ the language accepted by a state $\lambda \in \mathcal{P}(A^*)$ is the language λ itself. □

Example 155. Returning to the automaton of Example 120,

$$\begin{tikzcd}
s_0 \arrow[loop left, "b"] \arrow[r, bend left, "a"] & s_1 \arrow[l, bend left, "b"] \arrow[r, "a"] & \underline{s_2} \arrow[loop right, "a{,}b"]
\end{tikzcd}$$

we have

$$\begin{aligned}
[\![s_0]\!] &= \{vaaw \mid v, w \in \{a, b\}^*\,\} \\
[\![s_1]\!] &= \{aw \mid w \in \{a, b\}^*\,\} \cup \{bvaaw \mid v, w \in \{a, b\}^*\,\} \\
[\![s_2]\!] &= \{a, b\}^*
\end{aligned}$$

□

The final homomorphism from Theorem 153 maps every state to (a canonical representation of) its bisimilarity equivalence class, as expressed by the following theorem.

Theorem 156. *Let* $(S,\ \langle \mathsf{o}, \mathsf{tr}\rangle)$ *be an automaton. For all* $s, t \in S$,

$$s \ \sim\ t \quad\Longleftrightarrow\quad [\![s]\!] \ =\ [\![t]\!]$$

Proof: The proof is essentially the same as that of the corresponding Theorem 84, for streams. Consider a bisimulation relation on S, with projections $\pi_1, \pi_2 \colon R \to S$. By the definition of bisimulation, the projections are homomorphisms. Composition with $[\![-]\!]$ gives *two* homomorphisms from R to $\mathcal{P}(A^*)$. By finality, they must be equal, which proves the implication from left to right. The converse follows from the fact that the kernel of every homomorphism of automata is a bisimulation relation, Proposition 142. $\square$

We saw in the proof of Theorem 153 that $[\![s]\!] = l(s)$, which implies the following.

Corollary 157 (coinduction proof principle for automata). *Let* $(S,\ \langle \mathsf{o}, \mathsf{tr}\rangle)$ *be an automaton. For all* $s, t \in S$,

$$s \ \sim\ t \quad\Longleftrightarrow\quad l(s) \ =\ l(t) \qquad \square$$

According to Corollary 157, one can prove that two states s_1 and t_1 in S accept the same language by showing that they are bisimilar. In other words, if we want to prove $l(s_1) = l(t_1)$, it suffices to find a relation $R \subseteq S \times S$ with $(s_1, t_1) \in R$ that satisfies conditions (i) and (ii) from Definition 132: for all $(s,t) \in R$,

$$(i)\ \ \mathsf{o}_S(s) \ =\ \mathsf{o}_T(t) \quad \text{and} \qquad (ii)\ \ \forall\, a \in A: \quad (s_a, t_a) \in R$$

Similar to Theorem 96, Corollary 157 is directly based on finality, here of the automaton of languages. It is in a precise sense, explained in Chapter 4, dual to the principle of induction. Therefore we call it the *coinduction* proof principle for automata.

Example 158. Consider the following two automata:

$$\begin{tikzcd}
\underline{s_0} \arrow[d, bend left, "b"] \arrow[rd, "a"] & \underline{s_1} \arrow[r, bend left, "b"] \arrow[d, "a"] & s_2 \arrow[loop above, "b"] \arrow[d, bend left, "a"] \\
\underline{s_3} \arrow[u, bend left, "a"] \arrow[loop below, "b"] & \underline{s_4} \arrow[l, "b"'] \arrow[loop below, "a"] & s_5 \arrow[l, "a"'] \arrow[u, bend left, "b"]
\end{tikzcd}$$

$$\begin{tikzcd}
\underline{t_0} \arrow[loop left, "{a,b}"] & \underline{t_1} \arrow[l, "a"'] \arrow[r, "b"] & t_2 \arrow[loop above, "b"] \arrow[r, bend left, "a"] & t_3 \arrow[l, bend left, "b"] \arrow[r, "a"] & \underline{t_4} \arrow[loop above, "a"] \arrow[r, bend left, "b"] & \underline{t_5} \arrow[l, bend left, "a"] \arrow[loop right, "b"]
\end{tikzcd}$$

As a first illustration of the coinduction proof principle for automata, we will prove

$$l(s_0) \ =\ l(t_0)$$

by constructing a bisimulation relation containing the pair (s_0, t_0). We start our construction by considering the relation

$$R_0 \;=\; \{(s_0, t_0)\}$$

The pair (s_0, t_0) satisfies condition (i) of (8.1) above, since both s_0 and t_0 are accepting. In order to check condition (8.1)(ii), we compute the a- and b-derivatives of s_0 and t_0. This gives two new pairs (s_4, t_0) and (s_3, t_0), which are not in R_0. We extend R_0 by simply adding the new pairs:

$$R_1 \;=\; R_0 \cup \{(s_4, t_0),\, (s_3, t_0)\}$$

The newly added pairs both satisfy condition (i), since all of s_4, t_0 and s_3 are accepting. In order to check condition (ii), for the newly added pairs, we again compute the a- and b-derivatives, yielding (s_4, t_0) and (s_3, t_0), for (s_4, t_0), and (s_0, t_0) and (s_3, t_0), for (s_3, t_0). All of these pairs are contained in R_1, so we conclude that R_1 is a bisimulation relation. Now $l(s_0) = l(t_0)$ follows by coinduction, Corollary 157.

In a similar fashion, one can prove $l(s_1) = l(t_1)$ by constructing a bisimulation relation S with $(s_1, t_1) \in S$. Starting with that pair and adding all elementwise w-derivatives $((s_1)_w,\, (t_1)_w)$, for all $w \in A^*$, we obtain

$$S \;=\; \{\, (s_1, t_1),\, (s_4, t_0),\, (s_3, t_0),\, (s_0, t_0),\, (s_2, t_2),\, (s_5, t_3),\, (s_4, t_4),\, (s_3, t_5),\, (s_0, t_4)\,\}$$

By construction, S is closed under transitions. And since for all pairs $(x, y) \in S$, x is accepting iff y is accepting, we conclude that conditions (i) and (ii) from (8.1) are satisfied. Thus S is a bisimulation relation and $l(s_1) = l(t_1)$ follows by coinduction, Corollary 157.

Finally, we show how the attempt to construct a bisimulation between two states may also lead to a proof that they accept *different* languages. For a simple example, let us try to find a bisimulation relation containing the pair (s_1, t_4). Starting with the singleton relation $U = \{(s_1, t_4)\}$, we observe that condition (i) from (8.1) is satisfied since both s_1 and t_4 are accepting. Computing a- and b-derivatives gives the pairs (s_4, t_4) and (s_2, t_5). Adding these pairs to U will now *not* yield a bisimulation relation, however, since t_5 is accepting whereas s_2 is not. Since both pairs of derivatives (s_4, t_4) and (s_2, t_5) *have* to be contained in any bisimulation relation that contains the starting pair (s_1, t_4), we conclude that there exist no such relation. As a consequence, we conclude that $l(s_1) \neq l(t_4)$, again by Corollary 157.

On the positive side, we note that our attempt to construct a bisimulation relation for the pair (s_1, t_4) presents us with a witness for the inequality of $l(s_1)$ and $l(t_4)$: since $(s_1)_b = s_2$ and $(t_4)_b = t_5$, and since t_5 is accepting and s_2 is not, we observe that $b \in l(t_4)$ and $b \notin l(s_1)$. □

Exercise 159. Compute the bisimilarity relations for the automata in Example 158. □

We saw that the finality of the automaton of languages gives rise to a coinduction proof principle for automata, Theorem 156 and Corollary 157. Applying Theorem 156 to the final automaton of languages itself, gives a coinduction proof principle for languages. (Finality forms the basis for coinductive language *definitions* as well, cf. Section 11.4.)

Theorem 160 (coinduction proof principle for languages).

$$\kappa \sim \lambda \iff \kappa = \lambda \qquad (\kappa, \lambda \in \mathcal{P}(A^*))$$

Proof: Taking $(S, \langle \mathsf{o}, \mathsf{tr} \rangle) = (\mathcal{P}(A^*), \langle \mathsf{i}_l, \mathsf{d}_l \rangle)$ in Theorem 156, the theorem follows from the observation that the identity map is the only homomorphism from $(\mathcal{P}(A^*), \langle \mathsf{i}_l, \mathsf{d}_l \rangle)$ to itself. □

Exercise 161. Give an elementary proof of the implication from left to right in Theorem 160, using induction on the length of words. □

By the implication from left to right, one can prove that $\kappa = \lambda$ by showing that $\kappa \sim \lambda$, that is, by constructing a bisimulation relation $R \subseteq \mathcal{P}(A^*) \times \mathcal{P}(A^*)$ with $(\kappa, \lambda) \in R$. Spelling out Definition 132 for the automaton of languages, R is a bisimulation iff for all $(\mu, \nu) \in R$:

$$(i)\ \mu(\varepsilon) = \nu(\varepsilon) \quad \text{and} \quad (ii)\ \forall\, a \in A: \ (\mu_a, \nu_a) \in R \tag{8.4}$$

We shall see many examples of the coinduction proof principle for languages of Theorem 160 in Section 8.5.

8.4 Regular languages

Next we introduce the so-called *regular operators* on languages. These operators provide us with a vocabulary to define interesting examples and classes of languages, notably, the class of *regular languages*, which are – as we shall see in Theorem 189 – precisely those languages that can be represented by a *finite* automaton.

Definition 162 (regular operators). We define the constant languages

$$0 = \emptyset \qquad 1 = \{\varepsilon\} \qquad a = \{a\} \qquad (a \in A)$$

(Note that we use a to denote the *language* consisting of the *word* a, which consists of the single *letter* a.) We define the operators of *sum* and *concatenation* by

$$\kappa + \tau = \kappa \cup \tau \qquad \kappa \times \tau = \{vw \mid v \in \kappa,\ w \in \tau\} \qquad (\kappa, \lambda \in \mathcal{P}(A^*))$$

The operation of *Kleene star* is defined by

$$\kappa^* = \bigcup_{n \geqslant 0} \kappa^n \qquad (\kappa \in \mathcal{P}(A^*))$$

where $\kappa^0 = 1$ and $\kappa^{n+1} = \kappa \times \kappa^n$. □

The following elementary properties of the regular operators are an immediate consequence of their definitions.

Proposition 163. For all $\kappa, \lambda, \mu \in \mathcal{P}(A^*)$,

$$\begin{array}{rclrclrcl} \kappa + 0 & = & \kappa & \kappa \times 0 & = & 0 & 0 \times \kappa & = & 0 \\ \kappa + \kappa & = & \kappa & \kappa \times 1 & = & \kappa & 1 \times \kappa & = & 0 \\ \kappa + \lambda & = & \lambda + \kappa & (\kappa + \lambda) + \mu & = & \kappa + (\lambda + \mu) & (\kappa \times \lambda) \times \mu & = & \kappa \times (\lambda \times \mu) \end{array} \qquad \square$$

Languages that can be defined in terms of only the regular constants and operators are called *regular*.

Definition 164 (regular languages). Let A be finite. We call a language $\lambda \in \mathcal{P}(A^*)$ *regular* if it can defined by the following syntax:

$$\lambda ::= 0 \mid 1 \mid a \mid \lambda_1 + \lambda_2 \mid \lambda_1 \times \lambda_2 \mid \lambda^* \qquad \square$$

The following characterisation of the initial values and input derivatives of the regular operators will help us to construct bisimulation relations for our proofs by coinduction.

Lemma 165 (initial values and derivatives of regular operators). *For all $\kappa, \lambda \in \mathcal{P}(A^*)$ and $a, b \in A$, we have the following initial values*

$$\begin{array}{rclrclrcl} 0(\varepsilon) & = & 0 & 1(\varepsilon) & = & 1 & (\kappa + \lambda)(\varepsilon) & = & \max\{\kappa(\varepsilon), \lambda(\varepsilon)\} \\ b(\varepsilon) & = & 0 & (\kappa^*)(\varepsilon) & = & 1 & (\kappa \times \lambda)(\varepsilon) & = & \min\{\kappa(\varepsilon), \lambda(\varepsilon)\} \end{array}$$

and derivatives

$$\begin{array}{rclrclrcl} 0_a & = & 0 & 1_a & = & 0 & b_a & = & \begin{cases} 1 & \textit{if } b = a \\ 0 & \textit{if } b \neq a \end{cases} \\ (\kappa + \lambda)_a & = & \kappa_a + \lambda_a & (\kappa^*)_a & = & \kappa_a \times \kappa^* & (\kappa \times \lambda)_a & = & \begin{cases} \kappa_a \times \lambda & \textit{if } \kappa(\varepsilon) = 0 \\ (\kappa_a \times \lambda) + \lambda_a & \textit{if } \kappa(\varepsilon) = 1 \end{cases} \end{array}$$

$\square$

The identities in Lemma 165 are easy consequences of the definition of the regular operators. As we shall see later, in Section 11.4, Lemma 165 itself could also be taken as a (coinductive) *definition* of the regular operators.

Notation 166 ($\kappa\lambda = \kappa \times \lambda$). We shall usually simply write $\kappa\lambda$ for $\kappa \times \lambda$. For instance, $a^*(ba^*)^*$ will be short for $a^* \times (b \times a^*)^*$. $\square$

8.5 Proving language equality by coinduction

We prove various language equalities by coinduction, Theorem 160.

Example 167. Let $A = \{a, b\}$. We prove

$$(a+b)^* \;=\; a^*(ba^*)^*$$

by coinduction. Using Lemma 165 and some of the elementary properties from Proposition 163, we compute

$$\begin{aligned}
(a+b)^*_a &= (a+b)_a(a+b)^* \\
&= (a_a + b_a)(a+b)^* \\
&= (1+0)(a+b)^* \\
&= 1(a+b)^* \\
&= (a+b)^*
\end{aligned}
\qquad
\begin{aligned}
(a+b)^*_b &= (a+b)_b(a+b)^* \\
&= (a_b + b_b)(a+b)^* \\
&= (0+1)(a+b)^* \\
&= 1(a+b)^* \\
&= (a+b)^*
\end{aligned}$$

$$\begin{aligned}
(a^*(ba^*)^*)_a &= a^*_a(ba^*)^* + (ba^*)^*_a \\
&= a_a a^*(ba^*)^* + (ba^*)_a(ba^*)^* \\
&= 1a^*(ba^*)^* + b_a a^*(ba^*)^* \\
&= a^*(ba^*)^* + 0a^*(ba^*)^* \\
&= a^*(ba^*)^*
\end{aligned}
\qquad
\begin{aligned}
(a^*(ba^*)^*)_b &= a^*_b(ba^*)^* + (ba^*)^*_b \\
&= a_b a^*(ba^*)^* + (ba^*)_b(ba^*)^* \\
&= 0a^*(ba^*)^* + b_b a^*(ba^*)^* \\
&= 0 + 1a^*(ba^*)^* \\
&= a^*(ba^*)^*
\end{aligned}$$

We see that $((a+b)^*,\, a^*(ba^*)^*)$ satisfies condition (ii) in (8.4). Since

$$(a+b)^*(\varepsilon) \;=\; 1 \;=\; a^*(ba^*)^*(\varepsilon)$$

it also satisfies condition (8.4)(i). Thus

$$\{((a+b)^*,\, a^*(ba^*)^*)\,\}$$

is a bisimulation relation. The equality now follows by coinduction, Theorem 160.

For a second example, we prove

$$(a+b)^* \;=\; (a^*b^*)^*$$

again by coinduction. We define

$$R_0 \;=\; \{\,((a+b)^*,\, (a^*b^*)^*)\,\}$$

and check whether it is a bisimulation relation. The initial value of both languages is 1. We already computed the derivatives of $(a+b)^*$. For $(a^*b^*)^*$, we have

$$\begin{aligned}
(a^*b^*)^*_a &= (a^*b^*)_a(a^*b^*)^* \\
&= (a^*_a b^* + b^*_a)(a^*b^*)^* \\
&= (a_a a^*b^* + b_a b^*)(a^*b^*)^* \\
&= (1a^*b^* + 0b^*)(a^*b^*)^* \\
&= a^*b^*(a^*b^*)^*
\end{aligned}
\qquad
\begin{aligned}
(a^*b^*)^*_b &= (a^*b^*)_b(a^*b^*)^* \\
&= (a^*_b b^* + b^*_b)(a^*b^*)^* \\
&= (a_b a^*b^* + b_b b^*)(a^*b^*)^* \\
&= (0a^*b^* + 1b^*)(a^*b^*)^* \\
&= b^*(a^*b^*)^*
\end{aligned}$$

Thus R_0 is not closed under derivatives. Adding the new pairs, we get

$$R_1 \;=\; R_0 \cup \{\, ((a+b)^*,\, a^*b^*(a^*b^*)^*),\; ((a+b)^*,\, b^*(a^*b^*)^*) \,\}$$

and we check whether R_1 is a bisimulation. The initial value of all these languages is 1. Computing the derivatives of the newly added languages, we find

$$\begin{aligned} (a^*b^*(a^*b^*)^*)_a &= a^*b^*(a^*b^*)^* & \qquad (a^*b^*(a^*b^*)^*)_b &= b^*(a^*b^*)^* \\ (b^*(a^*b^*)^*)_a &= a^*b^*(a^*b^*)^* & \qquad (b^*(a^*b^*)^*)_b &= b^*(a^*b^*)^* \end{aligned}$$

Thus R_1 is a bisimulation, and $(a+b)^* = (a^*b^*)^*$ follows, by coinduction. □

Exercise 168. Prove by coinduction that $[(b^*a)^*ab^*]^* = 1+a(a+b)^*+(a+b)^*aa(a+b)^*$. □

Coinduction is not restricted to pairs of individual languages, but also applies to general laws.

Example 169. We prove by coinduction that

$$\lambda^* \;=\; 1 + \lambda\lambda^* \tag{8.5}$$

for all $\lambda \in \mathcal{P}(A^*)$. Let

$$R_0 \;=\; \{\, (\lambda^*,\, 1+\lambda\lambda^*) \mid \lambda \in \mathcal{P}(A^*) \,\}$$

For every $\lambda \in \mathcal{P}(A^*)$,

$$\lambda^*(\varepsilon) \;=\; 1 \;=\; (1+\lambda\lambda^*)(\varepsilon)$$

For every $a \in A$, $\lambda^*_a = \lambda_a\lambda^*$. For the computation of $(1+\lambda\lambda^*)_a$, we have to make a case distinction. If $\lambda(\varepsilon) = 0$ then

$$(1+\lambda\lambda^*)_a \;=\; 1_a + (\lambda\lambda^*)_a \;=\; 0 + \lambda_a\lambda^* \;=\; \lambda_a\lambda^*$$

And if $\lambda(\varepsilon) = 1$ then

$$(1+\lambda\lambda^*)_a \;=\; 1_a + (\lambda\lambda^*)_a \;=\; 0 + (\lambda_a\lambda^* + \lambda^*_a) \;=\; 0 + \lambda_a\lambda^* + \lambda_a\lambda^* \;=\; \lambda_a\lambda^*$$

We see that R_0 is not closed under derivatives: $(\lambda_a\lambda^*, \lambda_a\lambda^*) \notin R_0$. If we put

$$R_1 \;=\; R_0 \cup \{\, (\lambda, \lambda) \mid \lambda \in \mathcal{P}(A^*) \,\}$$

then R_1 *is* a bisimulation relation. Now $\lambda^* = 1 + \lambda\lambda^*$ follows, by coinduction.

For a second example, we show that concatenation distributes over sum:

$$\kappa(\lambda+\mu) \;=\; \kappa\lambda + \kappa\mu \tag{8.6}$$

for all $\kappa, \lambda, \mu \in \mathcal{P}(A^*)$. We define, as usual optimistically,

$$S_0 \;=\; \{\, (\kappa(\lambda + \mu),\; \kappa\lambda + \kappa\mu) \mid \kappa, \lambda, \mu \in \mathcal{P}(A^*) \,\}$$

and try to prove that S_0 is a bisimulation. One readily shows that

$$(\kappa(\lambda + \mu))(\varepsilon) \;=\; (\kappa\lambda + \kappa\mu)(\varepsilon) \qquad (\kappa, \lambda, \mu \in \mathcal{P}(A^*))$$

Because of the presence of concatenation, we again have to make a case distinction in the computation of derivatives. If $\kappa(\varepsilon) = 0$ then

$$(\kappa(\lambda + \mu))_a \;=\; \kappa_a(\lambda + \mu) \qquad \begin{aligned}(\kappa\lambda + \kappa\mu)_a \;&=\; (\kappa\lambda)_a + (\kappa\mu)_a \\ &=\; \kappa_a\lambda + \kappa_a\mu\end{aligned}$$

If $\kappa(\varepsilon) = 1$ then

$$\begin{aligned}(\kappa(\lambda + \mu))_a \;&=\; \kappa_a(\lambda + \mu) + (\lambda + \mu)_a \\ &=\; \kappa_a(\lambda + \mu) + (\lambda_a + \mu_a)\end{aligned} \qquad \begin{aligned}(\kappa\lambda + \kappa\mu)_a \;&=\; (\kappa\lambda)_a + (\kappa\mu)_a \\ &=\; (\kappa_a\lambda + \lambda_a) + (\kappa_a\mu + \mu_a) \\ &=\; (\kappa_a\lambda + \kappa_a\mu) + (\lambda_a + \mu_a)\end{aligned}$$

Thus S_0 is *not* a bisimulation but one can easily prove that the relation

$$S_1 \;=\; \{\, (\kappa(\lambda + \mu) + \nu,\; \kappa\lambda + \kappa\mu + \nu) \mid \kappa, \lambda, \mu, \nu \in \mathcal{P}(A^*) \,\}$$

which contains S_0 (by taking $\nu = 0$), *is*. Identity (8.6) now follows by coinduction.

For a third example, consider

$$\lambda^* \;=\; \lambda^*\lambda + 1 \tag{8.7}$$

for all $\lambda \in \mathcal{P}(A^*)$. Let

$$T_0 \;=\; \{\, (\lambda^*,\; \lambda^*\lambda + 1) \mid \lambda \in \mathcal{P}(A^*) \,\}$$

Initial values match, and computing derivatives gives $\lambda^*_a = \lambda_a\lambda^*$ and

$$(\lambda^*\lambda + 1)_a \;=\; (\lambda^*\lambda)_a + 1_a \;=\; (\lambda^*_a\lambda + \lambda_a) + 0 \;=\; \lambda_a\lambda^*\lambda + \lambda_a \;=\; \lambda_a(\lambda^*\lambda + 1)$$

where the last equality uses (8.6). As before, we see that our initial attempt at defining a bisimulation fails, but once again, the computation demonstrating this failure also shows the way out. If we extend T_0 to

$$T_1 \;=\; \{\, (\kappa\lambda^*,\; \kappa(\lambda^*\lambda + 1)) \mid \kappa, \lambda \in \mathcal{P}(A^*) \,\}$$

then one easily checks that T_1 a bisimulation, and Equation (8.7) follows by coinduction.

$\square$

Exercise 170 (derivative of concatenation). The case distinctions in the proofs in Example 169 are due to the following equality from Lemma 165:

$$(\kappa \times \lambda)_a \;=\; \begin{cases} \kappa_a \times \lambda & \text{if } \kappa(\varepsilon) = 0 \\ (\kappa_a \times \lambda) + \lambda_a & \text{if } \kappa(\varepsilon) = 1 \end{cases}$$

The following equivalent equation avoids such a case distinction:

$$(\kappa \times \lambda)_a \;=\; (\kappa_a \times \lambda) \;+\; (\kappa(\varepsilon) \times \lambda_a) \tag{8.8}$$

Here we interpret $\kappa(\varepsilon)$, which is either 0 or 1, as the *language* $0 = \emptyset$ or $1 = \{\varepsilon\}$, respectively; and we read $\kappa(\varepsilon) \times \lambda_a$ as language concatenation. With this new convention, now redo the proof of Equation (8.6) above. □

In all three coinductive proofs of Example 169, we start the construction of a bisimulation relation by taking the most obvious candidate: the relation containing precisely the pairs of languages that we want to prove equal. None of these relations R_0, S_0 and T_0 was closed under derivatives and in order to become a bisimulation, all of them had to be extended. All of these extensions were obvious, dictated by the shape of the computed derivatives. The extensions consisted of adding the diagonal to R_0; closing the relation S_0 under additional summands; and closing T_0 under concatenation with arbitrary languages.

We next introduce a general way of dealing with such extensions by introducing the notion of bisimulation *up-to-congruence*.

Definition 171 (bisimulation up-to-congruence). We call

$$R \;\subseteq\; \mathcal{P}(A^*) \times \mathcal{P}(A^*)$$

a *bisimulation up-to-congruence* (*bisimulation up-to*, for short) if, for all $(\kappa, \lambda) \in R$,

$$(i)\;\; \kappa(\varepsilon) \;=\; \lambda(\varepsilon) \quad \text{and} \qquad (ii)\;\; \forall\, a \in A: \quad (\kappa_a, \lambda_a) \in R^c \tag{8.9}$$

Here

$$R^c \;\subseteq\; \mathcal{P}(A^*) \times \mathcal{P}(A^*)$$

is the *congruence closure* of R (with respect to the regular operators), which is defined as the smallest relation such that

1. $R \subseteq R^c$
2. R^c is an equivalence relation
3. if (κ, λ), $(\mu, \nu) \in R^c$ then $(\kappa + \mu, \lambda + \nu)$, $(\kappa\mu, \lambda\nu)$, $(\kappa^*, \lambda^*) \in R^c$ □

Note that any bisimulation up-to-congruence is by definition also a bisimulation up-to-equivalence, Definition 138. All the relations R_0, S_0 and T_0, in Example 169, are examples of a bisimulation up-to-congruence.

There is again a corresponding coinduction principle.

Proposition 172 (coinduction up-to-congruence). If $R \subseteq \mathcal{P}(A^*) \times \mathcal{P}(A^*)$ is a bisimulation up-to-congruence and $(\kappa, \lambda) \in R$, then $\kappa = \lambda$.

Proof: Let R be a bisimulation up-to-congruence. It suffices to show that R^c is a bisimulation; the proposition then follows by coinduction Theorem 160. Thus we show, by induction on the defining clauses of R^c, that for all $(\mu, \nu) \in R^c$:

$$(i)\;\; \mu(\varepsilon) \;=\; \nu(\varepsilon) \quad \text{and} \qquad (ii)\;\; \forall\, a \in A: \quad (\mu_a, \nu_a) \in R^c \tag{8.10}$$

1. For all pairs in R, conditions (i) and (ii) are satisfied by the assumption that R is a bisimulation up-to-congruence. 2. Any pair (μ, μ) trivially satisfies (8.10). If (μ, κ) and (κ, ν) satisfy (8.10) then so do (κ, μ) and (μ, ν). 3. Suppose (κ, λ) and (μ, ν) satisfy (8.10). Then $(\kappa + \mu)(\varepsilon) = (\lambda + \nu)(\varepsilon)$, since by assumption, $\kappa(\varepsilon) = \lambda(\varepsilon)$ and $\mu(\varepsilon) = \nu(\varepsilon)$. Furthermore, $(\kappa_a, \lambda_a) \in R^c$ and $(\mu_a, \nu_a) \in R^c$, by assumption. Since R^c is closed under sum, this implies $(\kappa_a + \mu_a, \lambda_a + \nu_a) \in R^c$. Thus $(\kappa + \mu, \lambda + \nu)$ satisfies (8.10). The cases of concatenation and Kleene star can be proved similarly. $\square$

Example 173 (Arden's rule). Following [RBR13b, RBR16], we prove, for all languages $\kappa, \lambda, \mu \in \mathcal{P}(A^*)$,

$$\text{if} \quad \kappa \;=\; \lambda\kappa + \mu \quad \text{and} \quad \lambda(\varepsilon) \;=\; 0 \quad \text{then} \quad \kappa \;=\; \lambda^*\mu \tag{8.11}$$

by coinduction up-to-congruence. Consider κ, λ, μ with $\kappa = \lambda\kappa + \mu$ and $\lambda(\varepsilon) = 0$, and define $R = \{\, (\kappa,\; \lambda^*\mu) \,\}$. The initial values match, and computing the derivatives gives

$$\begin{aligned} \kappa_a &= (\lambda\kappa + \mu)_a \\ &= \lambda_a\kappa + \mu_a \\ &= \lambda_a(\kappa) + \mu_a \end{aligned} \qquad\qquad \begin{aligned} (\lambda^*\mu)_a &= \lambda^*_a\mu + \mu_a \\ &= (\lambda_a\lambda^*)\mu + \mu_a \\ &= \lambda_a(\lambda^*\mu) + \mu_a \end{aligned}$$

It follows that $(\kappa_a,\; (\lambda^*\mu)_a) \in R^c$. Thus R is a bisimulation up-to and $\kappa = \lambda^*\mu$ follows by Theorem 172. $\square$

Exercise 174. Prove by coinduction up-to-congruence that

$$(\lambda + \mu)\kappa \;=\; \lambda\kappa + \mu\kappa \tag{8.12}$$

for all $\kappa, \lambda, \mu \in \mathcal{P}(A^*)$. $\square$

Exercise 175. Prove by coinduction up-to-congruence that, for all $\kappa, \lambda, \mu \in \mathcal{P}(A^*)$,

$$\lambda\kappa + \mu \subseteq \kappa \quad \Longrightarrow \quad \lambda^*\mu \subseteq \kappa \tag{8.13}$$

$$\kappa\lambda + \mu \subseteq \kappa \quad \Longrightarrow \quad \mu\lambda^* \subseteq \kappa \tag{8.14}$$

using that $\kappa \subseteq \lambda$ if and only if $\kappa + \lambda = \lambda$. □

Exercise 176. Prove the following implications by coinduction up-to-congruence:

$$\lambda\lambda \; = \; 1 \quad \Longrightarrow \quad \lambda \; = \; 1 \qquad\qquad (\lambda \in \mathcal{P}(A^*))$$

$$\lambda\lambda \; = \; \lambda \quad \Longrightarrow \quad \lambda^* = 1 + \lambda$$

□

Exercise 177 (simulations and language inclusion). Here is an alternative way to prove language inclusions such as those of equations (8.13) and (8.14) in Exercise 175. A relation $R \subseteq \mathcal{P}(A^*) \times \mathcal{P}(A^*)$ is a *simulation* if for all $(\mu, \nu) \in R$:

$$(i) \;\; \mu(\varepsilon) \; \leqslant \; \nu(\varepsilon) \quad \text{and} \qquad (ii) \;\; \forall\, a \in A : \quad (\mu_a, \nu_a) \in R$$

Note that (i) says that if $\varepsilon \in \mu$ then $\varepsilon \in \nu$. Now prove the following variation on coinduction for languages, Theorem 160:

$$\mu \; \subseteq \; \nu \quad \Longleftrightarrow \quad (\mu, \nu) \in R \qquad\qquad \text{(for a simulation relation } R\text{)}$$

Use the implication from right to left to prove Equation (8.13). Improve on this by introducing *simulations up-to-congruence* (possibly consulting [RBR16] for details). □

8.6 Minimal automata

Every automaton can be *minimised* by identifying all states that accept the same language.

Definition 178 (minimal automata). An automaton S is *minimal* if the final homomorphism (from Theorem 153):

$$[\![-]\!] \colon S \to \mathcal{P}(A^*)$$

is a monomorphism. □

In other words, S is minimal if the final homomorphism is injective. The phrasing of Definition 178 is somewhat abstract, which has the advantage that it can be easily generalised to different kinds of systems, that is, coalgebras of different functors.

However, the meaning of Definition 178 in the present setting of automata is concrete enough: since $[\![s]\!] = l(s)$ (the language accepted by s), an automaton is minimal iff it does not contain any two different states that accept the same language.

By Corollary 157, two states of an automaton accept the same language iff they are bisimilar. Therefore, every automaton can be *minimised* by taking the quotient with respect its bisimilarity relation.

Theorem 179 (minimisation of automata). *For every automaton S, the quotient automaton $S/{\sim}$ is minimal.*

Proof: By Corollary 157, $\mathsf{ker}([\![-]\!]) = {\sim}$. By Theorem 150, we have

$$\begin{tikzcd}
S \arrow[rr, "{[\![-]\!]}"] \arrow[dr, bend right, "{[\![-]\!]}"] \arrow[ddr, bend right=40, "{e=[-]_\sim}"'] & & \mathcal{P}(A^*) \\
& {[\![S]\!]} \arrow[ur, bend right, "\subseteq"] & \\
& S/{\sim} \arrow[u, "\cong"'] \arrow[uur, bend right=40, "m"'] &
\end{tikzcd}$$

where the final homomorphism $m\colon S/{\sim} \longrightarrow \mathcal{P}(A^*)$ is a monomorphism. Thus $S/{\sim}$ is minimal, according to Definition 178. □

Note that by (the proof of) Theorem 179, the minimisation $S/{\sim}$ of an automaton S is isomorphic to the subautomaton $[\![S]\!]$ of $\mathcal{P}(A^*)$.

Exercise 180. Show that the following are equivalent: (i) an automaton S is minimal; (ii) $R \subseteq \Delta_S = \{(s,s) \mid s \in S\}$, for every bisimulation relation $R \subseteq S \times S$; (iii) every epimorphism $f\colon S \to T$ is an isomorphism. □

Exercise 181. Show that for every epimorphism $f\colon S \to T$ there is a unique epimorphism $g\colon T \to S/{\sim}$ such that

$$\begin{tikzcd}
S \arrow[r, "{[-]_\sim}"] \arrow[d, "\forall f"'] & S/{\sim} \\
T \arrow[ur, dashed, bend right, "\exists!\, g"'] &
\end{tikzcd}$$

□

Exercise 182. Compute the minimisation of each of the automata in Example 158 (cf. Exercise 159). □

The following theorem describes for every language how to construct a minimal automaton accepting it.

Theorem 183 (minimal automaton for a language). *For every language $\lambda \in \mathcal{P}(A^*)$, the automaton generated by λ:*

$$\langle\lambda\rangle \;=\; \{\lambda_w \mid w \in A^*\}$$

is a minimal automaton that accepts λ.

Proof: By Exercise 154, $l(\lambda) = \lambda$ so $\langle\lambda\rangle$ accepts λ. Since $\langle\lambda\rangle \subseteq \mathcal{P}(A^*)$ and inclusion of subautomata is a monomorphism, $\langle\lambda\rangle$ is minimal. □

Exercise 184 (Myhill-Nerode equivalence). Let $\lambda \in \mathcal{P}(A^*)$ and define the automaton $(A^*, \langle \mathsf{o}, \mathsf{tr}\rangle)$ by

$$\mathsf{o}(w) \;=\; \begin{cases} 1 & \text{if} \quad w \in \lambda \\ 0 & \text{if} \quad w \notin \lambda \end{cases} \qquad\qquad \mathsf{tr}(w)(a) \;=\; wa$$

Show that the final homomorphism $[\![-]\!]\colon A^* \to \mathcal{P}(A^*)$ satisfies, for all $w \in A^*$,

$$[\![w]\!] \;=\; \lambda_w \qquad\qquad [\![A^*]\!] \;=\; \langle\lambda\rangle$$

Define $R \subseteq A^* \times A^*$ by

$$vRw \quad \text{iff} \quad \forall u \in A^* :\; vu \in \lambda \iff wu \in \lambda$$

and show that $A^*/R \cong \langle\lambda\rangle$ is an isomorphism of automata. □

Example 185. Let $A = \{a, b, c\}$. We compute $\langle\kappa\rangle$ for $\kappa = \{a, ab, ac\}$:

$$\begin{array}{llll}
\kappa_a \;=\; \{\varepsilon, b, c\} & \kappa_{aa} \;=\; 0 & \kappa_{aba} \;=\; 0 & \kappa_{aca} \;=\; 0 \\
\kappa_b \;=\; 0 & \kappa_{ab} \;=\; 1 & \kappa_{abb} \;=\; 0 & \kappa_{acb} \;=\; 0 \\
\kappa_c \;=\; 0 & \kappa_{ac} \;=\; 1 & \kappa_{abc} \;=\; 0 & \kappa_{acc} \;=\; 0
\end{array}$$

Thus we find

$$\begin{tikzcd}
\kappa \arrow[r, "a"] \arrow[rrd, bend right, "b{,}c"'] & \underline{\kappa_a} \arrow[r, "b{,}c"] \arrow[rd, "a"] & \underline{1} \arrow[d, "a{,}b{,}c"] \\
& & 0 \arrow[loop right, "a{,}b{,}c"]
\end{tikzcd}$$

For a second example, let $A = \{a, b\}$ and $\lambda = (a+b)^*$. Then $\langle \lambda \rangle$ is

$$\begin{tikzcd}
\underline{\lambda} \arrow[loop right, "a{,}b"]
\end{tikzcd}$$

since we saw in Example 167 that $(a+b)^*_a = (a+b)^*_b = (a+b)^*$.

For a last example, we compute $\langle \mu \rangle$ for the language $\mu = (a^*b^*)^*$, whose derivatives were computed already in Example 167:

$$\begin{array}{lllllllll}
\mu_a & = & a^*b^*(a^*b^*)^* & \mu_{aa} & = & \mu_a & \mu_{ba} & = & \mu_a \\
\mu_b & = & b^*(a^*b^*)^* & \mu_{ab} & = & \mu_b & \mu_{bb} & = & \mu_b
\end{array}$$

This leads to the following *3-state* automaton:

$$\begin{tikzcd}
 & \underline{(a^*b^*)^*} \arrow[dl, bend right, "a"'] \arrow[dr, bend left, "b"] & \\
\underline{a^*b^*(a^*b^*)^*} \arrow[loop below, "a"] \arrow[rr, bend left, "b"] & & \underline{b^*(a^*b^*)^*} \arrow[loop below, "b"] \arrow[ll, bend left, "a"]
\end{tikzcd}$$

The state $(a^*b^*)^*$ above accepts the language $(a^*b^*)^*$. However, this automaton is *not* the *minimal* automaton $\langle (a^*b^*)^* \rangle$ we are after. We saw in Example 167 that

$$\{\, ((a+b)^*,\, (a^*b^*)^*),\ ((a+b)^*,\, a^*b^*(a^*b^*)^*),\ ((a+b)^*,\, b^*(a^*b^*)^*)\,\}$$

is a bisimulation, which implies that all three states in the automaton above are equal, to $(a+b)^*$. It follows that $\langle (a^*b^*)^* \rangle$ is identical to the *1-state* automaton $\langle (a+b)^* \rangle$ of the previous example, which is precisely the *minimisation* of the 3-state automaton above. $\square$

As we shall see below, in Kleene's Theorem 189, the minimal automaton $\langle \lambda \rangle$ is *finite* iff the language λ is regular (Definition 164).

Exercise 186. Let $\lambda \in \mathcal{P}(A^*)$ be a language satisfying the equation

$$\lambda \;=\; a\lambda b + 1$$

Compute the minimal automaton $\langle \lambda \rangle$ and compare the result with Example 125. $\square$

According to Theorem 179, computing the minimisation of an automaton amounts to the construction of the quotient with respect to its bisimilarity relation $\sim$. A naive procedure for the computation of $\sim$ on a given automaton S is to determine, for every pair (s, t) of states in S, whether s and t are bisimilar by trying to construct a bisimulation relation containing (s, t). One either finds a bisimulation containing (s, t), which proves $s \sim t$, or not, which implies $s \not\sim t$. Doing this for all $(s, t) \in S \times S$ computes $\sim$. If both the alphabet A and the set of states S are finite, then the procedure terminates.

The following algorithm for the computation of $\sim$ is more efficient than the naive procedure described above. It will *mark* (unordered) pairs of states $\{s, t\}$ with $s \neq t$ as soon as a reason is discovered why s and t are *not* bisimilar. For an automaton $(S, \langle \mathsf{o}, \mathsf{tr} \rangle)$, do the following:

1. Write down a table of pairs $\{s, t\}$, for all $s, t \in S$, $s \neq t$.
2. Mark all pairs $\{s, t\}$ with $\mathsf{o}(s) \neq \mathsf{o}(t)$.
3. Repeat the following until no more changes occur: if there exists an unmarked pair $\{s, t\}$ such that $\{s_a, t_a\}$ is marked, for some $a \in A$, then mark $\{s, t\}$.
4. When done, define $m = \{(s, t) \mid s, t \in S \ s.t. \ \{s, t\} \text{ is marked}\}$.

Proposition 187. Let A be finite. For a finite automaton $(S, \langle \mathsf{o}, \mathsf{tr} \rangle)$, the algorithm above terminates and computes the bisimilarity relation: for all $s, t \in S$,

$$s \sim t \iff \{s, t\} \notin m$$

Proof: Since there are only finitely many unordered pairs of elements in S, and because each time step 3 is executed, one more pair is marked, the algorithm terminates. For its correctness, note that

$$\begin{aligned} s \not\sim t &\iff l(s) \neq l(t) && \text{(Corollary 157)} \\ &\iff \exists w \in A^* \ s.t. \ \mathsf{o}(s_w) \neq \mathsf{o}(t_w) && (l(s) = \{w \in A^* \mid \mathsf{o}(s_w) = 1\}) \\ &\iff \text{the pair } \{s, t\} \text{ is marked} \end{aligned}$$

where the last equivalence can be proved by induction on the length of w. As a consequence, s and t are bisimilar if and only if the pair $\{s, t\}$ is unmarked. □

Exercise 188. Compute the bisimilarity relations of each of the two automata in Example 158, using Proposition 187. □

Kleene's celebrated theorem states that a language λ is regular (Definition 164) iff λ is accepted by a finite automaton. Since the automaton $\langle \lambda \rangle$ generated by λ is minimal, by Theorem 183, Kleene's theorem can be formulated, equivalently, as follows.

Theorem 189 (Kleene's Theorem). *Let A be finite. For every language $\lambda \in \mathcal{P}(A^*)$,*

$$\lambda \textit{ is regular} \iff \langle \lambda \rangle \textit{ is finite}$$

Proof: Let λ be regular. Since $\langle\lambda\rangle = \{\lambda_w \mid w \in A^*\}$, we have to show that the number of w-derivatives of λ is finite. We prove this by induction on the definition of regular language. If λ is 0, 1, or a, then $\langle\lambda\rangle$ is $\{0\}$, $\{0,1\}$, or $\{0,1,a\}$, respectively. Next we observe, for all $\kappa, \mu \in \mathcal{P}(A^*)$, that

$$\begin{aligned}
(\kappa+\mu)_w &= \kappa_w + \mu_w \\
(\kappa\mu)_w &= \kappa_w\mu + \mu_1 + \cdots + \mu_n && \text{(for some } n \geqslant 0 \text{ and } \mu_1,\ldots,\mu_n \in \langle\mu\rangle\text{)} \\
(\kappa^*)_w &= \kappa_1\kappa^* + \cdots + \kappa_m\kappa^* && \text{(for some } m \geqslant 1 \text{ and } \kappa_1,\ldots,\kappa_m \in \langle\kappa\rangle\text{)}
\end{aligned}$$

which easily follows by induction of the length of $w \in A^*$, using the computation rules for regular languages from Lemma 165. Suppose now that κ and μ are regular languages and assume that both $\langle\kappa\rangle$ and $\langle\mu\rangle$ are finite. Using associativity, commutativity and idempotency of the sum (Proposition 163), it follows from the three equations above that $\langle\lambda\rangle$ is finite if $\lambda = \kappa+\mu$, $\lambda = \kappa\mu$, or $\lambda = \kappa^*$.

Suppose, conversely, that $\langle\lambda\rangle = \{\lambda_1,\ldots,\lambda_l\}$, with $\lambda = \lambda_1$. We prove that λ is regular. Let $A = \{a_1,\ldots,a_k\}$. First we observe that every language $\kappa \in \mathcal{P}(A^*)$ satisfies

$$\kappa \;=\; a_1\kappa_{a_1} + \cdots + a_k\kappa_{a_k} + \kappa(\varepsilon) \tag{8.15}$$

where we read $\kappa(\varepsilon)$, whose value is either 0 or 1, as the corresponding *language*. Applying (8.15) to each of the languages in $\langle\lambda\rangle$ gives equations

$$\lambda_i \;=\; a_1(\lambda_i)_{a_1} + \cdots + a_k(\lambda_i)_{a_k} + \lambda_i(\varepsilon)$$

for all $1 \leqslant i \leqslant l$. Since $(\lambda_i)_{a_j} \in \langle\lambda\rangle$, for all $1 \leqslant j \leqslant k$, we have obtained a system of l equations in l many variables λ_i. Using Arden's rule (Example 173):

$$\text{if} \quad \alpha \;=\; \beta\alpha+\gamma \quad \text{and} \quad \beta(\varepsilon) \;=\; 0 \quad \text{then} \quad \alpha \;=\; \beta^*\gamma \tag{8.11}$$

and the distributivity of concatenation over sum:

$$\alpha(\beta+\gamma) \;=\; \alpha\beta+\alpha\gamma \qquad\qquad (\beta+\gamma)\alpha \;=\; \beta\alpha+\gamma\alpha \tag{8.16}$$

for all languages $\alpha,\beta,\gamma \in \mathcal{P}(A^*)$, one can show by induction on the number of equations that all the languages λ_i are regular. □

Example 190. Let $\lambda \in \mathcal{P}(A^*)$ be language for which $\langle\lambda\rangle$ is finite and given by

$$\begin{tikzcd}
\lambda \arrow[rr, bend left, "b"] \arrow[dr, "a"] & & \underline{\kappa} \arrow[ll, bend left, "a"] \arrow[dl, "b"] \\
& \mu \arrow[ul, bend left, "a"] \arrow[loop below, "b"] &
\end{tikzcd}$$

We show that λ is regular. By (8.15), we obtain the following system of equations:

$$\begin{aligned}\lambda &= a\mu + b\kappa \\ \kappa &= a\lambda + b\mu + 1 \\ \mu &= a\lambda + b\mu\end{aligned}$$

Applying Arden's rule (8.11) to the last equation (with $\alpha = \mu$, $\beta = b$, and $\gamma = a\lambda$) gives $\mu = b^*a\lambda$. Substituting μ in the second equation gives

$$\kappa = a\lambda + bb^*a\lambda + 1$$

Substituting μ and κ in the first equation gives

$$\begin{aligned}\lambda &= a\mu + b\kappa \\ &= ab^*a\lambda + b(a\lambda + bb^*a\lambda + 1) \\ &= ab^*a\lambda + ba\lambda + bbb^*a\lambda + b && (8.16) \\ &= (ab^*a + ba + bbb^*a)\lambda + b && (8.16)\end{aligned}$$

Applying (8.11) to the last equation (with $\alpha = \lambda$, $\beta = ab^*a + ba + bbb^*a$, and $\gamma = b$) gives $\lambda = (ab^*a + ba + bbb^*a)^*b$. □

Example 191 (finite versus minimal). Consider again the regular language $\mu = (a^*b^*)^*$, the last example from Example 185. There we saw that the automaton

$$\langle\mu\rangle = \{\mu_w \mid w \in A^*\} = \{(a^*b^*)^*,\ a^*b^*(a^*b^*)^*,\ b^*(a^*b^*)^*\}$$

is finite, in accordance with Kleene's Theorem 189. In that same Example 185, we also observed that all these three languages are identical, and that we are looking here in fact at a singleton set. We conclude that the computation rules for the derivatives of regular languages in Lemma 165, together with the use of associativity, commutativity and idempotency of the sum, are sufficient to guarantee that $\langle\lambda\rangle$ is *finite* whenever a language λ is regular, as we saw in the proof of Kleene's Theorem. However, in general not all possible identifications will have been made in the representation of $\langle\lambda\rangle$ thus found, and the *minimal* automaton for λ will only be obtained after computing its quotient with respect to bisimilarity. Which is precisely what we did for μ, in Example 185. □

Exercise 192. Compute a minimal automaton for the language $((b^*a)^*ab^*)^*$. □

Example 193 (non-regularity). Kleene's Theorem 189 can also be used to prove that a language is *not* regular, by showing that the number of its derivatives is *infinite*. For instance, let $\kappa = \{a^n b^n \mid n \geqslant 0\}$, where as usual $a^0 = 1$ and $a^{n+1} = aa^n$. Since all of the derivatives $\kappa_{a^k} = \{a^{n-k} b^n \mid n \geqslant k\}$ are different, $\langle \kappa \rangle$ is infinite. Thus κ is not regular.

For a second example, let $\lambda = \{w \in A^* \mid \sharp_a(w) = \sharp_b(w)\}$, consisting of all words containing an equal number of a's and b's. All of the derivatives λ_{a^k} are different, since $b^n \in \lambda_{a^k}$ iff $n = k$, for all n and k. Thus $\langle \lambda \rangle$ is infinite and λ is not regular. □

Exercise 194. Decide whether the languages $\{a^{2n} \mid n \geqslant 0\}$ and $\{a^{n^2} \mid n \geqslant 0\}$ are regular or not. □

8.7 Discussion

General references for the classical theory of deterministic automata and formal languages are [Eil74] and [HMU07]. The coalgebraic treatment in the present chapter is largely based on parts of [Rut98] and [RBR13b, RBR16]. Lemma 165 goes back to [Brz64] and, in the form presented here, to [Con71]. Proposition 187 has been taken from [Koz97]; minimization of deterministic automata has been extensively studied in the literature, including [Hop71]. Kleene's Theorem was proved in [Kle56]. The up-to techniques, in coinduction up-to-equivalence, Proposition 139, and coinduction up-to-congruence, Proposition 172, are based on [RBR13b, RBR16] and [BP13, BP15]. See also the brief discussion in Section 15.5.

Chapter 9

Partial automata

Systems without outputs, such as the dynamical systems (S, α) from Chapter 6 consisting of a set S of states and a transition function

$$\alpha \colon S \to S$$

are trivial in the sense that all states are behaviourally equivalent. This is reflected by the fact that the family of all dynamical systems has a trivial final coalgebra, namely, any dynamical system consisting of only one state (cf. Section 6.2).

Also systems with *inputs* but still without outputs, such as the *dynamical systems with inputs* introduced in the following exercise, are all behaviourally equivalent: they have again a trivial final coalgebra.

Exercise 195. Let A be a set. We define a dynamical system *with inputs from* A by

$$(S, \alpha) \qquad\qquad \alpha \colon S \to S^A$$

The transition function α maps a state $s \in S$ to a function $\alpha(s) \colon A \to S$, which assigns to every (input) element $a \in A$ a next state $\alpha(s)(a) \in S$. If (S, α) and (T, β) are dynamical systems with inputs from A, then a function $f \colon S \to T$ is a homomorphism if

$$\beta(f(s))(a) \;=\; f\,(\alpha(s)(a)) \qquad\qquad (s \in S,\, a \in A)$$

Prove the existence of a *final* dynamical system with inputs from A. $\square$

As we saw in Chapter 7, the situation changes when we consider dynamical systems *with outputs* in a set A, that is, stream systems

$$(S, \langle \mathsf{o}, \mathsf{tr} \rangle) \qquad \langle \mathsf{o}, \mathsf{tr} \rangle \colon S \to A \times S \qquad \mathsf{o} \colon S \to A \qquad \mathsf{tr} \colon S \to S$$

The output function o makes the states of such stream systems partially observable. As a consequence, not all stream systems are equivalent and the set A^ω of all streams is a non-trivial final stream system, Theorem 78. Similarly, dynamical systems with both *outputs* and inputs, such as the automata in Chapter 8, have non-trivial behaviour, as reflected by the fact that the set of all languages is a final automaton, Theorem 153.

In the present short chapter, we will look at yet another possible feature of dynamical systems that gives them non-trivial observable behaviour, namely systems of which the transitions are given by a *partial* function.

9.1 Partiality

A *partial (dynamical) system* is a pair

$$(S, \alpha) \qquad\qquad \alpha\colon S \to \mathbf{1} + S \qquad\qquad \mathbf{1} \;=\; \{*\}$$

where + denotes disjoint union. We use the following notation:

$$s\uparrow \;\Longleftrightarrow\; \alpha(s) \;=\; * \qquad\qquad s \longrightarrow s' \;\Longleftrightarrow\; \alpha(s) \;=\; s' \qquad\qquad (s, s' \in S)$$

If $\alpha(s) = *$ then we say that the dynamical system *terminates*; and if $\alpha(s) = s'$, then we say that the system *takes a transition step* to the new state s'. The structure map α is *partial* in that not all states are mapped to a next state.

It follows that for every state $s \in S$, there are two posibilities: either there exist $n \geqslant 0$ and states $s_0, s_1, \ldots, s_n$ in S such that

$$s = s_0 \longrightarrow s_1 \longrightarrow \cdots \longrightarrow s_n \uparrow \tag{9.1}$$

or there exists an infinite sequence of states $s_0, s_1, s_2, \ldots$ in S such that

$$s = s_0 \longrightarrow s_1 \longrightarrow s_2 \longrightarrow \cdots \tag{9.2}$$

That is, starting in s, the system either terminates after n transition steps or it never terminates, taking infinitely many transition steps. This global behaviour of partial systems is now reflected by the (existence of a) *final* partial system, which we shall describe next, in Theorem 198. First we introduce the notions of homomorphism and bisimulation.

Definition 196 (homomorphism and bisimulation of partial systems). A *homomorphism* $f\colon (S, \alpha) \to (T, \beta)$ of partial systems is a function $f\colon S \to T$ such that for all $s \in S$, either both s and $f(s)$ terminate, or both s and $f(s)$ take a transition step and $\beta(f(s)) = f(\alpha(s))$.

A relation $R \subseteq S \times T$ is a *bisimulation of partial systems* if for all $(s, t) \in R$, either both s and t terminate, or both s and t take a transition step and $(\alpha(s), \beta(t)) \in R$. We write $s \sim t$ if there exists a bisimulation $R \subseteq S \times T$ with $(s, t) \in R$. □

Exercise 197. Observe that partial systems are F-coalgebras of the functor $F\colon \mathsf{Set} \to \mathsf{Set}$ that is defined for sets S by $F(S) = \mathbf{1} + S$, and for functions $f\colon S \to T$ by

$$F(f) \;=\; id + f\colon (\mathbf{1} + S) \to (\mathbf{1} + T)$$

which maps $*$ to $*$ and $s \in S$ to $f(s) \in T$. Prove that with this definition of F, the notion of F-bisimulation, introduced in Definition 33, coincides with the definition of bisimulation given in Definition 196. □

Next we consider the set $\overline{\mathbb{N}}$ of the *extended natural numbers*:

$$\overline{\mathbb{N}} \;=\; \{0, 1, 2, \ldots, \} \cup \{\infty\}$$

consisting of the natural numbers augmented with an extra element ∞. The *predecessor* function, defined by

$$\mathsf{pre}\colon \overline{\mathbb{N}} \to \mathbf{1} + \overline{\mathbb{N}} \qquad \mathsf{pre}(0) \;=\; * \qquad \mathsf{pre}(n+1) \;=\; n \qquad \mathsf{pre}(\infty) \;=\; \infty$$

turns the set $\overline{\mathbb{N}}$ into a *final* partial system $(\overline{\mathbb{N}}, \mathsf{pre})$.

Theorem 198 (finality – extended natural numbers). *For every partial system (S, α) there exists a* unique *homomorphism* $[\![-]\!]\colon (S, \alpha) \to (\overline{\mathbb{N}}, \mathsf{pre})$*:*

$$\begin{tikzcd}
S \arrow[r, dashed, "\exists!\, [\![-]\!]"] \arrow[d, "\forall\, \alpha"'] & \overline{\mathbb{N}} \arrow[d, "\mathsf{pre}"] \\
\mathbf{1} + S \arrow[r, "1 + [\![-]\!]"'] & \mathbf{1} + \overline{\mathbb{N}}
\end{tikzcd}$$

Proof: We define $[\![-]\!]\colon S \to \overline{\mathbb{N}}$ by $[\![s]\!] = n$, if s terminates in n steps (as in (9.1)), or $[\![s]\!] = \infty$, if s does not terminate (as in (9.2)). This is the only function making the diagram above commute. □

The following example illustrates that finality gives rise to coinductive definitions.

Example 199. Consider the partial system $(\overline{\mathbb{N}} \times \overline{\mathbb{N}},\ f)$ with

$$f\colon \overline{\mathbb{N}} \times \overline{\mathbb{N}} \;\to\; \mathbf{1} + (\overline{\mathbb{N}} \times \overline{\mathbb{N}}) \qquad f(x, y) = \begin{cases} (\mathsf{pre}(x), y) & \text{if } x \neq 0 \\ (x, \mathsf{pre}(y)) & \text{if } x = 0,\ y \neq 0 \\ * & \text{if } x = 0 = y \end{cases}$$

for $x, y \in \overline{\mathbb{N}}$. By finality, there exists a (unique) homomorphism $\oplus\colon \overline{\mathbb{N}} \times \overline{\mathbb{N}} \to \overline{\mathbb{N}}$. It satisfies:

$$\mathsf{pre}(x \oplus y) = \begin{cases} \mathsf{pre}(x) \oplus y & \text{if } x \neq 0 \\ x \oplus \mathsf{pre}(y) & \text{if } x = 0,\ y \neq 0 \\ * & \text{if } x = 0 = y \end{cases}$$

It is easy to see that $n \oplus m = n + m$ for $n, m \in \mathbb{N}$ and $n \oplus \infty = \infty = \infty \oplus n$ (see Example 202 below), so that $\oplus$ behaves like a natural extension of addition on the natural numbers to the set of the extended natural numbers. □

As usual, finality gives rise to a coinduction proof principle.

Theorem 200 (coinduction proof principle for extended natural numbers).

$$x \sim y \iff x = y \qquad (x, y \in \overline{\mathbb{N}})$$

Exercise 201. Prove this. □

Example 202. Towards a proof by coinduction of

$$x \oplus \infty = \infty \qquad (x \in \overline{\mathbb{N}})$$

we define

$$R = \{ (x \oplus \infty, \infty) \mid x \in \overline{\mathbb{N}} \}$$

If $x = 0$ then

$$\mathsf{pre}(x \oplus \infty) = x \oplus \mathsf{pre}(\infty) = x \oplus \infty$$

If $x \neq 0$ then

$$\mathsf{pre}(x \oplus \infty) = \mathsf{pre}(x) \oplus \infty$$

Since $\mathsf{pre}(\infty) = \infty$, it follows that $(\mathsf{pre}(x \oplus \infty), \mathsf{pre}(\infty)) \in R$, for all $(x \oplus \infty, \infty) \in R$. Thus R is a bisimulation and the result follows by coinduction. □

Exercise 203 (commutativity of $\oplus$). Prove that $x \oplus y = y \oplus x$, for all $x, y \in \overline{\mathbb{N}}$, by coinduction. You will find it helpful to prove by coinduction, first, that

$$\mathsf{pre}(x) \oplus 1 = x$$

for all $x \in \overline{\mathbb{N}}$ with $x \neq 0$, and second, that

$$x \oplus (y \oplus 1) = (x \oplus 1) \oplus y$$

for all $x, y \in \overline{\mathbb{N}}$. □

Remark 204. We saw in Exercise 197 that partial systems are coalgebras of the functor $N \colon \mathsf{Set} \to \mathsf{Set}$ defined, for sets S, by $N(S) = \mathbf{1} + S$. In Chapter 4, we saw that the set $\mathbb{N}$ of natural numbers, with $[\mathsf{zero}, \mathsf{succ}] \colon \mathbf{1} + \mathbb{N} \to \mathbb{N}$ is an *initial N-algebra*. By Theorem 198, the set of extended natural numbers $\overline{\mathbb{N}}$, with $\mathsf{pre} \colon \overline{\mathbb{N}} \to \mathbf{1} + \overline{\mathbb{N}}$, is a *final N-coalgebra*. For a discussion of the close relationship between initial algebras and final coalgebras of functors from Set to Set, such as $\mathbb{N}$ and $\overline{\mathbb{N}}$, see [Bar93]. □

9.2 Partial systems with outputs

The following is a minor variation on the definition of partial system. A *partial system with outputs* in a set A is a pair (S, α) consisting of a set S of states and a structure map

$$\alpha\colon S \to A + S$$

The structure map α now either sends a state $s \in S$ to a value $a \in A$, in which case we write

$$s \uparrow a \quad \Longleftrightarrow \quad \alpha(s) \;=\; a \tag{9.3}$$

and say: s terminates with output a. Or α sends s, as before, to a new state $s' \in S$:

$$s \longrightarrow s' \quad \Longleftrightarrow \quad \alpha(s) \;=\; s' \tag{9.4}$$

In the following exercises, a few basic notions, properties and examples of partial systems with outputs are sketched.

Remark 205. Observe that partial systems with outputs in A are F-coalgebras of the functor $F\colon \mathsf{Set} \to \mathsf{Set}$ that is defined for sets S by $F(S) = A + S$ and, for functions $f\colon S \to T$ by

$$F(f) = (1_A + f)\colon (A + S) \to (A + T)$$

which acts as the identity function on A and as the function f on S. Formulate the notions of homomorphism and bisimulation of partial systems with outputs, either by instantiating Definition 33 or by simply varying Definition 196. $\square$

Exercise 206 (final partial system with outputs). We define

$$\overline{A \times \mathbb{N}} \;=\; (A \times \mathbb{N}) \cup \{\infty\}$$

and $\mathsf{pre}_A\colon \overline{A \times \mathbb{N}} \to A + \overline{A \times \mathbb{N}}$ by

$$\mathsf{pre}_A((a,0)) \;=\; a \qquad \mathsf{pre}_A((a,n+1)) \;=\; (a,n) \qquad \mathsf{pre}_A(\infty) \;=\; \infty$$

Prove that $(\overline{A \times \mathbb{N}}, \mathsf{pre}_A)$ is a final partial system with outputs in A. $\square$

Example 207 (while programs). Let Σ be an abstract set of program states and let the set Prog of *while programs* be given by the following syntax:

$$P \;::=\; a \;\mid\; P;Q \;\mid\; \mathsf{if}\ c\ \mathsf{then}\ P\ \mathsf{else}\ Q \;\mid\; \mathsf{while}\ c\ \mathsf{do}\ P \qquad (a \in \mathsf{Act},\ c \in \mathsf{Cond})$$

where the sets of *atomic actions* and *conditions* are defined by

$$\mathsf{Act} \;=\; \{\, a \mid a \colon \Sigma \to \Sigma \,\} \qquad\qquad \mathsf{Cond} \;=\; \{\, c \mid c \subseteq \Sigma \,\}$$

Clearly, more concrete versions of Σ, Act and Cond could be defined. States would typically be given as functions from variables to values, skip statements and assignments would be typical atomic actions, and Boolean expressions could be used as a syntax for conditions.

In order to define the behaviour of while programs by coinduction, we define a partial system with outputs in Σ:

$$(\mathsf{Prog} \times \Sigma, \alpha) \qquad\qquad \alpha \colon (\mathsf{Prog} \times \Sigma) \;\to\; \Sigma \;+\; (\mathsf{Prog} \times \Sigma)$$

where α is defined, using the notation from (9.3) and (9.4) above, as follows. Actions $a \in \mathsf{Act}$ terminate in one step:

$$(a, \sigma) \uparrow a(\sigma)$$

The behaviour of composite programs is defined by induction on their syntactic structure, by the following rules:

$$\frac{(P, \sigma) \longrightarrow (P', \sigma')}{(P; Q, \sigma) \longrightarrow (P'; Q, \sigma')} \qquad\qquad \frac{(P, \sigma) \uparrow \sigma'}{(P; Q, \sigma) \longrightarrow (Q, \sigma')}$$

$$\frac{\sigma \in c}{(\mathsf{if}\ c\ \mathsf{then}\ P\ \mathsf{else}\ Q, \sigma) \longrightarrow (P, \sigma)} \qquad\qquad \frac{\sigma \notin c}{(\mathsf{if}\ c\ \mathsf{then}\ P\ \mathsf{else}\ Q, \sigma) \longrightarrow (Q, \sigma)}$$

$$\frac{\sigma \in c}{(\mathsf{while}\ c\ \mathsf{do}\ P, \sigma) \longrightarrow (P; \mathsf{while}\ c\ \mathsf{do}\ P, \sigma)} \qquad\qquad \frac{\sigma \notin c}{\mathsf{while}\ c\ \mathsf{do}\ P, \sigma) \uparrow \sigma}$$

By finality of $\overline{\Sigma \times \mathbb{N}}$, there exists a unique homomorphism

$$[\![-]\!] \colon (\mathsf{Prog} \times \Sigma) \;\to\; \overline{\Sigma \times \mathbb{N}}$$

It sends a pair (P, σ) either to ∞, in case the computation is diverging, or to a pair (σ', n), consisting of an output σ' and a natural number n representing the number of steps that it took to compute σ'. □

Exercise 208. Coinduction may now be used to establish some familiar identities. To this end, first formulate a coinduction proof principle for $\overline{\Sigma \times \mathbb{N}}$. Then use it to prove, for instance, that

$$[\![(\mathsf{if}\ c\ \mathsf{then}\ P\ \mathsf{else}\ Q\ , \sigma)]\!] \;=\; [\![(\mathsf{if}\ \Sigma - c\ \mathsf{then}\ Q\ \mathsf{else}\ P\ , \sigma)]\!]$$

for all $\sigma \in \Sigma$. Observe that the present semantics of while programs is not very abstract. For instance, if 1 is the identity function on Σ (corresponding to a skip statement) then, generally,

$$[\![(1; P, \sigma)]\!] \neq [\![(P, \sigma)]\!]$$

See [Rut99c] for a semantics that abstracts away from the actual number of computation steps, and a discussion of a related coinductive proof principle based on so-called *weak bisimulation.* □

9.3 Partial automata

Partial automata combine the automata from Chapter 8, having both inputs and outputs, and the partial systems from Section 9.1, having a transition function that is partial.

Formally, they are defined as follows. A *partial automaton* with inputs from a set A and outputs in the set $2 = \{0, 1\}$ is a pair $(S, \langle \mathsf{o}, \mathsf{tr} \rangle)$, consisting of a set S of *states*, and a *structure map*

$$\langle \mathsf{o}, \mathsf{tr} \rangle \colon S \to 2 \times (\mathbf{1} + S)^A \qquad \mathsf{o} \colon S \to 2 \qquad \mathsf{tr} \colon S \to (\mathbf{1} + S)^A$$

The function o maps every $s \in S$ to an output $\mathsf{o}(s) \in 2$. The function tr assigns to s a function $\mathsf{tr}(s) \colon A \to (\mathbf{1} + S)$, which for an input $a \in A$ is either undefined: $\mathsf{tr}(s)(a) = *$, in which case we write

$$s \overset{a}{\not\to}$$

or for which $\mathsf{tr}(s)(a) \in S$, in which case we say that s *accepts* the input a, denoted by $\mathsf{tr}(s)(a) = s_a$, and write

$$s \xrightarrow{a} s_a$$

We will also write, for $w \in A^*$,

$$s \xrightarrow{w}$$

to denote that s consecutively accepts all the inputs in the word w.

Example 209. Let $A = \{a, b\}$ and $S = \{s_0, s_1, s_2, s_3, s_4, s_5\}$. The following diagram defines a partial automaton:

$$\begin{tikzcd}
s_0 \arrow[r, bend left, "b"] \arrow[d, bend right, "a"'] & s_2 \arrow[r, bend left, "b"] \arrow[d, bend left, "a"] & s_4 \arrow[d, bend left, "a"] \\
\underline{s_1} & \underline{s_3} \arrow[loop left, "b"] & s_5
\end{tikzcd}$$

where we have underlined, as in Chapter 8, all states $s \in S$ with $\mathsf{o}(s) = 1$. We observe that the states s_0 and s_2 accept both the inputs a and b, s_1 and s_5 accept neither a nor b, s_3 accepts b but not a, and s_4 accepts a but not b. □

Remark 210. For those who have already read Chapters 2 to 5: partial automata are coalgebras of the functor

$$\mathsf{pA}\colon \mathsf{Set} \to \mathsf{Set} \qquad \mathsf{pA}(S) \;=\; 2 \times (\mathbf{1} + S)^A \qquad (S \in \mathsf{Set})$$

which is defined for functions $f\colon S \to T$ by

$$\mathsf{pA}(f)\colon 2 \times (\mathbf{1} + S)^A \to 2 \times (\mathbf{1} + T)^A \qquad \mathsf{pA}(f) \;=\; 1 \times (id + f)^A$$

Here 1 is the identity function on the set $2 = \{0, 1\}$ and the function

$$(id + f)^A\colon (\mathbf{1} + S)^A \to (\mathbf{1} + T)^A$$

is defined by

$$\big(\,(id + f)^A(g)\,\big)\,(a) \;=\; \begin{cases} * & \text{if } g(a) = * \\ f(g(a)) & \text{if } g(a) \in S \end{cases}$$

for all $g\colon A \to (\mathbf{1} + S)$ and $a \in A$. □

Exercise 211. Use the above functor to define the notion of homomorphism for partial automata, instantiating the notion of F-homomorphism from Chapter 3. □

Intuitively, there are two aspects to the behaviour of a state s in a partial automaton S, when presented with an input word $w \in A^*$. First, there is the question whether

$$s \xrightarrow{w}$$

that is, whether all the input letters in the word w are accepted (by s and the subsequent states). If this is the case then there is, secondly, the question whether the state s_w thus reached on input w has $\mathsf{o}(s_w) = 0$ or $\mathsf{o}(s_w) = 1$.

The above intuition is captured formally by the following description of a *final* partial automaton.

Definition 212 (partial languages). We define the set

$$\mathsf{pL} \subseteq \mathcal{P}(A^*) \times \mathcal{P}(A^*)$$

of *partial languages* by

$$\mathsf{pL} \;=\; \{\; (\kappa, \lambda) \mid \kappa \subseteq \lambda \subseteq A^*, \;\; \lambda \neq \emptyset, \;\; \lambda \text{ is prefix-closed } \;\}$$

where λ is prefix-closed if $w \in \lambda$ and $w = v \cdot u$ implies $v \in \lambda$, for all $u, v, w \in A^*$. □

We see that partial languages are *pairs* of classical languages, that is, pairs of sets of words. The set pL of partial languages can be turned into a partial automaton

$$(\mathsf{pL},\ \langle \mathsf{i}_{pl}, \mathsf{d}_{pl}\rangle) \qquad \mathsf{i}_{pl} \colon \mathsf{pL} \to 2 \qquad \mathsf{d}_{pl} \colon \mathsf{pL} \to (\mathbf{1} + \mathsf{pL})^A$$

by defining the output and transition functions as follows: for all $(\kappa, \lambda) \in \mathsf{pL}$,

$$\mathsf{i}_{pl}(\kappa,\lambda) \;=\; \begin{cases} 0 & \text{if } \varepsilon \notin \kappa \\ 1 & \text{if } \varepsilon \in \kappa \end{cases} \qquad \mathsf{d}_{pl}(\kappa,\lambda)(a) \;=\; \begin{cases} (\kappa_a, \lambda_a) & \text{if } a \in \lambda \\ * & \text{if } a \notin \lambda \end{cases}$$

where we recall, from Section 8.3, the definition of a-derivative of a language:

$$\lambda_a \;=\; \{ w \in A^* \mid a \cdot w \in \lambda \}$$

Note that if $a \in \lambda$ and $(\kappa,\lambda) \in \mathsf{pL}$ then also $(\kappa_a, \lambda_a) \in \mathsf{pL}$.

The automaton $(\mathsf{pL},\ \langle \mathsf{i}_{pl}, \mathsf{d}_{pl}\rangle)$ is *final.*

Theorem 213 (finality – partial languages). *For every partial automaton* $(S,\ \langle \mathsf{o}, \mathsf{tr}\rangle)$ *there exists a* unique *homomorphism* $[\![-]\!] \colon (S,\ \langle \mathsf{o}, \mathsf{tr}\rangle) \to (\mathsf{pL},\ \langle \mathsf{i}_{pl}, \mathsf{d}_{pl}\rangle)$*:*

$$\begin{tikzcd} S \arrow[rr, dashed, "\exists!\, [\![-]\!]"] \arrow[d, "\forall\, \langle \mathsf{o}, \mathsf{tr}\rangle"'] & & \mathsf{pL} \arrow[d, "\langle \mathsf{i}_{pl}, \mathsf{d}_{pl}\rangle"] \\ 2 \times (\mathbf{1} + S)^A \arrow[rr, dashed, "id \times [\![-]\!]^A"'] & & 2 \times (\mathbf{1} + \mathsf{pL})^A \end{tikzcd}$$

Exercise 214. Define $[\![-]\!] \colon S \to \mathsf{pL}$, for $s \in S$, by

$$[\![s]\!] \;=\; (\, \{ w \in A^* \mid \ s \xrightarrow{w} \ \text{ and } \mathsf{o}(s_w) = 1 \},\ \{ w \in A^* \mid \ s \xrightarrow{w} \ \} \,)$$

Prove that $[\![-]\!]$ is the only function making the diagram above commute. □

Exercise 215. Prove that the set of partial languages pL is isomorphic to the set

$$\{\ f \colon A^* \to \mathbf{1} + 2 \mid f^{-1}(2) \neq \emptyset \quad \text{and} \quad f^{-1}(2) \text{ is prefix-closed } \ \}$$

Define a partial automaton structure on this set making it isomorphic to $(\mathsf{pL},\ \langle \mathsf{i}_{pl}, \mathsf{d}_{pl}\rangle)$. □

Example 216. Returning to Example 209,

$$\begin{tikzcd}
s_0 \arrow[r, bend left, "b"] \arrow[d, bend right, "a"'] & s_2 \arrow[r, bend left, "b"] \arrow[d, bend left, "a"] & s_4 \arrow[d, bend left, "a"] \\
\underline{s_1} & \underline{s_3} \arrow[loop left, "b"] & s_5
\end{tikzcd}$$

we have

$$[\![s_0]\!] \;=\; (a + bab^*,\, 1 + a + b + bab^* + bb + bba) \qquad \square$$

Exercise 217. In the diagram below,

$$\begin{tikzcd}
S \arrow[rr, "\kappa_2"] \arrow[d, "\langle \mathsf{o}, \mathsf{tr} \rangle"'] & & \mathbf{1} + S \arrow[rr, dashed, "\exists! h"] \arrow[dll, bend left, "\langle \overline{\mathsf{o}}, \overline{\mathsf{tr}} \rangle"] & & \mathcal{P}(A^*) \arrow[d, "\langle \mathsf{i}_l, \mathsf{d}_l \rangle"] \\
2 \times (\mathbf{1} + S)^A \arrow[rrrr, dashed, "1 \times h^A"'] & & & & 2 \times \mathcal{P}(A^*)^A
\end{tikzcd}$$

the function $\kappa_2 \colon S \to \mathbf{1} + S$ is the embedding of S into the disjoint union $\mathbf{1} + S$. Turn the partial automaton $(S,\, \langle \mathsf{o}, \mathsf{tr} \rangle)$ into a corresponding deterministic automaton, by defining functions $\overline{\mathsf{o}}$ and $\overline{\mathsf{tr}}$. The aim is to obtain by finality a homomorphism h such that the composition $h \circ \kappa_2$ defines a notion of behaviour in terms of ordinary languages that is "consistent" with the final semantics of Theorem 213 in terms of partial languages. (The diagram above closely resembles the one of the powerset construction for non-deterministic automata, which we shall see shortly, in Definition 221.) $\square$

9.4 Discussion

Partial automata have been used as models for so-called *supervised control* of discrete event systems [RW87] in [Rut99b, Rut00b]. There a variation of the standard coalgebraic notion of bisimulation for partial automata is defined, called *partial bisimulation.* Recent studies of partial bisimulations include [BvBL$^+$11, Mar16].

Chapter 10

Non-deterministic automata

A *non-deterministic automaton* (ndA) with inputs from a set A and outputs in $2 = \{0, 1\}$ is a pair $(S, \langle \mathsf{o}, \mathsf{tr} \rangle)$, consisting of a set S of *states* and a *structure map*

$$\langle \mathsf{o}, \mathsf{tr} \rangle \colon S \to 2 \times \mathcal{P}(S)^A \qquad \mathsf{o} \colon S \to 2 \qquad \mathsf{tr} \colon S \to \mathcal{P}(S)^A \tag{ndA}$$

The *output* function o maps every state $s \in S$ to an output $\mathsf{o}(s) \in 2$, and the *transition* function tr assigns to every $s \in S$ a function

$$\mathsf{tr}(s) \colon A \to \mathcal{P}(S)$$

that maps any $a \in A$ to a *set of possible next states* $\mathsf{tr}(s)(a) \subseteq S$.

We recall from Chapter 8 that deterministic automata (dA's) are of type

$$\langle \mathsf{o}, \mathsf{tr} \rangle \colon S \to 2 \times S^A \qquad \mathsf{o} \colon S \to 2 \qquad \mathsf{tr} \colon S \to S^A \tag{dA}$$

Non-deterministic automata generalise deterministic ones by allowing states to have, on a given input, zero, one or more successor states, rather than precisely one. As a consequence, ndA's are often smaller than corresponding (language equivalent) dA's.

In the present chapter, we shall study the behaviour of non-deterministic automata by transforming them into deterministic automata, using the so-called *powerset construction.* The language accepted by the resulting dA will then be taken as the definition of the behaviour of the original ndA.

The powerset construction is an interesting example of how one type of system can be transformed into another one. The powerset construction will lead, furthermore, to interesting further enhancements of the coinduction proof principle, called coinduction *up-to-union* and coinduction *up-to-congruence w.r.t. union.*

Remark 218. There is also an alternative treatment of ndA's that does not translate them into dA's. Instead, ndA's are viewed as coalgebras of the functor $\mathsf{ndA} \colon \mathsf{Set} \to \mathsf{Set}$, which is defined by $\mathsf{ndA}(S) = S \to \mathcal{P}(S)^A$ and which will be discussed at the end of this chapter, in Remark 233. □

Example 219. Here is an example of an ndA, with $A = \{a, b\}$:

$$(S,\ \langle \mathsf{o}, \mathsf{tr}\rangle) \quad = \quad \begin{tikzcd} s \arrow[loop left, "{a,b}"] \arrow[r, "a"] & t \arrow[r, "{a,b}"] & \underline{u} \end{tikzcd}$$

Underlining accepting states – with output value 1 – as before, the definition of o and tr can again be read from the diagram:

$$\begin{array}{rclcrclcrcl} \mathsf{o}(s) & = & 0 & \quad & \mathsf{o}(t) & = & 0 & \quad & \mathsf{o}(u) & = & 1 \\ \mathsf{tr}(s)(a) & = & \{s, t\} & & \mathsf{tr}(t)(a) & = & \{u\} & & \mathsf{tr}(u)(a) & = & \emptyset \\ \mathsf{tr}(s)(b) & = & \{s\} & & \mathsf{tr}(t)(b) & = & \{u\} & & \mathsf{tr}(u)(b) & = & \emptyset \end{array}$$

Note that there are two a-transitions from s and no transitions from u. □

We say that a state s *accepts a word* $w = a_0a_1 \cdots a_{n-1} \in A^*$ if there exists a sequence of transitions

$$s \;=\; s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} \cdots \xrightarrow{a_{n-1}} \underline{s_n} \qquad\qquad (s_{i+1} \in \mathsf{tr}(s_i)(a_i))$$

starting in s and leading to an accepting state s_n. In general, there can be more than one such sequence for a given word w. We denote the set of all words accepted by a state s by

$$l(s) \;=\; \{w \in A^* \mid s \text{ accepts the word } w\ \} \tag{10.1}$$

and we call $l(s)$ the *language accepted* by s. For instance, in Example 219, we have

$$l(s) \;=\; \{\, w \in A^* \mid w = vaa \text{ or } w = vab, \text{ for some } v \in A^* \,\}$$

Exercise 220. Let $A = \{a, b\}$ and consider the following ndA:

$$\begin{tikzcd} s_1 \arrow[loop left, "{a,b}"] \arrow[r, "a"] & s_2 \arrow[r, "{a,b}"] & s_3 \arrow[r, "{a,b}"] & \underline{s_4} \end{tikzcd}$$

Show that the language accepted by the state s_1 is

$$l(s_1) \;=\; \{\, w \in A^* \mid |w| > 2 \text{ and the third letter from the right is an } a \,\}$$

Use the methods from Section 8.6 to construct a minimal dA accepting the language $l(s_1)$. As it turns out, the resulting dA will have 8 states. □

10.1 The powerset construction

Every non-deterministic automaton can be turned into a deterministic one with the same behaviour. As we will see, the resulting deterministic automaton has in general more states than the original non-deterministic one. Thus ndA's have the advantage over dA's of offering more economic language representations.

Definition 221 (the powerset construction). For every non-deterministic automaton

$$\langle \mathsf{o}, \mathsf{tr}\rangle \colon S \to 2 \times \mathcal{P}(S)^A \qquad \mathsf{o}\colon S \to 2 \qquad \mathsf{tr}\colon S \to \mathcal{P}(S)^A$$

we define a deterministic automaton

$$\langle \overline{\mathsf{o}}, \overline{\mathsf{tr}}\rangle \colon \mathcal{P}(S) \to 2 \times \mathcal{P}(S)^A \qquad \overline{\mathsf{o}}\colon \mathcal{P}(S) \to 2 \qquad \overline{\mathsf{tr}}\colon \mathcal{P}(S) \to \mathcal{P}(S)^A$$

by

$$\overline{\mathsf{o}}(V) \;=\; \begin{cases} 1 & \text{if } \exists\, s \in V \text{ s.t. } \mathsf{o}(s) = 1 \\ 0 & \text{otherwise} \end{cases} \qquad \overline{\mathsf{tr}}(V)(a) \;=\; \bigcup \{\mathsf{tr}(s)(a) \mid s \in V\}$$

where the big union is defined as the union of all subsets $\mathsf{tr}(s)(a) \subseteq S$, for all $s \in V$. We sometimes call $\mathcal{P}(S)$ the *determinisation* of S. □

It follows from the diagram below that the deterministic automaton $(\mathcal{P}(S),\ \langle \overline{\mathsf{o}}, \overline{\mathsf{tr}}\rangle)$ has the same behaviour as the original non-deterministic automaton $(S,\ \langle \mathsf{o}, \mathsf{tr}\rangle)$:

$$\begin{tikzcd}
S \arrow[r, "\{\cdot\}"'] \arrow[rr, bend left, "l"] \arrow[d, "\langle \mathsf{o}, \mathsf{tr}\rangle"'] & \mathcal{P}(S) \arrow[r, dashed, "\exists!\, [\![-]\!]"'] \arrow[dl, bend left, "\langle \overline{\mathsf{o}}, \overline{\mathsf{tr}}\rangle"] & \mathcal{P}(A^*) \arrow[d, "\langle \mathsf{i}_l, \mathsf{d}_l\rangle"] \\
2 \times \mathcal{P}(S)^A \arrow[rr, dashed, "1 \times [\![-]\!]^A"] & & 2 \times \mathcal{P}(A^*)^A
\end{tikzcd}$$

where we have the following ingredients:

- the function $l\colon S \to \mathcal{P}(A^*)$, which was defined in (10.1), maps a state s in the ndA $(S,\ \langle \mathsf{o}, \mathsf{tr}\rangle)$ to the language $l(s)$ it accepts.
- the unique homomorphism of deterministic automata $[\![-]\!]\colon \mathcal{P}(S) \to \mathcal{P}(A^*)$ which is given by the finality of the set of languages, Theorem 153, maps a state V in the dA $(\mathcal{P}(S),\ \langle \overline{\mathsf{o}}, \overline{\mathsf{tr}}\rangle)$ to the language $[\![V]\!]$ it accepts.
- the function $\{\cdot\}\colon S \to \mathcal{P}(S)$ maps a state s in the ndA $(S,\ \langle \mathsf{o}, \mathsf{tr}\rangle)$ to the singleton set $\{s\} \subseteq S$, which is a state of the dA $(\mathcal{P}(S),\ \langle \overline{\mathsf{o}}, \overline{\mathsf{tr}}\rangle)$.

Theorem 222. *In the diagram above, everything commutes. In particular,*

$$[\![\{s\}]\!] = l(s) \tag{10.2}$$

That is, the language recognised by the state $\{s\}$ in the new automaton is the same as the language recognised by the state s in the original automaton. □

Exercise 223. Prove identity (10.2). □

Example 224. For the ndA of Example 219, the powerset construction yields a dA with $\mathcal{P}(\{s,t,u\})$ as the new set of states, containing 2^3 elements. If we take the subautomaton generated by the state $\{s\}$, we obtain

$$\begin{tikzcd}
\{s\} \arrow[loop above, "b"] \arrow[r, "a"] & \{s,t\} \arrow[d, "a"] \arrow[dl, bend left, "b"] \\
\underline{\{s,u\}} \arrow[u, "b"] \arrow[ur, bend left, "a"] & \underline{\{s,t,u\}} \arrow[l, "b"] \arrow[loop below, "a"]
\end{tikzcd}$$

This dA is minimal and has 4 states, which is one more than the original ndA. □

Exercise 225. Apply the powerset construction to the ndA of Exercise 220 and compare the result to the dA for the language $l(s_1)$ that was constructed in the same exercise. □

10.2 Language equivalence of non-deterministic automata

If we want to prove that two (states in two) ndA's are *language equivalent*, that is, accept the same language, then a naive approach would be to: (i) use the powerset construction to obtain two corresponding dA's; (ii) construct a bisimulation relation (of deterministic automata, Definition 131) relating the two dA's; and (iii) use the coinduction proof principle for deterministic automata, Corollary 157, to conclude that they accept the same language.

Example 226 (**[BP15]**)**.** Let $A = \{a\}$ and consider the following two ndA's:

$$\begin{tikzcd}
x & z \arrow[l, "a"] \arrow[r, bend left, "a"] & \underline{y} \arrow[l, bend left, "a"] \arrow[from=1-1, to=1-3, bend left=40, "a"]
\end{tikzcd}
\qquad
\begin{tikzcd}
u \arrow[r, bend left, "a"] \arrow[rr, bend left=40, "a"] & w \arrow[l, bend left, "a"] & \underline{v} \arrow[l, "a"]
\end{tikzcd}$$

Applying the powerset construction yields the following two corresponding dA's, where we have taken the subautomata generated by, respectively, $\{x\}$ and $\{u\}$:

$$\begin{tikzcd}
\{x\} \arrow[r, "a"] \arrow[d, dash, dashed, "1"'] & \underline{\{y\}} \arrow[r, "a"] \arrow[d, dash, dashed, "2"'] & \{z\} \arrow[r, "a"] \arrow[d, dash, dashed, "3"'] & \underline{\{x,y\}} \arrow[r, "a"] \arrow[d, dash, dashed, "4"'] & \underline{\{y,z\}} \arrow[r, "a"] \arrow[dl, dash, dashed, "5"'] & \underline{\{x,y,z\}} \arrow[loop below, "a"] \arrow[dll, dash, dashed, "6"] \\
\{u\} \arrow[r, "a"] & \underline{\{v,w\}} \arrow[r, "a"] & \{u,w\} \arrow[r, "a"] & \underline{\{u,v,w\}} \arrow[loop below, "a"] & &
\end{tikzcd}$$

The numbered, dashed lines above define a bisimulation relation of deterministic automata between the upper and the lower automaton. It follows by the coinduction proof principle for deterministic automata, Corollary 157, that $\{x\}$ and $\{u\}$ accept the same language. By identity (10.2) above, also x and u accept the same languages. □

Exercise 227. Let $A = \{a\}$ and consider the following two ndA's:

$$\begin{tikzcd}
\underline{x} \arrow[r, bend left, "a"] \arrow[rr, bend left=40, "a"] & \underline{y} \arrow[l, bend left, "a"] & z \arrow[l, "a"]
\end{tikzcd}
\qquad
\begin{tikzcd}
\underline{u} \arrow[loop above, "a"]
\end{tikzcd}$$

Apply the above procedure to prove that $l(x) = l(u)$. □

One can do better, meaning: construct a smaller relation and still conclude that x and u are language equivalent, by exploiting the special structure of the corresponding determinised automata. Their state sets are powersets and come equipped with the algebraic operation of *set union*:

$$\cup \colon \mathcal{P}(S) \times \mathcal{P}(S) \to \mathcal{P}(S) \qquad\qquad (V, W) \mapsto V \cup W \qquad\qquad \text{(union)}$$

This leads to the following notion.

Definition 228 (**bisimulation up-to-union**)**.** Let

$$(S, \langle \mathsf{o}_S, \mathsf{tr}_S \rangle) \qquad\qquad (T, \langle \mathsf{o}_T, \mathsf{tr}_T \rangle)$$

be two deterministic automata, and let

$$(\mathcal{P}(S),\ \langle \overline{\mathsf{o}_S}, \overline{\mathsf{tr}_S} \rangle) \qquad\qquad (\mathcal{P}(T),\ \langle \overline{\mathsf{o}_T}, \overline{\mathsf{tr}_T} \rangle)$$

be their corresponding determinisations. A relation

$$R \ \subseteq\ \mathcal{P}(S) \times \mathcal{P}(T)$$

is a *bisimulation up-to-union* if, for all $(V, W) \in R$,

$$(i)\ \ \overline{\mathsf{o}_S}(V) \ =\ \overline{\mathsf{o}_T}(W) \quad \text{and} \qquad (ii)\ \ \forall\, a \in A\colon \quad (\,V_a,\, W_a\,) \in R^{\cup}$$

where $V_a = \overline{\mathsf{tr}_S}(V)(a)$ and $W_a = \overline{\mathsf{tr}_T}(W)(a)$; and where

$$R^{\cup} \ \subseteq\ \mathcal{P}(S) \times \mathcal{P}(T)$$

is the *closure of R with respect to union*: the smallest relation such that

1. $R \subseteq R^{\cup}$
2. if $(C, D) \in R^{\cup}$ and $(E, F) \in R^{\cup}$ then $(C \cup E,\ D \cup F) \in R^{\cup}$ □

Proposition 229 (coinduction up-to-union). If $R \subseteq \mathcal{P}(S) \times \mathcal{P}(T)$ is a bisimulation up-to-union and $(V, W) \in R$, then $l(V) = l(W)$. □

Exercise 230. Give a proof of Proposition 229, along the same lines as the proof of Proposition 106, for streams, and Proposition 172, for deterministic automata. □

Example 231 ([BP15]). We saw in Example 226 that the relation $R = \{1, 2, 3, 4, 5, 6\}$, consisting of all six pairs of related subsets below:

$$\begin{tikzcd}
\{x\} \arrow[r, "a"] \arrow[d, dash, dashed, "1"'] & \underline{\{y\}} \arrow[r, "a"] \arrow[d, dash, dashed, "2"'] & \{z\} \arrow[r, "a"] \arrow[d, dash, dashed, "3"'] & \underline{\{x, y\}} \arrow[r, "a"] \arrow[d, dash, dotted, "4"'] & \underline{\{y, z\}} \arrow[r, "a"] \arrow[dl, dash, dotted, "5"'] & \underline{\{x, y, z\}} \arrow[loop above, "a"] \arrow[dll, dash, dotted, "6"] \\
\{u\} \arrow[r, "a"] & \underline{\{v, w\}} \arrow[r, "a"] & \{u, w\} \arrow[r, "a"] & \underline{\{u, v, w\}} \arrow[loop below, "a"] & &
\end{tikzcd}$$

is a bisimulation. The subset $S = \{1, 2, 3\} \subseteq R$ is a bisimulation up-to-union, since

$$\begin{aligned}
\{z\}_a \ &=\ \{x, y\} \\
&=\ \{x\} \cup \{y\} \\
&S^{\cup}\ \{u\} \cup \{v, w\} && \text{(since } \{x\}S\{u\} \text{ and } \{y\}S\{v, w\}) \\
&=\ \{u, v, w\} \\
&=\ \{u, w\}_a
\end{aligned}$$

The construction of the smaller relation S is sufficient to conclude that $\{x\}$ and $\{u\}$ accept the same language, by Proposition 229. □

Exercise 232 (bisimulation up-to-congruence w.r.t. union). Consider the following variation on Definition 228. A relation

$$R \subseteq \mathcal{P}(S) \times \mathcal{P}(T)$$

is a *bisimulation up-to-congruence* with respect to union if, for all $(V, W) \in R$,

$$(i)\ \overline{\mathsf{o}_S}(V) = \overline{\mathsf{o}_T}(W) \quad \text{and} \qquad (ii)\ \forall\, a \in A\colon \quad (\, V_a,\, W_a\,) \in R^c$$

where now

$$R^c \subseteq \mathcal{P}(S) \times \mathcal{P}(T)$$

is the *congruence closure* of R with respect to union: the smallest relation such that

1. $R \subseteq R^c$
2. R^c is an equivalence relation
3. if $(C, D) \in R^c$ and $(E, F) \in R^c$ then $(C \cup E,\ D \cup F) \in R^c$

Prove a corresponding principle of coinduction up-to-congruence. Apply it to the two automata from Exercise 227 to conclude that $l(x) = l(u)$, by proving that

$$\{\, (\{x\},\ \{u\}),\ (\{y, z\},\ \{u\})\,\}$$

is a bisimulation up-to-congruence. □

Remark 233 (coalgebraic bisimulation for non-deterministic automata). For those who have already read Chapters 2 to 5: we have defined the behaviour of ndA's in terms of language acceptance or, equivalently, by transforming them into dA's by means of the powerset construction, Definition 221. In doing so, we have followed the approach from the classical theory of automata.

One can alternatively develop a theory of ndA's without translating them to dA's, by viewing them as ndA-coalgebras for the functor $\mathsf{ndA}\colon \mathsf{Set} \to \mathsf{Set}$. This functor is defined, for sets S and functions $f\colon S \to T$, by

$$\mathsf{ndA}(S) = 2 \times \mathcal{P}(S)^A \qquad \mathsf{ndA}(f) = \left(1 \times \mathcal{P}(f)^A\right) : \left(2 \times \mathcal{P}(S)^A\right) \to \left(2 \times \mathcal{P}(T)^A\right)$$

where 1 is the identity function on the set 2 and where, for $g\colon A \to \mathcal{P}(S)$ and $a \in A$,

$$\mathcal{P}(f)^A\colon \mathcal{P}(S)^A \to \mathcal{P}(T)^A \qquad\qquad \mathcal{P}(f)^A(g)(a) = f(g(a))$$

which is the image of $g(a) \subseteq S$ under f.

If we view ndA's as ndA-coalgebras then a rather different picture arises, of which we will briefly sketch the main ingredients next. If we apply the general Definition 21 of F-homomorphism and the general Definition 33 of F-bisimulation to the functor ndA, then it is a little exercise to prove that a relation $R \subseteq S \times T$ is an ndA-bisimulation:

$$\begin{tikzcd}
S \arrow[d, "{\langle \mathsf{o}_S, \mathsf{tr}_S \rangle}"'] & R \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] \arrow[d, dashed, "{\exists\ \langle \mathsf{o}_R, \mathsf{tr}_R \rangle}"] & T \arrow[d, "{\langle \mathsf{o}_T, \mathsf{tr}_T \rangle}"] \\
2 \times \mathcal{P}(S)^A & 2 \times \mathcal{P}(R)^A \arrow[l, "1 \times \mathcal{P}(\pi_1)^A"] \arrow[r, "1 \times \mathcal{P}(\pi_2)^A"'] & 2 \times \mathcal{P}(T)^A
\end{tikzcd}$$

if and only if, for all $(s, t) \in R$,

(i) $\mathsf{o}_S(s) = \mathsf{o}_T(t)$

(ii) $\forall\ s \xrightarrow{a} s' \quad \exists\ t' \xleftarrow{a} t \quad$ s.t. $\quad (s', t') \in R$

(iii) $\forall\ t \xrightarrow{a} t' \quad \exists\ s' \xleftarrow{a} s \quad$ s.t. $\quad (s', t') \in R$

The notion of ndA-bisimilarity is rather different from language equivalence, as the following two ndA's illustrate:

$$\begin{tikzcd}
& s_1 \arrow[d, "a"] & \\
& s_2 \arrow[dl, bend right, "b"'] \arrow[dr, bend left, "c"] & \\
\underline{s_3} & & \underline{s_4}
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
& t_1 \arrow[dl, bend right, "a"'] \arrow[dr, bend left, "a"] & \\
t_2 \arrow[d, "b"'] & & t_3 \arrow[d, "c"] \\
\underline{t_4} & & \underline{t_5}
\end{tikzcd}$$

The states s_1 and t_1 are language equivalent, since they accept the same language $\{ab, ac\}$. However, they are not ndA-bisimilar: the state s_2, which has a choice of doing either a b-transition or a c-transition, cannot be related to any state on the right with which it would be bisimilar. The notion of ndA-bisimulation is sensitive to the *branching* structure of the automaton, whereas language equivalence is not. One can easily show that ndA-bisimilarity implies language equivalence; by the example above, the converse does not hold. □

10.3 Discussion

Non-deterministic automata and their translation to deterministic automata via the powerset construction are a classical subject in automata theory [HMU07]. The coalgebraic presentation of the powerset construction in Section 10.1 goes back to [RT93]. The discussion of language equivalence of non-deterministic automata, in Section 10.2, follows [BP13, BP15]. There Bonchi and Pous use coinduction up-to-union and coinduction up-to-congruence w.r.t. union for an optimisation of the classical algorithm by Hopcroft and Karp [HK71] for proving language equivalence of non-deterministic automata.

The notion of ndA-bisimulation from Remark 233 goes back to Park [Par81] and Milner [Mil80], and became a corner stone of the theory of *concurrency* [Mil89]; see Sangiorgi's chapter in [SR12] on the origin of bisimulation and coinduction. Interestingly, it was this notion of bisimulation that was generalised by Aczel and Mendler [Acz88, AM89] to the coalgebraic notion of F-bisimulation, Definition 33 in the present book.

Chapter 11

Stream differential equations

We introduce *stream differential equations* as a format for the coinductive definition of streams. Here is the main idea. In Chapter 7, we defined, for a stream $\sigma \in A^\omega$,

$$\begin{aligned} \sigma(0) &\in A && \text{(initial value)} \\ \sigma' &= (\sigma(1), \sigma(2), \sigma(3), \ldots) \in A^\omega && \text{(stream derivative)} \end{aligned}$$

Now let $A = \{a, b\}$ and consider the following two streams:

$$\sigma \in A^\omega \qquad \sigma(2n) = a \qquad \sigma(2n+1) = b \tag{11.1}$$

$$\tau \in A^\omega \qquad \tau(2n) = b \qquad \tau(2n+1) = a \tag{11.2}$$

Thus

$$\begin{aligned} \sigma &= (a, b, a, b, a, b, \ldots) \\ \tau &= (b, a, b, a, b, a, \ldots) \end{aligned}$$

and

$$\sigma(0) = a \qquad \sigma' = \tau \tag{11.3}$$

$$\tau(0) = b \qquad \tau' = \sigma \tag{11.4}$$

In analogy to calculus in mathematical analysis, we call identities (11.3) and (11.4) (a system of two) *stream differential equations*, or SDEs for short, since these equations describe the streams σ and τ in terms of their initial values and derivatives.

Note that in the above, we have first *defined* what the streams σ and τ *are*, by specifying their individual elements in equations (11.1) and (11.2). Next we have *characterised their behaviour*, in stream differential equations (11.3) and (11.4), by describing their initial values and derivatives or, in other words, their outputs and transitions.

In the present chapter, we will forget about elementwise definitions such as (11.1) and (11.2). Instead, we will use stream differential equations as *definitions* or *behavioural specifications* of streams. We will sometimes derive from the defining differential equations elementwise characterisations of the specified streams, but typically, we will not bother to do so: *the stream differential equations say it all.* More specifically, SDEs tell us what the initial values and derivatives of the defined streams are, which is all one needs to know to build suitable bisimulations in proofs by coinduction.

Using stream differential equations as definitions, one has to make sure that they are *well-formed.* More precisely, given a system of stream differential equations, one has to prove that it *has a solution*, that is, that there exist streams whose initial values and derivatives satisfy the given stream differential equations. Moreover, one has to show that these streams are the only ones with this property, in other words, one has to prove that the solution is *unique.*

With the phrase *solving a system of stream differential equations*, we refer to the process of finding its unique solution. The general methodology for solving a given system of stream differential equations will be to *define* its unique solution *by coinduction*, using the finality of A^ω. In the example of σ and τ above, we would solve the system of stream differential equations (11.3) and (11.4) by defining a stream system

$$\begin{tikzcd}(S, \langle \mathsf{o}, \mathsf{tr} \rangle) \; = \; s|a \arrow[r, bend left] & t|b \arrow[l, bend left] \end{tikzcd} \qquad (S \; = \; \{s, t\})$$

Next we would define two streams

$$\sigma \; = \; [\![s]\!] \qquad\qquad \tau \; = \; [\![t]\!] \tag{11.5}$$

where

$$[\![-]\!] \colon (S, \langle \mathsf{o}, \mathsf{tr} \rangle) \to (A^\omega, \langle \mathsf{i}_{st}, \mathsf{d}_{st} \rangle)$$

is the unique homomorphism given by the finality of A^ω, Theorem 78. Because $[\![-]\!]$ is a homomorphism of stream systems, we have

$$[\![s]\!](0) \; = \; \mathsf{o}(s) \qquad\qquad [\![s]\!]' \; = \; [\![\mathsf{tr}(s)]\!] \tag{11.6}$$

$$[\![t]\!](0) \; = \; \mathsf{o}(t) \qquad\qquad [\![t]\!]' \; = \; [\![\mathsf{tr}(t)]\!] \tag{11.7}$$

(recall that $\mathsf{i}_{st}(\rho) = \rho(0)$ and $\mathsf{d}_{st}(\rho) = \rho'$). Using the definition of $(S, \langle \mathsf{o}, \mathsf{tr} \rangle)$ and the definition of σ and τ in (11.5), we see that (11.6) and (11.7) are equivalent to

$$\sigma(0) \; = \; a \qquad\qquad \sigma' \; = \; \tau$$

$$\tau(0) \; = \; b \qquad\qquad \tau' \; = \; \sigma$$

This proves that σ and τ satisfy the system of stream differential equations (11.3) and (11.4). Because the homomorphism $[\![-]\!]$ is unique by finality, it follows that this solution is unique.

Exercise 234. Prove that σ and τ defined in (11.5) are the only streams satisfying (11.3) and (11.4). □

In the remainder of this chapter, we shall first look at a series of examples and non-examples of well-formed stream differential equations and their solutions, in Section 11.1. Then we shall describe a rather general way to solve stream differential equations, called the *syntactic solution method.* The syntactic method will be illustrated with an example first, in Section 11.2. This example is generic enough to understand how similar systems of stream differential equations can be solved, including all those that will play a role in the remainder of the book. Section 11.3 then gives a general account of the syntactic method, in all formal detail, for a large class of SDEs in so-called *GSOS* format.

11.1 Examples of stream differential equations

In the present section, we take

$$A = \mathbb{N}$$

and consider streams in $\mathbb{N}^\omega$. We will present various examples of SDEs illustrating their convenience and expressiveness as a formalism for defining streams and stream functions. For all equations, we will mention their solution but not yet how it was found, leaving the discussion of general methods for solving SDEs until Section 11.2. We have already encountered several of the SDEs below in Chapter 7, where they were presented as *properties* of various streams and stream functions. Instead, SDEs are taken now as *definitions.*

Our first SDE

$$\sigma(0) = 1 \qquad\qquad \sigma' = \sigma \tag{11.8}$$

defines the derivative of σ in terms of σ itself and has the stream

$$\mathsf{ones} = (1, 1, 1, \ldots) \tag{11.9}$$

as its solution. Next we define a *function* on streams:

$$+ \colon \mathbb{N}^\omega \times \mathbb{N}^\omega \to \mathbb{N}^\omega$$

by the following *system* of SDEs:

$$(\sigma + \tau)(0) = \sigma(0) + \tau(0) \qquad\qquad (\sigma + \tau)' = \sigma' + \tau' \tag{11.10}$$

We have one SDE for each pair of streams σ and τ. The derivative $(\sigma + \tau)'$ is defined in terms of the function $+$ itself, now applied to the derivatives of the arguments. The solution of (11.10) is the function of elementwise sum:

$$+ \colon \mathbb{N}^\omega \times \mathbb{N}^\omega \to \mathbb{N}^\omega \qquad\qquad (\sigma + \tau)(n) = \sigma(n) + \tau(n) \tag{11.11}$$

The following system of SDEs

$$(\sigma \odot \tau)(0) = \sigma(0) \cdot \tau(0) \qquad\qquad (\sigma \odot \tau)' = \sigma' \odot \tau' \tag{11.12}$$

defines, similarly, the elementwise product of streams:

$$\odot\colon \mathbb{N}^\omega \times \mathbb{N}^\omega \to \mathbb{N}^\omega \qquad (\sigma \odot \tau)(n) \;=\; \sigma(n) \cdot \tau(n) \tag{11.13}$$

also known as the *Hadamard product.* Next, the SDE

$$\sigma(0) \;=\; 1 \qquad \sigma' \;=\; \sigma + \sigma \tag{11.14}$$

defines the stream of all powers of 2:

$$(2^0, 2^1, 2^2, \ldots) \tag{11.15}$$

The function + is used again in the following system of two SDEs:

$$\begin{aligned} \sigma(0) \;&=\; 0 \qquad & \sigma' \;&=\; \tau \\ \tau(0) \;&=\; 1 \qquad & \tau' \;&=\; \sigma + \tau \end{aligned} \tag{11.16}$$

which has the stream of *Fibonacci numbers* [OEI, A000045]

$$\mathsf{fib} \;=\; (0, 1, 1, 2, 3, 5, 8, \ldots) \tag{11.17}$$

as its solution. More precisely, the streams $\sigma = \mathsf{fib}$ and $\tau = \mathsf{fib}'$ are a solution of (11.16). We observe that (11.16), which consists of *two* SDEs, can equivalently be presented as *one* so-called *higher-order* SDE:

$$\sigma(0) \;=\; 0 \qquad \sigma'(0) \;=\; 1 \qquad \sigma'' \;=\; \sigma + \sigma' \tag{11.18}$$

Exercise 235 (recurrence for the Fibonacci numbers). In mathematics, streams such as the Fibonacci numbers are often defined by induction, using *recurrence* relations. For instance, one can define a stream

$$(f_0, f_1, f_2, \ldots) \;\in\; \mathbb{N}^\omega \qquad f_0 \;=\; 0 \qquad f_1 \;=\; 1 \qquad f_{n+2} \;=\; f_n + f_{n+1}$$

Show that $(f_0, f_1, f_2, \ldots) = \mathsf{fib}$. □

In the next SDE,

$$\sigma(0) \;=\; 1 \qquad \sigma' \;=\; \sigma + \mathsf{ones} \tag{11.19}$$

the derivative of σ is defined in terms of σ, the constant stream ones (11.9), and the function + (11.11). The solution of (11.19) is given by the stream of positive natural numbers:

$$\mathsf{nat} \;=\; (1, 2, 3, \ldots) \tag{11.20}$$

The following system of SDEs defines again a function on pairs of streams:

$$f(\sigma,\tau)(0) \;=\; \sigma(0)\cdot\tau(0) \qquad\qquad f(\sigma,\tau)' \;=\; f(\sigma',\tau)+f(\sigma,\tau') \tag{11.21}$$

It follows from Exercise 94 that the solution of (11.21) is given by

$$f(\sigma,\tau) \;=\; \sigma\otimes\tau$$

where the *shuffle product* satisfies

$$\otimes\colon\; \mathbb{N}^\omega\times\mathbb{N}^\omega\to\mathbb{N}^\omega \qquad\qquad (\sigma\otimes\tau)(n) \;=\; \sum_{k=0}^{n}\binom{n}{k}\cdot\sigma(k)\cdot\tau(n-k) \tag{11.22}$$

The following SDE

$$\sigma(0) \;=\; 1 \qquad\qquad \sigma' \;=\; \sigma\otimes\sigma \tag{11.23}$$

has the stream of *factorial numbers* as its solution:

$$\mathsf{fac} \;=\; (0!,1!,2!,\ldots) \tag{11.24}$$

Exercise 236. Show that the stream fac could equivalently be defined by

$$\sigma(0) \;=\; 1 \qquad\qquad \sigma' \;=\; \sigma\odot\mathsf{nat} \tag{11.25}$$

(the Hadamard product $\odot$ was defined in (11.13)). □

Here is an example showing that not always, all is well. First we define a function on streams by the following system of SDEs:

$$f(\sigma)(0) \;=\; \sigma(0) \qquad\qquad f(\sigma)' \;=\; f(\sigma'') \tag{11.26}$$

The derivative $f(\sigma)'$ is defined in terms of the function f itself, now applied to the *second derivative* of its argument. The unique solution of (11.26) is the function

$$\mathsf{even}\colon\;\mathbb{N}^\omega\to\mathbb{N}^\omega \qquad\qquad \mathsf{even}(\sigma) \;=\; (\sigma(0),\sigma(2),\sigma(4),\ldots) \tag{11.27}$$

Next we consider the following SDE:

$$\sigma(0) \;=\; 0 \qquad\qquad \sigma' \;=\; \mathsf{even}(\sigma) \tag{11.28}$$

This equation, which uses the function even and thereby implicitly refers to SDE (11.26), has both $\sigma=(0,0,0,\ldots)$ and $\sigma=(0,0,1,1,1,\ldots)$ as solutions. Since we want our SDEs to have unique solutions, we consider this SDE (11.28) or, more precisely, this system of SDEs consisting of both (11.26) and (11.28), as unacceptable. In the remainder of this chapter, we will formulate conditions on the syntactic shape of SDEs that will ensure that they have a unique solution. As we will see, the present system of SDEs consisting of (11.26) and (11.28), does *not* satisfy those syntactic conditions.

Exercise 237. Describe all solutions of the SDE (11.28). □

The problem lies really with the defining equation for the function even, SDE (11.26). The righthand side of (11.26) uses σ'', the second derivative of σ. Even though in isolation, Equation (11.26) is well-formed – it has the function even as its unique solution – we just saw that it may nevertheless give rise to systems of SDEs with more than one solution, such as (11.28). The syntactic conditions that we are about to introduce, will forbid the use of second derivatives and will thereby exclude SDEs such as (11.26).

11.2 Introducing the syntactic solution method

We discuss a general method for showing that a system of SDEs has a unique solution. Because the method associates with each such system a set of *terms*, we call it *syntactic*. As we shall see in Section 11.3, the method works for a large class of systems of SDEs that satisfy a rather general condition on their (syntactic) shape. In the present section, we first introduce the syntactic solution method by means of an example.

The stream ham of the *Hamming numbers* consists of all natural numbers of the form $2^i 3^j 5^k$, for any $i, j, k \geqslant 0$, in increasing order:

$$\begin{aligned}\mathsf{ham} &= (1, 2, 3, 4, 5, 6, 8, 9, 10, \ldots) \\ &= (2^0 3^0 5^0,\, 2^1 3^0 5^0,\, 2^0 3^1 5^0,\, 2^2 3^0 5^0,\, 2^0 3^0 5^1,\, 2^1 3^1 5^0,\, 2^3 3^0 5^0,\, 2^0 3^2 5^0,\, 2^1 3^0 5^1,\, \ldots)\end{aligned}$$

Exercise 238. In Exercise 235, we saw a recurrence relation for the stream of Fibonacci numbers. Can you find a recurrence relation for the Hamming numbers? □

Formally, we define the stream ham by the following SDE:

$$\mathsf{ham}(0) = 1 \qquad \mathsf{ham}' = (2 \times \mathsf{ham}) \parallel (3 \times \mathsf{ham}) \parallel (5 \times \mathsf{ham}) \tag{11.29}$$

and we will prove below that it has a unique solution. In the SDE above, the functions of scalar multiplication

$$k \times (-) \colon \mathbb{N}^\omega \to \mathbb{N}^\omega \qquad (k = 2, 3, 5)$$

are defined by the SDEs

$$(k \times \sigma)(0) = k \cdot \sigma(0) \qquad (k \times \sigma)' = k \times \sigma' \tag{11.30}$$

It follows that

$$k \times \sigma = (k \cdot \sigma(0),\, k \cdot \sigma(1),\, k \cdot \sigma(2),\, \ldots)$$

The function of *ordered merge*

$$\parallel \colon \mathbb{N}^\omega \times \mathbb{N}^\omega \to \mathbb{N}^\omega$$

is defined by the SDEs:

$$(\sigma \parallel \tau)(0) \;=\; \begin{cases} \sigma(0) & \text{if } \sigma(0) < \tau(0) \\ \sigma(0) & \text{if } \sigma(0) = \tau(0) \\ \tau(0) & \text{if } \sigma(0) > \tau(0) \end{cases} \qquad (\sigma \parallel \tau)' \;=\; \begin{cases} \sigma' \parallel \tau & \text{if } \sigma(0) < \tau(0) \\ \sigma' \parallel \tau' & \text{if } \sigma(0) = \tau(0) \\ \sigma \parallel \tau' & \text{if } \sigma(0) > \tau(0) \end{cases} \qquad (11.31)$$

For instance,

$$(0, 2, 4, 6, 8, \ldots) \;\parallel\; (0, 3, 6, 9, 12, \ldots) \;=\; (0, 2, 3, 4, 6, 8, 9, \ldots)$$

We will now prove the existence of a unique solution for the system of SDEs (11.29), (11.30), and (11.31). Let us suppose for a moment that this system of SDEs indeed has a solution. According to Lemma 77, any stream σ is given by the initial values of its successive derivatives $\sigma'(0), \sigma^{(2)}(0), \sigma^{(3)}(0), \ldots$. So we can investigate the behaviour of the stream ham by computing its derivatives:

$$\begin{aligned} \mathsf{ham}' &= (2 \times \mathsf{ham}) \parallel (3 \times \mathsf{ham}) \parallel (5 \times \mathsf{ham}) \\ \mathsf{ham}'' &= (2 \times \mathsf{ham}') \parallel (3 \times \mathsf{ham}) \parallel (5 \times \mathsf{ham}) \\ &= (2 \times ((2 \times \mathsf{ham}) \parallel (3 \times \mathsf{ham}) \parallel (5 \times \mathsf{ham}))) \parallel (3 \times \mathsf{ham}) \parallel (5 \times \mathsf{ham}) \end{aligned}$$

and so on. We see that these derivatives are given by syntactic expressions denoting repeated applications of the functions $\parallel$ and the three scalar multiplication functions, to the stream ham. Forgetting again about the existence of the stream ham, the syntactic method now consists of the following four steps:

(1) We model the collection of all the above derivatives as a set Term of *syntactic terms.*

(2) Then we use the SDEs (11.29), (11.30), and (11.31) to turn the set Term into a stream system

$$(\mathsf{Term}, \langle \mathsf{o}, \mathsf{tr} \rangle) \qquad\qquad \mathsf{o} \colon \mathsf{Term} \to \mathbb{N} \qquad\qquad \mathsf{tr} \colon \mathsf{Term} \to \mathsf{Term}$$

by inductively defining the functions o and tr.

(3) Next we use the unique homomorphism

$$[\![-]\!] \colon (\mathsf{Term}, \langle \mathsf{o}, \mathsf{tr} \rangle) \to (\mathbb{N}^\omega,\ \langle \mathsf{i}_{st}, \mathsf{d}_{st} \rangle)$$

given by finality, Theorem 78, to construct a candidate solution for our SDEs.

(4) Finally we prove that the proposed solution satisfies the SDEs and is unique.

We will now explain each of these four steps in detail.

(1) In order to define the set Term, we first introduce syntactic symbols representing the stream and the operators that we are after, by simply underlining their names. So we introduce a constant symbol $\underline{\mathsf{ham}}$, three unary term constructors $\underline{k\times}(-)$ for $k = 2, 3, 5$, and one binary term constructor $(-)\underline{\|}(-)$. Since we want to define our functions on arbitrary streams, we shall furthermore include in the set Term a constant symbol $\underline{\sigma}$ for each $\sigma \in \mathbb{N}^\omega$.

Formally, we define the set Term of all terms t as the smallest set that contains the constants $\underline{\mathsf{ham}}$ and $\underline{\sigma}$ (for all $\sigma \in \mathbb{N}^\omega$) and that is closed under the application of the term constructors:

$$\mathsf{Term} \ni \;\; t ::= \; \underline{\mathsf{ham}} \;\mid\; \underline{\sigma} \;\mid\; \underline{2\times}(t) \;\mid\; \underline{3\times}(t) \;\mid\; \underline{5\times}(t) \;\mid\; t_1 \,\underline{\|}\, t_2 \qquad (\sigma \in \mathbb{N}^\omega)$$

(2) Next we define

$$\mathsf{o}\colon \mathsf{Term} \to \mathbb{N} \qquad\qquad \mathsf{tr}\colon \mathsf{Term} \to \mathsf{Term}$$

by induction on the structure of our terms. For the constant $\underline{\mathsf{ham}}$, we put

$$\mathsf{o}(\underline{\mathsf{ham}}) \;=\; 1 \qquad\qquad \mathsf{tr}(\underline{\mathsf{ham}}) \;=\; \underline{2\times}(\underline{\mathsf{ham}}) \,\underline{\|}\, \underline{3\times}(\underline{\mathsf{ham}}) \,\underline{\|}\, \underline{5\times}(\underline{\mathsf{ham}})$$

following the defining SDE (11.29). For the constants $\underline{\sigma}$, we put

$$\mathsf{o}(\underline{\sigma}) \;=\; \sigma(0) \qquad\qquad \mathsf{tr}(\underline{\sigma}) \;=\; \underline{\sigma'}$$

following the definitions of the functions

$$\mathsf{i}_{st}\colon \mathbb{N}^\omega \to \mathbb{N} \qquad\qquad \mathsf{d}_{st}\colon \mathbb{N}^\omega \to \mathbb{N}^\omega$$

which are given (in Section 7.2, (7.5) and (7.6)) by

$$\begin{array}{ll} \mathsf{i}_{st}\colon \mathbb{N}^\omega \to \mathbb{N} \qquad\qquad & \mathsf{i}_{st}(\sigma) \;=\; \sigma(0) \\ \mathsf{d}_{st}\colon \mathbb{N}^\omega \to \mathbb{N}^\omega \qquad\qquad & \mathsf{d}_{st}(\sigma) \;=\; \sigma' \end{array}$$

For the operators of scalar multiplication, consider $t \in \mathsf{Term}$ and assume that $\mathsf{o}(t)$ and $\mathsf{tr}(t)$ have already been defined. Then we define

$$\mathsf{o}(\underline{k\times}(t)) \;=\; k \cdot \mathsf{o}(t) \qquad\qquad \mathsf{o}(\underline{k\times}(t)) \;=\; \underline{k\times}(\mathsf{tr}(t)) \qquad\qquad (k = 2, 3, 5)$$

following the defining SDE (11.30). For the ordered merge, let $t_1, t_2 \in \mathsf{Term}$ and assume that $\mathsf{o}(t_1)$, $\mathsf{tr}(t_1)$, $\mathsf{o}(t_2)$ and $\mathsf{tr}(t_2)$ have already been defined. We define

$$\mathsf{o}(t_1 \,\underline{\|}\, t_2) \;=\; \begin{cases} \mathsf{o}(t_1) & \text{if } \mathsf{o}(t_1) < \mathsf{o}(t_2) \\ \mathsf{o}(t_1) & \text{if } \mathsf{o}(t_1) = \mathsf{o}(t_2) \\ \mathsf{o}(t_2) & \text{if } \mathsf{o}(t_1) > \mathsf{o}(t_2) \end{cases} \quad \mathsf{tr}(t_1 \,\underline{\|}\, t_2) \;=\; \begin{cases} \mathsf{tr}(t_1) \,\underline{\|}\, t_2 & \text{if } \mathsf{o}(t_1) < \mathsf{o}(t_2) \\ \mathsf{tr}(t_1) \,\underline{\|}\, \mathsf{tr}(t_2) & \text{if } \mathsf{o}(t_1) = \mathsf{o}(t_2) \\ t_1 \,\underline{\|}\, \mathsf{tr}(t_2) & \text{if } \mathsf{o}(t_1) > \mathsf{o}(t_2) \end{cases}$$

where we followed SDE (11.31).

(3) We define a candidate solution of our system of SDEs with the help of

$$[\![-]\!] \colon (\mathsf{Term}, \langle \mathsf{o}, \mathsf{tr} \rangle) \to (\mathbb{N}^\omega,\ \langle \mathsf{i}_{st}, \mathsf{d}_{st} \rangle)$$

which is the final homomorphism already mentioned above, by

$$\mathsf{ham} \ = \ [\![\underline{\mathsf{ham}}]\!] \tag{11.32}$$

and, for all $\sigma, \tau \in \mathbb{N}^\omega$, by

$$2 \times \sigma \ = \ [\![\underline{2\times}(\underline{\sigma})]\!] \quad 3 \times \sigma \ = \ [\![\underline{3\times}(\underline{\sigma})]\!] \quad 5 \times \sigma \ = \ [\![\underline{5\times}(\underline{\sigma})]\!] \quad \sigma \parallel \tau \ = \ [\![\underline{\sigma} \ \underline{\parallel} \ \underline{\tau}]\!] \tag{11.33}$$

(4) Finally we show that the syntactic method introduced above is correct.

Theorem 239 (correctness). *The stream and stream functions defined in equations* (11.32) *and* (11.33) *satisfy the original system of defining SDEs* (11.29), (11.30)*, and* (11.31)*, and they are the only ones.*

Proof: We show that the stream ham satisfies the SDE (11.29), leaving the other cases to the reader. For the initial value, we have

$$\mathsf{ham}(0) \ = \ [\![\underline{\mathsf{ham}}]\!](0) \ = \ \mathsf{o}(\underline{\mathsf{ham}}) \ = \ 1$$

where the second equality follows from the fact that $[\![-]\!]$ is a homomorphism. Furthermore,

$$\begin{aligned}
\mathsf{ham}' \ &= \ [\![\underline{\mathsf{ham}}]\!]' \\
&= \ [\![\mathsf{tr}(\underline{\mathsf{ham}})]\!] && ([\![-]\!] \text{ is a homomorphism}) \\
&= \ [\![\underline{2\times}(\underline{\mathsf{ham}}) \ \underline{\parallel} \ \underline{3\times}(\underline{\mathsf{ham}}) \ \underline{\parallel} \ \underline{5\times}(\underline{\mathsf{ham}})]\!] && (\text{definition } \mathsf{tr}) \\
&= \ [\![\underline{2\times}(\underline{\mathsf{ham}})]\!] \ \parallel \ [\![\underline{3\times}(\underline{\mathsf{ham}})]\!] \ \parallel \ [\![\underline{5\times}(\underline{\mathsf{ham}})]\!] && (\text{Proposition 244 below}) \\
&= \ 2 \times [\![\underline{\mathsf{ham}}]\!] \ \parallel \ 3 \times [\![\underline{\mathsf{ham}}]\!] \ \parallel \ 5 \times [\![\underline{\mathsf{ham}}]\!] && (\text{Proposition 244 below}) \\
&= \ 2 \times \mathsf{ham} \ \parallel \ 3 \times \mathsf{ham} \ \parallel \ 5 \times \mathsf{ham} && (\text{definition of } \mathsf{ham}, (11.32))
\end{aligned}$$

as desired. We leave the proof of the uniqueness of the solution as Exercise 240. □

Exercise 240 (uniqueness of solution). Use coinduction to prove that any two solutions of the SDEs (11.29), (11.30), and (11.31) – each consisting of a constant $\sigma \in \mathbb{N}^\omega$; functions of type $f_k \colon \mathbb{N}^\omega \to \mathbb{N}^\omega$ (for $k = 2, 3, 5$); and a function of type $m \colon \mathbb{N}^\omega \times \mathbb{N}^\omega \to \mathbb{N}^\omega$ – are equal. □

The proof of Theorem 239 uses the fact that $[\![-]\!]$ is *compositional*, which we shall prove as Proposition 244 below. We shall use two lemma's. Recall from Definition 64 that

$$\sim \ \subseteq \ \mathsf{Term} \times \mathsf{Term}$$

denotes the greatest bisimulation relation on Term, and that $\sim$ is equal to the union of all bisimulations.

Lemma 241. *The function* $\underline{(-)}\colon \mathbb{N}^\omega \to \mathsf{Term}$ *that maps* σ *to* $\underline{\sigma}$ *is a homomorphism of stream systems. As a consequence,*

$$t \;\sim\; \underline{[\![\, t \,]\!]}$$

for all $t \in \mathsf{Term}$*.*

Proof: By Proposition 69, the function $\underline{(-)}$ is a homomorphism if and only if

$$\{\ (\sigma,\ \underline{\sigma}) \ \mid\ \sigma \in \mathbb{N}^\omega\ \}$$

is a bisimulation relation. Since the latter is immediate by the definition of $\underline{(-)}$, this proves the first statement of the lemma. Since also $[\![-]\!]\colon \mathsf{Term} \to \mathbb{N}^\omega$ is a homomorphism and since the composition of two homomorphisms is again a homomorphism, the second statement follows, again by Proposition 69. □

Exercise 242. Prove that $[\![\, \underline{\sigma} \,]\!] = \sigma$, for all $\sigma \in \mathbb{N}^\omega$. □

Lemma 243 (bisimilarity is a congruence). *For all* $s, s_1, s_2, t, t_1, t_2 \in \mathsf{Term}$*,* $k = 2, 3, 5$*,*

$$\textit{if}\ \ s \sim t\ \ \textit{then}\ \ \underline{k\times}(s)\ \sim\ \underline{k\times}(t) \tag{11.34}$$

$$\textit{if}\ \ s_1 \sim t_1\ \ \textit{and}\ \ s_2 \sim t_2\ \ \textit{then}\ \ s_1 \,\underline{\|}\, t_1\ \sim\ s_2 \,\underline{\|}\, t_2 \tag{11.35}$$

We say that bisimilarity is a congruence *with respect to the operators* $\underline{k\times}$ *and* $\underline{\|}$*.*

Proof: Let

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

be a bisimulation relation and let

$$\overline{R} \ \subseteq\ \mathsf{Term} \times \mathsf{Term}$$

be defined as the smallest set such that

$$R \ \subseteq\ \overline{R} \tag{\textbf{i}}$$

$$\text{if } (s,t) \in \overline{R} \ \text{ then } \ (\underline{k\times}(s),\ \underline{k\times}(t)) \ \in\ \overline{R} \tag{\textbf{ii}}$$

$$\text{if } (s_1,t_1),(s_2,t_2) \in \overline{R} \ \text{ then } \ (s_1 \,\underline{\|}\, t_1\ , s_2 \,\underline{\|}\, t_2) \ \in\ \overline{R} \tag{\textbf{iii}}$$

For the proof of (11.34) and (11.35), it suffices to show that $\overline{R}$ is again a bisimulation. We have to show, for all $(s,t) \in \overline{R}$, that

$$\text{(a)} \quad \mathsf{o}(s)\ =\ \mathsf{o}(t) \qquad\qquad \text{(b)} \quad (\mathsf{tr}(s),\, \mathsf{tr}(t))\ \in\ \overline{R}$$

We will prove this by induction on the definition of $\overline{R}$:

(i) For $(s, t) \in R$, (a) and (b) follow from the assumption that R is a bisimulation.

(ii) Next let $(s, t) \in \overline{R}$ and assume that we have already proved (a) and (b) for (s, t). We show that also $(\underline{k\times}(s),\ \underline{k\times}(t))$ satisfies (a) and (b). For (a), we have

$$\begin{aligned} \mathsf{o}(\underline{k\times}(s)) &= \mathsf{o}(s) && \text{(definition } \mathsf{o}\text{)} \\ &= \mathsf{o}(t) && \text{(by assumption (a) for } (s,t)\text{)} \\ &= \mathsf{o}(\underline{k\times}(t)) && \text{(definition } \mathsf{o}\text{)} \end{aligned}$$

For (b), we note that, by assumption (b) for (s, t),

$$(\mathsf{tr}(s), \mathsf{tr}(t)) \ \in \ \overline{R}$$

By clause **(ii)** in the definition of $\overline{R}$, we have

$$(\underline{k\times}(\mathsf{tr}(s)),\ \underline{k\times}(\mathsf{tr}(t))) \ \in \ \overline{R}$$

It now follows that

$$\begin{aligned} \mathsf{tr}(\underline{k\times}(s)) &= \underline{k\times}(\mathsf{tr}(s)) && \text{(definition } \mathsf{tr}\text{)} \\ &\ \overline{R}\ \ \underline{k\times}(\mathsf{tr}(s)) && \\ &= \mathsf{tr}(\underline{k\times}(t)) && \text{(definition } \mathsf{tr}\text{)} \end{aligned}$$

(where we have used the infix notation for the relation $\overline{R}$). This proves (a) and (b) for $(\underline{k\times}(s)\ ,\ \underline{k\times}(t))$.

(iii) The case of $(s_1 \underline{\|}\ t_1,\ s_2 \underline{\|}\ t_2)$ is similar to the previous one.

This concludes the proof that $\overline{R}$ is a bisimulation. □

Proposition 244 (compositionality of $[\![-]\!]$).

$$[\![\, \underline{k\times}(t)\,]\!] \ = \ k \times [\![\, t\,]\!] \qquad [\![t_1 \ \underline{\|}\ t_2]\!] \ = \ [\![t_1]\!] \ \| \ [\![t_2]\!] \qquad (t, t_1, t_2 \in \mathsf{Term},\ k = 2, 3, 5)$$

Proof: By Lemma 241, $t \sim \underline{[\![\, t\,]\!]}$, which implies, by Lemma 243, that

$$\underline{k\times}(t) \ \sim \ \underline{k\times}(\underline{[\![\, t\,]\!]})$$

Since $[\![-]\!]$ is a homomorphism of stream systems, we have, for all $s, t \in \mathsf{Term}$,

$$s \ \sim \ t \quad \Longleftrightarrow \quad [\![\, s\,]\!] \ = \ [\![\, t\,]\!]$$

by Theorem 84. As a consequence,

$$\begin{aligned} [\![\, \underline{k\times}(t)\,]\!] &= [\![\, \underline{k\times}(\underline{[\![\, t\,]\!]})\,]\!] && \\ &= k \times [\![\, t\,]\!] && \text{((11.33), definition of the function } k\times\text{)} \end{aligned}$$

which proves the first equality of the proposition. The second is proved similarly. □

This concludes our proof of the correctness of the syntactic method, Theorem 239.

Exercise 245 (algebras and coalgebras in the syntactic method). We define a functor $F\colon \mathsf{Set} \to \mathsf{Set}$ by

$$F(S) \;=\; 1 \,+\, S \,+\, S \,+\, S \,+\, S^2$$

for every set S. Complete this definition by describing how F acts on functions. An *F-algebra* (discussed in Chapter 3) is a pair (S, α) consisting of a set S and a function $\alpha\colon F(S) \to S$. A *homomorphism* of F-algebras (S, α) and (T, β) is a function $f\colon S \to T$ such that $f \circ \alpha = \beta \circ F(f)$.

Show that, equivalently, an F-algebra is a pair $(S, [s, g, h, i, j])$, consisting of a set S and a 5-tuple consisting of an element $s \in S$, three functions $g, h, i\colon S \to S$, and a function $j\colon S^2 \to S$. Prove that the function $[\![-]\!]\colon \mathsf{Term} \to \mathbb{N}^\omega$ is both a homomorphism of F-algebras and a homomorphism of G-coalgebras:

$$\begin{tikzcd}
1 + \mathsf{Term} + \mathsf{Term} + \mathsf{Term} + \mathsf{Term}^2 \arrow[rr] \arrow[d, "{[\,\underline{\mathsf{ham}},\ \underline{2\times},\ \underline{3\times},\ \underline{5\times},\ \underline{\parallel}\,]}"'] & & 1 + \mathbb{N}^\omega + \mathbb{N}^\omega + \mathbb{N}^\omega + (\mathbb{N}^\omega)^2 \arrow[d, "{[\,\mathsf{ham}\ ,2\times,\ 3\times,\ 5\times,\ \parallel\,]}"'] \\
\mathsf{Term} \arrow[rr, "{[\![-]\!]}"'] \arrow[d, "{\langle \mathsf{o}, \mathsf{tr} \rangle}"] & & \mathbb{N}^\omega \arrow[d, "{\langle \mathsf{i}_{st}, \mathsf{d}_{st} \rangle}"] \\
\mathbb{N} \times \mathsf{Term} \arrow[rr] & & \mathbb{N} \times \mathbb{N}^\omega
\end{tikzcd}$$

where we define the functor $G\colon \mathsf{Set} \to \mathsf{Set}$ by $G(S) = \mathbb{N} \times S$. $\square$

Exercise 246. We define the functions $f, g\colon \mathbb{N}^\omega \times \mathbb{N}^\omega \to \mathbb{N}^\omega$ by

$$\begin{aligned}
f(\sigma,\tau)(0) &= \sigma(0) + \tau(0) & \qquad f(\sigma,\tau)' &= f(\sigma',\tau') \\
g(\sigma,\tau)(0) &= \sigma(0) \cdot \tau(0) & \qquad g(\sigma,\tau)' &= f(\, g(\sigma',\tau),\, g(\sigma,\tau')\,)
\end{aligned}$$

Use the syntactic method to prove that the above system of SDEs has a unique solution. $\square$

11.3 GSOS: a general format for stream differential equations

Next we will explain that the syntactic solution method of the previous section works, far more generally, for all systems of stream differential equations in the so-called *GSOS* format. This acronym is often read as *guarded structural operational semantics* and stems from the theory of concurrency (see Section 11.5).

Terms over a signature: Because we want to be precise about the syntax of the GSOS format, we begin with a bit of background on term algebras. A *signature* Σ is a collection of operation symbols f, together with a function $\mathsf{ar}\colon \Sigma \to \mathbb{N}$ that assigns to every f its *arity* $\mathsf{ar}(f)$. If $\mathsf{ar}(f) = 0$ then we call f a constant.

For a given set X, we define the set $T_\Sigma(X)$ of all *terms* over the signature Σ with *generators* or *variables* in X as the smallest set that contains X and that is closed under the operations in Σ:

$$T_\Sigma(X) \ \ni \quad t ::= \ x \ \mid \ f(t_1, \ldots, t_k) \qquad\qquad (x \in X,\ f \in \Sigma,\ \mathsf{ar}(f) = k)$$

A Σ-*algebra* (S, α) consists of a set S and a collection of functions

$$\alpha \ = \ \{ \ f_\alpha\colon S^k \to S \ \mid \ f \in \Sigma, \ \mathsf{ar}(f) = k \ \}$$

containing for every operation symbol f an *interpretation* f_α. A Σ-*homomorphism*

$$h\colon (S, \alpha) \to (T, \beta)$$

is a function $h\colon S \to T$ that respects the algebra structure:

$$f_\beta(\, h(s_1), \ldots, h(s_k)\,) \ = \ h(f_\alpha(s_1, \ldots, s_k)) \qquad (f \in \Sigma,\ \mathsf{ar}(f) = k,\ s_1, \ldots, s_k \in S)$$

Exercise 247. Define for every signature Σ a functor $F_\Sigma\colon \mathsf{Set} \to \mathsf{Set}$ such that the notions of Σ-algebra and Σ-homomorphism above coincide with those of F_Σ-algebras, as defined in Section 3.1. $\square$

The set of terms $T_\Sigma(X)$ with generators in X carries a Σ-algebra structure $(T_\Sigma(X), \gamma)$, where for every $f \in \Sigma$ the function $f_\gamma\colon T_\Sigma(X)^k \to T_\Sigma(X)$ is simply given by the syntactic construction of terms:

$$f_\gamma(t_1, \ldots, t_k) \ = \ f(t_1, \ldots, t_k)$$

The algebra $(T_\Sigma(X), \gamma)$ is called the *free* Σ-*algebra over the set* X because of the following universal property: for all Σ-algebras (S, α) and for every function $g\colon X \to S$ embedding the generators in X into the algebra S, there exists a unique Σ-homomorphism

$$g^*\colon (T_\Sigma(X), \gamma) \to (S, \alpha)$$

extending g, that is, such that $g^*(x) = g(x)$, for all $x \in X$. We can define g^* by induction as follows:

$$\begin{aligned} g^*(x) \ &= \ g(x) && (\text{all } x \in X) \\ g^*(\, f(t_1, \ldots, t_k)\,) \ &= \ f_\alpha(\, g^*(t_1), \ldots, g^*(t_k)\,) && (\text{all } f \in \Sigma,\ \mathsf{ar}(f) = k) \end{aligned}$$

Exercise 248. Use Exercise 247 to show that the notion of free Σ-algebra above is an instance of the following categorical notion. Consider a functor $F\colon \mathsf{Set} \to \mathsf{Set}$, a set X, an F-algebra (T, β), and an injective function $i\colon X \to T$. We call (T, β) *free on* X if for every F-algebra (S, α) and every function $g\colon X \to S$ there exists a unique F-homomorphism $g^*\colon (T, \beta) \to (S, \alpha)$ such that

$$\begin{tikzcd}
F(T) \arrow[r, dashed, "F(g^*)"'] \arrow[d, "\beta"'] & F(S) \arrow[d, "\alpha"] \\
T \arrow[r, dashed, "\exists!\, g^*"] & S \\
X \arrow[u, "i"] \arrow[ur, bend right, "\forall\, g"'] &
\end{tikzcd}$$

that is, $g^* \circ i = g$. □

Exercise 249. Prove that $(T_\Sigma(\emptyset), \gamma)$ is initial in the category of all Σ-algebras and Σ homomorphisms. □

Finally, a *substitution* is a Σ-homomorphism

$$s\colon T_\Sigma(X) \to T_\Sigma(Y)$$

Note that for

$$X \;=\; \{x_1, \ldots, x_k\} \qquad\qquad \{t_1, \ldots t_k\} \;\subseteq\; T_\Sigma(Y)$$

every function

$$g\colon X \to T_\Sigma(Y) \qquad\qquad g(x_i) = t_i$$

induces a substitution

$$g^*\colon T_\Sigma(X) \to T_\Sigma(Y) \tag{11.36}$$

using the fact that $(T_\Sigma(\mathsf{X}), \gamma)$ is free. We shall use the following notation:

$$g^*(t) \;=\; t\,[t_i/x_i]_{i \leqslant k} \qquad\qquad (t \in T_\Sigma(X))$$

Introducing the GSOS format: Next we introduce a general syntactic format for SDEs. For the rest of this section, let Σ be an arbitrary but fixed signature. We shall consider streams over an arbitrary set A.

Definition 250 (GSOS format). A system of stream differential equations in *GSOS format* for the signature Σ contains for every $f \in \Sigma$ an SDE of the form:

$$f(\sigma_1, \ldots, \sigma_k)(0) \;=\; o_f(\sigma_1(0), \ldots, \sigma_k(0)) \tag{11.37}$$

$$f(\sigma_1, \ldots, \sigma_k)' \;=\; d_f(\sigma_1(0), \ldots, \sigma_k(0))[\,\sigma_i/x_i,\; \sigma_i'/y_i\,]_{i \leqslant k} \tag{11.38}$$

where $k = \mathsf{ar}(f)$ and where the initial value and the derivative are given by functions

$$o_f \colon A^k \to A \qquad\qquad d_f \colon A^k \to T_\Sigma(\,\{x_1, \ldots, x_k, y_1, \ldots, y_k\})$$ □

We note that

$$d_f(\sigma_1(0), \ldots, \sigma_k(0))[\,\sigma_i/x_i,\; \sigma_i'/y_i\,]_{i \leqslant k} \;\in\; T_\Sigma(\,A^\omega\,)$$

since the substitution $[\,\sigma_i/x_i,\; \sigma_i'/y_i\,]_{i \leqslant k}$ is a function of type

$$[\,\sigma_i/x_i,\; \sigma_i'/y_i\,]_{i \leqslant k} : T_\Sigma(\,\{x_1, \ldots, x_k, y_1, \ldots, y_k\}\,) \to T_\Sigma(\,A^\omega\,)$$

By Definition 250, the GSOS format allows functions $f \colon (A^\omega)^k \to A^\omega$ to be specified by SDEs in which, according to (11.37), the initial value may depend on the initial value of the arguments $\sigma_1, \ldots, \sigma_k \in A^\omega$, and in which the derivative, according to (11.38), may depend on the initial values of the arguments, the arguments themselves, and the derivatives of the arguments.

Example 251. The system of SDEs (11.29), (11.30), and (11.31) defining the Hamming numbers can be seen to be in GSOS format, by taking

$$\Sigma \;=\; \{\underline{\mathsf{ham}},\, \underline{2\times},\, \underline{3\times},\, \underline{5\times},\, \underline{\|}\,\}$$

with arities 0, 1, 1, 1, and 2, respectively; and by defining

$$\begin{aligned}
o_{\underline{\mathsf{ham}}} &= 1 & d_{\underline{\mathsf{ham}}} &= \underline{2\times}(\underline{\mathsf{ham}}) \;\underline{\|}\; \underline{3\times}(\underline{\mathsf{ham}}) \;\underline{\|}\; \underline{5\times}(\underline{\mathsf{ham}}) \\
o_{\underline{k\times}}(n) &= k \cdot n & d_{\underline{k\times}}(n) &= \underline{k\times}(y_1) \\
o_{\underline{\|}}(n,m) &= \begin{cases} n & \text{if } n < m \\ n & \text{if } n = m \\ m & \text{if } n > m \end{cases} & d_{\underline{\|}}(n,m) &= \begin{cases} y_1 \;\underline{\|}\; x_2 & \text{if } n < m \\ y_1 \;\underline{\|}\; y_2 & \text{if } n = m \\ x_1 \;\underline{\|}\; y_2 & \text{if } n > m \end{cases}
\end{aligned}$$

for $k = 2, 3, 5$ and for all $n, m \geqslant 0$. □

Defining the notion of solution: A *solution* of a system of SDEs in GSOS format, as in Definition 250, is now defined to be a Σ-algebra structure

$$(A^\omega, \alpha) \qquad\qquad \alpha \;=\; \{\; f_\alpha \colon (A^\omega)^k \to A^\omega \;\mid\; f \in \Sigma,\; \mathsf{ar}(f) = k \;\}$$

satisfying the stream differential equations of (11.37) and (11.38): for all $f \in \Sigma$,

$$\begin{aligned} f_\alpha(\sigma_1, \ldots, \sigma_k)(0) &= o_f(\sigma_1(0), \ldots, \sigma_k(0)) \\ f_\alpha(\sigma_1, \ldots, \sigma_k)' &= id^*(\, d_f(\sigma_1(0), \ldots, \sigma_k(0))[\, \sigma_i/x_i,\ \sigma_i'/y_i\,]_{i \leqslant k}\,) \end{aligned}$$

where

$$id^* \colon (\ T_\Sigma(\, A^\omega\,),\ \gamma\ ) \to (A^\omega, \alpha)$$

is the unique Σ-algebra homomorphism that is induced, by the fact that ($T_\Sigma(\, A^\omega\,)$ is free, by the identity function $id\colon A^\omega \to A^\omega$.

A syntactic stream system: Similar to the example of the Hamming numbers, we shall next introduce a set of terms, turn it into a stream system, and then use finality to construct a candidate solution for our system of SDEs. Let

$$\underline{A^\omega} \ = \ \{\, \underline{\sigma} \mid \sigma \in A^\omega\, \}$$

The set $\underline{A^\omega}$ contains for every stream σ a constant symbol $\underline{\sigma}$. Every system of SDEs in GSOS format now induces a stream system

$$(\, T_\Sigma(\, \underline{A^\omega}\,), \langle \mathsf{o}, \mathsf{tr} \rangle)$$

where the functions $\mathsf{o}\colon T_\Sigma(\, \underline{A^\omega}\,) \to A$ and $\mathsf{tr}\colon T_\Sigma(\, \underline{A^\omega}\,) \to T_\Sigma(\, \underline{A^\omega}\,)$ are inductively defined, for all $\sigma \in A^\omega$, by

$$\mathsf{o}(\underline{\sigma}) \ = \ \sigma(0) \qquad\qquad \mathsf{tr}(\underline{\sigma}) \ = \ \underline{\sigma'}$$

and, for all $f \in \Sigma$ of arity $\mathsf{ar}(f) = k$ and all terms $t_1, \ldots, t_k \in T_\Sigma(\, \underline{A^\omega}\,)$, by

$$\begin{aligned} \mathsf{o}(\, f(t_1, \ldots, t_k)\,) &= o_f(\, \mathsf{o}(t_1), \ldots, \mathsf{o}(t_k)\,) \\ \mathsf{tr}(\, f(t_1, \ldots, t_k)\,) &= d_f(\, \mathsf{o}(t_1), \ldots, \mathsf{o}(t_k)\,)\, [\, t_i/x_i,\ \mathsf{tr}(t_i)/y_i\,]_{i \leqslant k} \end{aligned}$$

By finality of $(A^\omega,\ \langle \mathsf{i}_{st}, \mathsf{d}_{st} \rangle)$, there exists a unique homomorphism of stream systems

$$[\![-]\!] \colon (\, T_\Sigma(\, \underline{A^\omega}\,), \langle \mathsf{o}, \mathsf{tr} \rangle) \to (A^\omega,\ \langle \mathsf{i}_{st}, \mathsf{d}_{st} \rangle)$$

We use it to define a Σ-algebra structure (A^ω, α) on the set of streams, which will be the candidate solution for our system of SDEs. For every $f \in \Sigma$ of arity $\mathsf{ar}(f) = k$, and for every $\sigma_1, \ldots, \sigma_k \in A^\omega$, we define

$$f_\alpha(\sigma_1, \ldots, \sigma_k) \ = \ [\![\, f(\, \underline{\sigma_1}, \ldots, \underline{\sigma_1}\,)\,]\!] \tag{11.39}$$

Theorem 252 (correctness of the syntactic method). *The Σ-algebra (A^ω, α) defined by Equation* (11.39) *is the unique solution of the system of SDEs in Definition 250.* □

This theorem can be proved in essentially the same way as Theorem 239 for the Hamming numbers, using again the fact that bisimilarity is a congruence, as in Lemma 243, and the fact that the final semantics is compositional, as in Proposition 244. The proof is omitted, details can be found in [HKR17].

Exercise 253. Show that all the SDEs in Section 11.1 – apart from (11.27), which defines the function even – are in GSOS format and hence have unique solutions. Show that also the system of SDEs for the Hamming numbers, consisting of the equations in (11.29), (11.30), and (11.31), is in GSOS format. □

Any signature Σ with a corresponding set of operators on A^ω that are defined by a system of SDEs in GSOS format, comes equipped with the notions of *bisimulation up-to-*Σ and *coinduction up-to-*Σ. These notions generalise earlier definitions of *up-to* such as bisimulation and coinduction up-to-sum, for streams, in Definition 105 and Proposition 106. They generalise also the definitions of bisimulation and coinduction up-to-equivalence in Definition 138 and Proposition 139, and bisimulation and coinduction up-to-congruence, in Definition 171 and Proposition 172, for languages (when restricted to a one-letter alphabet).

Definition 254 (bisimulation up-to-Σ). Consider a signature Σ and a corresponding collection of operations on A^ω defined by a system of SDEs in GSOS format. A relation

$$R \;\subseteq\; A^\omega \times A^\omega$$

is called a *bisimulation up-to-*Σ if

$$(i)\;\; \sigma(0) \;=\; \tau(0) \quad \text{and} \quad (ii)\;\; (\sigma', \tau') \in R^\Sigma \qquad\qquad ((\sigma,\tau) \in R)$$

where

$$R^\Sigma \;\subseteq\; A^\omega \times A^\omega$$

is the smallest relation such that

1. $R \subseteq R^\Sigma$
2. R^Σ is an equivalence relation
3. R^Σ is closed under the operations in Σ

(For instance, if $+ \in \Sigma$ and $(\alpha_1, \beta_1), (\alpha_2, \beta_2) \in R^\Sigma$ then $(\alpha_1 + \alpha_2,\, \beta_1 + \beta_2) \in R^\Sigma$.) □

Proposition 255 (coinduction up-to-Σ). If $R \subseteq A^\omega \times A^\omega$ is a bisimulation up-to-Σ and $(\sigma, \tau) \in R$, then $\sigma = \tau$.

Proof: If R is a bisimulation up-to-Σ, then one shows by stuctural induction on its definition that R^{Σ} is a bisimulation relation. Since $(\sigma, \tau) \in R \subseteq R^{\Sigma}$, coinduction Theorem 96 implies $\sigma = \tau$. □

We shall see many examples of proofs by coinduction up-to-Σ in Chapter 12. The following exercise gives a characterisation of precisely which stream functions are definable by SDEs in GSOS format.

Exercise 256 (GSOS and causal stream functions). For streams $\sigma, \tau \in A^{\omega}$ and for $n \geqslant 1$, we define

$$\sigma \equiv_n \tau \qquad \Longleftrightarrow \qquad \sigma(0) = \tau(0)\,,\ \ldots\ ,\ \sigma(n-1) = \tau(n-1)$$

We call a stream function $f \colon (A^{\omega})^k \to A^{\omega}$ *causal* if

$$\text{if} \quad \sigma_1 \equiv_n \tau_1\,,\ \ldots\ ,\ \sigma_k \equiv_n \tau_k \quad \text{then} \quad f(\sigma_1, \ldots, \sigma_k)(n) = f(\tau_1, \ldots, \tau_k)(n)$$

for all $\sigma_1, \ldots, \sigma_k, \tau_1, \ldots, \tau_k \in A^{\omega}$, and for all $n \geqslant 0$ (see also Definition 343). Then:

$$f \text{ can be defined by a system of SDEs in GSOS format} \qquad \Longleftrightarrow \qquad f \text{ is causal}$$

Reconstruct a proof of this fact consulting [HKR17]. Which of the functions even, odd, and zip (defined in Example 86) can be defined by a SDE in GSOS format? □

Example 257. Recalling the definitions of elementwise addition (11.10):

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

and of scalar multiplication (11.30):

$$(k \times \sigma)(0) \ = \ k \cdot \sigma(0) \qquad\qquad (k \times \sigma)' \ = \ k \times \sigma'$$

we consider the following SDEs:

$$\begin{aligned}
\mathsf{ones}(0) &= 1 & \mathsf{ones}' &= \mathsf{ones} \\
\mathsf{nat}(0) &= 1 & \mathsf{nat}' &= \mathsf{ones} + \mathsf{nat} \\
\mathsf{squares}(0) &= 1 & \mathsf{squares}' &= \mathsf{ones} + (2 \times \mathsf{nat}) + \mathsf{squares} \\
\mathsf{cubes}(0) &= 1 & \mathsf{cubes}' &= \mathsf{ones} + (3 \times \mathsf{nat}) + (3 \times \mathsf{squares}) + \mathsf{cubes}
\end{aligned}$$

These equations are in GSOS format and define the following streams:

$$\begin{aligned}
\mathsf{ones} &= (1, 1, 1, \ldots) & \mathsf{nat} &= (1, 2, 3, \ldots) \\
\mathsf{squares} &= (1^2, 2^2, 3^2, \ldots) & \mathsf{cubes} &= (1^3, 2^3, 3^3, \ldots)
\end{aligned}$$

For a formal proof of $\mathsf{squares} = (1^2, 2^2, 3^2, \ldots)$, we observe that

$$\mathsf{squares}(1) \;=\; (\mathsf{squares}')(0) \;=\; (\mathsf{ones} + (2 \times \mathsf{nat}) + \mathsf{squares})(0) \;=\; 2^2$$

For $\mathsf{squares}(2)$, we compute:

$$\begin{aligned}
\mathsf{squares}^{(2)} &= (\mathsf{ones} + (2 \times \mathsf{nat}) + \mathsf{squares})' \\
&= \mathsf{ones}' + (2 \times \mathsf{nat})' + \mathsf{squares}' \\
&= \mathsf{ones} + (2 \times \mathsf{nat}') + \mathsf{ones} + (2 \times \mathsf{nat}) + \mathsf{squares} \\
&= \mathsf{ones} + (2 \times (\mathsf{ones} + \mathsf{nat})) + \mathsf{ones} + (2 \times \mathsf{nat}) + \mathsf{squares} \\
&= \mathsf{ones} + (2 \times \mathsf{ones}) + (2 \times \mathsf{nat}) + \mathsf{ones} + (2 \times \mathsf{nat}) + \mathsf{squares} \\
&= (4 \times \mathsf{ones}) + (4 \times \mathsf{nat}) + \mathsf{squares}
\end{aligned}$$

As a consequence, $\mathsf{squares}(2) = \mathsf{squares}^{(2)}(0) = 4 + 4 + 1 = 3^2$. One proves that

$$\mathsf{squares}^{(n)} \;=\; (n^2 \times \mathsf{ones}) + (2n \times \mathsf{nat}) + \mathsf{squares}$$

for any $n \geqslant 2$, by induction on n. It follows that $\mathsf{squares}(n) = \mathsf{squares}^{(n)}(0) = (n+1)^2$. □

Exercise 258. Prove, similarly, that $\mathsf{cubes} = (1^3, 2^3, 3^3, \ldots)$. Use the equality

$$(1^n, 2^n, 3^n, \ldots)' \;=\; ((1+1)^n, (1+2)^n, (1+3)^n, \ldots)$$

to find an SDE for the stream $(1^n, 2^n, 3^n, \ldots)$, for any $n \geqslant 2$. □

Many more examples of SDEs in GSOS format can be found in Chapter 12, on stream calculus.

11.4 Behavioural differential equations

Behavioural differential equations (BDEs) are for *languages* what SDEs are for streams. Here is a very brief sketch. Recall from Section 8.3 the set of languages over a set A:

$$\mathcal{P}(A^*) \;=\; \{\lambda \mid \lambda \subseteq A^*\}$$

and from Definition 151, the notions of initial value and a-derivative of a language:

$$\lambda(\varepsilon) \;=\; \begin{cases} 1 & \text{if } \; \varepsilon \in \lambda \\ 0 & \text{if } \; \varepsilon \notin \lambda \end{cases} \qquad \lambda_a \;=\; \{w \in A^* \mid aw \in \lambda\} \qquad (\lambda \in \mathcal{P}(A^*),\, a \in A)$$

where ε is the empty word. Note that instead of just one derivative, we now have a derivative for every $a \in A$. Here is an elementary example of a BDE (with $A = \{a, b\}$):

$$\lambda(\varepsilon) \;=\; 1 \qquad\qquad \lambda_a \;=\; b \times \lambda \qquad\qquad \lambda_b = 0$$

This BDE defines what the language λ is by specifying its behaviour, which consists of its initial value $\lambda(\varepsilon)$ and its input derivatives λ_a and λ_b. The unique solution of the BDE is $\lambda = (ab)^*$, which is the only language satisfying the above identities.

Exercise 259. Consider the following system of two BDEs:

$$\begin{array}{lll} \lambda(\varepsilon) = 1 & \lambda_a = \mu & \lambda_b = 0 \\ \mu(\varepsilon) = 0 & \mu_a = 0 & \mu_b = \lambda \end{array}$$

What is its solution? □

Similarly, we can define *functions* on languages. For instance,

$$f(\lambda)(\varepsilon) = 1 \qquad f(\lambda)_a = \lambda_a \times f(\lambda)$$

is a system of BDEs, containing the above two identities for every $\lambda \in \mathcal{P}(A^*)$ and $a \in A$. It defines the function

$$f\colon \mathcal{P}(A^*) \to \mathcal{P}(A^*) \qquad f(\lambda) = \lambda^*$$

Exercise 260. What is the solution of the following system of BDEs:

$$f(\kappa,\lambda)(\varepsilon) = \min\{\kappa(\varepsilon),\, \lambda(\varepsilon)\} \qquad f(\kappa,\lambda)_a = f(\kappa_a,\lambda) + f(\kappa(\varepsilon),\lambda_a)$$

(where we interpret the value of $\kappa(\varepsilon)$, which is 0 or 1, as the corresponding *language*)? □

In Section 11.3, we proved that any system of *stream* differential equations in GSOS format has a unique solution. This solution was constructed by coinduction, using the fact that the set of streams is a final stream system and applying what was called the syntactic method. All of this can be readily adapted for *languages*: a GSOS format can be defined for systems of BDEs, and a corresponding syntactic solution method can be used to construct unique solutions by coinduction, this time exploiting the fact that the set $\mathcal{P}(A^*)$ of all languages is a final automaton. More generally still, all of this applies as well to *weighted languages*, also known as formal power series, which are a common generalisation of both streams and languages. Weighted languages were introduced in Exercise 354, where it was observed that they form a final Moore automaton. Also for weighted languages, finality can again be taken as a basis for the solution of behavioural differential equations. None of this is worked out in detail here, but references will be mentioned below.

11.5 Discussion

The present chapter is largely based on the overview paper [HKR17]. Stream differential equations and behavioural differential equations were introduced in [Rut99a, Rut01] and [Rut00a, Rut03a], building on [Brz64], [Con71] and [PE98].

The GSOS format was first introduced in the framework of structural operational semantics [Plo81, BIM95, AFV01], which studies rule formats that guarantee compositional semantics. It was then generalised to *abstract GSOS*, in [TP97, Bar03, Bar04], using the categorical framework of bialgebras and distributive laws. The survey paper [Kli11] gives an excellent introduction to abstract GSOS. [HKR17, Section 9], which itself is mainly based on [Bar04, Kli11, LPW04], presents a brief account of the categorical underpinnings of the GSOS format and the syntactic solution method for streams. For more on GSOS and distributive laws, see [BHKR15].

All SDEs that we discussed in the present chapter are based on the standard final coalgebra structure on the set of streams, given by initial value and stream derivative. [HKR17, Section 7] discusses several alternative final coalgebra structures on the set of streams, each of which gives rise to non-standard types of differential equations; see also Exercises 307 and 314 in Chapter 12.

Chapter 12

A calculus of streams

We develop a *calculus* of streams $\sigma \in \mathbb{R}^\omega$ in close analogy to mathematical analysis, with as main ingredients the notions of *initial value* $\sigma(0)$ and *stream derivative* σ', which were introduced in Section 7.2. Definitions of streams and stream functions will be typically given by means of *stream differential equations*, introduced in Chapter 11. And as always, proofs will typically be given by coinduction or coinduction up-to, Proposition 255.

We will formulate stream calculus versions of classical notions and results from mathematics such as, for instance, differentiation and integration, the fundamental theorem of calculus, Taylor series, and the exponential function. The resulting stream calculus turns out to be conceptually simple yet surprisingly expressive.

12.1 Basic stream operations

We begin by observing that the set $\mathbb{R}$ of real numbers is a commutative ring:

$$(\mathbb{R},\, +,\, -,\, \times,\, 0,\, 1\,)$$

with the operations of sum, minus and multiplication, and with the constants 0 and 1. We shall next introduce corresponding operations and constants for *streams* of real numbers:

$$(\mathbb{R}^\omega,\, +,\, -,\, \times,\, [0],\, [1]\,)$$

with which the set $\mathbb{R}^\omega$ is a commutative ring as well, as we will prove in Proposition 265 below. Note that we are using the same function symbols for the operations on real numbers and the corresponding operations on *streams* of real numbers.

Definition 261 (sum, minus, convolution product). We define the operations of *sum*, *minus* and *convolution* product:

$$+\colon\, \mathbb{R}^\omega \times \mathbb{R}^\omega \to \mathbb{R}^\omega \qquad\qquad -\colon\, \mathbb{R}^\omega \to \mathbb{R}^\omega \qquad\qquad \times\colon\, \mathbb{R}^\omega \times \mathbb{R}^\omega \to \mathbb{R}^\omega$$

together with a function $[-]\colon \mathbb{R} \to \mathbb{R}^\omega$ that embeds the set $\mathbb{R}$ into the set $\mathbb{R}^\omega$, by means of the following system of stream differential equations:

$$\begin{array}{rclrcll}
[r](0) & = & r & [r]' & = & [0] & (r \in \mathbb{R}) \\
(\sigma + \tau)(0) & = & \sigma(0) + \tau(0) & (\sigma + \tau)' & = & \sigma' + \tau' & (12.1) \\
(-\sigma)(0) & = & -\sigma(0) & (-\sigma)' & = & -(\sigma') & (12.2) \\
(\sigma \times \tau)(0) & = & \sigma(0) \times \tau(0) & (\sigma \times \tau)' & = & (\sigma' \times \tau) + ([\sigma(0)] \times \tau') & (12.3)
\end{array}$$

Since the above system of SDEs is in GSOS format, the unique existence of these operators follows from Theorem 252. □

Note that (12.3) defines $(\sigma \times \tau)'$ in terms of (one of) the original arguments, their derivatives, and (one of) their initial values, thus using the full expressiveness that the GSOS format allows us. As usual, we define

$$\sigma^0 = [1] \qquad\qquad \sigma^{n+1} = \sigma \times \sigma^n$$

It is straightforward to see that

$$[0] = (0, 0, 0, \ldots) \qquad [1] = (1, 0, 0, 0, \ldots) \qquad [r] = (r, 0, 0, 0, \ldots) \qquad (r \in \mathbb{R})$$

The functions minus and sum on streams are elementwise extensions of the corresponding functions on real numbers: for all $n \geqslant 0$,

$$(-\sigma)(n) = -\sigma(n) \tag{12.4}$$

$$(\sigma + \tau) = \sigma(n) + \tau(n) \tag{12.5}$$

Traditionally, the convolution product of two streams is defined elementwise, by

$$(\sigma \times \tau)(n) = \sum_{k=0}^{n} \sigma(k) \cdot \tau(n-k) \tag{12.6}$$

It is a little exercise to show that both (12.6) and the SDE (12.3) define the same function. However, we prefer the definition by the SDE, since it will allow us to prove several of the properties of the convolution product by coinduction.

Exercise 262 (convolution product versus function product). One motivation for the definition of the convolution product of streams can be found in the world of real-valued functions, which is an important source of inspiration for stream calculus in general. Consider a function $f\colon \mathbb{R} \to \mathbb{R}$ and a stream $\sigma \in \mathbb{R}^\omega$ (of coefficients) such that

$$f(x) = \sum_{i=0}^{\infty} \sigma(i) \times x^i$$

assuming that this infinite sum exists, for instance, for real numbers x close to 0. Defining

$$\overline{f}(x) \;=\; \sum_{i=0}^{\infty} \sigma(i+1) \times x^i$$

and observing that $f(0) = \sigma(0)$, we have

$$f(x) \;=\; f(0) + (x \times \overline{f}(x))$$

Consider, similarly, a second function $g\colon \mathbb{R} \to \mathbb{R}$ with a stream τ of coefficients. Now show that the elementwise product of functions

$$(f \times g)(x) \;=\; f(x) \times g(x) \qquad\qquad (x \in \mathbb{R})$$

satisfies

$$(f \times g)(0) \;=\; f(0) \times g(0) \qquad\qquad (\overline{f \times g})(x) \;=\; (\overline{f}(x) \times g(x)) \,+\, (f(0) \times \overline{g}(x))$$

and compare this equality to the SDE (12.3) defining the convolution product of streams. Based on the analogy between functions and streams, one might have expected that

$$(\sigma \times \tau)' \;=\; (\sigma' \times \tau) + (\sigma \times \tau') \tag{12.7}$$

Observe that this equality is generally *not* valid. Apparently, stream differentiation does not quite correspond to function differentiation in analysis. Rather, it corresponds to the transformation of a function f into $\overline{f}$, an operation that is not usually present in analysis. We shall see later, in Section 12.4, that the operation of *shuffle* product *does* satisfy equality (12.7) above. □

Exercise 263 (scalar multiplication as convolution product). The following SDE defines scalar stream multiplication, for every $r \in \mathbb{R}$:

$$(r \times \sigma)(0) \;=\; r \cdot \sigma(0) \qquad\qquad (r \times \sigma)' \;=\; r \times \sigma' \tag{12.8}$$

(It generalises our earlier definition (11.30), which was for natural numbers.) Clearly, it satisfies

$$r \times \sigma \;=\; (r \cdot \sigma(0),\, r \cdot \sigma(1),\, r \cdot \sigma(2),\, \ldots)$$

Prove the following equality, for all $r \in \mathbb{R}$:

$$r \times \sigma \;=\; [r] \times \sigma \tag{12.9}$$

Here the symbol $\times$ on the left denotes scalar multiplication as defined in (12.8); on the right, $\times$ denotes the convolution product of two streams as defined in (12.3). □

Notation 264 (writing r for $[r]$). Equation (12.9) offers a justification for the following abuse of notation: in the future, we shall often write $r \times \sigma$ for $[r] \times \sigma$. More generally, we shall often simply write r instead of $[r]$ in *any* context of streams and stream operations. For instance,

$$r + \sigma \;=\; [r] + \sigma \qquad\qquad r\sigma \;=\; r \times \sigma \;=\; [r] \times \sigma$$

where in the latter case, we additionally have omitted the multiplication symbol. □

Proposition 265 ($\mathbb{R}^\omega$ with convolution product is a commutative ring). The set $\mathbb{R}^\omega$ of streams of real numbers with the operations

$$(\mathbb{R}^\omega,\, +,\, -,\, \times,\, [0],\, [1]\,)$$

is a commutative ring.

Proof: All of the ring properties can be proved by coinduction up-to, see Exercises 266 and 274 below. As an example, we prove

$$(\sigma + \tau) \times \rho \;=\; (\sigma \times \rho) \,+\, (\tau \times \rho) \tag{12.10}$$

for all $\sigma, \tau, \rho \in \mathbb{R}^\omega$. We define

$$R \;=\; \{(\,(\sigma+\tau)\times\rho, (\sigma\times\rho) \,+\, (\tau\times\rho)\,) \mid \sigma,\tau,\rho \in \mathbb{R}^\omega\}$$

and show that R is a bisimulation up-to. For the initial values, we have:

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

and for the stream derivatives:

$$\begin{aligned}
((\sigma+\tau)\times\rho)' &= ((\sigma+\tau)'\times\rho) \,+\, ((\sigma+\tau)(0)\times\rho') \\
&= ((\sigma'+\tau')\times\rho) \,+\, ((\sigma(0)+\tau(0))\times\rho') \\
&\mathrel{R^c} ((\sigma'\times\rho) \,+\, (\tau'\times\rho)) \,+\, ((\sigma(0)\times\rho') \,+\, (\tau(0)\times\rho')) \\
&= ((\sigma'\times\rho) \,+\, (\sigma(0)\times\rho')) \,+\, ((\tau'\times\rho) \,+\, (\tau(0)\times\rho')) \\
&= (\sigma\times\rho)'
\end{aligned}$$

Here R^c is the closure of R with respect to the operators $+$ and $\times$; in the last but one equality, we have used the fact that $+$ is commutative and associative. Thus, R is a bisimulation up-to and (12.10) follows by coinduction up-to, Proposition 255. □

Exercise 266. Prove the following identities: for all $\sigma, \tau, \rho \in \mathbb{R}^\omega$, $r, t \in \mathbb{R}$:

$$\begin{aligned} 1 \times \sigma &= \sigma & 0 \times \sigma &= 0 \\ \sigma + \tau &= \tau + \sigma & (\sigma + \tau) + \rho &= \sigma + (\tau + \rho) \\ (r + t) \times \sigma &= (r \times \sigma) + (t \times \sigma) & \sigma \times (\tau + \rho) &= (\sigma \times \tau) + (\sigma \times \rho) \end{aligned}$$

where we follow Notation 264, writing r for $[r]$ and t for $[t]$. □

The following constant stream plays a central role in the calculus of streams.

Definition 267 (the constant stream X). Let the stream X be given by

$$\mathsf{X} = (0, 1, 0, 0, 0, \ldots)$$

or, equivalently, by the following SDE:

$$\mathsf{X}(0) = 0 \qquad\qquad \mathsf{X}' = [1]$$

Note the difference between $\mathsf{X} = (0, 1, 0, 0, 0, \ldots)$ and $[1] = (1, 0, 0, 0, \ldots)$. □

We have the following elementary properties: for all $\sigma \in \mathbb{R}^\omega$ and $n \geqslant 0$,

$$\begin{aligned} \mathsf{X} \times \sigma &= (0, \sigma(0), \sigma(1), \sigma(2), \ldots) && (12.11) \\ \mathsf{X}^2 &= (0, 0, 1, 0, 0, 0, \ldots) \\ \mathsf{X}^n &= (\underbrace{0, \ldots, 0}_{n \text{ times}}, 1, 0, 0, 0, \ldots) && (12.12) \end{aligned}$$

As a consequence,

$$\begin{aligned} (\mathsf{X} \times \sigma)' &= \sigma && (12.13) \\ \left(\mathsf{X}^{n+1}\right)' &= \mathsf{X}^n && (12.14) \end{aligned}$$

Example 268. Here are some examples of streams:

$$\begin{aligned} 1 - \mathsf{X} &= (1, -1, 0, 0, 0, \ldots) \\ 1 + 7\mathsf{X}^2 - \sqrt{2}\mathsf{X}^5 &= (1, 0, 7, 0, 0, -\sqrt{2}, 0, 0, 0, \ldots) \\ (1 - 3\mathsf{X}) \times (2 + 5\mathsf{X}^2) &= 2 - 6\mathsf{X} + 5\mathsf{X}^2 - 15\mathsf{X}^3 \end{aligned}$$

□

Multiplying a stream σ by the stream X can be seen as an elementary form of stream *integration*, as is expressed by the following theorem.

Theorem 269 (fundamental theorem of stream calculus).

$$\sigma \;=\; \sigma(0) + (\mathsf{X} \times \sigma') \qquad (\sigma \in \mathbb{R}^\omega)$$

Proof: The equality trivially follows from the observation that, on the right, $\sigma(0)$ denotes the stream $[\sigma(0)] = (\sigma(0), 0, 0, \ldots)$ and, furthermore,

$$\begin{aligned} \mathsf{X} \times \sigma' &= \mathsf{X} \times (\sigma(1), \sigma(2), \sigma(3), \ldots) \\ &= (0, \sigma(1), \sigma(2), \sigma(3), \ldots) \end{aligned}$$

Equivalently, the equality can be proved by coinduction. □

Theorem 269 tells us how to compute a stream σ from its initial value $\sigma(0)$ and its derivative σ', and is thus analogous to the *fundamental theorem of calculus*, which for functions $f\colon \mathbb{R} \to \mathbb{R}$ and $x \in \mathbb{R}$ reads – under conditions – as follows:

$$f(x) \;=\; f(0) + \int_0^x f'(t)\, dt$$

For another characterisation of streams inspired by classical analysis, we recall from Lemma 77 that for any $\sigma \in \mathbb{R}^\omega$ and $n \geqslant 0$, we have

$$\sigma^{(n)}(0) \;=\; \sigma(n) \tag{12.15}$$

where the higher-order derivatives of σ are defined by

$$\sigma^{(0)} \;=\; \sigma \qquad\qquad \sigma^{(n+1)} \;=\; (\sigma^{(n)})'$$

Thus

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

We call the latter the stream of *Taylor coefficients* of σ (with respect to stream derivation) and observe that, trivially, streams are identical to the stream of their Taylor coefficients.

There is also the corresponding Taylor *series* representation for streams, for which we need to introduce the operation of infinite sum first.

Definition 270 (infinite sum). Let I be a (finite or) infinite set and let $\{\sigma_i\}_{i\in I}$ be a family of streams that is *summable*: for all $n \geqslant 0$, the set

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

is finite. We define the *infinite* sum $\Sigma_{i\in I}\, \sigma_i$ by the following system of SDEs:

$$(\Sigma_{i\in I}\, \sigma_i)(0) \;=\; \Sigma_{i\in I}\; \sigma_i(0) \qquad\qquad (\Sigma_{i\in I}\, \sigma_i)' \;=\; \Sigma_{i\in I}\; \sigma_i'$$

Note that the sum of initial values $\sigma_i(0)$ is finite because we assume the family $\{\sigma_i\}_{i\in I}$ to be summable. □

Theorem 271 (Taylor series). *For every $\sigma \in \mathbb{R}^\omega$,*

$$\sigma \;=\; \sum_{i=0}^{\infty} \sigma^{(i)}(0) \times \mathsf{X}^i \;=\; \sum_{i=0}^{\infty} \sigma(i) \times \mathsf{X}^i$$

We shall sometimes write this infinite sum also as

$$\sigma \;=\; \sigma(0) \,+\, \sigma(1)\mathsf{X} \,+\, \sigma(2)\mathsf{X}^2 \,+\, \cdots \tag{12.16}$$

Proof: The second equality follows from equality (12.15), the first from (12.12). Equivalently, the theorem follows by coinduction. □

Exercise 272. Prove that for any sequence of streams $\tau_0, \tau_1, \tau_2, \ldots \in \mathbb{R}^\omega$, the family $\{\tau_i \times X^i\}_{i\geqslant 0}$ is summable. In particular, the family $\{\sigma^{(i)}(0) \times X^i\}_{i\geqslant 0}$ above is. □

Remark 273 (generating functions). In mathematics, Taylor series are a means to study the behaviour of functions $f\colon \mathbb{R} \to \mathbb{R}$. Sometimes, Taylor series are also used as *representations* of infinite sequences, that is, streams. In that case, f is called a *generating function* for the stream it represents. For instance, the generating function

$$f(x) \;=\; \frac{1}{1-2x} \;=\; 1 + 2x + 2^2x^2 + 2^3x^3 + \cdots$$

represents the stream $(1, 2, 2^2, 2^3, \ldots)$. Note that one has to be careful: the function f above is not everywhere defined and the infinite sum diverges for certain values of x. A way out is to treat x not as a real variable but as an *indeterminate*, and to treat the infinite sum as a *formal* series. Here is how the Wikipedia entry on the notion of generating function has it:

> In mathematics, the term generating function is used to describe an infinite sequence of numbers (a_n) by treating them as the coefficients of a series expansion. The sum of this infinite series is the generating function. Unlike an ordinary series, this *formal* series is allowed to diverge, meaning that the generating function *is not always a true function* and the *variable* is actually an *indeterminate.*

(A general reference on generating functions is [Wil94].) In contrast, identity 12.16 of Theorem 271 tells us that every stream *is identical* to its own Taylor series. For instance,

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

where the second equality will be made sense of in Section 12.2 below, once the operation of convolution inverse will have been introduced. Note that all we have in these two equalities are streams (such as $1 = [1]$ and X) and stream functions (such as convolution product, infinite sum, and convolution inverse). Notably, notions such as *formal*, *representation* or *divergence* simply do not come into the picture. □

We shall see some examples of Taylor series in Example 279 below.

Exercise 274 (commutativity of the convolution product). Using Equation (12.6), it is trivial to prove that the convolution product is commutative:

$$\sigma \times \tau \; = \; \tau \times \sigma \tag{12.17}$$

for all $\sigma, \tau \in \mathbb{R}^\omega$. A proof by coinduction, not using (12.6) but based on the defining SDE (12.3), is not at all easy. The reason is that (12.3) is not "symmetric" regarding the roles of σ and τ (whereas (12.6) *is*). As it happens, (12.3) is a special case of a more general definition of product, which generalises also *non* commutative cases, such as language concatenation. The commutativity of the convolution product *of streams* is, in other words, a bit of a coincidence. Here is another, now *symmetric* SDE for the convolution product:

$$(\sigma \times \tau)(0) \; = \; \sigma(0) \cdot \tau(0) \qquad (\sigma \times \tau)' \; = \; (\sigma' \times \tau) + (\sigma \times \tau') - (\mathsf{X} \times \sigma' \times \tau') \tag{12.18}$$

It uses the constant stream X introduced in Definition 267. Show that both (12.18) and (12.3) have the same solution. Then prove (12.17) by coinduction. □

12.2 Convolution inverse

The set $\mathbb{R}$ of real numbers carries not only a ring structure but it is also a *field*: for every $r \in \mathbb{R}$ such that $r \neq 0$ there exists $r^{-1} \in \mathbb{R}$, also written as $1/r$ or $\frac{1}{r}$, such that

$$r \times r^{-1} \; = \; 1 \qquad\qquad r^{-1} \times r \; = \; 1$$

As before, such an operation on real numbers can be extended to an operation on *streams* of real numbers. For every stream $\sigma \in \mathbb{R}^\omega$ such that $\sigma(0) \neq 0$, we shall define a stream σ^{-1}, sometimes also written as $1/\sigma$ or $\frac{1}{\sigma}$, such that

$$\sigma \times \sigma^{-1} \; = \; 1 \qquad\qquad \sigma^{-1} \times \sigma \; = \; 1 \tag{12.19}$$

On the right, 1 denotes the stream $[1] = (1, 0, 0, 0, \dots)$, as usual following Notation 264. The stream σ^{-1} is called the (convolution) *inverse* of σ. We can use the equation

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

which states a *property* of the inverse of σ, as a *specification* from which the definition of σ^{-1} can be derived, as follows. If we take initial values on both sides, then we obtain

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

which implies

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

by the defining SDEs for $\times$ and [1] in Definition 261. It follows that

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

This defines the initial value of σ^{-1}, and explains at the same time the condition $\sigma(0) \neq 0$. Computing derivatives in the equation $\sigma \times \sigma^{-1} = 1$, we find, on the left,

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

again by the defining SDE for $\times$; on the right, we have $1' = 0$. This implies

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

This defines what the derivative of σ^{-1} should be. All in all, we have obtained an SDE with which we can now formally define the operation of convolution inverse.

Definition 275 (convolution inverse). For every $\sigma \in \mathbb{R}^\omega$ such that $\sigma(0) \neq 0$ we define the (convolution) *inverse* σ^{-1} by the following system of SDEs:

$$\sigma^{-1}(0) \;=\; \sigma(0)^{-1} \qquad\qquad (\sigma^{-1})' \;=\; -\sigma(0)^{-1} \times \sigma' \times \sigma^{-1} \tag{12.20}$$

Since this system is in GSOS format, it has a unique solution. □

As usual, we shall write

$$\sigma^{-n} \;=\; (\sigma^{-1})^n \qquad\qquad \frac{\tau}{\sigma} \;=\; \tau \times \sigma^{-1} \qquad\qquad (n \geqslant 0,\, \sigma \in \mathbb{R}^\omega,\, \sigma(0) \neq 0)$$

There are the following elementary properties (whose proof is left to the reader).

Proposition 276. For all $\sigma, \tau \in \mathbb{R}^\omega$ with $\sigma(0) \neq 0$ and $\tau(0) \neq 0$,

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

□

Exercise 277 ($\mathbb{R}^\omega$ is an integral domain). We mentioned above that

$$(\mathbb{R},\, +,\, -,\, \times,\, (-)^{-1},\, 0,\, 1\,)$$

is a field. Although we lifted the operation of inverse to $\mathbb{R}^\omega$, this does *not* make

$$(\mathbb{R}^\omega,\, +,\, -,\, \times,\, (-)^{-1},\, [0],\, [1]\,)$$

a field. (Why not?) Prove that $\mathbb{R}^\omega$ is, instead, an *integral domain without zero divisors.* □

Exercise 278 (a recurrence relation for convolution inverse). We saw in equations (12.5) and (12.6) an explicit formula for the nth element of the sum and product of two streams. There does not seem to exist a similar closed formula for $\sigma^{-1}(n)$ for streams $\sigma \in \mathbb{R}^\omega$ with $\sigma(0) \neq 0$. However, there is the following recurrence relation:

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

Derive this equality from $\sigma \times \sigma^{-1} = 1$ and identity (12.6). □

Here are some examples of convolution inverse and of Taylor series.

Example 279. One can compute the inverse of a stream by repeatedly taking its derivative. For an elementary example, we consider

$$1 - \mathsf{X} \;=\; (0, -1, 0, 0, 0, \ldots)$$

We compute the derivative of its inverse using the defining stream differential Equation (12.20) for inverse:

$$\begin{aligned} \left(\frac{1}{1-\mathsf{X}}\right)' &= -1^{-1} \times (1-\mathsf{X})' \times \frac{1}{1-\mathsf{X}} \\ &= -1 \times -1 \times \frac{1}{1-\mathsf{X}} \\ &= \frac{1}{1-\mathsf{X}} \end{aligned}$$

Together with Theorem 271, this implies

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

Similarly, one computes

$$\begin{aligned} \frac{1}{1-2\mathsf{X}} &= 1 + 2\mathsf{X} + 2^2\mathsf{X}^2 + 2^3\mathsf{X}^3 + \cdots \;=\; (1, 2, 2^2, 2^3, \ldots) \\ \frac{1}{1-\mathsf{X}^2} &= 1 + \mathsf{X}^2 + \mathsf{X}^4 + \mathsf{X}^6 + \cdots \;=\; (1, 0, 1, 0, 1, 0, \ldots) \\ \frac{1}{(1-\mathsf{X})^2} &= 1 + 2\mathsf{X} + 3\mathsf{X}^2 + 4\mathsf{X}^3 + \cdots \;=\; (1, 2, 3, 4, \ldots) \end{aligned}$$ □

Exercise 280. Prove by coinduction that for all $\sigma \in \mathbb{R}^\omega$,

$$\frac{1}{1-\mathsf{X}} \times \sigma \;=\; \Sigma(\sigma) \;=\; (\sigma(0),\ \sigma(0)+\sigma(1),\ \sigma(0)+\sigma(1)+\sigma(2),\ \ldots)$$

where Σ is the operation of partial sums, defined in identity (7.14) and characterised by identity (7.18). □

With the following, surprisingly helpful rule, one can easily compute derivatives of fractions σ/τ, without having to remember the rules for the derivative of the convolution product and the inverse.

Proposition 281. For all $\sigma \in \mathbb{R}^\omega$: $\quad \sigma' = (\sigma - \sigma(0))'$.

Proof: This follows from

$$\begin{aligned} (\sigma - \sigma(0))' &= (\mathsf{X} \times \sigma')' && \text{(by Theorem 269)} \\ &= \sigma' && \text{(by Equation (12.13))} \end{aligned}$$

or, equivalently, from $(\sigma - \sigma(0))' = \sigma' - \sigma(0)' = \sigma' - 0 = \sigma'$. □

Here is an example of the use of Proposition 281 in the calculation of the derivative of a fraction:

$$\begin{aligned} \left(\frac{2+\mathsf{X}^3}{1-\mathsf{X}-\mathsf{X}^2}\right)' &= \left(\frac{2+\mathsf{X}^3}{1-\mathsf{X}-\mathsf{X}^2} - 2\right)' && \text{(Proposition 281)} \\ &= \left(\frac{2+\mathsf{X}^3}{1-\mathsf{X}-\mathsf{X}^2} - \frac{2-2\mathsf{X}-2\mathsf{X}^2}{1-\mathsf{X}-\mathsf{X}^2}\right)' \\ &= \left(\frac{2\mathsf{X}+2\mathsf{X}^2+\mathsf{X}^3}{1-\mathsf{X}-\mathsf{X}^2}\right)' \\ &= \left(\mathsf{X} \times \frac{2+2\mathsf{X}+\mathsf{X}^2}{1-\mathsf{X}-\mathsf{X}^2}\right)' \\ &= \frac{2+2\mathsf{X}+\mathsf{X}^2}{1-\mathsf{X}-\mathsf{X}^2} && \text{(by Equation (12.13))} \end{aligned}$$

Exercise 282. Compute the (higher-order) derivatives $\sigma^{(n)}$ and $\tau^{(n)}$, for all $n \geqslant 0$, of

$$\sigma \;=\; \frac{1}{(1-\mathsf{X})^2} \qquad\qquad \tau \;=\; \frac{1+\mathsf{X}}{(1-\mathsf{X})^3}$$

and prove that $\sigma = (1, 2, 3, \ldots)$ and $\tau = (1^2, 2^2, 3^2, \ldots)$. □

Exercise 283. Prove the following identity:

$$\left(\frac{1}{(1-r\mathsf{X})^n}\right)' \;=\; \frac{1}{(1-r\mathsf{X})^n} + \frac{1}{(1-r\mathsf{X})^{n-1}} + \cdots + \frac{1}{1-r\mathsf{X}}$$

for all $r \in \mathbb{R}$ and $n \geqslant 1$. □

With the help of convolution product and inverse, we define the following two classes of streams.

Definition 284 (polynomial, rational streams). We call a stream $\sigma \in \mathbb{R}^\omega$ *polynomial* if it is of the form

$$\sigma \;=\; a_0 + a_1\mathsf{X} + a_2\mathsf{X}^2 + \cdots + a_n\mathsf{X}^n \qquad (n \geqslant 0,\ a_i \in \mathbb{R})$$

We call σ *rational* if it is a fraction of two polynomials:

$$\sigma \;=\; \frac{a_0 + a_1\mathsf{X} + a_2\mathsf{X}^2 + \cdots + a_n\mathsf{X}^n}{b_0 + b_1\mathsf{X} + b_2\mathsf{X}^2 + \cdots + b_m\mathsf{X}^m} \qquad (n, m \geqslant 0,\ a_i, b_j \in \mathbb{R},\ b_0 \neq 0)$$

□

Here are examples of a polynomial and a rational stream:

$$\begin{aligned} 2 - 6\mathsf{X} + 5\mathsf{X}^2 - 15\mathsf{X}^3 &= (2, -6, 5, -15, 0, 0, 0, \ldots) \\ \frac{1}{(1-\mathsf{X})^2} &= 1 + 2\mathsf{X} + 3\mathsf{X}^2 + 4\mathsf{X}^3 + \cdots \;=\; (1, 2, 3, 4, \ldots) \end{aligned}$$

We shall study rational streams in much detail later, in Section 12.8, where we shall see that rational streams are the solutions of so-called finite *linear systems* of SDEs; and in Chapter 14, where Theorem 402 will give various additional characterisations.

12.3 Stream composition

We have seen how the constant stream $\mathsf{X} = (0, 1, 0, 0, 0, \ldots)$ can be used as a basic building block to construct new streams and to formulate general properties such as the fundamental theorem of stream calculus and Taylor series representations. As it turns out, X can under certain conditions also be viewed as a *place holder*. The following definition will help to make this precise.

Definition 285 (stream composition). We define the composition of streams

$$\circ \colon \mathbb{R}^\omega \times \mathbb{R}^\omega \to \mathbb{R}^\omega$$

by the following stream differential equation:

$$(\sigma \circ \tau)(0) \;=\; \sigma(0) \qquad\qquad (\sigma \circ \tau)' \;=\; \tau' \times (\sigma' \circ \tau)$$

The above SDE is in GSOS format, and so $\circ$ is well-defined, by Theorem 252. □

We shall use stream composition in Section 12.5, to relate the convolution product and the *shuffle* product (which will be introduced in Section 12.4). A few elementary properties and examples of stream composition are given below.

Proposition 286. For all $r \in \mathbb{R}$ and all $\rho, \sigma, \tau \in \mathbb{R}^\omega$,

$$\begin{aligned} [r] \circ \tau &= [r] & \qquad (\rho + \sigma) \circ \tau &= (\rho \circ \tau) + (\sigma \circ \tau) \\ \sigma^{-1} \circ \tau &= (\sigma \circ \tau)^{-1} & \qquad (\rho \times \sigma) \circ \tau &= (\rho \circ \tau) \times (\sigma \circ \tau) \end{aligned}$$

and similarly for the infinite sum. Moreover, for all τ with $\tau(0) = 0$, we have

$$\mathsf{X} \circ \tau \;=\; \tau.$$

Proof: All identities follow by coinduction up-to. We give a proof of

$$(\rho + \sigma) \circ \tau \;=\; (\rho \circ \tau) + (\sigma \circ \tau)$$

by showing that the relation

$$R \;=\; \{\, (\, (\rho + \sigma) \circ \tau,\; (\rho \circ \tau) + (\sigma \circ \tau)\,) \mid \rho, \sigma, \tau \in \mathbb{R}^\omega \,\}$$

is a bisimulation-up-to. For the initial values,

$$((\rho + \sigma) \circ \tau)(0) \;=\; ((\rho \circ \tau) + (\sigma \circ \tau))(0)$$

follows from the definitions of $+$ and $\circ$. For the derivative, we have

$$\begin{aligned} ((\rho + \sigma) \circ \tau)' &= \tau' \times ((\rho + \sigma)' \circ \tau) && \text{(definition } \circ) \\ &= \tau' \times ((\rho' + \sigma') \circ \tau) && \text{(definition } +) \\ &\mathrel{R^c} \tau' \times ((\rho' \circ \tau) + (\sigma' \circ \tau)) \\ &= \tau' \times (\rho' \circ \tau) + \tau' \times (\sigma' \circ \tau) \\ &= ((\rho \circ \tau) + (\sigma \circ \tau))' && \text{(definitions of } + \text{ and } \circ) \end{aligned}$$

where R^c is the context closure of R. This proves that R is a bisimulation up-to and the equality follows by coinduction up-to, Proposition 255. The other identities are proved similarly. □

Proposition 286 shows that composing with τ distributes over sum, product and inverse. Stream composition acts as identity on the constant stream $[r]$, but it replaces X with τ when $\tau(0) = 0$. As a consequence, if $\tau(0) = 0$ then the composition $\sigma \circ \tau$ is obtained by replacing every X in σ with τ.

Example 287. The following equality is immediate by Proposition 286:

$$\frac{\mathsf{X}}{1-\mathsf{X}-\mathsf{X}^2} \circ \frac{\mathsf{X}}{1+\mathsf{X}} = \frac{\frac{\mathsf{X}}{1+\mathsf{X}}}{1-\frac{\mathsf{X}}{1+\mathsf{X}}-(\frac{\mathsf{X}}{1+\mathsf{X}})^2} = \frac{\mathsf{X}(1+\mathsf{X})}{1+\mathsf{X}-\mathsf{X}^2} \qquad \square$$

Exercise 288 (composition preserves rationality). Prove, more generally, that composition preserves rationality. $\square$

Exercise 289. Let $\sigma, \tau \in \mathbb{R}^\omega$ with $\tau(0) = 0$. Prove the following identities:

$$\begin{aligned}
\sigma(\mathsf{X}) \;&=\; \sigma \\
\sigma(-\mathsf{X}) \;&=\; \sigma(0) \,-\, \sigma(1)\mathsf{X} \,+\, \sigma(2)\mathsf{X}^2 \,-\, \sigma(3)\mathsf{X}^3 \,+\, \cdots \\
\frac{\sigma(\mathsf{X}) \,+\, \sigma(-\mathsf{X})}{2} \;&=\; \sigma(0) \,+\, \sigma(2)\mathsf{X}^2 \,+\, \sigma(4)\mathsf{X}^4 \,+\, \cdots \\
\sigma(\mathsf{X}^2) \;&=\; \sigma(0) \,+\, \sigma(1)\mathsf{X}^2 \,+\, \sigma(2)\mathsf{X}^4 \,+\, \sigma(3)\mathsf{X}^6 \,+\, \cdots
\end{aligned}$$

where we have used the notation

$$\sigma(\tau) \;=\; \sigma \circ \tau$$

suggesting that substitution behaves similar to function application. $\square$

12.4 Shuffle product

We introduce and study the so-called *shuffle* product (the name derives from a similar operator on formal languages), which we already encountered, for streams of natural numbers, in Exercise 94. Replacing convolution product by shuffle product will provide an alternative ring structure on $\mathbb{R}^\omega$. Furthermore, the combined use of both products greatly enhances the expressiveness of our stream calculus.

Definition 290 (shuffle product). We define the operation of *shuffle* product

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

by means of the following SDE:

$$(\sigma \otimes \tau)(0) \;=\; \sigma(0) \times \tau(0) \qquad\qquad (\sigma \otimes \tau)' \;=\; (\sigma' \otimes \tau) \,+\, (\sigma \otimes \tau') \tag{12.21}$$

We shall use the following notation: for all $n \geqslant 0$,

$$\sigma^{\underline{0}} \;=\; 1 \qquad\qquad \sigma^{\underline{n+1}} \;=\; \sigma \otimes \sigma^{\underline{n}} \tag{12.22}$$

where, as always following Notation 264, 1 stands for the stream [1]. □

Since the SDE above is in GSOS format, it follows from Theorem 252 that the shuffle product is well-defined. Replacing convolution product by shuffle product in Proposition 265 gives the following.

Proposition 291 (**$\mathbb{R}^\omega$ with shuffle product is a commutative ring).** The set $\mathbb{R}^\omega$ of streams of real numbers with the operations

$$(\mathbb{R}^\omega,\, +,\, -,\, \otimes,\, [0],\, [1]\,)$$

is a commutative ring.

Proof: The proofs of the ring identies are again by coinduction up-to, Proposition 255, similar to Proposition 265, which stated that $\mathbb{R}^\omega$ with *convolution* product is a commutative ring. The proof of commutativity is simpler for the shuffle product than for the convolution product, because the defining SDE for $\sigma \otimes \tau$ is symmetric with respect to the role of σ and τ. As a consequence,

$$R \;=\; \{(\,\sigma \otimes \tau,\; \tau \otimes \sigma\,) \mid \sigma, \tau \in \mathbb{R}^\omega\}$$

is readily seen to be a bisimulation up-to. □

Remark 292. There is the following closed form for the shuffle product (which we already saw in Exercise 94 for streams of natural numbers), which is a variation on equality (12.6) for the convolution product: for $\sigma, \tau \in \mathbb{R}^\omega$ and $n \geqslant 0$,

$$(\sigma \otimes \tau)(n) \;=\; \sum_{k=0}^{n} \binom{n}{k} \cdot \sigma(k) \cdot \tau(n-k) \tag{12.23}$$

This formula will play no role in what follows, because proofs by coinduction, based on the defining SDE (12.21) for the shuffle product, will not involve binomial coefficients and, as a consequence, will be simpler. □

Here are some elementary properties of the shuffle product.

Proposition 293. For all $\sigma \in \mathbb{R}^\omega$, $r, s \in \mathbb{R}$, $n \geqslant 0$,

$$r \otimes \sigma \;=\; r \times \sigma \tag{12.24}$$

$$\left(\sigma^{\underline{n+1}}\right)' \;=\; (n+1) \otimes \sigma' \otimes \sigma^{\underline{n}} \tag{12.25}$$

$$\mathsf{X}^{\underline{n}} \;=\; n! \times \mathsf{X}^n \tag{12.26}$$

$$\mathsf{X} \otimes \sigma \;=\; \sigma(0)\mathsf{X} + 2\sigma(1)\mathsf{X}^2 + 3\sigma(2)\mathsf{X}^3 + \cdots \tag{12.27}$$

$$(\mathsf{X} \otimes \sigma')' \;=\; \sigma(1) + 2\sigma(2)\mathsf{X} + 3\sigma(3)\mathsf{X}^2 + \cdots \tag{12.28}$$

$$\frac{1}{1-r\mathsf{X}} \otimes \frac{1}{1-s\mathsf{X}} \;=\; \frac{1}{1-(r+s)\mathsf{X}} \tag{12.29}$$

Proof: Proofs are elementary, using induction or coinduction up-to, Proposition 255. For instance, since

$$\begin{aligned}
\left(\frac{1}{1-r\mathsf{X}} \otimes \frac{1}{1-s\mathsf{X}}\right)' &= \left(\left(\frac{1}{1-r\mathsf{X}}\right)' \otimes \frac{1}{1-s\mathsf{X}}\right) + \left(\frac{1}{1-r\mathsf{X}} \otimes \left(\frac{1}{1-s\mathsf{X}}\right)'\right) \\
&= \left(\frac{r}{1-r\mathsf{X}} \otimes \frac{1}{1-s\mathsf{X}}\right) + \left(\frac{1}{1-r\mathsf{X}} \otimes \frac{s}{1-s\mathsf{X}}\right) \\
&= (r+s) \times \left(\frac{1}{1-r\mathsf{X}} \otimes \frac{1}{1-s\mathsf{X}}\right)
\end{aligned}$$

and

$$\left(\frac{1}{1-(r+s)\mathsf{X}}\right)' \;=\; (r+s) \times \frac{1}{1-(r+s)\mathsf{X}}$$

it follows that

$$R \;=\; \left\{ \left(\frac{1}{1-r\mathsf{X}} \otimes \frac{1}{1-s\mathsf{X}} \;,\; \frac{1}{1-(r+s)\mathsf{X}} \right) \mid r, s \in \mathbb{R} \right\}$$

is a bisimulation up-to, from which (12.29) then follows by coinduction up-to. □

Exercise 294 (Taylor series with shuffle product). Prove the following identity:

$$\sigma \;=\; \sum_{i=0}^{\infty} \frac{\sigma^{(i)}(0)}{i!} \times \mathsf{X}^{\underline{i}} \;=\; \sum_{i=0}^{\infty} \frac{\sigma(i)}{i!} \times \mathsf{X}^{\underline{i}}$$

for every $\sigma \in \mathbb{R}^\omega$. □

Like the convolution product, also shuffle product has a partial inverse.

Definition 295 (shuffle inverse). Let the *shuffle inverse* $\sigma^{-\underline{1}}$ of a stream $\sigma \in \mathbb{R}^\omega$ with $\sigma(0) \neq 0$ be defined by the following SDE:

$$\sigma^{-\underline{1}}(0) \;=\; \sigma(0)^{-1} \qquad\qquad (\sigma^{-\underline{1}})' \;=\; -\sigma' \otimes \sigma^{-\underline{1}} \otimes \sigma^{-\underline{1}} \tag{12.30}$$

We shall use the following notation:

$$\sigma^{-\underline{n}} \;=\; \left(\sigma^{-\underline{1}}\right)^{\underline{n}} \tag{12.31}$$

for all $n \geqslant 0$. □

Exercise 296. Prove the following identities:

$$\begin{aligned} \sigma \otimes \sigma^{-\underline{1}} &= 1 & \left(\sigma^{-\underline{1}}\right)^{-\underline{1}} &= \sigma \\ (\sigma \otimes \tau)^{-\underline{1}} &= \sigma^{-\underline{1}} \otimes \tau^{-\underline{1}} & (1 - \mathsf{X})^{-\underline{1}} &= 1 + 1!\mathsf{X} + 2!\mathsf{X}^2 + \cdots \end{aligned} \tag{12.32}$$

by coinduction up-to. □

Exercise 297 (stream exponentiation). In analysis, the exponential function

$$e^{-} : \mathbb{R} \to \mathbb{R}$$

satisfies, for any function $f : \mathbb{R} \to \mathbb{R}$,

$$\left(e^{f(x)}\right)' \;=\; f'(x) \times e^{f(x)}$$

where $\times$ is the elementwise product of functions. We define *stream exponentiation*

$$\mathsf{e}^{-} : \mathbb{R}^\omega \to \mathbb{R}^\omega$$

by the following system of SDEs:

$$(\mathsf{e}^{\sigma})(0) \;=\; e^{\sigma(0)} \qquad\qquad (\mathsf{e}^{\sigma})' \;=\; \sigma' \otimes \mathsf{e}^{\sigma} \tag{12.33}$$

Prove the following familiar identities:

$$\begin{aligned} \left(\mathsf{e}^{\mathsf{X}}\right)' &= \mathsf{e}^{\mathsf{X}} & \mathsf{e}^{\sigma} \otimes \mathsf{e}^{\tau} &= \mathsf{e}^{\sigma + \tau} \\ \mathsf{e}^{-\sigma} &= (\mathsf{e}^{\sigma})^{-\underline{1}} & \mathsf{e}^{\sigma} &= 1 + \frac{\sigma^{\underline{1}}}{1!} + \frac{\sigma^{\underline{2}}}{2!} + \cdots \end{aligned}$$

for all $\sigma, \tau \in \mathbb{R}^\omega$; and the following, more surprising one, for all $r \in \mathbb{R}$:

$$\mathsf{e}^{r\mathsf{X}} \;=\; \frac{1}{1 - r\mathsf{X}}$$ □

Things get more complicated rather quickly. For instance, it is by no means clear what a closed expression for the stream $\mathsf{e}^{\mathsf{X}^2}$ could be. We shall see in Exercise 378 that this stream can be represented by an *infinite* weighted stream automaton.

12.5 Laplace transform

The convolution product and the shuffle product can be related by means of the so-called *Laplace transform* of streams, which we shall define using a new type of stream derivation.

Definition 298 (analytic stream derivative). We define the *analytic stream derivative* of a stream $\sigma \in \mathbb{R}^\omega$ by

$$\frac{\mathsf{d}}{\mathsf{dX}}(\sigma) \;=\; (\mathsf{X} \otimes \sigma')'$$

which we shall also write as

$$\frac{\mathsf{d}\sigma}{\mathsf{dX}} \;=\; \frac{\mathsf{d}}{\mathsf{dX}}(\sigma) \qquad \square$$

The following equalities illustrate that analytic stream derivative behaves for convolution product and inverse in the same way as function derivation in analysis.

Proposition 299. For all $\sigma, \tau \in \mathbb{R}^\omega$,

$$\begin{aligned}
\frac{\mathsf{d}(\sigma \times \tau)}{\mathsf{dX}} &= \left(\frac{\mathsf{d}\sigma}{\mathsf{dX}} \times \tau\right) + \left(\sigma \times \frac{\mathsf{d}\tau}{\mathsf{dX}}\right) \\
\frac{\mathsf{d}(\sigma^{-1})}{\mathsf{dX}} &= -\frac{\mathsf{d}\sigma}{\mathsf{dX}} \times \sigma^{-1} \times \sigma^{-1} \\
\frac{\mathsf{d}\sigma}{\mathsf{dX}} &= \sigma(1) + 2\sigma(2)\mathsf{X} + 3\sigma(3)\mathsf{X}^2 + \cdots
\end{aligned} \tag{12.34}$$

assuming $\sigma(0) \neq 0$ for the second equality.

Proof: The first two equalities are readily proved by coinduction up-to. The third equality is identity (12.28), and explains the name of this new type of derivative. $\square$

Next we use the operation of analytic stream derivative to define the so-called *Laplace* transform, with which we shall be able to relate the convolution and shuffle products.

Definition 300 (Laplace transform). Let $\Lambda \colon \mathbb{R}^\omega \to \mathbb{R}^\omega$ be defined by the following stream differential equation:

$$\Lambda(\sigma)(0) \;=\; \sigma(0) \qquad\qquad \Lambda(\sigma)' \;=\; \Lambda\left(\frac{\mathsf{d}\sigma}{\mathsf{dX}}\right) \qquad \square$$

One easily shows that

$$\Lambda(\sigma) \;=\; 0!\sigma(0) \,+\, 1!\sigma(1)\mathsf{X} \,+\, 2!\sigma(2)\mathsf{X}^2 \,+\, \cdots$$

As a consequence, we can define the inverse of the Laplace transform by

$$\Lambda^{-1}(\sigma) \;=\; \frac{\sigma(0)}{0!} \,+\, \frac{\sigma(1)}{1!}\mathsf{X} \,+\, \frac{\sigma(2)}{2!}\mathsf{X}^2 \,+\, \cdots$$

Exercise 301 (Laplace as coalgebra isomorphism). Prove that

$$\Lambda^{-1}(\sigma)(0) \;=\; \sigma(0) \qquad\qquad \frac{\mathsf{d}\;(\Lambda^{-1}(\sigma))}{\mathsf{dX}} \;=\; \Lambda^{-1}(\sigma')$$

for every $\sigma \in \mathbb{R}^\omega$. Show that Λ is an isomorphism of stream systems:

$$\Lambda\colon \left(\mathbb{R}^\omega,\, \left\langle\, (-)(0), \frac{\mathsf{d}}{\mathsf{dX}} \,\right\rangle\right) \;\cong\; (\,\mathbb{R}^\omega,\, \langle\, (-)(0), (-)' \,\rangle\,)$$

with on the left, the analytic and on the right, the ordinary stream derivative. □

The Laplace transform changes the convolution product and inverse into the shuffle product and inverse.

Proposition 302. For all $r \in \mathbb{R}$ and $\sigma, \tau \in \mathbb{R}^\omega$,

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

Proof: The first two equalities are trivial, the one but last is the definition of Λ, and the other three can be proved by a straightforward coinduction up-to. □

Exercise 303. Prove similar properties for Λ^{-1}. □

All in all, we have obtained the following.

Theorem 304 (Laplace as ring isomorphism). *The Laplace transform is an isomorphism of rings:*

$$\Lambda\colon (\mathbb{R}^\omega,\, +,\, -,\, \times,\, [0],\, [1]\,) \;\cong\; (\mathbb{R}^\omega,\, +,\, -,\, \otimes,\, [0],\, [1]\,)$$ □

Thus the Laplace transform and its inverse let us switch between the two isomorphic ring structures on $\mathbb{R}^\omega$, transforming convolution product into shuffle product and back. Often, however, it makes perfect sense to work with *both* ring structures at the same time, as is illustrated by the following identities, in which both the convolution and the shuffle product occur. Since the shuffle product occurs on the left in these equations, and not on the right, they allow us to *eliminate* the shuffle product.

Theorem 305 (shuffle product elimination). *For all $\sigma \in \mathbb{R}^\omega$, $r \in \mathbb{R}$,*

$$\begin{aligned}
\frac{1}{1+r\mathsf{X}} \otimes \frac{1}{1-r\mathsf{X}} &= 1 && (12.35)\\
\mathsf{X} \otimes \sigma &= \left(\mathsf{X}^2 \times \frac{\mathsf{d}\sigma}{\mathsf{dX}}\right) + (\mathsf{X} \times \sigma) && (12.36)\\
\frac{1}{1-r\mathsf{X}} \otimes \sigma &= \frac{1}{1-r\mathsf{X}} \times \left(\sigma \circ \frac{\mathsf{X}}{1-r\mathsf{X}}\right) && (12.37)\\
&= \frac{\sigma(0)}{1-r\mathsf{X}} + \frac{\sigma(1)\mathsf{X}}{(1-r\mathsf{X})^2} + \frac{\sigma(2)\mathsf{X}^2}{(1-r\mathsf{X})^3} + \cdots && (12.38)
\end{aligned}$$

Proof: The first equality is an instance of identity (12.29) in Proposition 293. For (12.36), we compute as follows:

$$\begin{aligned}
\mathsf{X} \otimes \sigma &= \mathsf{X} \times (\mathsf{X} \otimes \sigma)' && \text{(Theorem 269, } (\mathsf{X} \times \sigma)(0) = 0)\\
&= \mathsf{X} \times ((\mathsf{X} \otimes \sigma') + \sigma) && \text{(definition } \otimes)\\
&= \mathsf{X} \times (\mathsf{X} \otimes \sigma') + (\mathsf{X} \times \sigma) && (\times \text{ distributes over } +)\\
&= \mathsf{X} \times (\mathsf{X} \times (\mathsf{X} \otimes \sigma')') + (\mathsf{X} \times \sigma) && \text{(Theorem 269, } (\mathsf{X} \times \sigma')(0) = 0)\\
&= \mathsf{X}^2 \times (\mathsf{X} \otimes \sigma')' + (\mathsf{X} \times \sigma)\\
&= \left(\mathsf{X}^2 \times \frac{\mathsf{d}\sigma}{\mathsf{dX}}\right) + (\mathsf{X} \times \sigma)
\end{aligned}$$

Equality (12.37) can be readily proved by coinduction up-to, and (12.38) follows from Theorem 271 and the properties of stream composition, Proposition 286. □

Here is an illustration of identity (12.36) above:

$$\begin{aligned}
\mathsf{X} \otimes \frac{1}{1+\mathsf{X}^2} &= \mathsf{X}^2 \times \frac{\mathsf{d}}{\mathsf{dX}}\left(\frac{1}{1+\mathsf{X}^2}\right) + \left(\mathsf{X} \times \frac{1}{1+\mathsf{X}^2}\right) && \text{(identity (12.36))}\\
&= \mathsf{X}^2 \times \frac{-2\mathsf{X}}{(1+\mathsf{X}^2)^2} + \frac{\mathsf{X}}{1+\mathsf{X}^2} && \text{(identity (12.34))}\\
&= \frac{\mathsf{X}-\mathsf{X}^3}{(1+\mathsf{X}^2)^2}
\end{aligned}$$

Exercise 306. Use Theorem 305 to prove the following equalities:

$$\mathsf{X}^2 \otimes \frac{1}{1+\mathsf{X}^2} = \frac{\mathsf{X}^2 - 3\mathsf{X}^4}{(1+\mathsf{X}^2)^3} \qquad \mathsf{X}^2 \otimes \frac{\mathsf{X}}{1+\mathsf{X}^2} = \frac{-\mathsf{X}^3 - 3\mathsf{X}^5}{(1+\mathsf{X}^2)^3}$$
$$\frac{1}{1-r\mathsf{X}} \otimes \mathsf{X}^n = \frac{\mathsf{X}^n}{(1-r\mathsf{X})^{n+1}} \qquad \frac{1}{1-r\mathsf{X}} \otimes (1+r\mathsf{X})^n = \frac{1}{(1-r\mathsf{X})^{n+1}}$$

for all $n \geqslant 0$ and $r \in \mathbb{R}$. □

Exercise 307 (non-standard SDEs with analytic derivative). We can formulate *non-standard* stream differential equations by using the analytic stream derivative instead of the standard *tail* derivative. For instance, we can define a stream $\sigma \in \mathbb{R}^\omega$ by

$$\sigma(0) = 1 \qquad \frac{\mathsf{d}\sigma}{\mathsf{dX}} = \sigma$$

Compute σ. □

12.6 Solving analytic differential equations

The method of *undetermined coefficients* [BR78] is a classical technique in mathematical analysis for the solution of differential equations that define *analytic* functions. The idea is quickly explained by means of an example. In order to solve the differential equation

$$f(0) = 0 \qquad f'(0) = 1 \qquad f + f'' = 0 \tag{12.39}$$

one assumes the solution to be of the form

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

Computing f' and f'' gives

$$f'(x) = s_1 + \frac{s_2}{1!}x + \frac{s_3}{2!}x^2 + \cdots \qquad f''(x) = s_2 + \frac{s_3}{1!}x + \frac{s_4}{2!}x^2 + \cdots$$

Substituting these expressions in (12.39) above, one obtains the following *difference equation* for the stream $(s_0, s_1, s_2, \ldots)$ of coefficients of f:

$$s_0 = 0 \qquad s_1 = 1 \qquad s_{n+2} + s_n = 0 \qquad (n \geqslant 0)$$

Thus the problem of solving a differential equation for f is reduced to the problem of solving a difference equation for the Taylor coefficients of f.

Although conceptually clear, the method of undetermined coefficients has two major drawbacks. First, more interesting differential equations quickly lead to complicated difference equations. Second, there is no general technique for translating the solution of the difference equation (if found at all) back into a workable expressions for f.

Here we present a variation on the above method, as an application of the calculus of streams developed in the present chapter. We define

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

and observe that such functions are infinitely differentiable in 0. Our main tool will be the function

$$\mathsf{T}\colon \mathcal{A} \to \mathbb{R}^\omega \qquad\qquad \mathsf{T}(f) \;=\; (f(0),\, f'(0),\, f''(0),\, \ldots)$$

that sends an analytic function to its Taylor series. We call $\mathsf{T}(f)$ the *Taylor transform* of f. Formally, we define T by the stream differential equation

$$\mathsf{T}(f)(0) \;=\; f(0) \qquad\qquad \mathsf{T}(f)' \;=\; \mathsf{T}(f') \tag{12.40}$$

where in the second equality, we have stream derivation, on the left, and function derivation, on the right.

Exercise 308. Show that T is a monomorphism of stream systems. □

Our method, then, consists of three steps:

(1) T transforms a differential equation for f into an SDE for the Taylor series of f;

(2) this SDE is solved in stream calculus;

(3) the resulting solution is translated back into an expression for f.

As we shall see, this approach works well for a number of interesting examples; at the same time, we do not have a characterisation of precisely which class of differential equations can be solved in this manner. We define, as usual,

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

for all $x \in \mathbb{R}$. The following proposition shows that T is well-behaved.

Proposition 309. For all $r \in \mathbb{R}$, $f, g \in \mathcal{A}$,

$$\begin{aligned} \mathsf{T}(r) &= [r] & \mathsf{T}(x) &= \mathsf{X} \\ \mathsf{T}(f+g) &= \mathsf{T}(f) + \mathsf{T}(g) & \mathsf{T}(f \cdot g) &= \mathsf{T}(f) \otimes \mathsf{T}(g) \\ \mathsf{T}\left(f^{-1}\right) &= (\mathsf{T}(f))^{-\underline{1}} & \mathsf{T}\left(e^{f}\right) &= \mathsf{e}^{\mathsf{T}(f)} \end{aligned}$$

where the r in $\mathsf{T}(r)$ denotes the constant function r; the x in $\mathsf{T}(x)$ denotes the identity function; $(-)^{-\underline{1}}$ denotes the operation of shuffle inverse, Definition 295; and $\mathsf{e}^{(-)}$ denotes stream exponentiation, introduced in Exercise 297.

Proof: Fairly straightforward by coinduction up-to. □

Here are a few identities on the Taylor transforms of some well-known functions that will be helpful when solving differential equations.

Proposition 310. For all $n \geqslant 0$, $r \in \mathbb{R}$,

$$\mathsf{T}(x^n) = n!\mathsf{X}^n \qquad \mathsf{T}(e^{rx}) = \frac{1}{1 - r\mathsf{X}} \tag{12.41}$$

$$\mathsf{T}(\sin(rx)) = \frac{r\mathsf{X}}{1 + r^2\mathsf{X}^2} \qquad \mathsf{T}(\cos(rx)) = \frac{1}{1 + r^2\mathsf{X}^2} \tag{12.42}$$

Furthermore, for all $f \in \mathcal{A}$,

$$\mathsf{T}(f \cdot x) = \left(\mathsf{X}^2 \times \frac{\mathsf{d}\mathsf{T}(f)}{\mathsf{d}\mathsf{X}}\right) + (\mathsf{X} \times \mathsf{T}(f)) \tag{12.43}$$

$$\mathsf{T}(f \cdot e^{rx}) = \left(\frac{1}{1 - r\mathsf{X}}\right) \times \left(\mathsf{T}(f) \circ \frac{\mathsf{X}}{1 - r\mathsf{X}}\right) \tag{12.44}$$

Proof: For (12.43), we have

$$\begin{aligned} \mathsf{T}(f \cdot x) &= \mathsf{T}(f) \otimes \mathsf{T}(x) && \text{(Proposition 309)} \\ &= \mathsf{T}(f) \otimes \mathsf{X} && \text{(Proposition 309)} \\ &= \left(\mathsf{X}^2 \times \frac{\mathsf{d}\mathsf{T}(f)}{\mathsf{d}\mathsf{X}}\right) + (\mathsf{X} \times \mathsf{T}(f)) && \text{(identity (12.36))} \end{aligned}$$

And for (12.44),

$$\begin{aligned} \mathsf{T}(f \cdot e^{rx}) &= \mathsf{T}(f) \otimes \mathsf{T}(e^{rx}) && \text{(Proposition 309)} \\ &= \mathsf{T}(f) \otimes \left(\frac{1}{1 - r\mathsf{X}}\right) && \text{(identity (12.41))} \\ &= \left(\frac{1}{1 - r\mathsf{X}}\right) \times \left(\mathsf{T}(f) \circ \frac{\mathsf{X}}{1 - r\mathsf{X}}\right) && \text{(identity (12.37))} \end{aligned}$$

The four equations in (12.41) and (12.42) are straightforward by coinduction up-to. □

We are now sufficiently prepared to tackle a variety of differential equations.

Example 311. Applying T to the differential equation at the beginning of this section,

$$f(0) \;=\; 0 \qquad f'(0) \;=\; 1 \qquad f + f'' \;=\; 0 \tag{12.39}$$

we obtain the following SDE:

$$\sigma(0) \;=\; 0 \qquad \sigma'(0) \;=\; 1 \qquad \sigma + \sigma'' \;=\; 0 \qquad (\text{where } \sigma = \mathsf{T}(f))$$

Solving this SDE in stream calculus, we find

$$\mathsf{T}(f) \;=\; \sigma \;=\; \frac{\mathsf{X}}{1+\mathsf{X}^2} \;=\; \mathsf{X} - \mathsf{X}^3 + \mathsf{X}^5 - \mathsf{X}^7 + \cdots$$

Consulting Proposition 310, we obtain

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

For a second example, consider the following (so-called non-homogeneous) equation:

$$f(0) \;=\; 1 \qquad f' - f \;=\; e^x$$

Applying T gives

$$\sigma(0) \;=\; 1 \qquad \sigma' - \sigma \;=\; \frac{1}{1-\mathsf{X}} \qquad (\text{where } \sigma = \mathsf{T}(f))$$

Solving this SDE in stream calculus gives

$$\sigma \;=\; \frac{1}{(1-\mathsf{X})^2} \;=\; 1 + 2\mathsf{X} + 3\mathsf{X}^2 + 4\mathsf{X}^3 + \cdots$$

We can rewrite σ as

$$\sigma \;=\; \frac{1}{(1-\mathsf{X})^2} \;=\; (1+\mathsf{X}) \otimes \frac{1}{1-\mathsf{X}} \qquad (\text{Exercise 306})$$

Consulting Proposition 310, we then obtain

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

For a third example, consider

$$f(0) \;=\; 1 \qquad f'(0) \;=\; 1 \qquad f'' + r^2 f \;=\; 1$$

Applying T gives

$$\sigma(0) \;=\; 1 \qquad \sigma'(0) \;=\; 1 \qquad \sigma'' + r^2\sigma \;=\; 1 \qquad (\text{where } \sigma = \mathsf{T}(f))$$

Next we rewrite the solution of this SDE as follows:

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

Consulting as before Proposition 310, we obtain

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

Exercise 312. Solve the following differential equation:

$$f(0) = 1 \qquad f'(0) = 0 \qquad f'' - f' = 2 + 6x \qquad \square$$

12.7 Newton transform

We look at yet another stream transformation, called the *Newton transform*, with which various new stream representations can be expressed. For the definition of the Newton transform, we will need a new notion of stream derivative, different from both the *tail* and the *analytic* derivatives.

Definition 313 (difference operator). We define $\Delta\colon \mathbb{R}^\omega \to \mathbb{R}^\omega$ by

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

for all $\sigma \in \mathbb{R}^\omega$. $\square$

Exercise 314 (non-standard SDEs with the difference operator). In Exercise 307, we formulated *non-standard* SDEs using the analytic stream derivative instead of the tail derivative. Similarly, we can formulate non-standard SDEs with the difference operator as derivative. For instance, we can define a stream $\sigma \in \mathbb{R}^\omega$ by

$$\sigma(0) = 1 \qquad \Delta(\sigma) = \sigma$$

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

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

Prove this identity and use it to compute a closed expression for the stream σ above. $\square$

Definition 315 (Newton transform). Let $\mathsf{N}\colon \mathbb{R}^\omega \to \mathbb{R}^\omega$ be defined by the following (standard) stream differential equation:

$$\mathsf{N}(\sigma)(0) \;=\; \sigma(0) \qquad\qquad \mathsf{N}(\sigma)' \;=\; \mathsf{N}\,(\,\Delta(\sigma)\,)$$

We call $\mathsf{N}(\sigma)$ the stream of *Newton coefficients* of σ. It follows from the definition that

$$\mathsf{N}(\sigma) \;=\; (\,\Delta^0(\sigma)(0),\, \Delta^1(\sigma)(0),\, \Delta^2(\sigma)(0),\, \ldots\,) \tag{12.45}$$

where $\Delta^0(\sigma) = \sigma$ and $\Delta^{n+1}(\sigma) = \Delta(\Delta^n(\sigma))$. □

For concrete calculations of the Newton coefficients $\mathsf{N}(\sigma)$ of a stream $\sigma \in \mathbb{R}^\omega$, one can use the following theorem, which gives an easy characterisation of N in terms of the shuffle product, and which, at the same time, shows that N is bijective.

Theorem 316 (Newton transform and shuffle product). *The function* N *is bijective and satisfies*

$$\mathsf{N}(\sigma) \;=\; \frac{1}{1+\mathsf{X}} \otimes \sigma \qquad\qquad \mathsf{N}^{-1}(\sigma) \;=\; \frac{1}{1-\mathsf{X}} \otimes \sigma \qquad\qquad (\sigma \in \mathbb{R}^\omega)$$

Proof: The first equality follows by coinduction from the fact that

$$R \;=\; \left\{ \left(\mathsf{N}(\sigma),\, \frac{1}{1+\mathsf{X}} \otimes \sigma \right) \;\mid\; \sigma \in \mathbb{R}^\omega \right\}$$

is a bisimulation, which we show next. For $\sigma \in \mathbb{R}^\omega$, we have $\mathsf{N}(\sigma)(0) = ((1/1+\mathsf{X}) \otimes \sigma)(0)$. For the derivatives, we have

$$\begin{aligned}
\left(\frac{1}{1+\mathsf{X}} \otimes \sigma \right)' \;&=\; \left(-\frac{1}{1+\mathsf{X}} \otimes \sigma \right) \,+\, \left(\frac{1}{1+\mathsf{X}} \otimes \sigma' \right) \\
&=\; \frac{1}{1+\mathsf{X}} \otimes (\sigma' - \sigma) \\
&=\; \frac{1}{1+\mathsf{X}} \otimes \Delta\sigma
\end{aligned}$$

This proves that R is a bisimulation, since

$$\mathsf{N}(\sigma)' \;=\; \mathsf{N}(\Delta(\sigma)) \;\; R \;\; \left(\frac{1}{1+\mathsf{X}} \otimes \Delta(\sigma) \right) \;=\; \left(\frac{1}{1+\mathsf{X}} \otimes \sigma \right)'$$

The second equality of the theorem follows from identity (12.35) in Theorem 305. □

Exercise 317. Use Theorem 316 and identity (12.23) in Remark 292 to prove the following closed formula for the Newton series of a stream $\sigma \in \mathbb{R}^\omega$: for all $n \geqslant 0$,

$$\mathsf{N}(\sigma)(n) \;=\; \sum_{k=0}^{n} \binom{n}{k} \cdot (-1)^{n-k} \cdot \sigma(k)$$

Prove a similar formula for $\mathsf{N}^{-1}(\sigma)(n)$. □

Exercise 318 (Newton transform as coalgebra isomorphism). Prove that

$$\mathsf{N}^{-1}(\sigma)(0) \;=\; \sigma(0) \qquad\qquad \Delta\left(\mathsf{N}^{-1}(\sigma)\right) \;=\; \mathsf{N}^{-1}(\sigma')$$

for every $\sigma \in \mathbb{R}^\omega$. Show that N is an isomorphism of stream systems:

$$\mathsf{N}\colon\; (\,\mathbb{R}^\omega,\; \langle\,(-)(0), \Delta\,\rangle\,) \;\cong\; (\,\mathbb{R}^\omega,\; \langle\,(-)(0), (-)'\,\rangle\,)$$

with on the left, the difference operator and on the right, the standard tail derivative. □

Theorem 316 can be combined with the identities for shuffle product elimination from Theorem 305 to compute the Newton series of concrete streams.

Example 319. For the Fibonacci numbers $\mathsf{X}/1-\mathsf{X}-\mathsf{X}^2 \;=\; (0,1,1,2,3,5,8,\ldots)$, we have

$$\begin{aligned}
\mathsf{N}\left(\frac{\mathsf{X}}{1-\mathsf{X}-\mathsf{X}^2}\right) \;&=\; \frac{1}{1+\mathsf{X}} \otimes \frac{\mathsf{X}}{1-\mathsf{X}-\mathsf{X}^2} && \text{(Theorem 316)}\\
&=\; \frac{1}{1+\mathsf{X}} \times \left(\frac{\mathsf{X}}{1-\mathsf{X}-\mathsf{X}^2} \circ \frac{\mathsf{X}}{1+\mathsf{X}}\right) && \text{(identity (12.37))}\\
&=\; \frac{1}{1+\mathsf{X}} \times \left(\frac{\frac{\mathsf{X}}{1+\mathsf{X}}}{1-\frac{\mathsf{X}}{1+\mathsf{X}}-(\frac{\mathsf{X}}{1+\mathsf{X}})^2}\right) && \text{(Proposition 286)}\\
&=\; \frac{\mathsf{X}}{1+\mathsf{X}-\mathsf{X}^2}
\end{aligned}$$

For all $r \in \mathbb{R}$, the stream of powers of r is transformed as follows:

$$\begin{aligned}
\mathsf{N}(1, r, r^2, \ldots) \;&=\; \mathsf{N}\left(\frac{1}{1-r\mathsf{X}}\right)\\
&=\; \frac{1}{1+\mathsf{X}} \otimes \frac{1}{1-r\mathsf{X}} && \text{(Theorem 316)}\\
&=\; \frac{1}{1+\mathsf{X}} \times \left(\frac{1}{1-r\mathsf{X}} \circ \frac{\mathsf{X}}{1+\mathsf{X}}\right) && \text{(identity (12.37))}\\
&=\; \frac{1}{1-(r-1)\mathsf{X}} && \text{(Proposition 286)}\\
&=\; (1, (r-1), (r-1)^2, \ldots)
\end{aligned}$$

□

Exercise 320. Compute $\mathsf{N}(0,1,0,1,0,1,\ldots)$ and $\mathsf{N}(0,1,2,3,\ldots)$. □

Exercise 321 (Newton transform of stream of factorial numbers). We saw in Exercise 296 that

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

where $(-)^{-\underline{1}}$ is the operation of shuffle inverse introduced in Exercise 295. Use Theorem 316, identity (12.37), and (an extension of) Proposition 286 to prove that

$$\mathsf{N}(0!, 1!, 2!, \ldots) \;=\; \frac{1}{1+\mathsf{X}} \times \left(\frac{1}{1+\mathsf{X}}\right)^{-\underline{1}}$$

Note that the latter expression combines convolution inverse, convolution product, and shuffle inverse. As it turns out, $\mathsf{N}(0!, 1!, 2!, \ldots)$ is the stream of so-called *derangements* [OEI, A000166]. □

Exercise 322 (Newton transform preserves rationality). Prove that σ is rational iff $\mathsf{N}(\sigma)$ is rational, for all $\sigma \in \mathbb{R}^\omega$. □

Next we show that the Newton transform is an isomorphism between yet two other ring structures on the set $\mathbb{R}^\omega$, featuring the following two product operators on streams.

Definition 323 (Hadamard and infiltration product). We define the operations of *Hadamard* and *infiltration* product

$$\odot\colon\ \mathbb{R}^\omega \times \mathbb{R}^\omega \to \mathbb{R}^\omega \qquad\qquad \uparrow\colon\ \mathbb{R}^\omega \times \mathbb{R}^\omega \to \mathbb{R}^\omega$$

by means of the following SDEs:

$$(\sigma \odot \tau)(0) \;=\; \sigma(0) \times \tau(0) \qquad (\sigma \odot \tau)' \;=\; \sigma' \odot \tau' \tag{12.46}$$

$$(\sigma \uparrow \tau)(0) \;=\; \sigma(0) \times \tau(0) \qquad (\sigma \uparrow \tau)' \;=\; (\sigma' \uparrow \tau) + (\sigma \uparrow \tau') + (\sigma' \uparrow \tau') \tag{12.47}$$

□

The Hadamard product and the infiltration product give rise to two additional ring structures on the set of streams:

$$(\mathbb{R}^\omega,\ +,\ -,\ \odot,\ [0],\ \mathsf{ones}\,) \qquad\qquad (\mathbb{R}^\omega,\ +,\ -,\ \uparrow,\ [0],\ [1]\,)$$

where $\mathsf{ones} = \frac{1}{1-\mathsf{X}} = (1,1,1,\ldots)$. A proof of the ring identities, for both of these ring structures, can as always be readily given by coinduction up-to.

Exercise 324. Prove the following equalities:

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

□

The Newton transform, and its inverse, change Hadamard products into infiltration products, and back.

Theorem 325 (Newton as ring isomorphism). *The Newton transform is an isomorphism of rings:*

$$\mathsf{N}\colon (\mathbb{R}^\omega,\, +,\, -,\, \odot,\, [0],\, \mathsf{ones}\,) \;\cong\; (\mathbb{R}^\omega,\, +,\, -,\, \uparrow,\, [0],\, [1]\,)$$

Proof: We have $\mathsf{N}([0]) = [0]$ and $\mathsf{N}(\mathsf{ones}) = [1]$. The identities for the operators, such as $\mathsf{N}(-\sigma) = -\mathsf{N}(\sigma)$, are all by coinduction up-to. In particular, for

$$\mathsf{N}(\sigma \odot \tau) \;=\; \mathsf{N}(\sigma) \uparrow \mathsf{N}(\tau)$$

one proves the identities

$$\begin{aligned}
\mathsf{N}(\sigma \odot \tau)' &= \mathsf{N}(\sigma' \odot \tau') \;-\; \mathsf{N}(\sigma \odot \tau) \\
(\mathsf{N}(\sigma) \uparrow \mathsf{N}(\tau))' &= \mathsf{N}(\sigma') \uparrow \mathsf{N}(\tau') \;-\; \mathsf{N}(\sigma) \uparrow \mathsf{N}(\tau)
\end{aligned}$$

from which it follows that

$$\{\, \langle\, \mathsf{N}(\sigma \odot \tau),\; \mathsf{N}(\sigma) \uparrow \mathsf{N}(\tau)\,\rangle \;\mid\; \sigma, \tau \in \mathbb{R}^\omega \,\}$$

is a bisimulation up-to. □

The Newton transform gives rise to the following expansion theorem for streams (see [Sch68] for a *finitary* version of thereof).

Theorem 326 (Euler expansion for streams). *For all $\sigma \in \mathbb{R}^\omega$,*

$$\begin{aligned}
\sigma &= \sum_{i=0}^{\infty} \frac{\mathsf{N}(\sigma)(i) \times \mathsf{X}^i}{(1-\mathsf{X})^{i+1}} \\
&= \frac{\Delta^0(\sigma)(0)}{1-\mathsf{X}} + \frac{\Delta^1(\sigma)(0) \times \mathsf{X}}{(1-\mathsf{X})^2} + \frac{\Delta^2(\sigma)(0) \times \mathsf{X}^2}{(1-\mathsf{X})^3} + \cdots
\end{aligned}$$

Proof:

$$\begin{aligned}
\sigma &= \frac{1}{1-\mathsf{X}} \otimes \frac{1}{1+\mathsf{X}} \otimes \sigma && \text{((12.35) in Theorem 305)}\\
&= \frac{1}{1-\mathsf{X}} \otimes \mathsf{N}(\sigma) && \text{(Theorem 316)}\\
&= \frac{\mathsf{N}(\sigma)(0)}{1-\mathsf{X}} + \frac{\mathsf{N}(\sigma)(1)\times\mathsf{X}}{(1-\mathsf{X})^2} + \frac{\mathsf{N}(\sigma)(2)\times\mathsf{X}^2}{(1-\mathsf{X})^3} + \cdots && \text{((12.38) in Theorem 305)}\\
&= \frac{\Delta^0(\sigma)(0)}{1-\mathsf{X}} + \frac{\Delta^1(\sigma)(0)\times\mathsf{X}}{(1-\mathsf{X})^2} + \frac{\Delta^2(\sigma)(0)\times\mathsf{X}^2}{(1-\mathsf{X})^3} + \cdots && \text{(identity (12.45))}
\end{aligned}$$

□

Example 327. We illustrate the use of Theorem 326 in an easy derivation of a rational expression for the stream of cubes:

$$(1^3, 2^3, 3^3, \ldots) = \frac{1+4\mathsf{X}+\mathsf{X}^2}{(1-\mathsf{X})^4}$$

As before, let $\mathsf{ones} = (1,1,1,\ldots)$ and $\mathsf{nat} = (1,2,3,\ldots)$. We shall write the *nth* Hadamard power of a stream $\sigma \in \mathbb{R}^\omega$ as

$$\sigma^{\langle 0\rangle} = \mathsf{ones} \qquad\qquad \sigma^{\langle n+1\rangle} = \sigma^{\langle n\rangle} \odot \sigma$$

With this notation, $(1^3, 2^3, 3^3, \ldots) = \mathsf{nat}^{\langle 3\rangle}$. We compute the respective values of $\Delta^n(\mathsf{nat}^{\langle 3\rangle})$ as follows. By definition, $\Delta^0(\mathsf{nat}^{\langle 3\rangle}) = \mathsf{nat}^{\langle 3\rangle}$. Next,

$$\begin{aligned}
\Delta^1(\mathsf{nat}^{\langle 3\rangle}) &= \left(\mathsf{nat}^{\langle 3\rangle}\right)' - \mathsf{nat}^{\langle 3\rangle}\\
&= \left(\mathsf{nat}'\right)^{\langle 3\rangle} - \mathsf{nat}^{\langle 3\rangle} && \text{(SDE (12.46))}\\
&= \left(\mathsf{nat}+\mathsf{ones}\right)^{\langle 3\rangle} - \mathsf{nat}^{\langle 3\rangle} && \text{(since } \mathsf{nat}' = \mathsf{nat}+\mathsf{ones}\text{)}\\
&= \left(\mathsf{nat}^{\langle 3\rangle} + 3\mathsf{nat}^{\langle 2\rangle}\odot\mathsf{ones} + 3\mathsf{nat}\odot\mathsf{ones}^{\langle 2\rangle} + \mathsf{ones}^{\langle 3\rangle}\right) - \mathsf{nat}^{\langle 3\rangle}\\
&= 3\mathsf{nat}^{\langle 2\rangle} + 3\mathsf{nat} + \mathsf{ones}
\end{aligned}$$

Similarly, one finds

$$\Delta^2(\mathsf{nat}^{\langle 3\rangle}) = 6\mathsf{nat} + 6\mathsf{ones} \qquad \Delta^3(\mathsf{nat}^{\langle 3\rangle}) = 6\mathsf{ones} \qquad \Delta^i(\mathsf{nat}^{\langle 3\rangle}) = 0$$

for all $i \geqslant 4$. By Theorem 326, we obtain the following rational expression:

$$\begin{aligned}
(1^3, 2^3, 3^3, \ldots) &= \mathsf{nat}^{\langle 3\rangle}\\
&= \frac{1}{1-\mathsf{X}} + \frac{7\mathsf{X}}{(1-\mathsf{X})^2} + \frac{12\mathsf{X}^2}{(1-\mathsf{X})^3} + \frac{6\mathsf{X}^3}{(1-\mathsf{X})^4}\\
&= \frac{1+4\mathsf{X}+\mathsf{X}^2}{(1-\mathsf{X})^4}
\end{aligned}$$

□

Exercise 328. Prove, more generally, that $\Delta^i\left(\mathsf{nat}^{\langle n\rangle}\right) = 0$, for all $n \geqslant 1$ and $i > n$, and show that

$$\mathsf{nat}^{\langle n\rangle} \;=\; \frac{\sum_{m=0}^{n-1} A(n,m) \times \mathsf{X}^m}{(1-\mathsf{X})^{n+1}}$$

Here $A(n,m)$ are the so-called *Eulerian numbers*, which are defined, for every $n \geqslant 1$ and $0 \leqslant m \leqslant n-1$, by the following recurrence relation:

$$A(n,m) \;=\; (n-m)A(n-1,m-1) \;+\; (m+1)A(n-1,m) \qquad \square$$

12.8 Rational streams and linear systems of SDEs

We will discuss various properties of the well-behaved class of *rational* streams of real numbers, introduced in Definition 284. In particular, we show that rational streams are the solutions of finite *linear systems* of SDEs. Later, in Chapter 14, Theorem 402, we will see a number of additional characterisations of rational streams (in terms of so-called linear *stream* systems and weighted stream automata). We begin with the relevant definitions.

Definition 329 (linear systems of SDEs). A linear system of SDEs, of dimension $n \geqslant 1$, consists of n stream differential equations of the form

$$\begin{array}{rclcrcl} \sigma_1(0) & = & r_1 & \qquad & \sigma_1' & = & r_{11}\sigma_1 + \cdots + r_{1n}\sigma_n \\ & \vdots & & & & \vdots & \\ \sigma_n(0) & = & r_n & & \sigma_n' & = & r_{n1}\sigma_1 + \cdots + r_{nn}\sigma_n \end{array}$$

with $r_i, r_{ij} \in \mathbb{R}$. $\square$

Example 330. Before proving the correspondence between rational streams and linear systems of SDEs in general, let us look at an example. Consider the following linear system of dimension 2:

$$\begin{array}{rclcrcl} \sigma_1(0) & = & 0 & \qquad & \sigma_1' & = & \sigma_2 \\ \sigma_2(0) & = & 1 & & \sigma_2' & = & \sigma_1 + \sigma_2 \end{array}$$

Since any linear system of SDEs is in GSOS format, we know the system above has a unique solution. In order to compute this solution explicitly, we apply the fundamental theorem of stream calculus, both to σ_1 and σ_2:

$$\sigma_1 \;=\; \sigma_1(0) \,+\, (\mathsf{X} \times \sigma_1') \qquad \sigma_2 \;=\; \sigma_2(0) \,+\, (\mathsf{X} \times \sigma_2') \qquad \text{(Theorem 269)}$$

and compute as follows:

$$\begin{aligned}
\sigma_2 &= \sigma_2(0) + (\mathsf{X} \times \sigma_2') \\
&= 1 + (\mathsf{X} \times (\sigma_1 + \sigma_2)) && (\sigma_2(0) = 1 \text{ and } \sigma_2' = \sigma_1 + \sigma_2) \\
&= 1 + (\mathsf{X} \times ((\mathsf{X} \times \sigma_2) + \sigma_2)) && (\sigma_1 = \mathsf{X} \times \sigma_2) \\
&= 1 + (\mathsf{X} \times \sigma_2) + (\mathsf{X}^2 \times \sigma_2)
\end{aligned}$$

This implies

$$\begin{aligned}
1 &= \sigma_2 - (\mathsf{X} \times \sigma_2) - (\mathsf{X}^2 \times \sigma_2) \\
&= (1 - \mathsf{X} - \mathsf{X}^2) \times \sigma_2
\end{aligned}$$

As a consequence, we find

$$\sigma_1 = \frac{\mathsf{X}}{1 - \mathsf{X} - \mathsf{X}^2} \qquad\qquad \sigma_2 = \frac{1}{1 - \mathsf{X} - \mathsf{X}^2}$$

We see that the solution of our linear system of SDEs consists of rational streams. Conversely, any rational stream can be defined by a finite linear system of SDEs. Consider, for instance,

$$\tau_1 = \frac{1}{(1 - \mathsf{X})^2} \tag{12.48}$$

Defining $\tau_2 = \tau_1'$, we have

$$\begin{aligned}
\tau_2 &= \tau_1' & \tau_2' &= \left(\frac{2 - \mathsf{X}}{(1 - \mathsf{X})^2}\right)' \\
&= \left(\frac{1}{(1 - \mathsf{X})^2}\right)' & &= \frac{3 - 2\mathsf{X}}{(1 - \mathsf{X})^2} \\
&= \frac{2 - \mathsf{X}}{(1 - \mathsf{X})^2} & &= -\tau_1 + 2\tau_2
\end{aligned}$$

It follows that τ_1 and τ_2 satisfy the following linear system of SDEs of dimension 2:

$$\begin{aligned}
\tau_1(0) &= 1 & \tau_1' &= \tau_2 \\
\tau_2(0) &= 2 & \tau_2' &= -\tau_1 + 2\tau_2
\end{aligned}$$ □

Exercise 331. Show that there exist other linear systems of SDEs of which the stream τ_1 in (12.48) is a solution; for instance, by considering the identities

$$\tau_1' = \frac{2 - \mathsf{X}}{(1 - \mathsf{X})^2} = \frac{1}{1 - \mathsf{X}} + \frac{1}{(1 - \mathsf{X})^2}$$

and by defining $\tau_2 = 1/(1 - \mathsf{X})$. □

The examples above can be readily generalised to arbitrary rational streams.

Theorem 332 (rational streams and linear systems of SDEs). *For every $\sigma \in \mathbb{R}^\omega$, the following are equivalent:*

(1) σ is rational.

(2) σ can be defined by a finite linear system of SDEs.

Proof: From *(1)* to *(2)*: Since for all $\tau \in \mathbb{R}^\omega$ with $\tau(0) \neq 0$, and $n \geqslant 1$,

$$\left(\frac{\mathsf{X}^n}{\tau}\right)^{(n)} = \frac{1}{\tau} \qquad \text{(by identity (12.13))}$$

it suffices to understand the situation for rational streams of the form

$$\sigma = \frac{1}{\tau} \qquad\qquad \tau = 1 + b_1\mathsf{X} + b_2\mathsf{X}^2 + \cdots + b_m\mathsf{X}^m$$

where we have assumed $b_0 = 1$, for computational convenience. Computing σ', we find

$$\begin{aligned}
\sigma' &= \left(\frac{1}{\tau}\right)' \\
&= -\tau(0)^{-1} \times \tau' \times \frac{1}{\tau} \\
&= -1 \times \left(b_1 + b_2\mathsf{X} + \cdots + b_m\mathsf{X}^{m-1}\right) \times \frac{1}{\tau} \\
&= \frac{-b_1 - b_2\mathsf{X} - \cdots - b_m\mathsf{X}^{m-1}}{\tau} \qquad \text{(by identity (12.13))} \\
&= -b_1\frac{1}{\tau} - b_2\frac{\mathsf{X}}{\tau} - \cdots - b_m\frac{\mathsf{X}^{m-1}}{\tau}
\end{aligned}$$

If we define

$$\sigma_1 = \sigma = 1/\tau \qquad \sigma_2 = \mathsf{X}/\tau \qquad \ldots \qquad \sigma_m = \mathsf{X}^{m-1}/\tau$$

then it follows that σ is (part of) the solution of the following linear system of SDEs:

$$\begin{array}{rclrcl}
\sigma_1(0) &=& 1 & \sigma_1' &=& -b_1\sigma_1 - b_2\sigma_2 - \cdots - b_m\sigma_m \\
\sigma_2(0) &=& 0 & \sigma_2' &=& \sigma_1 \\
&\vdots& & &\vdots& \\
\sigma_m(0) &=& 0 & \sigma_m' &=& \sigma_{m-1}
\end{array}$$

From *(2)* to *(1)*: As our proof will use some linear algebra, we will first identify the relevant algebraic structure in which we can do matrix manipulations. Let

$$\mathsf{M}_n(\mathbb{R}^\omega) = \{\ n\text{-by-}n \text{ matrices with values in } \mathbb{R}^\omega\ \}$$

As usual, we write

$$M_{i,j} \;\; (\in \mathbb{R}^\omega)$$

for the entry of the matrix $M \in \mathsf{M}_n(\mathbb{R}^\omega)$ at row i and column j. We observe that $\mathsf{M}_n(\mathbb{R}^\omega)$ is a ring under matrix addition and matrix multiplication, which are defined as usual and written here, for $N, M \in \mathsf{M}_n(\mathbb{R}^\omega)$, as

$$N + M \qquad\qquad\qquad N \cdot M$$

The operation of *scalar* multiplication is defined by

$$(\sigma \cdot M)_{i,j} \;=\; \sigma \times M_{i,j} \qquad\qquad (\sigma \in \mathbb{R}^\omega,\; M \in \mathsf{M}_n(\mathbb{R}^\omega))$$

where $\times$ on the right denotes the convolution product of streams. Scalar multiplication is compatible with matrix multiplication (both denoted by $\cdot$) in the following sense:

$$\sigma \cdot (M \cdot N) \;=\; (\sigma \cdot M) \cdot N \;=\; M \cdot (\sigma \cdot N)$$

Now consider a linear system of SDEs of dimension $n \geqslant 1$:

$$\begin{array}{rclrcl} \sigma_1(0) & = & r_1 & \sigma_1' & = & r_{11}\sigma_1 + \cdots + r_{1n}\sigma_n \\ \vdots & & & \vdots & & \\ \sigma_n(0) & = & r_n & \sigma_n' & = & r_{n1}\sigma_1 + \cdots + r_{nn}\sigma_n \end{array} \tag{12.49}$$

with $r_i, r_{ij} \in \mathbb{R}$. Any such system can be written in matrix form as

$$\begin{pmatrix} \sigma_1 \\ \vdots \\ \sigma_n \end{pmatrix}(0) \;=\; N \qquad\qquad \begin{pmatrix} \sigma_1 \\ \vdots \\ \sigma_n \end{pmatrix}' \;=\; M \cdot \begin{pmatrix} \sigma_1 \\ \vdots \\ \sigma_n \end{pmatrix}$$

where initial value and derivative are taken element-wise, and with

$$N \;=\; \begin{pmatrix} r_1 \\ \vdots \\ r_n \end{pmatrix} \qquad\qquad M \;=\; \begin{pmatrix} r_{11} & \cdots & r_{1n} \\ \vdots & & \vdots \\ r_{n1} & \cdots & r_{nn} \end{pmatrix}$$

(As usual following Notation 264, real numbers r_{ij} denote here the corresponding *streams* $[r_{ij}]$.) Applying the fundamental theorem of stream calculus to $\sigma_1, \ldots, \sigma_n$, we find

$$\begin{aligned} \begin{pmatrix} \sigma_1 \\ \vdots \\ \sigma_n \end{pmatrix} &= \begin{pmatrix} \sigma_1 \\ \vdots \\ \sigma_n \end{pmatrix}(0) + \mathsf{X} \cdot \begin{pmatrix} \sigma_1 \\ \vdots \\ \sigma_n \end{pmatrix}' && \text{(Theorem 269)} \\ &= N + \mathsf{X} \cdot M \cdot \begin{pmatrix} \sigma_1 \\ \vdots \\ \sigma_n \end{pmatrix} \end{aligned}$$

(Note that $\mathsf{X} = (0, 1, 0, 0, 0, \ldots)$ is a scalar stream.) In $\mathsf{M}_n(\mathbb{R}^\omega)$, this is equivalent to

$$(I - (\mathsf{X} \cdot M)) \cdot \begin{pmatrix} \sigma_1 \\ \vdots \\ \sigma_n \end{pmatrix} = N$$

where I is the identity matrix. The solution to our linear system (12.49) is then obtained as:

$$\begin{pmatrix} \sigma_1 \\ \vdots \\ \sigma_n \end{pmatrix} = (I - (\mathsf{X} \cdot M))^{-1} \cdot N \tag{12.50}$$

The matrix inverse $(I - (\mathsf{X} \cdot M))^{-1}$ exists, since one can easily show that the determinant of $I - (\mathsf{X} \cdot M)$ is always equal to 1 (see Exercise 333 below). Now $(I - (\mathsf{X} \cdot M))^{-1}$ can be computed as usual, by performing elementary row operations on the identity matrix, which consist of multiplying or dividing by a rational stream, and of adding rows. It follows that since the matrix $I - (\mathsf{X} \cdot M)$ has rational streams as entries, so does $(I - (\mathsf{X} \cdot M))^{-1}$. As a consequence, the solution of the linear system (12.49) consists of rational streams. □

Exercise 333. Let R be a commutative ring. Prove that $\mathsf{M}_n(R)$ is an associative R-algebra, referring to [Lan02] for details. In general, an element of a matrix ring $\mathsf{M}_n(R)$ is invertible if its determinant has a multiplicative inverse in R. Hence $I - (\mathsf{X} \cdot M)$ has an inverse in $\mathsf{M}_n(\mathbb{R}^\omega)$ if its determinant is a stream whose initial value is non-zero. Prove that

$$\det(I - (\mathsf{X} \cdot M))\,(0) = 1$$

for any $M \in \mathsf{M}_n(\mathbb{R}^\omega)$. □

Example 334. Let us again compute the solution of the linear system of SDEs

$$\begin{aligned} \sigma_1(0) &= 0 & \qquad \sigma_1' &= \sigma_2 \\ \sigma_2(0) &= 1 & \qquad \sigma_2' &= \sigma_1 + \sigma_2 \end{aligned}$$

from Example 330, now using the matrix solution method from the proof of Theorem 332. Writing this system in matrix form, we obtain:

$$\begin{pmatrix} \sigma_1 \\ \sigma_2 \end{pmatrix}(0) = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \qquad \begin{pmatrix} \sigma_1 \\ \sigma_2 \end{pmatrix}' = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} \sigma_1 \\ \sigma_2 \end{pmatrix}$$

Its solution is given by identity (12.50):

$$\begin{aligned}\begin{pmatrix}\sigma_1\\ \sigma_2\end{pmatrix} &= \begin{pmatrix}1 & -\mathsf{X}\\ -\mathsf{X} & 1-\mathsf{X}\end{pmatrix}^{-1} \cdot \begin{pmatrix}0\\ 1\end{pmatrix}\\ &= \begin{pmatrix}1-\mathsf{X}/1-\mathsf{X}-\mathsf{X}^2 & \mathsf{X}/1-\mathsf{X}-\mathsf{X}^2\\ \mathsf{X}/1-\mathsf{X}-\mathsf{X}^2 & 1/1-\mathsf{X}-\mathsf{X}^2\end{pmatrix} \cdot \begin{pmatrix}0\\ 1\end{pmatrix}\\ &= \begin{pmatrix}\mathsf{X}/1-\mathsf{X}-\mathsf{X}^2\\ 1/1-\mathsf{X}-\mathsf{X}^2\end{pmatrix}\end{aligned}$$

which is, as expected, what we found earlier, in Example 330. □

Exercise 335. Show that $\sigma_1 = (0, 1, 1, 2, 3, 5, 8, 13, \ldots)$, in Example 334. □

Example 336. We compute a finite linear system of SDEs of which the stream

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

is a solution. Using Proposition 281, we compute

$$\sigma' = \left(\frac{1+\mathsf{X}}{(1-\mathsf{X})^3} - 1\right)' = \left(\frac{4\mathsf{X}-3\mathsf{X}^2+\mathsf{X}^3}{(1-\mathsf{X})^3}\right)' = \frac{4-3\mathsf{X}+\mathsf{X}^2}{(1-\mathsf{X})^3}$$

Similarly, we find

$$\sigma^{(2)} = \frac{9-11\mathsf{X}+4\mathsf{X}^2}{(1-\mathsf{X})^3} \qquad \sigma^{(3)} = \frac{16-23\mathsf{X}+9\mathsf{X}^2}{(1-\mathsf{X})^3}$$

There is the following dependence:

$$\sigma^{(3)} = \sigma - 3\sigma' + 3\sigma^{(2)}$$

This leads to the following linear system of SDEs:

$$\begin{aligned} \sigma_1(0) &= 1 & \sigma_1' &= \sigma_2\\ \sigma_2(0) &= 4 & \sigma_2' &= \sigma_3\\ \sigma_3(0) &= 9 & \sigma_3' &= \sigma_4\\ \sigma_4(0) &= 16 & \sigma_4' &= \sigma_1 - 3\sigma_2 + 3\sigma_3 \end{aligned}$$

of which $\sigma = \sigma_1$ is (part of) the solution. □

Exercise 337. Compute the solution of the following linear system of SDEs

$$\begin{aligned} \sigma_1(0) &= 1 & \sigma_1' &= \sigma_1 - \sigma_2 \\ \sigma_2(0) &= 3 & \sigma_2' &= 2\sigma_1 + \sigma_2 - \sigma_3 \\ \sigma_3(0) &= 2 & \sigma_3' &= \sigma_1 + 3\sigma_2 + 2\sigma_3 \end{aligned}$$

□

Exercise 338. Find a linear system of SDEs of which

$$\sigma = \frac{1 + 4\mathsf{X} + \mathsf{X}^2}{(1 - \mathsf{X})^4} = (1^3, 2^3, 3^3, \ldots)$$

is a solution. □

Remark 339 (context-free streams). We saw in Theorem 332 that a stream is rational if and only if it is defined by a linear system of SDEs. For instance,

$$\begin{aligned} \sigma_1(0) &= 0 & \sigma_1' &= \sigma_2 \\ \sigma_2(0) &= 1 & \sigma_2' &= \sigma_1 + \sigma_2 \end{aligned}$$

defines the rational stream of the Fibonacci numbers:

$$\sigma_1 = \frac{\mathsf{X}}{1 - \mathsf{X} - \mathsf{X}^2} = (0, 1, 1, 2, 3, 5, 8, \ldots)$$

A larger class of streams can be defined by allowing the defining SDEs to contain in the definition of the derivatives not only variables (such as σ_1 and σ_2 above) and sums (such as $\sigma_1 + \sigma_2$), but also convolution products, as in the following two examples:

$$\sigma(0) = 1 \qquad \sigma' = \sigma \times \sigma \tag{12.51}$$

$$\tau(0) = 1 \qquad \tau' = \tau + (\tau \times \tau) \tag{12.52}$$

One can prove that these SDEs define the streams of the *Catalan* numbers [OEI, A000108] and the *(large) Schröder* numbers [OEI, A006318]:

$$\begin{aligned} \sigma &= (1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862, \ldots) \\ \tau &= (1, 2, 6, 22, 90, 394, 1806, 8558, 41586, 206098, \ldots) \end{aligned}$$

Streams that are defined by SDEs of this type are called *context-free* or *algebraic*. The class of context-free streams is strictly larger than the class of rational streams: every rational stream is trivially context-free, and in Chapter 14, Exercise 401, we will see that the above streams of Catalan and Schroeder numbers are *not* rational.

In this book, we do not discuss the class of context-free streams in further detail. More on the coalgebraic approach to context-free streams and context-free languages can be found in [WBR11], [WBR13], [Win14], [WBR15], [BHKR15] and [HKR17]. □

Exercise 340 (square root of a stream). Rational streams can be represented as fractions of two polynomial streams (which was the way we actually defined them, in Definition 284). For context-free streams, there seems to be no such kind of general representation. For the specific example of the Catalan numbers, defined by SDE (12.51) above, there *does* exist a pleasant closed form in stream calculus. One can define the *square root* $\sqrt{\sigma}$ of a stream $\sigma \in \mathbb{R}^\omega$ by the following SDE:

$$\sqrt{\sigma}(0) \;=\; \sqrt{\sigma(0)} \qquad\qquad \left(\sqrt{\sigma}\right)' \;=\; \frac{\sigma'}{\sqrt{\sigma(0)} \,+\, \sqrt{\sigma}} \tag{12.53}$$

assuming that $\sigma(0) > 0$. Prove that

$$\sqrt{\sigma} \times \sqrt{\sigma} \;=\; \sigma$$

Compute $\sqrt{1 - 4\mathsf{X}}$. Prove that the stream

$$\frac{2}{1 \,+\, \sqrt{1 - 4\mathsf{X}}}$$

is a solution of SDE (12.51) and hence is equal to the stream of the Catalan numbers. □

12.9 Discussion

The present chapter is based on [Rut01, Rut05a, Rut08], which in turn were in part inspired by Conway's *differential calculus of events* [Con71, Chapter 5] and Pavlović and Escardó's [PE98]. Section 12.7, on the Newton transform, builds moreover on [BHPR17]. The Laplace transform in Definition 300 is also know as the *Laplace-Carson* transform ([BLL98, p.350] and [Com74, p.48]). See [Sch68] for a *finitary* version of the Euler expansion for streams, Theorem 326. For more on coalgebra and differential equations, see [Bor17].

Theorem 332 is a reformulation in terms of SDEs of a classical result in automata theory, called the theorem of Kleene, Schützenberger and Eilenberg [Eil76].

There exist further characterisations of rational streams in terms of so-called stream circuits, which are also known as signal flow graphs; see [Rut05b, NR13] for details.

The present calculus of streams can be widely generalised in two respects. First, if one replaces $\mathbb{R}$ by *any* field, or even any *ring*, or even any *semi*-ring B, then many of the results presented above still apply to the set B^ω of streams over B. Secondly, one can furthermore look at Moore automata with *inputs from a set* A and outputs in B:

$$\langle \mathsf{o}, \mathsf{tr} \rangle \colon S \to B \times S^A$$

We saw in Section 13.3 that Moore automata generalise both deterministic automata and stream systems. In Exercise 354, we showed that the set B^{A^*} is a *final* Moore automaton. If the set B of outputs is a semi-ring (or ring or field) then the elements of B^{A^*} are called *formal power series* or *weighted languages*. And again, much of the present stream calculus can be generalised to weighted languages. See [Rut05a], [Rut03a], and [BHPR17] for more details.

Chapter 13

Mealy automata

A *Mealy automaton* (MA) with inputs from a set A and outputs in a set B is a pair (S, α), consisting of a set S of states, and a structure map

$$\alpha \colon S \to (B \times S)^A$$

The function α maps any state $s \in S$ to a function

$$\alpha(s) \colon A \to (B \times S) \qquad\qquad \alpha(s)(a) = (b, t) \qquad\qquad (a \in A)$$

where the pair (b, t) consists of an output $b \in B$ and a next state $t \in S$. Mealy automata differ from deterministic automata in two ways: outputs belong to an (arbitrary) set B rather than the set 2; and outputs may in general depend on inputs.

We shall see that the behaviour of Mealy automata consists of *causal stream functions* rather than languages. The set Γ of all causal stream functions will be shown to be a final Mealy automaton, whose structure map will be defined in terms of *stream function derivatives*. We shall use these stream function derivatives to construct, for a given causal stream function, a minimal Mealy automaton *realising* it.

Remark 341. For those who have already read Chapters 2 to 5: Mealy automata are coalgebras for the functor

$$\mathsf{MA} \colon \mathsf{Set} \to \mathsf{Set} \qquad\qquad \mathsf{MA}(S) \;=\; (B \times S)^A \qquad\qquad (S \in \mathsf{Set})$$

which is defined for functions $f \colon S \to T$, $\phi \colon A \to (B \times S)$, and $a \in A$, by

$$\mathsf{MA}(f) \colon (B \times S)^A \to (B \times T)^A \qquad\qquad (\, \mathsf{MA}(f)(\phi)\,)\,(a) \;=\; (b, f(s))$$

where $(b, s) = \phi(a)$. $\square$

For a Mealy automaton (S, α), we will write

$$s \xrightarrow{a|b} t \quad\iff\quad \alpha(s)(a) = (b, t)$$

We define the (input-output) *behaviour* of a state $s \in S$ as the stream function

$$\mathsf{beh}_\alpha(s) \colon A^\omega \to B^\omega$$

that maps a stream of inputs $(a_0, a_1, a_2, \ldots) \in A^\omega$ to the stream $(b_0, b_1, b_2, \ldots) \in B^\omega$ of outputs that is given by the (uniquely defined) sequence of transitions

$$s \;=\; s_0 \xrightarrow{a_0|b_0} s_1 \xrightarrow{a_1|b_1} s_2 \xrightarrow{a_2|b_2} \cdots \qquad\qquad \alpha(s_i)(a_i) = (b_i, s_{i+1})$$

We say that $s \in S$ *realises* a stream function $f \colon A^\omega \to B^\omega$ if $\mathsf{beh}_\alpha(s) = f$.

Example 342 (binary Mealy automaton). If $A = B = 2 = \{0, 1\}$ then we call a Mealy automaton *binary*. Here is an example:

$$\begin{tikzcd}
s \arrow[loop above, "0|0"] \arrow[r, bend right, "1|1"] & t \arrow[loop above, "0|1"] \arrow[r, bend left, "1|0"] & u \arrow[l, bend left, "1|1"] \arrow[loop above, "0|0"]
\end{tikzcd}$$

The state s above realises the bitstream function

$$f \colon 2^\omega \to 2^\omega \qquad\qquad f(\sigma)(n) \;=\; \left(\sum_{k=0}^{n} \sigma(i) \right) \bmod 2 \qquad (13.1)$$

which counts the number of ones in the first $n+1$ inputs $\sigma(0), \ldots, \sigma(n)$, modulo 2. $\square$

13.1 Causal stream functions

The behaviour of Mealy automata consists of *causal* stream functions.

Definition 343 (causal stream functions). We define

$$\sigma \equiv_n \tau \quad\Longleftrightarrow\quad \sigma(0) = \tau(0), \ldots, \sigma(n) = \tau(n) \qquad\qquad (\sigma, \tau \in A^\omega,\ n \geqslant 0)$$

and call a stream function $f \colon A^\omega \to B^\omega$ *causal* if

$$\sigma \equiv_n \tau \quad\Longrightarrow\quad f(\sigma)(n) \;=\; f(\tau)(n)$$

Intuitively, a stream function is causal if the nth value of its result depends on only the first n values of its argument. $\square$

Exercise 344. Identify among the various examples we saw in Chapter 7 examples of both causal and non-causal stream functions. □

The set of all causal stream functions is denoted by

$$\Gamma \;=\; \{ f \mid f\colon A^\omega \to B^\omega \text{ and } f \text{ is causal } \} \tag{13.2}$$

It is straightforward to show that the behaviour function

$$\mathsf{beh}_\alpha(s)\colon A^\omega \to B^\omega$$

of any state s in a Mealy automaton (S, α) is causal. As a consequence, the behaviour function beh_α has type

$$\mathsf{beh}_\alpha\colon S \to \Gamma$$

Conversely, any causal stream function is realised by a state in a Mealy automaton. This will follow from the fact that the set Γ is a *final* Mealy automaton, which we show next. We will use the following notation:

$$a{:}\sigma \;=\; (a, \sigma(0), \sigma(1), \sigma(2), \ldots) \qquad\qquad (a \in A,\ \sigma \in A^\omega)$$

Definition 345 (stream function derivative). We turn the set Γ of all causal functions into a Mealy automaton

$$(\Gamma, \gamma) \qquad \gamma\colon \Gamma \to (B \times \Gamma)^A \qquad \gamma(f)(a) \;=\; (f[a],\ f_a) \qquad (f \in \Gamma,\ a \in A)$$

where $f[a]$ is the *initial output of f on input a*, defined by

$$f[a] \in B \qquad\qquad f[a] \;=\; f(a{:}\sigma)(0)$$

and f_a is the *stream function derivative of f on input a*, defined by

$$f_a\colon A^\omega \to B^\omega \qquad\qquad f_a(\sigma) \;=\; f(a{:}\sigma)'$$ □

Note that because f is causal, the value of $f[a]$ is independent of σ; furthermore, the function f_a is again causal and hence belongs to Γ.

Theorem 346 (finality – causal stream functions). *The Mealy automaton (Γ, γ) is* final*: for every Mealy automaton (S, α), the function $\mathsf{beh}_\alpha\colon S \to \Gamma$*

$$\begin{tikzcd}
S \arrow[r, dashed, "\mathsf{beh}_\alpha"] \arrow[d, "\forall\,\alpha"'] & \Gamma \arrow[d, "\gamma"] \\
(B \times S)^A \arrow[r, dashed, "(1 \times \mathsf{beh}_\alpha)^A"'] & (B \times \Gamma)^A
\end{tikzcd}$$

is the unique function making the diagram above commute. □

Exercise 347. Prove Theorem 346. Begin your proof by defining the notion of homomorphism of Mealy automata, either by an educated guess or by applying the categorical definition of F-homomorphism from Definition 21, to the functor MA defined in Remark 341. □

13.2 Constructing minimal Mealy automata

The notions of generated subautomaton, Definition 146, and of minimal automaton, Definition 178, generalise straightforwardly from the setting of deterministic automata, in Chapter 8, to Mealy automata. The same applies to Theorem 183, about the construction of a minimal deterministic automaton for a given language, which for causal functions and Mealy automata reads as follows.

Theorem 348 (minimal Mealy automaton for causal function). *For every causal stream function $f \in \Gamma$, the Mealy machine* generated by f*:*

$$\langle f \rangle \;=\; \{ f_w \mid w \in A^* \}$$

where

$$f_\varepsilon \;=\; f \qquad\qquad f_{wa} \;=\; (f_w)_a \qquad\qquad (w \in A^*)$$

is a minimal Mealy machine realising f.

Proof: It is straightforward to show that $\mathsf{beh}_\gamma \colon \Gamma \to \Gamma$ is the identity function. As a consequence, the (state f in the) generated Mealy automaton $\langle f \rangle \subseteq \Gamma$ realises the function f. The Mealy automaton $\langle f \rangle$ is minimal since the inclusion of Mealy automata always is an injective homomorphism. □

Example 349. We illustrate the notions and results above with the binary counter from Example 342. With f as defined in (13.1), we have $\mathsf{beh}(s) = f$. Defining

$$g \;=\; \mathsf{beh}(t) \qquad\qquad h \;=\; \mathsf{beh}(u)$$

one easily shows that

$$\mathsf{beh}(s) \;=\; f \;=\; h \;=\; \mathsf{beh}(u) \qquad\qquad \mathsf{beh}(t) \;=\; g \;=\; 1 - f$$

Moreover, we have the following initial outputs and stream function derivatives:

$$\begin{array}{llll} f[0] \;=\; 0 & f[1] \;=\; 1 & f_0 \;=\; f & f_1 \;=\; g \\ g[0] \;=\; 1 & g[1] \;=\; 0 & g_0 \;=\; g & g_1 \;=\; f \end{array}$$

We see that beh maps the Mealy automaton $\langle s\rangle$, on the left,

$$\begin{tikzcd}
s \arrow[loop above, "0|0"] \arrow[r, bend right, "1|1"] & t \arrow[loop above, "0|1"] \arrow[r, bend left, "1|0"] & u \arrow[l, bend left, "1|1"] \arrow[loop above, "0|0"] & {} \arrow[r, mapsto, "\mathsf{beh}"] & {} & f \arrow[loop above, "0|0"] \arrow[r, bend left, "1|1"] & g \arrow[l, bend left, "1|0"] \arrow[loop above, "0|1"]
\end{tikzcd}$$

to its minimisation $\langle f\rangle \subseteq \Gamma$, on the right. $\square$

Example 350. Let $g\colon A \to B$ be a function and let

$$f\colon A^\omega \to B^\omega \qquad f(\sigma) \;=\; (g(\sigma(0)),\, g(\sigma(1)),\, g(\sigma(2)),\, \ldots) \qquad (\sigma \in A^\omega)$$

The function f, which applies g elementwise, satisfies

$$f(\sigma)(0) \;=\; g(\sigma(0)) \qquad\qquad f(\sigma)' \;=\; f(\sigma')$$

For the construction of $\langle f\rangle$, we calculate

$$f[a] \;=\; f(a:\sigma)(0) \;=\; g(a) \qquad (a \in A,\, \sigma \in A^\omega)$$

and

$$f_a(\sigma) \;=\; f(a:\sigma)' \;=\; f((a:\sigma)') \;=\; f(\sigma) \qquad (a \in A,\, \sigma \in A^\omega)$$

As a consequence, we obtain the following one-state Mealy automaton

$$\langle f\rangle \;=\; \begin{tikzcd} f \arrow[loop above right, "a|g(a)"] \end{tikzcd}$$

with a transition from f to f for every $a \in A$. $\square$

Exercise 351 (an infinite Mealy automaton). Stream functions are generally realised by Mealy automata that have *infinitely* many states. For instance, let $A = B = \mathbb{N}$ and consider the function

$$f\colon \mathbb{N}^\omega \to \mathbb{N}^\omega \qquad f(\sigma)(n) \;=\; \sum_{k=0}^{n} \sigma(k) \qquad (\sigma \in \mathbb{N}^\omega,\, n \geqslant 0)$$

Prove the following identities: for all $n, m \geqslant 0$ and $\sigma \in \mathbb{N}^\omega$,

$$\begin{aligned} f[n] \;&=\; n & f_n(\sigma) \;&=\; f(\sigma) + \overline{n} \\ f_n[m] \;&=\; n+m & (f_n)_m(\sigma) \;&=\; f_{n+m}(\sigma) \end{aligned}$$

where + denotes elementwise addition of streams and $\overline{n} = (n, n, n, \ldots)$, and conclude that

$$\begin{tikzcd}
\langle f \rangle \; = & f \arrow[loop left, "0|0"] \arrow[r, "1|1"] \arrow[rr, bend right, "2|2"'] & f_1 \arrow[loop above, "0|1"] \arrow[r, "1|2"] \arrow[rr, bend left, "2|3"] & f_2 \arrow[loop below, "0|2"] \arrow[r, "1|3"] & \cdots
\end{tikzcd}$$

is a minimal Mealy automaton realising f. □

Exercise 352. Recall the function $\mathsf{zip}\colon (A^\omega \times A^\omega) \to A^\omega$ from Section 7.3, satisfying

$$\mathsf{zip}(\sigma, \tau) \;=\; (\sigma(0), \tau(0), \sigma(1), \tau(1), \ldots)$$

for all $\sigma, \tau \in A^\omega$. Construct a minimal Mealy automaton realising zip. □

13.3 Moore automata

Here is a minor variation on the notion of Mealy automaton. A *Moore automaton* with inputs from a set A and outputs in a set B is a pair

$$(S, \langle \mathsf{o}, \mathsf{tr} \rangle) \qquad \langle \mathsf{o}, \mathsf{tr} \rangle\colon S \to B \times S^A \qquad \mathsf{o}\colon S \to B \qquad \mathsf{tr}\colon S \to S^A$$

The difference with Mealy automata is that in a Moore automaton, outputs depend only on the present state and not on the present input.

Remark 353. For those who have already read Chapters 2 to 5: Moore automata are coalgebras for the functor

$$\begin{array}{lll}
\mathsf{MoA}\colon \mathsf{Set} \to \mathsf{Set} & \mathsf{MoA}(S) \;=\; B \times S^A & (S \in \mathsf{Set}) \\
\mathsf{MoA}(f)\colon B \times S^A \to B \times T^A & \mathsf{MoA}(f) \;=\; 1_B \times f^A & (f\colon S \to T)
\end{array}$$

where 1_B is the identity function on B and the function $f^A\colon S^A \to T^A$ is as before. □

Moore automata generalise deterministic automata with inputs from A and outputs in 2, which we find as a special case by simply taking $B = 2$. Moore automata also generalise stream systems: if we take $A = 1$, the one-element set, then

$$B \times S^A \;=\; B \times S^1 \;=\; B \times S$$

Thus Moore automata with inputs from 1 and outputs in B are stream systems with outputs in B. All of the theory of automata and stream systems straightforwardly generalises to Moore automata.

Exercise 354 (formal power series aka weighted languages). We saw in Section 8.3 that the set $\mathcal{P}(A^*) \cong 2^{A^*}$ of all languages over a set A is a final automaton. Prove that the set

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

can be turned into a final Moore automaton. If B is a semi-ring or a field, then such functions $f\colon A^* \to B$ are called *formal power series*, also known as *weighted languages*. $\square$

Exercise 355 (an alternative final Mealy automaton). Prove that the set Γ of causal stream functions, defined in (13.2), is isomorphic to the set

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

where A^+ is the set of non-empty words. Show that, as a consequence, this set can be turned into a final Mealy automaton. $\square$

Exercise 356 (relating Mealy and Moore automata). Show that every Mealy automaton can be transformed into a Moore automaton and vice versa. For those with a background in category theory: show that these transformations form an adjunction between the categories of Mealy and Moore automata. $\square$

13.4 A calculus of bitstreams

A *bitstream* is a stream over the set $2 = \{0, 1\}$ of Booleans, that is, an element

$$\sigma \in 2^\omega$$

We shall use bitstreams as representations of (certain) rational numbers (by taking infinitary base-2 expansions, as we shall see below). Having bitstreams represent rational numbers, one can then use *bitstream functions*

$$f\colon 2^\omega \to 2^\omega$$

to represent (certain) functions on rational numbers. In the present section, we shall introduce various operators on 2^ω, similar to the stream calculus of Chapter 12, with which such functions can be expressed. In Section 13.5, we shall then show how to generate from (an expression for) a bitstream function f a Mealy machine that realises f.

We begin by showing how the set of rational numbers with odd denominator

$$\mathbb{Q}_{odd} = \{ p/q \mid p, q \in \mathbb{Z},\ q \text{ is odd } \}$$

can be included into 2^ω by taking infinitary base-2 expansions.

Definition 357 (binary representations of rational numbers). The function

$$\mathsf{bin}\colon \mathbb{Q}_{odd} \to 2^\omega$$

is defined by the following system of stream differential equations:

$$\mathsf{bin}(q)(0) \;=\; odd(q) \qquad \mathsf{bin}(q)' \;=\; \mathsf{bin}\left(\frac{q - odd(q)}{2}\right) \qquad (q \in \mathbb{Q}_{odd})$$

where $odd(n/2m+1) = n \bmod 2$, for $n, m \in \mathbb{Z}$. □

For a positive integer $n \in \mathbb{N}$, the stream $\mathsf{bin}(n)$ is just the binary representation of n (least significant bit on the left) padded with a tail of zeros; for instance,

$$\begin{aligned}\mathsf{bin}(2) &= (0,1,0,0,0,\ldots) = 01(0)^\omega\\ \mathsf{bin}(5) &= (1,0,1,0,0,\ldots) = 101(0)^\omega\end{aligned}$$

Binary representations of negative integers end with an infinite sequence of ones and rational numbers (with odd denominator) have binary representations that are eventually periodic; for instance,

$$\begin{aligned}\mathsf{bin}(-1) &= (1,1,1,1,\ldots) = 1^\omega\\ \mathsf{bin}(-5) &= (1,1,0,1,1,1,\ldots) = 110(1)^\omega\\ \mathsf{bin}(1/5) &= (1,0,1,1,0,0,1,1,0,0,1,1,0,\ldots) = 1(0110)^\omega\end{aligned}$$

(To represent rationals with *even* denominator, one needs *pairs* consisting of a finite sequence in 2^* together with a bitstream or, equivalently, formal power series of the form $\sum_{i=k}^{\infty} a_i 2^i$ where $k < 0$; they are not treated here.)

We recall that the set of rational numbers (with odd denominator) is an integral domain (see Section 16.7 for the definition). Next we shall turn the set 2^ω into an integral domain as well, by introducing operations of addition, multiplication, minus and inverse. But first we present two examples illustrating how addition and multiplication are typically defined in the literature: for addition,

$$\begin{array}{rcr} {\scriptstyle 1\;1\;1\;\cdots} & & \text{(carry)}\\ 1\,0\,1\,0\,0\,0\,\cdots & = & 5\\ +\,0\,1\,1\,1\,1\,1\,\cdots & = & -2\\ \hline 1\,1\,0\,0\,0\,0\,\cdots & = & 3\end{array}$$

and for multiplication,

$$\begin{array}{lr} 1\,1\,0\,0\,0\,\cdots\;\times\;0\,1\,1\,1\,1\,\cdots & =\;3\times(-2)\\ \hline 0\,0\,0\,0\,0\,0\,\cdots & \\ \quad 1\,1\,0\,0\,0\,\cdots & \\ \qquad 1\,1\,0\,0\,0\,\cdots & \\ \qquad\quad 1\,1\,0\,0\,0\,\cdots & \\ \qquad\qquad 1\,1\,0\,0\,0\,\cdots & \\ +\qquad\qquad \vdots & \\ \hline 0\,1\,0\,1\,1\,1\,\cdots & =\;-6\end{array}$$

In contrast, we shall define these operators, as usual, by stream differential equations. First, we recall the definitions of the Boolean operators *and* and *exclusive or* :

$$\begin{array}{ll} \wedge\colon 2\times 2\to 2 & \oplus\colon 2\times 2\to 2 \\ a\wedge b \;=\; \min\{a,b\} & a\oplus b \;=\; 1 \iff (\,a=0,\, b=1\,) \text{ or } (\,a=1,\, b=0\,) \end{array}$$

Definition 358 (operators on bitstreams). We define the operations of *sum*, *minus*, *product* and *inverse* :

$$+\colon\; 2^{\omega}\times 2^{\omega}\to 2^{\omega} \qquad -\colon\; 2^{\omega}\to 2^{\omega} \qquad \times\colon\; 2^{\omega}\times 2^{\omega}\to 2^{\omega} \qquad 1/(-)\colon\; 2^{\omega}\to 2^{\omega}$$

together with a function $[-]\colon 2\to 2^{\omega}$ that embeds 2 into the set 2^{ω}, by means of the following system of stream differential equations: for $a\in 2$, $\alpha,\beta\in 2^{\omega}$,

$$\begin{array}{rclrcll} [a](0) &=& a & [a]' &=& [0] & \\ (\alpha+\beta)(0) &=& \alpha(0)\oplus\beta(0) & (\alpha+\beta)' &=& (\alpha'+\beta') + [\,\alpha(0)\wedge\beta(0)\,] & \\ (-\alpha)(0) &=& \alpha(0) & (-\alpha)' &=& -(\alpha' + [\alpha(0)]) & \\ (\alpha\times\beta)(0) &=& \alpha(0)\wedge\beta(0) & (\alpha\times\beta)' &=& (\alpha'\times\beta) + (\,[\alpha(0)]\times\beta'\,) & \\ (1/\alpha)(0) &=& 1 & (1/\alpha)' &=& -\,(\,\alpha'\times(1/\alpha)\,) & (\alpha(0)=1) \end{array}$$

Since the above system of SDEs is in GSOS format, the unique existence of these operators follows from Theorem 252. □

We briefly explain the intuition behind the equations above. As before, we have

$$[a] \;=\; (a,\, 0,\, 0,\, 0,\, \ldots)$$

which we shall, as before adhering to Notation 264, often simply write as

$$a \;=\; [a]$$

The equation for sum shows that a carry term must be added in case the two initial values are both 1. The equation for minus can be derived from the requirement that

$$\alpha \,+\, (-\alpha) \;=\; [0]$$

The equation for the product states that $\alpha\times\beta$ can be calculated using the base-2 version of shift-add-multiplication known from the multiplication in decimal notation. The equation for $1/\alpha$ can be derived from the requirement that

$$\alpha \,\times\, (1/\alpha) \;=\; [1]$$

which will also explain the condition that $\alpha(0)=1$.

Thus we have turned 2^{ω} into an integral domain

$$(\,2^{\omega},\, +,\, -,\, \times,\, /,\, [0],\, [1]\,)$$

Moreover, these operations have been devised in such a way that $\mathsf{bin}\colon \mathbb{Q}_{odd}\to 2^{\omega}$ is a homomorphism of integral domains.

Proposition 359 (a homomorphism of integral domains). The function

$$\mathsf{bin} : \mathbb{Q}_{odd} \to 2^\omega$$

satisfies, for all $p, q \in \mathbb{Q}_{odd}$,

$$\begin{aligned} \mathsf{bin}(p+q) &= \mathsf{bin}(p) + \mathsf{bin}(q) & \qquad \mathsf{bin}(-q) &= -\mathsf{bin}(q) \\ \mathsf{bin}(p \times q) &= \mathsf{bin}(p) \times \mathsf{bin}(q) & \qquad \mathsf{bin}(1/q) &= 1/\mathsf{bin}(q) \end{aligned}$$

Exercise 360. Prove Proposition 359 by coinduction up-to. □

We recall the constant stream $\mathsf{X} \in \mathbb{R}^\omega$ from Definition 267, which plays a central role in the calculus of streams in Chapter 12. Using the same symbol, we (re)define

$$\mathsf{X} \in 2^\omega \qquad\qquad \mathsf{X} = (0, 1, 0, 0, 0, \ldots)$$

and observe that

$$\mathsf{X} = \mathsf{bin}(2)$$

As before, multiplying a stream σ by the stream X can be seen as an elementary form of stream *integration*, giving us a fundamental theorem of bitstream calculus, similar to Theorem 269: for $\alpha \in 2^\omega$,

$$\alpha = \alpha(0) + (\mathsf{X} \times \alpha') \tag{13.3}$$

Also the following identity will be useful in what follows:

$$\alpha + \alpha = \mathsf{X} \times \alpha \tag{13.4}$$

Exercise 361. Prove identity 13.4 by coinduction up-to. □

Example 362. We shall use the calculus above to compute (again) the values of $\mathsf{bin}(5)$, $\mathsf{bin}(-5)$, and $\mathsf{bin}(1/5)$. For integers, things are straightforward:

$$\mathsf{bin}(5) = \mathsf{bin}(1 + 2^2) = \mathsf{bin}(1) + \mathsf{bin}(2)^2 = 1 + X^2$$

Similarly, we have

$$\mathsf{bin}(-5) = -\mathsf{bin}(5) = -1 - X^2 = 1 + \mathsf{X} - X^3 = 110(1)^\omega$$

where we have used

$$-\mathsf{X}^k = \mathsf{X}^k - \mathsf{X}^{k+1} \qquad\qquad \mathsf{bin}(-1) = (1, 1, 1, \ldots) = (1)^\omega$$

For

$$\mathsf{bin}(1/5) \;=\; 1/\mathsf{bin}(5) \;=\; 1/(1+X^2)$$

we shall compute the respective derivatives of the latter rational expression. Using the defining stream differential equations from Definition 358, as well as the following equalities:

$$\begin{aligned}(1\,+\,\mathsf{X}^2)' \;&=\; 1' \,+\, (\mathsf{X}^2)' \,+\, [1(0) \wedge \mathsf{X}^2(0)]\\ &=\; 0 \,+ \mathsf{X} \,+\, 0\\ &=\; \mathsf{X}\\ (-1\,-\,\mathsf{X})' \;&=\; (-1)' \,+\, (-\mathsf{X})' \,+\, [-1(0) \wedge -\mathsf{X}(0)]\\ &=\; (-1) \,+ (-1) \,+\, 0\\ &=\; -\mathsf{X}\end{aligned}$$

one obtains

$$\begin{aligned}(1/1+\mathsf{X}^2)' \;&=\; -\mathsf{X}/1+\mathsf{X}^2\\ (-\mathsf{X}/1+\mathsf{X}^2)' \;&=\; -1/1+\mathsf{X}^2\\ (-1/1+\mathsf{X}^2)' \;&=\; (-1-\mathsf{X})/1+\mathsf{X}^2\\ ((-1-\mathsf{X})/1+\mathsf{X}^2)' \;&=\; -\mathsf{X}^2/1+\mathsf{X}^2\\ (-\mathsf{X}^2/1+\mathsf{X}^2)' \;&=\; -\mathsf{X}/1+\mathsf{X}^2\end{aligned}$$

These are all the different stream derivatives of $1/1+\mathsf{X}^2$. Using Lemma 77,

$$\alpha(n) \;=\; \alpha^{(n)}(0) \qquad\qquad (\alpha \in 2^\omega)$$

one then finds

$$\mathsf{bin}(1/5) \;=\; 1(0110)^\omega$$ □

Exercise 363. Prove, more generally, that rational bitstreams have only finitely many different derivatives and that, as a consequence, the binary representations of rational numbers are eventually periodic. □

13.5 Mealy machines for bitstream functions

Next we shall construct for various bitstream functions $f\colon 2^\omega \to 2^\omega$ a minimal Mealy machine realising it. According to Theorem 348, the states of a minimal Mealy machine for f are given by

$$\langle f\rangle \;=\; \{f_w \mid w \in 2^*\}$$

that is, the set of all repeated stream function derivatives of f. We shall use the following identities:

$$0 : \alpha \;=\; \mathsf{X} \times \alpha \qquad\qquad 1 : \alpha \;=\; 1 + (\mathsf{X} \times \alpha) \qquad\qquad (\alpha \in 2^\omega)$$

With this notation, the stream function derivatives for a function $f\colon 2^\omega \to 2^\omega$ are

$$\begin{aligned} f_0(\alpha) &= f(0:\alpha)' & \qquad f_1(\alpha) &= f(1:\alpha)' \\ &= f(\mathsf{X}\times\alpha)' & &= f(1 + (\mathsf{X}\times\alpha))' \end{aligned}$$

For a first example, we consider the function

$$f\colon 2^\omega \to 2^\omega \qquad\qquad f(\alpha) \;=\; (1+\mathsf{X}) \times \alpha$$

The function f multiplies a stream α, which could represent a rational number $q \in \mathbb{Q}_{odd}$ with $\alpha = \mathsf{bin}(q)$, with $(1+\mathsf{X}) = \mathsf{bin}(3)$, the binary representation of the number 3. We compute all of the repeated stream function derivatives of f:

$$\begin{aligned} f_0(\alpha) &= f(\mathsf{X}\times\alpha)' \\ &= (\,(1+\mathsf{X}) \times (\mathsf{X}\times\alpha)\,)' \\ &= (\,(1+\mathsf{X})' \times (\mathsf{X}\times\alpha)\,) + (\mathsf{X}\times\alpha)' && \text{(using that } (1+\mathsf{X})(0)=1\text{)} \\ &= (\mathsf{X}\times\alpha) + \alpha \\ &= f(\alpha) \\ f_1(\alpha) &= f(1 + (\mathsf{X}\times\alpha))' \\ &= (\,(1+\mathsf{X}) \times (1+(\mathsf{X}\times\alpha))\,)' \\ &= (\,(1+\mathsf{X})' \times (1+(\mathsf{X}\times\alpha))\,) + (1+(\mathsf{X}\times\alpha))' && \text{(using that } (1+\mathsf{X})(0)=1\text{)} \\ &= 1 + (\mathsf{X}\times\alpha) + \alpha \\ &= f(\alpha) + 1 \end{aligned}$$

Continuing with the stream function derivatives of f_1, we obtain, by similar computations,

$$f_{10}(\alpha) = f(\alpha) \qquad f_{11}(\alpha) = f(\alpha) + \mathsf{X} \qquad f_{110}(\alpha) = f_1(\alpha) \qquad f_{111}(\alpha) = f_{11}(\alpha)$$

It follows that

$$\langle f \rangle \;=\; \{f,\, f_1,\, f_{11}\}$$

Computing initial outputs,

$$\begin{aligned} f[0] &= 0 & \qquad f_1[0] &= 1 & \qquad f_{11}[0] &= 0 \\ f[1] &= 1 & f_1[1] &= 0 & f_{11}[1] &= 1 \end{aligned}$$

we obtain the following minimal Mealy machine realising f:

$$\begin{tikzcd} f \arrow[loop left, "0|0"] \arrow[r, bend left, "1|1"] & f_1 \arrow[l, bend left, "0|1"] \arrow[r, bend left, "1|0"] & f_{11} \arrow[l, bend left, "0|0"] \arrow[loop right, "1|1"] \end{tikzcd}$$

For a second example, let

$$g\colon 2^\omega \to 2^\omega \qquad g(\alpha) \;=\; \frac{-\mathsf{X} + ((1+\mathsf{X}) \times \alpha)}{1+\mathsf{X}^3} \;=\; \frac{\mathsf{bin}(-2) + (\mathsf{bin}(3) \times \alpha)}{\mathsf{bin}(9)}$$

For notational convenience, we will leave out the symbol bin and simply write

$$g(\alpha) \;=\; \frac{-2 + (3 \times \alpha)}{9} \;=\; (-2/9) \,+\, (3 \times \alpha)/9$$

Computing the first two stream function derivatives, one finds

$$g_0(\alpha) \;=\; (-1/9) \,+\, (3 \times \alpha)/9 \qquad\qquad g_1(\alpha) \;=\; (-4/9) \,+\, (3 \times \alpha)/9$$

As it turns out, every stream function derivative of g is of the form

$$g_w(\alpha) \;=\; (d(w)/9) \,+\, (3 \times \alpha)/9 \qquad\qquad (d(w) \in \{1, 2, -1, -2, -4, -5, -7, -8\})$$

Computing initial outputs and denoting g_w simply by the value of $d(w)$, we obtain

$$\begin{tikzcd}
1 \arrow[r, "0|1"'] \arrow[dd, bend left, "1|0"] & -4 \arrow[dd, bend left, "0|0"] \arrow[drr, bend left, "1|1"] & -8 \arrow[l, "0|0"'] \arrow[dd, bend left, "1|1"] & & \\
 & & & -5 \arrow[r, bend left, "1|0"] \arrow[dl, bend left, "0|1"] & -1 \arrow[ullll, bend right, "1|0"'] \arrow[l, bend left, "0|1"] \\
2 \arrow[uu, bend left, "0|0"] \arrow[r, "1|1"] & g \arrow[uu, bend left, "1|1"] \arrow[urrr, bend right, "0|1"'] & -7 \arrow[l, "1|0"'] \arrow[uu, bend left, "0|1"] & &
\end{tikzcd}$$

as a minimal Mealy machine representing g.

The examples above generalise as follows.

Proposition 364 (rational bitstream functions). A bitstream function $f\colon 2^\omega \to 2^\omega$ is *rational* if it is of the form

$$f(\alpha) \;=\; \frac{d + (m \times \alpha)}{n}$$

with $d, m, n \in \mathbb{Z}$, n odd. Its stream function derivatives are of the form

$$f_w(\alpha) \;=\; \frac{d(w) + (m \times \alpha)}{n} \qquad\qquad (w \in 2^*,\ \alpha \in 2^\omega)$$

where $d(w) \in \mathbb{Z}$ satisfies

$$\min\{\, d,\ -n+1,\ -n+m+1 \,\} \;\leqslant\; d(w) \;\leqslant\; \max\{\, d,\ m-1,\ 0 \,\} \qquad \square$$

It follows from Proposition 364 that rational bitstream functions are realised by finite Mealy machines.

Exercise 365. Prove Proposition 364 (possibly consulting [HR10]). □

Exercise 366. Show that the following Mealy machine

$$\begin{tikzcd}
 & & & 1 \arrow[ddlll, bend right, "0|1"'] \arrow[ddll, bend right, "1|1"] \\
 & & 2 \arrow[ur, "0|0"] \arrow[r, "1|0"] & 5 \arrow[l, bend left, "0|1"] \arrow[ddl, "1|1"'] \\
h \arrow[loop below, "0|0"] \arrow[r, "1|0"] & 4 \arrow[ur, "0|0"] \arrow[dr, "1|0"] & & \\
 & & 6 \arrow[r, "0|0"] \arrow[dr, "1|0"'] & 3 \arrow[uu, "1|1"] \arrow[uuu, bend right, "0|1"'] \\
 & & & 7 \arrow[loop below, "1|1"] \arrow[u, "0|1"]
\end{tikzcd}$$

is a minimal realisation for the function

$$h\colon 2^\omega \to 2^\omega \qquad\qquad h(\alpha) \;=\; \mathsf{X}^3 \times \alpha \;=\; 8 \times \alpha$$ □

Exercise 367 (fundamental theorem for binary stream functions). According to Definition 345, there is the following final coalgebra structure on the set Γ of all causal bitstream functions:

$$(\Gamma, \gamma) \qquad \gamma\colon \Gamma \to (2 \times \Gamma)^2 \qquad \gamma(f)(a) \;=\; (f[a],\, f_a) \qquad (f \in \Gamma,\, a \in 2)$$

Prove the following identity, for all causal functions $f\colon 2^\omega \to 2^\omega$:

$$f(\alpha) \;=\; (1 \oplus \alpha(0)) \times (\, f[0] + (X \times f_0(\alpha'))\,) \;+\; \alpha(0) \times (\, f[1] + (X \times f_1(\alpha'))\,)$$

This identity is similar to the Fundamental Theorem 269 of stream calculus and follows from the fact that γ is an isomorphism. □

Exercise 368. Let $f\colon 2^\omega \to 2^\omega$ be a function such that

$$f[0] \;=\; 0 \qquad f[1] \;=\; 1 \qquad f_0(\alpha) \;=\; (1+\mathsf{X}) \times \alpha \qquad f_1(\alpha) \;=\; 1 + \alpha + (\mathsf{X} \times \alpha)$$

for all $\alpha \in 2^\omega$. Use the identity from Exercise 367 to prove that

$$f(\alpha) \;=\; (1+\mathsf{X}) \times \alpha \qquad \square$$

Exercise 369. Compute a minimal Mealy machine realising

$$k\colon 2^\omega \to 2^\omega \qquad\qquad k(\alpha) \;=\; -\alpha$$

Show that there is *no* finite realisation of the function

$$l\colon 2^\omega \to 2^\omega \qquad\qquad l(\alpha) \;=\; \alpha \times \alpha \qquad \square$$

13.6 Discussion

We refer to [Ran58] and [Eil74] for a classical treatment of Mealy automata and their minimisation. Our coalgebraic presentation of Mealy automata is based on [Rut06] and [HR10], where the notion of stream function derivative, Definition 345 above, and its use in the description of minimal Mealy automata, in Theorem 348, were introduced.

In their role of representations of rational numbers, bitstreams are sometimes referred to as *2-adic integers*, see [Gou93]. More on infinitary base-2 expansions can be found in [HH79]. In [HCR06, HR10], an algorithm is presented for the symbolic computation of minimal Mealy machines for rational functions, as well as a characterisation of its complexity and an implementation in Haskell.

Chapter 14

Weighted stream automata

Weighted stream automata generalise the stream systems from Chapter 7, by allowing states to have multiple transitions, rather than precisely one. Weighted stream automata are, as it were, non-deterministic versions of stream systems. As a consequence, streams can often be represented by a weighted stream automaton that is dramatically smaller than a corresponding stream system. Each transition will have a *weight*, which in the present chapter will be a real number but which, more generally, could be a value in an arbitrary field (or even a ring or semi-ring). Computationally, these weights may have various interpretations, such as cost, duration, multiplicity etc.

The behaviour of weighted stream automata will be defined by transforming them into stream systems, by generalising the powerset construction (Definition 221 from Section 10.1), which was used to transform non-deterministic automata into deterministic ones. The resulting stream systems will be *linear*, which means that their state space consists of a *vector space* over $\mathbb{R}$. This opens the way for enhancements of the coinduction proof method by using so-called bisimulations *up-to-linearity*. Furthermore, we shall characterise the behaviour of *finite dimensional* linear stream systems in terms of *rational streams* (which were introduced in Section 12.2, Definition 284).

14.1 Representing streams by weighted automata

Formally, a *weighted stream automaton* with outputs and weights in $\mathbb{R}$ is a pair

$$(S,\ \langle \mathsf{o}, \mathsf{tr}\rangle) \qquad \langle \mathsf{o}, \mathsf{tr}\rangle \colon S \to \mathbb{R} \times \mathbb{R}^S_\omega \qquad \mathsf{o}\colon S \to \mathbb{R} \qquad \mathsf{tr}\colon S \to \mathbb{R}^S_\omega$$

Here $\mathbb{R}^S_\omega$ is the set of functions from S to $\mathbb{R}$ of *finite support*, defined by

$$\mathbb{R}^S_\omega \;=\; \{\,\phi\colon S \to \mathbb{R} \mid \mathsf{supp}(\phi) \text{ is finite}\} \qquad \mathsf{supp}(\phi) \;=\; \{\, s \in S \mid \phi(s) \neq 0\,\} \tag{14.1}$$

The output function o maps every state $s \in S$ to an output $\mathsf{o}(s) \in \mathbb{R}$. The transition function tr assigns to every state $s \in S$ a function $\mathsf{tr}(s)\colon S \to \mathbb{R}$ that maps a state $t \in S$ to $\mathsf{tr}(s)(t) \in \mathbb{R}$. We call $\mathsf{tr}(s)(t)$ the *weight* of the transition from s to t. If $\mathsf{tr}(s)(t) = 0$

then there is no transition from s to t. Since $\mathsf{tr}(s)$ has finite support, there are only finitely many transitions leaving from s.

In Section 14.6 below, we shall briefly mention weighted automata *with inputs*, which can be viewed as a common generalisation of both stream systems and deterministic and non-deterministic automata. There we shall also discuss the coalgebraic type of weighted stream automata and of weighted automata with inputs.

For $s, t \in S$ in a weighted stream automaton $(S,\ \langle \mathsf{o}, \mathsf{tr} \rangle)$, and for $k \in \mathbb{R}$, we will write

$$s|k \quad \Longleftrightarrow \quad \mathsf{o}(s) = k \qquad\qquad s \xrightarrow{k} t \quad \Longleftrightarrow \quad \mathsf{tr}(s)(t) = k$$

In pictures, we will often omit transitions with weight 0.

As we mentioned above, the behaviour of weighted automata will be defined by transforming them into stream systems. However, here is already a first, elementary and operationally intuitive definition of the behaviour of weighted stream automata.

Definition 370 (stream behaviour of weighted stream automaton). Let $(S,\ \langle \mathsf{o}, \mathsf{tr} \rangle)$ be a weighted stream automaton. We define the function $\mathsf{str}\colon S \to \mathbb{R}^\omega$, for all $s \in S$ and all $n \geqslant 0$, by

$$\mathsf{str}(s)(n) \;=\; \sum \{\, k_0 \times \cdots \times k_{n-1} \times k \;\mid\; s = s_0 \xrightarrow{k_0} \cdots \xrightarrow{k_{n-1}} s_n|k \,\}$$

We call $\mathsf{str}(s)$ *the stream represented by* s. □

The nth value of the stream $\mathsf{str}(s)$ is obtained by considering all transition sequences of length n starting in the state s; for each such sequence, one multiplies the weights of all the transitions in the sequence with the output value of the last state of the sequence; then one adds up the numbers thus obtained for all transition sequences. Note that only finitely many of these numbers will be different from 0 since $\mathsf{tr}(s)\colon S \to \mathbb{R}$ has finite support. As a consequence, the sum in Definition 370 is well-defined.

Example 371. Consider the following weighted stream automaton:

$$\begin{tikzcd}
s|1 \arrow[loop left, "0"] \arrow[r, bend left, "1"] & t|2 \arrow[loop right, "2"] \arrow[l, bend left, "-1"] & u|1 \arrow[loop left, "1"] \arrow[r, bend left, "1"] & v|1 \arrow[loop right, "1"] \arrow[l, bend left, "0"] & (S = \{s, t, u, v\})
\end{tikzcd}$$

By counting finite transition sequences with infinite patience, we find

$$\begin{aligned} \mathsf{str}(s) &= (1, 2, 3, \ldots) & \qquad \mathsf{str}(u) &= (1, 2, 3, \ldots) \\ \mathsf{str}(t) &= (2, 3, 4 \ldots) & \qquad \mathsf{str}(v) &= (1, 1, 1, \ldots) \end{aligned}$$

according to Definition 370. Here are two examples of infinite weighted automata:

$$\begin{tikzcd}
s_0|1 \arrow[loop left, "1"] \arrow[r, bend left, "1"] & s_1|0 \arrow[loop above, "3"] \arrow[l, bend left, "1"] \arrow[r, bend left, "2"] & s_2|0 \arrow[loop above, "5"] \arrow[l, bend left, "2"] \arrow[r, bend left, "3"] & \cdots \arrow[l, bend left, "3"]
\end{tikzcd}$$

$$\begin{tikzcd}
t_0|1 \arrow[loop left, "1"] \arrow[r, bend left, "1"] & t_1|0 \arrow[loop above, "2"] \arrow[l, bend left, "1"] \arrow[r, bend left, "1"] & t_2|0 \arrow[loop above, "2"] \arrow[l, bend left, "1"] \arrow[r, bend left, "1"] & \cdots \arrow[l, bend left, "1"]
\end{tikzcd}$$

where we have left out all transitions with weight 0. Applying Definition 370, we find

$$\mathsf{str}(s_0) \;=\; (0!, 1!, 2!, \ldots) \qquad\qquad \mathsf{str}(t_0) \;=\; (1, 1, 2, 5, 14, 42, 132, \ldots)$$

which are the streams of the factorial and the *Catalan* numbers [OEI, A000108]. □

Exercise 372. Consider the following weighted stream automaton:

$$\begin{tikzcd}
u_1|0 \arrow[loop left, "0"] \arrow[r, bend left, "1"] & u_2|1 \arrow[loop right, "1"] \arrow[l, bend left, "1"]
\end{tikzcd}$$

Which streams are represented by u_1 and by u_2? □

14.2 Splitting stream derivatives

Splitting the derivative of a stream $\sigma \in \mathbb{R}^\omega$ into two (or more) summands gives a method for constructing a weighted stream automaton representing σ: if

$$\sigma(0) \;=\; s \qquad\qquad \sigma' \;=\; (t \times \tau) + (r \times \rho) \qquad\qquad (\sigma, \tau, \rho \in \mathbb{R}^\omega,\ s, t, r \in \mathbb{R})$$

then

$$\begin{tikzcd}
 & \sigma \mid s \arrow[dl, bend right, "t"'] \arrow[dr, bend left, "r"] & \\
\tau & & \rho
\end{tikzcd}$$

defines part of a weighted stream automaton for σ. Continuing the construction of the automaton in the same way for τ and ρ, one obtains in the end an automaton of the following kind.

Definition 373 (semantic automata). A weighted stream automaton

$$(S,\ \langle \mathsf{o}, \mathsf{tr} \rangle) \qquad \langle \mathsf{o}, \mathsf{tr} \rangle \colon S \to \mathbb{R} \times \mathbb{R}^S_\omega \qquad \mathsf{o} \colon S \to \mathbb{R} \qquad \mathsf{tr} \colon S \to \mathbb{R}^S_\omega$$

is called *semantic* if $S \subseteq \mathbb{R}^\omega$ and, for all $\sigma \in S$,

$$\sigma(0) \ = \ \mathsf{o}(\sigma) \qquad \sigma' \ = \ \sum_{\tau \in S} \mathsf{tr}(\sigma)(\tau) \times \tau$$ □

One possible justification of the term *semantic* lies in the fact that in a semantic automaton, states coincide with their behaviour.

Proposition 374. If $(S,\ \langle \mathsf{o}, \mathsf{tr} \rangle)$ is semantic then, for all $\sigma \in S$,

$$\mathsf{str}(\sigma) \ = \ \sigma$$ □

Exercise 375. Prove Proposition 374. □

Example 376 (examples of semantic weighted stream automata). Every stream $\sigma \in \mathbb{R}^\omega$ trivially gives rise to the following semantic weighted stream automaton:

$$\begin{tikzcd} \sigma \mid \sigma(0) \arrow[r, "1"] & \sigma' \mid \sigma(1) \arrow[r, "1"] & \sigma'' \mid \sigma(2) \arrow[r, "1"] & \cdots \end{tikzcd}$$

Streams can be represented by infinitely many different semantic automata. For instance, recalling our notation for constant streams:

$$\overline{r} \ = \ (r, r, r, \ldots) \qquad (r \in \mathbb{R})$$

all of the following automata represent the stream of ones:

$$\begin{tikzcd} \overline{1} \mid 1 \arrow[r, "1"] & \overline{1} \mid 1 \arrow[r, "1"] & \overline{1} \mid 1 \arrow[r, "1"] & \cdots \end{tikzcd} \qquad \begin{tikzcd} \overline{1} \mid 1 \arrow[loop above, "1"] \end{tikzcd}$$

$$\begin{tikzcd} \overline{1} \mid 1 \arrow[r, bend left, "1/r"] & \overline{r} \mid r \arrow[l, bend left, "r"] \end{tikzcd} \qquad \begin{tikzcd} & \overline{1} \mid 1 \arrow[dl, bend right, "1"'] \arrow[dr, bend left, "2"] & \\ \overline{1/2} \mid 1/2 \arrow[rr, bend left, "2"] & & \overline{1/4} \mid 1/4 \arrow[ll, bend left, "1/2"] \end{tikzcd}$$

where $r \neq 0$. For the stream of positive natural numbers:

$$\mathsf{nat} \;=\; (1, 2, 3, \ldots)$$

we have the following semantic weighted stream automaton:

$$\begin{tikzcd}
\mathsf{nat} \mid 1 \arrow[loop above, "1"] \arrow[r, "1"] & \overline{1} \mid 1 \arrow[loop above, "1"]
\end{tikzcd}$$

The latter example is taken from Section 7.6, where we also saw the following automaton (omitting the output values, which are all 1):

$$\begin{tikzcd}[column sep=large]
\overline{1} \arrow[loop left, "\binom{0}{0}"] & \mathsf{nat} \arrow[l, "\binom{1}{0}"'] \arrow[loop below, "\binom{1}{1}"] & \mathsf{nat}^{\langle 2\rangle} \arrow[l, "\binom{2}{1}"'] \arrow[ll, bend right=40, "\binom{2}{0}"'] \arrow[loop above, "\binom{2}{2}"] & \cdots \arrow[l, "\binom{3}{2}"'] & \mathsf{nat}^{\langle n\rangle} \arrow[l, "\binom{n}{n-1}"'] \arrow[loop above, "\binom{n}{n}"] \arrow[ll, bend left=10, "\binom{n}{2}"] \arrow[lll, bend left=25, "\binom{n}{1}"] \arrow[llll, bend left=40, "\binom{n}{0}"]
\end{tikzcd}$$

It is a representation of the n-fold Hadamard product

$$\mathsf{nat}^{\langle n\rangle} \;=\; (1^n, 2^n, 3^n, \ldots)$$

For a last example, we consider the stream of the Fibonacci numbers, and its derivative:

$$\begin{aligned}
\sigma_1 &= \frac{\mathsf{X}}{1 - \mathsf{X} - \mathsf{X}^2} = (0, 1, 1, 2, 3, 5, 8, \ldots) \\
\sigma_2 &= \frac{1}{1 - \mathsf{X} - \mathsf{X}^2} = (1, 1, 2, 3, 5, 8, \ldots)
\end{aligned}$$

As we saw in Example 334, they are the solution of the following linear system of SDEs:

$$\begin{aligned}
\sigma_1(0) &= 0 & \qquad \sigma_1' &= \sigma_2 \\
\sigma_2(0) &= 1 & \qquad \sigma_2' &= \sigma_1 + \sigma_2
\end{aligned}$$

With these equations, we can now construct a semantic weighted stream automaton representing σ_1 and σ_2:

$$\begin{tikzcd}[column sep=huge]
\sigma_1 \mid 0 \arrow[r, bend left, "1"] & \sigma_2 \mid 1 \arrow[l, bend left, "1"] \arrow[loop above, "1"]
\end{tikzcd}$$

□

Exercise 377. The last example can be easily generalised. Use the proof of Theorem 332 in Section 12.8 to construct a finite semantic weighted stream automaton for any given rational stream (Definition 284). □

Exercise 378. Recall the function of stream exponentiation, from Exercise 297. By computing and suitably splitting the derivatives of the stream $\mathsf{e}^{\mathsf{X}^2}$, show that it can be represented by the state s_0 in the following *infinite* weighted stream automaton:

$$\begin{tikzcd}
s_0|1 \arrow[r, bend left, "1"] & s_1|0 \arrow[l, bend left, "1!"] \arrow[r, bend left, "1"] & s_2|0 \arrow[l, bend left, "2!"] \arrow[r, bend left, "1"] & \cdots \arrow[l, bend left, "3!"]
\end{tikzcd}$$
□

14.3 Weighted automata and linear stream systems

The definition of the function $\mathsf{str}\colon S \to \mathbb{R}^\omega$, in Definition 370, may be operationally intuitive but is, obviously, computationally inconvenient. We can do better by transforming weighted stream automata into stream systems, which were introduced in Chapter 7. As the state space of the resulting stream system will be a *vector space*, we will call them *linear* stream systems. The vector space structure of such linear stream systems will allow the formulation, in the present section, of coinduction *up-to-linearity*, which is an enhancement of the usual coinductive proof method. Then, in Section 14.4, we will use the vector space structure to characterise the behaviour of *finite dimensional* linear stream systems by rational streams.

Definition 379 (linear stream system). We call a stream system

$$(V,\ \langle \mathsf{o}, \mathsf{tr}\rangle) \qquad \langle \mathsf{o}, \mathsf{tr}\rangle\colon V \to \mathbb{R}\times V \qquad \mathsf{o}\colon V \to \mathbb{R} \qquad \mathsf{tr}\colon V \to V$$

linear if V is a vector space over $\mathbb{R}$ and both o and tr are linear maps. □

We recall from Theorem 78 that the *set* of streams forms a *final* stream system: for every stream system $(V,\ \langle \mathsf{o}, \mathsf{tr}\rangle)$ with outputs in $\mathbb{R}$, there exists a unique homomorphism

$$\begin{tikzcd}
V \arrow[r, dashed, "\exists!\, [\![-]\!]"] \arrow[d, "\forall\, \langle \mathsf{o}, \mathsf{tr}\rangle"'] & \mathbb{R}^\omega \arrow[d, "\langle \mathsf{i}_{st}, \mathsf{d}_{st}\rangle"] \\
\mathbb{R}\times V \arrow[r, dashed, "1_{\mathbb{R}} \times [\![-]\!]"'] & \mathbb{R}\times\mathbb{R}^\omega
\end{tikzcd} \tag{14.2}$$

Since $\mathbb{R}$ is a *field*, all of the above lives in the world of vector spaces and linear maps:

Proposition 380 (final linear stream system). The stream system $(\mathbb{R}^\omega,\ \langle \mathsf{i}_{st}, \mathsf{d}_{st}\rangle)$ is linear. If $(V,\ \langle \mathsf{o}, \mathsf{tr}\rangle)$ is linear then $[\![-]\!] \colon V \to \mathbb{R}^\omega$ is a linear map. □

Proof: The set $\mathbb{R}^\omega$ is a vector space, where the operations of sum and scalar multiplication on $\mathbb{R}^\omega$ are defined elementwise:

$$(\,k \cdot \sigma \ + \ l \cdot \tau\,)\,(n) \ = \ k \times (\sigma(n)) \ + \ \ l \times (\tau(n)) \qquad\qquad (\sigma, \tau \in \mathbb{R}^\omega,\ k, l \in \mathbb{R},\ n \geqslant 0)$$

Linearity of $[\![-]\!]$ follows from that of o and tr. □

We will now transform weighted stream automata into linear stream systems in a way that closely resembles the transformation of non-deterministic automata into deterministic automata, in Section 10.1, for which we used the powerset construction (Definition 221).

We first give a characterisation of the set $\mathbb{R}^S_\omega$, defined in (14.1) above.

Proposition 381 ($\mathbb{R}^S_\omega$ is a free vector space). For any set S, the set

$$\mathbb{R}^S_\omega \ = \ \{\, \phi \colon S \to \mathbb{R} \mid \mathsf{supp}(\phi) \text{ is finite } \}$$

is a vector space over $\mathbb{R}$. Moreover, $\mathbb{R}^S_\omega$ is *free on the set* S: for every vector space V and for every *function* $f \colon S \to V$, there exists a unique *linear map* $f^\sharp \colon \mathbb{R}^S_\omega \to V$ extending f from S to $\mathbb{R}^S_\omega$:

$$\begin{tikzcd} S \arrow[r, "\hat{\{\cdot\}}"] \arrow[d, "\forall f"'] & \mathbb{R}^S_\omega \arrow[dl, dashed, "\exists!\, f^\sharp"] \\ V & \end{tikzcd} \qquad\qquad f^\sharp(\hat{s}) \ = \ f(s)$$

Here, the function $\hat{\{\cdot\}} \colon S \to \mathbb{R}^S_\omega$ is defined, for $s, t \in S$, by

$$\hat{\{\cdot\}}(s) \ = \ \hat{s} \qquad\qquad \hat{s} \colon S \to \mathbb{R} \qquad\qquad \hat{s}(t) \ = \ \begin{cases} 1 & \text{if } t = s \\ 0 & \text{if } t \neq s \end{cases} \tag{14.3}$$

Proof: The operations of sum and scalar multiplication on $\mathbb{R}^S_\omega$ are defined elementwise:

$$(\,k \cdot \phi \ + \ l \cdot \psi\,)\,(s) \ = \ k \times (\phi(s)) \ + \ l \times (\psi(s)) \qquad\qquad (\phi, \psi \in \mathbb{R}^S_\omega,\ k, l \in \mathbb{R},\ s \in S)$$

We observe that the set

$$\hat{S} \ = \ \{\hat{s} \mid s \in S\}$$

forms a basis for the vector space $\mathbb{R}^S_\omega$. As a consequence, we can write

$$\phi \ = \ \phi(s_1) \cdot \hat{s_1} \ + \ \cdots \ + \ \phi(s_n) \cdot \hat{s_n}$$

for any $\phi \in \mathbb{R}^S_\omega$, with $\mathsf{supp}(\phi) = \{s_1, \ldots, s_n\}$. This leaves

$$f^\sharp(\phi) \ = \ \ \phi(s_1) \cdot f(s_1) \ + \cdots + \ \phi(s_n) \cdot f(s_n)$$

as the only possible choice for the definition of $f^\sharp$. □

Definition 382 (generalised powerset construction for weighted automata). We can transform a *weighted stream automaton*

$$(S,\, \langle \mathsf{o}, \mathsf{tr}\rangle) \qquad \langle \mathsf{o}, \mathsf{tr}\rangle\colon S \to \mathbb{R} \times \mathbb{R}^S_\omega \qquad \mathsf{o}\colon S \to \mathbb{R} \qquad \mathsf{tr}\colon S \to \mathbb{R}^S_\omega$$

into a *linear stream system*

$$(\mathbb{R}^S_\omega,\, \langle \mathsf{o}^\sharp,\, \mathsf{tr}^\sharp\rangle) \qquad \langle \mathsf{o}^\sharp,\, \mathsf{tr}^\sharp\rangle\colon \mathbb{R}^S_\omega \to \mathbb{R} \times \mathbb{R}^S_\omega \qquad \mathsf{o}^\sharp\colon \mathbb{R}^S_\omega \to \mathbb{R} \qquad \mathsf{tr}^\sharp\colon \mathbb{R}^S_\omega \to \mathbb{R}^S_\omega$$

by observing that $\mathbb{R} \times \mathbb{R}^S_\omega$ is a vector space (being the product of two), and by taking

$$V \;=\; \mathbb{R} \times \mathbb{R}^S_\omega \qquad\qquad f \;=\; \langle \mathsf{o}, \mathsf{tr}\rangle$$

in Proposition 381 above:

$$\begin{tikzcd} S \arrow[r, "\hat{\{\cdot\}}"] \arrow[d, "\forall f"'] & \mathbb{R}^S_\omega \arrow[dl, dashed, bend left, "\exists!\, f^\sharp"] \\ V & \end{tikzcd} \qquad \begin{tikzcd} S \arrow[r, "\hat{\{\cdot\}}"] \arrow[d, "\langle \mathsf{o},\mathsf{tr}\rangle"'] & \mathbb{R}^S_\omega \arrow[dl, dashed, bend left, "\langle \mathsf{o}^\sharp, \mathsf{tr}^\sharp\rangle"] \\ \mathbb{R} \times \mathbb{R}^S_\omega & \end{tikzcd} \tag{14.4}$$

to obtain the functions $\mathsf{o}^\sharp$ and $\mathsf{tr}^\sharp$. □

To be precise, Proposition 381 gives us, in diagram (14.4) above, a linear map

$$\langle \mathsf{o}, \mathsf{tr}\rangle^\sharp\colon \mathbb{R}^S_\omega \to \mathbb{R} \times \mathbb{R}^S_\omega$$

which we have written as $\langle \mathsf{o}^\sharp,\, \mathsf{tr}^\sharp\rangle$. Spelling out the definitions of $\mathsf{o}^\sharp$ and $\mathsf{tr}^\sharp$, we find, for all $\phi \in \mathbb{R}^S_\omega$ with $\mathsf{supp}(\phi) = \{s_1, \ldots, s_n\} \subseteq S$,

$$\begin{aligned} \mathsf{o}^\sharp(\phi) \;&=\; \phi(s_1) \times \mathsf{o}(s_1) \;+\; \cdots \;+\; \phi(s_n) \times \mathsf{o}(s_n) \\ \mathsf{tr}^\sharp(\phi) \;&=\; \phi(s_1) \cdot \mathsf{tr}(s_1) \;+\; \cdots \;+\; \phi(s_n) \cdot \mathsf{tr}(s_n) \end{aligned}$$

Note that above, $\phi(s_i) \times \mathsf{o}(s_i)$ denotes multiplication of the real numbers $\phi(s_i)$ and $\mathsf{o}(s_i)$, and $\phi(s_i) \cdot \mathsf{tr}(s_i)$ denotes scalar multiplication of the real number $\phi(s_i)$ and the vector $\mathsf{tr}(s_i)$. Also note that $\mathsf{o}^\sharp(\hat{s}) = \mathsf{o}(s)$ and $\mathsf{tr}^\sharp(\hat{s}) = \mathsf{tr}(s)$, for all $s \in S$.

Having thus entered the world of linear stream systems, we can combine diagrams (14.2) and (14.4) above, as follows:

$$\begin{tikzcd} S \arrow[r, "\hat{\{\cdot\}}"'] \arrow[rr, bend left, "\mathsf{str}"] \arrow[d, "\langle \mathsf{o}, \mathsf{tr}\rangle"'] & \mathbb{R}^S_\omega \arrow[r, dashed, "\exists!\, [\![-]\!]"'] \arrow[dl, bend left, "\langle \mathsf{o}^\sharp,\, \mathsf{tr}^\sharp\rangle"] & \mathbb{R}^\omega \arrow[d, "\langle \mathsf{i}_{st}, \mathsf{d}_{st}\rangle"] \\ \mathbb{R} \times \mathbb{R}^S_\omega \arrow[rr, dashed, "1 \times [\![-]\!]"] & & \mathbb{R} \times \mathbb{R}^\omega \end{tikzcd} \tag{14.5}$$

where we have the following ingredients:

- the function $\mathsf{str}\colon S \to \mathbb{R}^\omega$, from Definition 370, maps a state $s \in S$ in the weighted stream automaton $(S, \langle \mathsf{o}, \mathsf{tr}\rangle)$ to the stream $\mathsf{str}(s)$ it represents.
- the unique homomorphism of stream systems $[\![-]\!]\colon \mathbb{R}^S_\omega \to \mathbb{R}^\omega$ is given by the finality of the stream system $(\mathbb{R}^\omega, \langle \mathsf{i}_{st}, \mathsf{d}_{st}\rangle)$, Theorem 78, and maps a state $\phi \in \mathbb{R}^S_\omega$ in the linear stream system $(\mathbb{R}^S_\omega, \langle \mathsf{o}^\sharp, \mathsf{tr}^\sharp\rangle)$ to the stream $[\![\phi]\!]$ it represents.
- the function $\hat{\{\cdot\}}\colon S \to \mathbb{R}^S_\omega$, which is defined in Equation (14.3) above, is an embedding of the weighted stream automaton S into the stream system $\mathbb{R}^S_\omega$.

By Propositions 380 and 381, all of the right hand square in diagram (14.5) lives in the world of vector spaces and linear maps. Furthermore, everyting in the diagram commutes.

Theorem 383. *In diagram* (14.5) *above, we have* $\mathsf{str}(s) = [\![\, \hat{s} \,]\!]$*, for all* $s \in S$*.*

Proof: We prove $\mathsf{str}(s)(n) = [\![\, \hat{s} \,]\!](n)$, for all $s \in S$ and $n \geqslant 0$, by induction on n:

$$\begin{aligned} \mathsf{str}(s)(0) &= \mathsf{o}(s) && \text{(definition } \mathsf{str}) \\ &= \mathsf{o}^\sharp(\hat{s}) && \text{(definition } \mathsf{o}^\sharp) \\ &= [\![\, \hat{s} \,]\!](0) && ([\![-]\!] \text{ is a homomorphism}) \end{aligned}$$

Next we assume, for all $r \in S$, that

$$\mathsf{str}(r)(n) = [\![\, \hat{r} \,]\!](n) \qquad \text{(induction hypothesis)}$$

We will prove, for all $s \in S$,

$$\mathsf{str}(s)(n+1) = [\![\, \hat{s} \,]\!](n+1)$$

Let $s \in S$ and let $\mathsf{supp}(\mathsf{tr}(s)) = \{s_1, \ldots, s_l\} \subseteq S$. We can write $\mathsf{tr}(s) \in \mathbb{R}^S_\omega$ as

$$\mathsf{tr}(s) = k_1 \cdot \hat{s_1} + \cdots + k_l \cdot \hat{s_l} \tag{14.6}$$

Let $\mathsf{tr}(s)(s_i) = k_i$, for $i = 1, \ldots, l$. Then

$$\begin{aligned} \mathsf{str}(s)(n+1) &= k_1 \times \mathsf{str}(s_1)(n) + \cdots + k_l \times \mathsf{str}(s_l)(n) && \text{(definition } \mathsf{str}) \\ &= k_1 \times [\![\, \hat{s_1} \,]\!](n) + \cdots + k_l \times [\![\, \hat{s_l} \,]\!](n) && \text{(by the ind. hyp.)} \\ &= [\![\, k_1 \cdot \hat{s_1} + \cdots + k_l \cdot \hat{s_l} \,]\!](n) && ([\![-]\!] \text{ is linear}) \\ &= [\![\, \mathsf{tr}(s) \,]\!](n) && \text{(identity (14.6) above)} \\ &= [\![\, \mathsf{tr}^\sharp(\hat{s}) \,]\!](n) && \text{(definition } \mathsf{tr}^\sharp) \\ &= [\![\, \hat{s} \,]\!]'(n) && ([\![-]\!] \text{ is a homomorphism}) \\ &= [\![\, \hat{s} \,]\!](n+1) \end{aligned}$$

which concludes the proof. □

Thus (the state $s \in S$ in) the original weighted stream automaton $(S, \langle \mathsf{o}, \mathsf{tr}\rangle)$ has the same behaviour as (the state $\hat{s} \in \mathbb{R}^S_\omega$ in) the linear stream system $(\mathbb{R}^S_\omega, \langle \mathsf{o}^\sharp, \mathsf{tr}^\sharp\rangle)$ into which it is transformed.

Example 384. The linear stream system $(\mathbb{R}^S_\omega,\ \langle \mathsf{o}^\sharp,\ \mathsf{tr}^\sharp \rangle)$ generated by the automaton

$$\begin{tikzcd}
s|1 \arrow[loop left, "0"] \arrow[r, bend left, "1"] & t|2 \arrow[l, bend left, "-1"] \arrow[loop right, "2"] & u|1 \arrow[loop left, "1"] \arrow[r, bend left, "1"] & v|1 \arrow[l, bend left, "0"] \arrow[loop right, "1"]
\end{tikzcd}
\qquad (S = \{s, t, u, v\})$$

from Example 371, is a 4-dimensional vector space with basis $\{\hat{s}, \hat{t}, \hat{u}, \hat{v}\}$. The values of

$$\mathsf{o}^\sharp \colon \mathbb{R}^S_\omega \to \mathbb{R} \qquad\qquad \mathsf{tr}^\sharp \colon \mathbb{R}^S_\omega \to \mathbb{R}^S_\omega$$

are therefore determined by

$$\begin{aligned}
\mathsf{o}^\sharp(\hat{s}) &= 1 & \qquad \mathsf{tr}^\sharp(\hat{s}) &= \hat{t} \\
\mathsf{o}^\sharp(\hat{t}) &= 2 & \qquad \mathsf{tr}^\sharp(\hat{t}) &= 2\hat{t} - \hat{s} \\
\mathsf{o}^\sharp(\hat{u}) &= 1 & \qquad \mathsf{tr}^\sharp(\hat{u}) &= \hat{u} + \hat{v} \\
\mathsf{o}^\sharp(\hat{v}) &= 1 & \qquad \mathsf{tr}^\sharp(\hat{v}) &= \hat{v}
\end{aligned}$$

Looking at the transition sequences starting in $\hat{s}$ and $\hat{u}$, we find

$$\begin{tikzcd}
\hat{s}|1 \arrow[r] \arrow[d, dash, dashed] & \hat{t}|2 \arrow[r] \arrow[d, dash, dashed] & (2\hat{t} - \hat{s})|3 \arrow[r] \arrow[d, dash, dashed] & (3\hat{t} - 2\hat{s})|4 \arrow[r] \arrow[d, dash, dashed] & (4\hat{t} - 3\hat{s})|5 \arrow[r] \arrow[d, dash, dashed] & \cdots \\
\hat{u}|1 \arrow[r] & (\hat{u} + \hat{v})|2 \arrow[r] & (\hat{u} + 2\hat{v})|3 \arrow[r] & (\hat{u} + 3\hat{v})|4 \arrow[r] & (\hat{u} + 4\hat{v})|5 \arrow[r] & \cdots
\end{tikzcd}$$

from which it follows that

$$[\![\, \hat{s} \,]\!] = (1, 2, 3, \ldots) \qquad\qquad [\![\, \hat{u} \,]\!] = (1, 2, 3, \ldots)$$

Similarly, one finds

$$[\![\, \hat{t} \,]\!] = (2, 3, 4 \ldots) \qquad\qquad [\![\, \hat{v} \,]\!] = (1, 1, 1, \ldots)$$

For a formal proof of the equality of $[\![\, \hat{s} \,]\!]$ and $[\![\, \hat{u} \,]\!]$, we have indicated with the vertical dashed lines above a stream bisimulation relation on $\mathbb{R}^S_\omega$:

$$\{\, \langle \hat{s}, \hat{u} \rangle \,\} \cup \{\, \langle \hat{t}, \hat{u} + \hat{v} \rangle \,\} \cup \{\, \langle k\hat{t} - (k-1)\hat{s},\ \hat{u} + k\hat{v} \rangle \ \mid\ k \geqslant 2 \,\}$$

By coinduction Theorem 96, $[\![\, \hat{s} \,]\!] = [\![\, \hat{u} \,]\!]$, and by Theorem 383, $\mathsf{str}(s) = \mathsf{str}(u)$. □

Exercise 385. Consider again the weighted stream automaton

$$\begin{tikzcd}
s_0|1 \arrow[loop left, "1"] \arrow[r, bend left, "1"] & s_1|0 \arrow[l, bend left, "1"] \arrow[loop above, "3"] \arrow[r, bend left, "2"] & s_2|0 \arrow[l, bend left, "2"] \arrow[loop above, "5"] \arrow[r, bend left, "3"] & \cdots \arrow[l, bend left, "3"]
\end{tikzcd}
\qquad (S = \{s_0, s_1, s_2, \ldots\})$$

from Example 371. Show that in the corresponding stream automaton $\mathbb{R}^S_\omega$ (which is a vector space of infinite dimension), the sequence of transitions starting in $\hat{s_0}$ is of the form

$$\hat{s_0} = \phi_0|0! \longrightarrow \phi_1|1! \longrightarrow \phi_2|2! \longrightarrow \cdots$$

where, for all $n \geqslant 0$,

$$\phi_n \;=\; n!\left(\binom{n}{0}\hat{s_0} + \binom{n}{1}\hat{s_1} + \cdots + \binom{n}{n}\hat{s_n}\right) \qquad \square$$

We saw in Example 384 that it took an infinite bisimulation relation to prove that $[\![\, \hat{s} \,]\!] = [\![\, \hat{u} \,]\!]$, from which $\mathsf{str}(s) = \mathsf{str}(u)$ followed. We can do better with the help of the following up-to technique.

Definition 386 (bisimulation up-to-linearity). Let $(V, \langle \mathsf{o}, \mathsf{tr}\rangle)$ be a linear stream system over $\mathbb{R}$. A relation

$$R \;\subseteq\; V \times V$$

is a *bisimulation up-to-linearity* if

$$(i)\;\; \mathsf{o}(v) \;=\; \mathsf{o}(w) \quad \text{and} \qquad (ii)\;\; (\,\mathsf{tr}(v),\, tr(w)\,) \in \overline{R}$$

for all $(v, w) \in R$. Here

$$\overline{R} \;\subseteq\; V \times V$$

is the *linear closure* of R: the smallest relation such that

1. $R \subseteq \overline{R}$
2. if (v, w), $(x, y) \in \overline{R}$ then $(\,(k \cdot v) + (l \cdot x),\, (k \cdot w) + (l \cdot y)\,) \in \overline{R}$

for all $v, w, x, y \in V$, and $k, l \in \mathbb{R}$. $\square$

Proposition 387 (coinduction up-to-linearity). If $R \subseteq V \times W$ is a bisimulation up-to-linearity and $(v, w) \in R$, then $[\![\, v \,]\!] = [\![\, w \,]\!]$.

Proof: If R is a bisimulation up-to-linearity then one easily shows that $\overline{R}$ is an ordinary stream bisimulation, by induction on its definition. Since $(v, w) \in R \subseteq \overline{R}$, it follows from Theorem 84 that $[\![\, v \,]\!] = [\![\, w \,]\!]$. $\square$

Example 388. Returning again to the automaton in Example 384, we observe that

$$\begin{aligned}
\mathsf{tr}^\sharp(\hat{s}) &= \hat{t} & \mathsf{tr}^\sharp(\hat{t}) &= 2\hat{t} - \hat{s} \\
\mathsf{tr}^\sharp(\hat{u}) &= \hat{u} + \hat{v} & \mathsf{tr}^\sharp(\hat{u}+\hat{v}) &= \hat{u} + 2\hat{v} = 2(\hat{u}+\hat{v}) - \hat{u}
\end{aligned}$$

This shows that $\{\, (\hat{s},\, \hat{u}),\ (\hat{t},\, \hat{u}+\hat{v}) \,\}$ is a bisimulation up-to-linearity. It follows from Proposition 387 that this finite witness suffices to conclude that $[\![\, \hat{s} \,]\!] = [\![\, \hat{u} \,]\!]$ and hence, as before, that $\mathsf{str}(s) = \mathsf{str}(u)$. □

Exercise 389. Consider the following weighted stream automaton:

$$\begin{tikzcd}
s_0|0 \arrow[r, "1"] & s_1|1 \arrow[loop above, "1/2"] \arrow[r, "1/2"] & s_2|1 \arrow[loop above, "1"]
\end{tikzcd}
\qquad
\begin{tikzcd}
t_0|0 \arrow[r, "1/2"] \arrow[rr, bend right, "1/2"'] & t_1|1 \arrow[loop above, "1/2"] \arrow[r, "1/2"] & t_2|1 \arrow[loop above, "1"]
\end{tikzcd}$$

Use coinduction up-to-linearity to prove that $\mathsf{str}(s_0) = \mathsf{str}(t_0)$. □

Using bisimulations and bisimulations up-to-linearity, in Examples 384 and 388, to prove the equivalence of states of linear stream automata, offers a useful exercise in coinduction and its enhancements.

14.4 Rational streams and linear stream systems

The behaviour of *finite-dimensional* linear stream systems can be characterised in terms of rational streams (Definition 284), using the fact that rational streams are solutions of finite linear systems of SDEs (Theorem 332 in Section 12.8). As a corollary, we will obtain yet another method for proving the equivalence of linear stream systems, as an alternative to the method of enhanced coinduction (up-to-linearity) from Section 14.3

We call a linear stream system

$$(V,\, \langle \mathsf{o}, \mathsf{tr} \rangle) \qquad \langle \mathsf{o}, \mathsf{tr} \rangle \colon V \to \mathbb{R} \times V \qquad \mathsf{o} \colon V \to \mathbb{R} \qquad \mathsf{tr} \colon V \to V$$

finite dimensional if the vector space V is of finite dimension. Without loss of generality, we may assume that

$$V = \mathbb{R}^n \qquad (n \geqslant 1)$$

The linear maps

$$\mathsf{o} \colon \mathbb{R}^n \to \mathbb{R} \qquad \mathsf{tr} \colon \mathbb{R}^n \to \mathbb{R}^n$$

can be written in matrix form as

$$\mathsf{o}(\overline{v}) \;=\; \overline{v} \cdot N \qquad\qquad \mathsf{tr}(\overline{v}) \;=\; \overline{v} \cdot M \tag{14.7}$$

where we read a vector $\overline{v} \in \mathbb{R}^n$ as a real-valued matrix of dimension $1 \times n$, and where N and M are real-valued matrices of dimension $n \times 1$ and $n \times n$:

$$\overline{v} \;=\; \begin{pmatrix} v_1 & \cdots & v_n \end{pmatrix} \qquad N \;=\; \begin{pmatrix} r_1 \\ \vdots \\ r_n \end{pmatrix} \qquad M \;=\; \begin{pmatrix} r_{11} & \cdots & r_{1n} \\ \vdots & & \vdots \\ r_{n1} & \cdots & r_{nn} \end{pmatrix} \qquad (v_i, r_i, r_{ij} \in \mathbb{R})$$

The global behaviour of our linear stream system is given by the final homomorphism into the set of streams, according to Theorem 78:

$$\begin{tikzcd} \mathbb{R}^n \arrow[r, dashed, "\exists!\, [\![-]\!]"] \arrow[d, "{\langle N,\, M\rangle}"'] & \mathbb{R}^\omega \arrow[d, "{\langle \mathsf{i}_{st}, \mathsf{d}_{st}\rangle}"] \\ \mathbb{R} \times \mathbb{R}^n \arrow[r, dashed, "1 \times [\![-]\!]"] & \mathbb{R} \times \mathbb{R}^\omega \end{tikzcd}$$

Here (14.7) has allowed us to write N and M for o and tr:

$$\langle N,\, M\rangle \;=\; \langle \mathsf{o}, \mathsf{tr}\rangle$$

We will now show how $[\![-]\!]$ can be expressed in terms of the matrices N and M.

Proposition 390 (the behaviour of finite dimensional linear stream systems). For a finite dimensional linear stream system $(\mathbb{R}^n, \langle N,\, M\rangle)$:

$$[\![-]\!] \colon \mathbb{R}^n \longrightarrow \mathbb{R}^\omega \qquad\qquad [\![\overline{v}]\!] \;=\; \overline{v} \cdot (I - (\mathsf{X} \cdot M))^{-1} \cdot N \qquad\qquad (\overline{v} \in \mathbb{R}^n)$$

Proof: Let us denote the vectors of the standard basis for $\mathbb{R}^n$ by

$$e_1 \;=\; \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \end{pmatrix} \quad e_2 \;=\; \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \end{pmatrix} \quad \cdots \quad e_n \;=\; \begin{pmatrix} 0 & 0 & \cdots & 0 & 1 \end{pmatrix}$$

The value of $[\![-]\!]$ is determined by its value on $e_1, \ldots, e_n$. We define

$$\sigma_i \;=\; [\![e_i]\!] \qquad\qquad (i = 1, \ldots, n)$$

The initial value of σ_i is given by

$$\begin{aligned} \sigma_i(0) \;&=\; [\![e_i]\!](0) \\ &=\; e_i \cdot N && ([\![-]\!] \text{ is a homomorphism}) \\ &=\; r_i \end{aligned}$$

For the derivative of σ_i, we have

$$\begin{aligned}
\sigma_i' &= [\![e_i]\!]' \\
&= [\![e_i \cdot M]\!] && ([\![-]\!] \text{ is a homomorphism}) \\
&= [\![\, \begin{pmatrix} r_{i1} & \cdots & r_{in} \end{pmatrix} \,]\!] \\
&= [\![\, r_{i1}e_1 + \cdots + r_{in}e_n \,]\!] \\
&= r_{i1}[\![e_1]\!] + \cdots + r_{in}[\![e_n]\!] && ([\![-]\!] \text{ is linear, Proposition 380}) \\
&= r_{i1}\sigma_1 + \cdots + r_{in}\sigma_n
\end{aligned}$$

It follows that the streams $\sigma_1, \ldots, \sigma_n$ satisfy

$$\begin{pmatrix} \sigma_1 \\ \vdots \\ \sigma_n \end{pmatrix}(0) = N \qquad\qquad \begin{pmatrix} \sigma_1 \\ \vdots \\ \sigma_n \end{pmatrix}' = M \cdot \begin{pmatrix} \sigma_1 \\ \vdots \\ \sigma_n \end{pmatrix} \tag{14.8}$$

which is the matrix representation of a *linear system of SDEs*, as in (12.49) of Section 12.8. The solution of (14.8) is given, according to identity (12.50) in the proof of Theorem 332, by

$$\begin{pmatrix} \sigma_1 \\ \vdots \\ \sigma_n \end{pmatrix} = (I - (\mathsf{X} \cdot M))^{-1} \cdot N \tag{14.9}$$

Writing $\overline{v} \in \mathbb{R}^n$ as

$$\overline{v} = \begin{pmatrix} v_1 & \cdots & v_n \end{pmatrix} = v_1e_1 + \cdots + v_ne_n$$

we have

$$\begin{aligned}
[\![\overline{v}]\!] &= [\![v_1e_1 + \cdots + v_ne_n]\!] \\
&= v_1[\![e_1]\!] + \cdots + v_n[\![e_n]\!] && ([\![-]\!] \text{ is linear, Proposition 380}) \\
&= v_1\sigma_1 + \cdots + v_n\sigma_n && (\text{definition of } \sigma_i) \\
&= \overline{v} \cdot \begin{pmatrix} \sigma_1 \\ \vdots \\ \sigma_n \end{pmatrix} \\
&= \overline{v} \cdot (I - (\mathsf{X} \cdot M))^{-1} \cdot N && (\text{by (14.9)})
\end{aligned}$$

which concludes the proof. □

There is the following immediate consequence of Proposition 390.

Theorem 391 (rational streams and finite dimensional linear stream systems). *For every $\sigma \in \mathbb{R}^\omega$, the following are equivalent:*

(1) σ is rational.

(2) σ can be defined by a finite dimensional linear stream system.

Proof: By the proof of Proposition 390, finite dimensional linear stream systems are in one-to-one correspondence to finite linear systems of SDEs: both are determined by two real-valued matrices N and M of dimension $n \times 1$ and $n \times n$, for some $n \geqslant 1$. The present theorem now follows from Theorem 332, which states that a stream is rational iff it can be defined by a finite linear system of SDEs. $\square$

Example 392. The linear stream system $(\mathbb{R}^S_\omega, \langle \mathsf{o}^\sharp, \mathsf{tr}^\sharp \rangle)$ from Example 384, which is generated by the weighted stream automaton

$$\begin{tikzcd} s|1 \arrow[loop left, "0"] \arrow[r, bend left, "1"] & t|2 \arrow[loop right, "2"] \arrow[l, bend left, "-1"] & u|1 \arrow[loop left, "1"] \arrow[r, bend left, "1"] & v|1 \arrow[loop right, "1"] \arrow[l, bend left, "0"] \end{tikzcd} \qquad (S = \{s, t, u, v\})$$

is a 4-dimensional vector space $\mathbb{R}^S_\omega \cong \mathbb{R}^4$ for which we choose the following basis:

$$\begin{array}{lllll} \hat{s} = e_1 = (1,0,0,0) & \qquad & \hat{u} = e_3 = (0,0,1,0) \\ \hat{t} = e_2 = (0,1,0,0) & \qquad & \hat{v} = e_4 = (0,0,0,1) \end{array}$$

The matrices N and M corresponding to $\mathsf{o}^\sharp \colon \mathbb{R}^4 \to \mathbb{R}$ and $\mathsf{tr}^\sharp \colon \mathbb{R}^4 \to \mathbb{R}^4$ are

$$\mathsf{o}^\sharp = N = \begin{pmatrix} 1 \\ 2 \\ 1 \\ 1 \end{pmatrix} \qquad\qquad \mathsf{tr}^\sharp = M = \begin{pmatrix} 0 & 1 & 0 & 0 \\ -1 & 2 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Computing the inverse of $I - (\mathsf{X} \cdot M)$ gives

$$(I - (\mathsf{X} \cdot M))^{-1} = \begin{pmatrix} 1 - 2\mathsf{X}/(1-\mathsf{X})^2 & \mathsf{X}/(1-\mathsf{X})^2 & 0 & 0 \\ -\mathsf{X}/(1-\mathsf{X})^2 & 1/(1-\mathsf{X})^2 & 0 & 0 \\ 0 & 0 & 1/1-\mathsf{X} & \mathsf{X}/(1-\mathsf{X})^2 \\ 0 & 0 & 0 & 1/1-\mathsf{X} \end{pmatrix}$$

As a consequence,

$$\begin{aligned} [\![(v_1, v_2, v_3, v_4)]\!] &= \overline{v} \cdot (I - (\mathsf{X} \cdot M))^{-1} \cdot N \\ &= v_1 \cdot \frac{1}{(1-\mathsf{X})^2} + v_2 \cdot \frac{2-\mathsf{X}}{(1-\mathsf{X})^2} + v_3 \cdot \frac{1}{(1-\mathsf{X})^2} + v_4 \cdot \frac{1}{1-\mathsf{X}} \end{aligned}$$

Thus $[\![\, \hat{s} \,]\!] = [\![\, e_1 \,]\!] = [\![\, e_3 \,]\!] = [\![\, \hat{u} \,]\!]$ and, as before, by Theorem 383, $\mathsf{str}(s) = \mathsf{str}(u)$. $\square$

Exercise 393. Compute the behaviour of the linear stream system $(\mathbb{R}^S_\omega, \langle \mathsf{o}^\sharp, \mathsf{tr}^\sharp \rangle)$ that is generated by the following weighted stream automaton

$$\begin{tikzcd} s|1 \arrow[r, "1"] & t|2 \arrow[loop above, "2"] \arrow[r, bend right, "-1"'] & u|1 \arrow[loop above, "1"] \arrow[r, "1"] & v|1 \arrow[loop above, "1"] \end{tikzcd} \qquad (S = \{s, t, u, v\})$$

where as usual, all transitions with weight 0 have been omitted. □

There is also the following corollary of Theorem 391.

Theorem 394 (rational streams and finite weighted stream automata). *For every $\sigma \in \mathbb{R}^\omega$, the following are equivalent:*

(1) σ is rational.

(2) σ can be represented by a finite weighted stream automaton.

Proof: By the generalised powerset construction from Definition 382, every finite weighted automaton is transformed into a finite dimensional stream system, which has rational behaviour, by Theorem 391. Conversely, any finite dimensional linear stream system $(\mathbb{R}^n, \langle N, M \rangle)$ can be obtained by applying the generalised powerset construction to a finite weighted stream automaton, the outputs and transitions of which are defined by the matrices N and M. □

14.5 Minimisation of linear stream systems

We look again at the final homomorphism $[\![-]\!] \colon V \to \mathbb{R}^\omega$ from a stream system V to the final stream system $\mathbb{R}^\omega$, and recall from Theorem 84 that

$$v \sim w \iff [\![v]\!] = [\![w]\!]$$

for all $v, w \in V$. We also recall, from Exercise 85, that this implies that $\mathsf{im}([\![-]\!])$, which is the image of V under the homomorphism $[\![-]\!]$, is a subsystem of $\mathbb{R}^\omega$ that is *minimal* with respect to bisimulation equivalence.

The above applies to arbirary, not necessarily linear, stream systems. If we are dealing with a *linear* stream system $(V, \langle \mathsf{o}, \mathsf{tr} \rangle)$, then we have, for all $v, w \in V$,

$$\begin{aligned} v \sim w &\iff [\![v]\!] = [\![w]\!] && \text{(Theorem 84)} \\ &\iff [\![v]\!] - [\![w]\!] = 0 \\ &\iff [\![v - w]\!] = 0 && \text{($[\![-]\!]$ is linear, Proposition 380)} \\ &\iff v - w \in \mathsf{ker}([\![-]\!]) \end{aligned}$$

If our linear stream system $(V, \langle \mathsf{o}, \mathsf{tr} \rangle)$ is, moreover, *finite dimensional*, then both $\mathsf{ker}([\![-]\!])$ and $\mathsf{im}([\![-]\!])$ can be computed explicitly. Here is an example.

Example 395. We saw that the behaviour map $[\![-]\!] \colon \mathbb{R}^4 \to \mathbb{R}^\omega$ of the linear stream system $(\mathbb{R}^4\,,\, \langle \mathsf{o}^\sharp, \mathsf{tr}^\sharp \rangle)$ in Example 392 satisfies, for all $\overline{v} = (v_1, v_2, v_3, v_4) \in \mathbb{R}^4$,

$$[\![\overline{v}]\!] \;=\; v_1 \cdot \frac{1}{(1-\mathsf{X})^2} + v_2 \cdot \frac{2-\mathsf{X}}{(1-\mathsf{X})^2} + v_3 \cdot \frac{1}{(1-\mathsf{X})^2} + v_4 \cdot \frac{1}{1-\mathsf{X}}$$

Thus $\mathsf{im}([\![-]\!]) \subseteq \mathbb{R}^\omega$ is a 2-dimensional subspace, with basis, for instance,

$$\{ \frac{1}{(1-\mathsf{X})^2},\ \frac{2-\mathsf{X}}{(1-\mathsf{X})^2} \}$$

Computing $\mathsf{ker}([\![-]\!])$, we find

$$\begin{aligned}
[\![\overline{v}]\!] \;&=\; 0 \\
\iff \quad & v_1 \cdot \frac{1}{(1-\mathsf{X})^2} + v_2 \cdot \frac{2-\mathsf{X}}{(1-\mathsf{X})^2} + v_3 \cdot \frac{1}{(1-\mathsf{X})^2} + v_4 \cdot \frac{1}{1-\mathsf{X}} \;=\; 0 \\
\iff \quad & \frac{(v_1 + 2v_2 + v_3 + v_4) \;-\; (v_2 + v_4)\mathsf{X}}{(1-\mathsf{X})^2} \;=\; 0 \\
\iff \quad & v_1 + 2v_2 + v_3 + v_4 \;=\; 0 \quad \text{and} \quad v_2 + v_4 \;=\; 0 \\
\iff \quad & v_1 + v_2 + v_3 \;=\; 0 \quad \text{and} \quad v_2 + v_4 \;=\; 0
\end{aligned}$$

It follows that $\mathsf{ker}([\![-]\!]) \subseteq \mathbb{R}^4$ is a 2-dimensional subspace, for which, for instance,

$$\{ (1, -1, 0, 1),\ (1, 0, -1, 0) \}$$

is a basis. □

Exercise 396. Explain why the dimensions of the kernel and the image of $[\![-]\!]$ in Example 395 add up to 4, the dimension of $\mathbb{R}^4$, by recalling, from linear algebra, that

$$L \colon V \to W \qquad\qquad \dim(V) \;=\; \dim(\mathsf{ker}(L)) \;+\; \dim(\mathsf{im}(L))$$

for any linear map between finite dimensional vector spaces V and W. □

Exercise 397. Consider the following weighted stream automaton:

$$\begin{tikzcd}
s|1 \arrow[r, "1"] & t|2 \arrow[r, bend left, "1"] & u|3 \arrow[l, "1"] \arrow[ll, bend left, "-1"] \arrow[loop right, "1"]
\end{tikzcd} \qquad (S = \{s, t, u\})$$

(omitting all transitions with weight 0). Construct $\mathbb{R}^S_\omega \cong \mathbb{R}^3$ and compute $\mathsf{ker}([\![-]\!])$ and $\mathsf{im}([\![-]\!])$ of the corresponding behaviour map $[\![-]\!] \colon \mathbb{R}^3 \to \mathbb{R}^\omega$. □

Here is yet another characterisation of rational streams, which is equivalent to Theorem 391. Let $\sigma \in \mathbb{R}^\omega$ and let the *subspace generated by* σ:

$$\langle \sigma \rangle \;\subseteq\; \mathbb{R}^\omega$$

be defined as the smallest subspace of the vector space $\mathbb{R}^\omega$ that contains σ. It has the following property.

Theorem 398 (generated subspace of rational streams). *For every $\sigma \in \mathbb{R}^\omega$, the following are equivalent:*

(1) σ is rational.

(2) the subspace $\langle \sigma \rangle \subseteq \mathbb{R}^\omega$ generated by σ is finite dimensional.

Proof: It follows from Theorem 391 that any rational stream is represented as $\sigma = [\![v]\!]$ by a state v of a finite dimensional linear stream system V. Since $\langle \sigma \rangle \subseteq \mathsf{im}([\![-]\!])$ and the latter is finite dimensional, it follows that $\langle \sigma \rangle$ is finite dimensional, too. Conversely, if $\langle \sigma \rangle \subseteq \mathbb{R}^\omega$ is finite dimensional then σ is represented by the finite dimensional linear stream system $\langle \sigma \rangle$ under the inclusion homomorphism from $\langle \sigma \rangle$ to $\mathbb{R}^\omega$. It follows, by Theorem 391, that σ is rational. □

Theorem 398 can be used to prove that a given stream $\sigma \in \mathbb{R}^\omega$ is *not* rational: it suffices to show that the set of derivatives

$$\{\, \sigma^{(0)}, \, \sigma^{(1)}, \, \sigma^{(2)}, \, \ldots \} \;\subseteq\; \mathbb{R}^\omega \tag{14.10}$$

contains infinitely many linearly independent vectors. Here is an example.

Example 399. Consider the stream $\sigma \in \mathbb{R}^\omega$ defined by

$$\begin{aligned} \sigma &= (1,1,0,1,0,0,1,0,0,0,1,0,0,0,0,\ldots) \\ &= 1 + \mathsf{X} + \mathsf{X}^3 + \mathsf{X}^6 + \mathsf{X}^{10} + \cdots \\ &= \sum_{k=0}^{\infty} \mathsf{X}^{k(k+1)/2} \end{aligned}$$

The set (14.10) above contains the following infinitely many linearly independent streams:

$$\begin{array}{l} (1,\ldots) \\ (0,1,\ldots) \\ (0,0,1,\ldots) \\ (0,0,0,1,\ldots) \\ \cdots \end{array}$$

Thus σ is not rational. □

Exercise 400. Decide whether the streams

$$\sigma \;=\; \sum_{k=0}^{\infty} \mathsf{X}^{2k} \qquad\qquad \tau \;=\; \sum_{k=0}^{\infty} \mathsf{X}^{k^2}$$

are rational or not. (Can you explain the connection with Exercise 194?) □

Exercise 401. In Section 339, we defined the streams of the *Catalan* numbers [OEI, A000108] and the *(large) Schröder* numbers [OEI, A006318] by the SDEs

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

Show that both σ and τ are non-rational. □

For future reference, we collect all the characterisations of rational streams that we have obtained in Theorems 332, 391, 394, and 398, in one theorem:

Theorem 402 (characterising rational streams). *For every $\sigma \in \mathbb{R}^\omega$, the following are equivalent:*

(1) σ is rational.

(2) σ can be defined by a finite linear system of SDEs.

(3) σ can be defined by a finite dimensional linear stream system.

(4) σ can be represented by a finite weighted stream automaton.

(5) the subspace $\langle\sigma\rangle \subseteq \mathbb{R}^\omega$ generated by σ is finite dimensional.

Exercise 403. Applying the generalised powerset construction to each of the following two weighted stream automata

$$\begin{tikzcd} s_0|1 \arrow[loop left, "1"] \arrow[r, bend left, "1"] & s_1|0 \arrow[loop above, "3"] \arrow[l, bend left, "1"] \arrow[r, bend left, "2"] & s_2|0 \arrow[loop above, "5"] \arrow[l, bend left, "2"] \arrow[r, bend left, "3"] & \cdots \arrow[l, bend left, "3"] \end{tikzcd}$$

$$\begin{tikzcd} t_0|1 \arrow[loop left, "1"] \arrow[r, bend left, "1"] & t_1|0 \arrow[loop above, "2"] \arrow[l, bend left, "1"] \arrow[r, bend left, "1"] & t_2|0 \arrow[loop above, "3"] \arrow[l, bend left, "2"] \arrow[r, bend left, "1"] & \cdots \arrow[l, bend left, "3"] \end{tikzcd}$$

yields a linear stream system of infinite dimension. Show that their minimisation is still infinite dimensional and conclude that the behaviour of both automata is non-rational. (The state s_0 represents the stream of the factorial numbers, whereas the state t_0 represents the *Bell* numbers [OEI, A000110]. For more examples of similar infinite weighted stream automata, see [Rut03b].) □

14.6 Weighted automata with inputs

There is a more general notion of weighted automaton that we shall briefly mention here. Details can be found in the references below.

A *weighted automaton with inputs* from a set A and weights in $\mathbb{R}$ is a pair

$$(S,\, \langle \mathsf{o}, \mathsf{tr} \rangle) \qquad \langle \mathsf{o}, \mathsf{tr} \rangle \colon S \to \mathbb{R} \times (\mathbb{R}^S_\omega)^A \qquad \mathsf{o} \colon S \to \mathbb{R} \qquad \mathsf{tr} \colon S \to (\mathbb{R}^S_\omega)^A$$

where the set $(\mathbb{R}^S_\omega)^A$ is the set of all functions from A to $\mathbb{R}^S_\omega$ (the latter was defined in (14.1)). The transition function tr assigns to any input $a \in A$ and state $s \in S$ a function $\mathsf{tr}(a)(s) \colon S \to \mathbb{R}$, of finite support. This function $\mathsf{tr}(a)(s)$ maps any state $t \in S$ to $\mathsf{tr}(a)(s)(t) \in \mathbb{R}$, which we call the weight of the transition from s to t *on input* a.

Weighted automata with inputs generalise weighted *stream* automata, which are obtained as a special case by taking $A = \mathbf{1}$, the one element set.

The generalised powerset construction of Definition 382, which transformed weighted stream automata to linear stream systems, can be easily adapted to take the inputs from A into account, with the obvious definitions of $\widehat{\{\cdot\}}$ and $\langle \mathsf{o}^\sharp,\, \mathsf{tr}^\sharp \rangle$:

$$\begin{tikzcd} S \arrow[r, "\widehat{\{\cdot\}}"] \arrow[d, "{\langle \mathsf{o}, \mathsf{tr} \rangle}"'] & (\mathbb{R}^S_\omega)^A \arrow[dl, dashed, "{\langle \mathsf{o}^\sharp,\, \mathsf{tr}^\sharp \rangle}"] \\ \mathbb{R} \times (\mathbb{R}^S_\omega)^A & \end{tikzcd} \tag{14.11}$$

This construction transforms $(S,\, \langle \mathsf{o}, \mathsf{tr} \rangle)$ into $((\mathbb{R}^S_\omega)^A,\, \langle \mathsf{o}^\sharp,\, \mathsf{tr}^\sharp \rangle)$, which is a *Moore* automaton, as defined in Section 13.3, with inputs from A and outputs in $\mathbb{R}$. Because $(\mathbb{R}^S_\omega)^A$ is again a free vector space on S, it is sometimes also called a *linear weighted automaton*.

As before, the behaviour of S can now be defined by finality, composing the embedding $\widehat{\{\cdot\}}$ with the unique homomorphism from $(\mathbb{R}^S_\omega)^A$ to the final Moore automaton:

$$S \xrightarrow{\widehat{\{\cdot\}}} (\mathbb{R}^S_\omega)^A \xrightarrow{[\![-]\!]} \mathbb{R}^{A^*}$$

which, as we saw in Exercise 354, is given by the set $\mathbb{R}^{A^*}$ of all weighted languages.

We conclude this section with a discussion of the coalgebraic type of weighted stream automata and weighted automata with inputs.

Remark 404. For those who have already read Chapters 2 to 5: weighted stream automata with weights in $\mathbb{R}$ are wsA-coalgebras for the functor

$$\begin{array}{lrcll} \mathsf{wsA} \colon \mathsf{Set} \to \mathsf{Set} & \mathsf{wsA}(S) & = & \mathbb{R} \times \mathbb{R}^S_\omega & (S \in \mathsf{Set}) \\ \mathsf{wsA}(f) \colon \mathbb{R} \times \mathbb{R}^S_\omega \to \mathbb{R} \times \mathbb{R}^T_\omega & \mathsf{wsA}(f) & = & 1_\mathbb{R} \times \mathbb{R}^f_\omega & (f \colon S \to T) \end{array}$$

Here $1_\mathbb{R} \colon \mathbb{R} \to \mathbb{R}$ is the identity function on $\mathbb{R}$ and

$$\mathbb{R}^f_\omega \colon \mathbb{R}^S_\omega \to \mathbb{R}^T_\omega$$

is defined, for $\phi \in \mathbb{R}^S_\omega$, $t \in T$, by

$$\mathbb{R}^f_\omega(\phi)\colon T \to \mathbb{R} \qquad\qquad \mathbb{R}^f_\omega(\phi)(t) \;=\; \sum_{s \in f^{-1}(t)} \phi(s) \tag{14.12}$$

Weighted automata with inputs from A and weights in $\mathbb{R}$ are wA-coalgebras for the functor

$$\begin{array}{lll} \mathsf{wA}\colon \mathsf{Set} \to \mathsf{Set} & \mathsf{wA}(S) \;=\; \mathbb{R} \times (\mathbb{R}^S_\omega)^A & (S \in \mathsf{Set}) \\ \mathsf{wA}(f)\colon \mathbb{R} \times (\mathbb{R}^S_\omega)^A \to \mathbb{R} \times (\mathbb{R}^T_\omega)^A & \mathsf{wA}(f) \;=\; 1_\mathbb{R} \times (\mathbb{R}^f_\omega)^A & (f\colon S \to T) \end{array}$$

where the definition of $(\mathbb{R}^f_\omega)^A$ is the obvious extension of the definition of $\mathbb{R}^f_\omega$ in (14.12).

In the present chapter, we have defined the behaviour of weighted stream automata by transforming them into (linear) stream systems and above, we have described a similar procedure for weighted automata with inputs. In both cases, the coalgebraic types – wsA and wA defined above – do not play a role. The situation is in that respect similar to our discussion in Remark 233 of the coalgebraic semantics of non-deterministic automata.

At the same time, it *is* possible to develop a coalgebraic semantics for weighted (stream) automata, leading to notions of wsA- and wA-bisimulation that *do* take the branching structure of the transitions into account, similar again to what we saw for non-deterministic automata. Details can be found in [BBB$^+$12]. □

14.7 Discussion

Weighted automata (with inputs) were introduced by Schützenberger [Sch61]. A broad overview of the classical theory of weighted automata can be found in the handbook [DKV09]. The present chapter on weighted *stream* automata is based on parts of [Rut08], [BBB$^+$12]and [HKR17]; see also [BKK17]. We assumed that weights are real numbers, but one can be more general by taking weights in any *field*, or even a *ring* or *semi-ring*. Parts of the present story still apply, see again [BBB$^+$12] and [HKR17] for further details. The powerset construction from Definition 382 generalises the powerset construction for non-deterministic automata from Definition 221. At the same time, Definition 382 is a special instance of a more general coalgebraic construction, see [SBBR13]. For a formulation of Theorem 398 in classical automata theory, see [BR88]. For weighted automata with inputs, coinduction up-to-linearity was studied by Winter in [Win14, Win15], where bisimulation up-to-linearity is called bisimulation up to linear combinations. For a brief discussion on general coalgebraic up-to techniques, see Section 15.5.

Chapter 15

Universal coalgebra

Universal coalgebra studies properties that hold for many different types of coalgebras at the same time. Although a systematic account of universal coalgebra is beyond the scope of this book, we will discuss a number of elementary examples of such universal properties. At the end of this chapter, in Section 15.6, we will mention references to various general accounts of universal coalgebra.

Reasoning in universal coalgebra uses the language of category theory, which was introduced in Chapter 2. More generally, the present chapter builds on Chapters 2 to 5.

15.1 Lambek's Lemma

Here is a first and elegant example of a property that holds for *any type* of coalgebra. Looking at the various instances of final coalgebras that we have encountered so far, we see that their structure map is always an isomorphism.

Exercise 405. Prove that the structure map of the final coalgebra of streams

$$\langle \mathsf{i}_{st}, \mathsf{d}_{st} \rangle \colon A^{\omega} \to (A \times A^{\omega})$$

introduced in Section 7.2, is an isomorphism. Prove that both

$$\langle \mathsf{i}_{l}, \mathsf{d}_{l} \rangle \colon \mathcal{P}(A^{*}) \to \big(2 \times \mathcal{P}(A^{*})^{A}\big)$$

which is the final automaton, from Section 8.3, and

$$\langle \mathsf{i}_{pl}, \mathsf{d}_{pl} \rangle \colon \mathsf{pL} \to \big(2 \times (\mathsf{ones} + \mathsf{pL}^{A})\big)$$

which is the final partial automaton, from Theorem 213), are isomorphisms. □

As it turns out, the structure map of *any* final coalgebra is an isomorphism.

Theorem 406 (Lambek's Lemma [Lam68]**).** *Let (B, β) be a* final *F-coalgebra. Then the structure map $\beta\colon B \to F(B)$ is an isomorphism.*

Proof: If (B, β) is an F-coalgebra then so is $(F(B), F(\beta))$. We observe that the structure map $\beta\colon B \to F(B)$ is a homomorphism of coalgebras:

$$\begin{tikzcd} B \arrow[r, "\beta"] \arrow[d, "\beta"'] & F(B) \arrow[d, "F(\beta)"] \\ F(B) \arrow[r, "F(\beta)"'] & F(F(B)) \end{tikzcd}$$

Since (B, β) is final, there exists a unique homomorphism:

$$\begin{tikzcd} F(B) \arrow[r, dashed, "\exists!\,\alpha"] \arrow[d, "F(\beta)"'] & B \arrow[d, "\beta"] \\ F(F(B)) \arrow[r, "F(\alpha)"'] & F(B) \end{tikzcd}$$

Composing two homomorphisms gives again a homomorphism, so $\alpha \circ \beta\colon B \to B$ is a homomorphism. Since the identity $1_B\colon B \to B$ is always a homomorphism and since by finality of (B, β), there is only one, it follows that $\alpha \circ \beta = 1_B$. For the converse,

$$\begin{aligned} \beta \circ \alpha &= F(\alpha) \circ F(\beta) && (\alpha \text{ is a homomorphism}) \\ &= F(\alpha \circ \beta) && (\text{functors preserve composition}) \\ &= F(1_B) \\ &= 1_{F(B)} && (\text{functors preserve identities}) \end{aligned}$$

□

Exercise 407. Lambek's Lemma was originally formulated for algebras: the structure map of every *initial algebra* is an isomorphism. Show that this follows from Theorem 406 by duality. □

Exercise 408 (non-existence of final coalgebras). Lambek's Lemma can be used to show that for certain types, a final coalgebra does *not* exist. For instance, consider the *powerset* functor $\mathcal{P}\colon \mathsf{Set} \to \mathsf{Set}$ (see Section 16.10 for its definition). A $\mathcal{P}$-coalgebra (S, α) consists of a set S of states and a transition function

$$\alpha\colon S \to \mathcal{P}(S)$$

which maps a state $s \in S$ to a *set* $\alpha(s) \subseteq S$ of successor states. Prove that there does not exist a final $\mathcal{P}$-coalgebra. Next consider the functor

$$\mathcal{P}_+\colon \mathsf{Set} \to \mathsf{Set} \qquad \mathcal{P}_+(X) = \{V \mid V \subseteq X \text{ and } V \neq \emptyset\}$$

Does there exist a final $\mathcal{P}_+$-coalgebra? □

15.2 Homomorphisms

Many of the elementary properties of homomorphisms of stream systems and automata hold for coalgebras of any type. Here is a basic example.

Theorem 409. *Let $f\colon (A,\alpha) \to (B,\beta)$ be an F-homomorphism and let $g\colon B \to A$ be an arrow in $\mathcal{C}$ that is an inverse of f:*

$$g \circ f \;=\; 1_A \qquad\qquad f \circ g \;=\; 1_B$$

Then g is an F-homomorphism.

Proof: If g is an inverse of f then

$$\begin{aligned}
\alpha \circ g &= 1_{F(A)} \circ \alpha \circ g \\
&= F(1_A) \circ \alpha \circ g \\
&= F(g \circ f) \circ \alpha \circ g \\
&= F(g) \circ F(f) \circ \alpha \circ g \\
&= F(g) \circ \beta \circ f \circ g && (f \text{ is a homomorphism}) \\
&= F(g) \circ \beta \circ 1_B \\
&= F(g) \circ \beta
\end{aligned}$$

which proves that g is an F-homomorphism. □

Exercise 410. Here is another example of a general property of homomorphisms of coalgebras. Let $F\colon \mathsf{Set} \to \mathsf{Set}$ be a functor. We call an F-homomorphism $f\colon (A,\alpha) \to (B,\beta)$ *monic* if, for all $g, h\colon (C,\gamma) \to (A,\alpha)$,

$$f \circ g \;=\; f \circ h \quad\Longrightarrow\quad g \;=\; h$$

Prove that if a homomorphism is injective then it is monic. For the converse, consider a homomorphism $f\colon (A,\alpha) \to (B,\beta)$ that is monic. We shall see later, in Corollary 422, that if the functor F *preserves kernel pairs* then the kernel of any homomorphism is an F-bisimulation. Use this fact to prove that for such F, monics are injective. □

15.3 Coproducts of coalgebras

A *coproduct* of two objects $A, B \in \mathcal{C}$ is an object $A + B \in \mathcal{C}$ together with two arrows

$$\kappa_1\colon A \to A + B \qquad\qquad \kappa_2\colon B \to A + B$$

(called embeddings) with the following *universal property*: for all objects C and arrows $f\colon A \to C$ and $g\colon B \to C$ there exists a unique arrow (which we call) $[f, g]\colon A + B \to C$ satisfying

$$\begin{tikzcd} A \arrow[r, "\kappa_1"] \arrow[dr, bend right, "f"'] & A+B \arrow[d, dashed, "{[f,g]}"] & B \arrow[l, "\kappa_2"'] \arrow[dl, bend left, "g"] \\ & C & \end{tikzcd} \tag{15.1}$$

that is, $[f, g] \circ \kappa_1 = f$ and $[f, g] \circ \kappa_2 = g$.

Exercise 411. Prove that any two coproducts are isomorphic. □

Exercise 412. Describe the coproduct of two automata from Chapter 8. □

There is the following general fact: if the category $\mathcal{C}$ has coproducts then the category Coalg_F of F-coalgebras and F-homomorphisms has coproducts as well.

Theorem 413. *Consider two F-coalgebras (A, α) and (B, β) and let $A+B$ be a coproduct of A and B in the category $\mathcal{C}$. Then there exists a unique structure map*

$$\gamma\colon (A + B) \to F(A + B)$$

such that the coalgebra $(A+B, \gamma)$ is a coproduct of (A, α) and (B, β) in the category Coalg_F.

Proof: If we take in (15.1) $C = F(A + B)$, $f = F(\kappa_1) \circ \alpha$ and $g = F(\kappa_2) \circ \beta$,

$$\begin{tikzcd} A \arrow[r, "\kappa_1"] \arrow[d, "\alpha"'] & A+B \arrow[d, dashed, "\gamma", "\exists!"'] & B \arrow[l, "\kappa_2"'] \arrow[d, "\beta"] \\ F(A) \arrow[r, "F(\kappa_1)"'] & F(A+B) & F(B) \arrow[l, "F(\kappa_2)"] \end{tikzcd}$$

then the unique existence of the arrow

$$\gamma \;=\; [F(\kappa_1) \circ \alpha,\; F(\kappa_2) \circ \beta]$$

follows from the universal property of $A + B$. It is now a little exercise to show that the F-coalgebra $(A + B, \gamma)$ has the universal property of a coproduct in Coalg_F. □

Theorem 413 tells us not only that the coproduct of two coalgebras exists whenever coproducts exist in the underlying category, but also that the carrier of the coproduct of two coalgebras consists of the coproduct of their carriers.

Exercise 414. Prove that the F-coalgebra $(A + B, \gamma)$ in the proof of Theorem 413 is a coproduct in Coalg_F. □

Exercise 415. Be courageous and formulate and prove a version of Theorem 413 for other examples of so-called *categorical colimits* – of which coproducts are an example – such as push-outs or co-equalisers. Check one of the references in Section 15.6 below to see how Theorem 413 can be generalised to *arbitrary* colimits. □

Exercise 416. Describe the *product* of two automata from Chapter 8. What does the dualisation of Theorem 413 tell us about *products* of coalgebras? □

15.4 Bisimulation and finality

For well-behaved types of coalgebras, we shall prove that homomorphisms from a given coalgebra into a final coalgebra map every state of that coalgebra to – a canonical representation of – its bisimilarity equivalence class.

The well-behavedness mentioned above will be formulated with the help of the notion of *kernel pair*, which is a categorical generalization of the *equivalence relation* induced by a mapping of one set into another.

Definition 417 (kernel pair). A *kernel pair* for an arrow $f\colon A \to B$ in a category $\mathcal{C}$ is a pair of arrows in $\mathcal{C}$

$$(\,\pi_1\colon K \to A,\ \pi_2\colon K \to A\,)$$

such that $f \circ \pi_1 = f \circ \pi_2$ and such that the following universal property holds:

$$\begin{tikzcd}
\forall C \arrow[rd, dashed, "\exists! g"] \arrow[rrd, bend left, "\forall l"] \arrow[rdd, bend right, "\forall k"'] & & \\
& K \arrow[r, "\pi_2"] \arrow[d, "\pi_1"'] & A \arrow[d, "f"] \\
& A \arrow[r, "f"'] & B
\end{tikzcd}$$

That is, for all pairs $(k\colon C \to A,\ l : C \to A)$ such that $f \circ k = f \circ l$, there exists a unique arrow $g\colon C \to K$ such that $k = \pi_1 \circ g$ and $l = \pi_2 \circ g$. □

Exercise 418. Describe kernel pairs in the category Set of sets and functions. □

We say that a functor $F\colon \mathcal{C} \to \mathcal{C}$ *preserves kernel pairs* if for every kernel pair

$$(\pi_1\colon K \to A,\ \pi_2\colon K \to A)$$

for an arrow $f\colon A \to B$, the pair

$$(\,F(\pi_1)\colon F(K) \to F(A),\ F(\pi_2)\colon F(K) \to F(A)\,)$$

is a kernel pair for the arrow $F(f)\colon F(A) \to F(B)$.

Exercise 419. Show that the functors for stream systems in Chapter 7 and automata in Chapter 8 preserve kernel pairs. □

Theorem 420 (kernel pairs of F-homomorphisms). *Let $F\colon \mathcal{C} \to \mathcal{C}$ be a functor, $f\colon (A,\alpha) \to (B,\beta)$ an F-homomorphism, and let*

$$(\,\pi_1\colon K \to A,\ \pi_2\colon K \to A\,)$$

be a kernel pair for the arrow $f\colon A \to B$ in the category $\mathcal{C}$. *If the functor F preserves kernel pairs then $(\,\pi_1\colon K \to A,\ \pi_2\colon K \to A\,)$ is a kernel pair* in the category Coalg_F *for the F-homomorphism $f\colon (A,\alpha) \to (B,\beta)$.*

Proof: If F preserves kernel pairs and $(\pi_1\colon K \to A,\ \pi_2\colon K \to A)$ is a kernel pair for the arrow $f\colon A \to B$ in the category $\mathcal{C}$, then

$$(\,F(\pi_1)\colon F(K) \to F(A),\ F(\pi_2)\colon F(K) \to F(A)\,)$$

is a kernel pair for the arrow $F(f)\colon F(A) \to F(B)$. Since

$$\begin{aligned} F(f) \circ \alpha \circ \pi_1 &= \beta \circ f \circ \pi_1 && (\alpha \text{ is an } F\text{-homomorphism}) \\ &= \beta \circ f \circ \pi_2 && ((\pi_1, \pi_2) \text{ is a kernel pair}) \\ &= F(f) \circ \alpha \circ \pi_2 && (\alpha \text{ is an } F\text{-homomorphism}) \end{aligned}$$

it follows from the universal property of the kernel pair for $F(f)$ that there exists a unique arrow $\gamma\colon K \to F(K)$ such that

$$\begin{tikzcd} K \arrow[dr, dashed, "\exists!\gamma"] \arrow[drr, bend left, "\alpha\circ\pi_2"] \arrow[ddr, bend right, "\alpha\circ\pi_1"'] & & \\ & F(K) \arrow[r, "F(\pi_2)"] \arrow[d, "F(\pi_1)"] & F(A) \arrow[d, "F(f)"] \\ & F(A) \arrow[r, "F(f)"'] & F(B) \end{tikzcd}$$

The arrow γ turns K into an F-coalgebra (K,γ) and the arrows π_1 and π_2 into F-homomorphisms. It is now easily verified that the pair $(\pi_1\colon K \to A,\ \pi_2\colon K \to A)$ has the universal property of a kernel pair in the category Coalg_F. □

Exercise 421. Prove the following variation on Theorem 420: Let $F\colon \mathcal{C} \to \mathcal{C}$ be a functor and (A, α) and (B, β) two F-coalgebras, and let $A \times B$ be a *product* of A and B in the category $\mathcal{C}$. If the functor F preserves products (define what this means) then there exists a unique structure map

$$\gamma\colon (A \times B) \to F(A \times B)$$

such that $(A \times B, \gamma)$ is the product of (A, α) and (B, β) *in the category* Coalg_F.

Now take courage again and formulate and prove similar variations for other examples of so-called *categorical limits* – of which kernel pairs and products are examples – such as pullbacks and equalisers. Check one of the references in Section 15.6 to see that Theorem 420 can be generalised to *arbitrary* limits. □

Corollary 422 (kernel of F-homomorphism is a bisimulation). *Let $F\colon \mathsf{Set} \to \mathsf{Set}$ be a functor preserving kernel pairs. If $f\colon (S, \alpha) \to (T, \beta)$ is an F-homomorphism, then*

$$\mathsf{ker}(f) \;=\; \{(s, s') \in S \times S \mid f(s) = f(s')\}$$

is an F-bisimulation equivalence.

Proof: The kernel $\mathsf{ker}(f)$ of any function is an equivalence relation. By Theorem 420, there exists a function $\gamma\colon \mathsf{ker}(f) \to F(\mathsf{ker}(f))$ such that $(\mathsf{ker}(f), \gamma)$ is an F-coalgebra and $(\pi_1\colon \mathsf{ker}(f) \to S,\ \pi_2\colon \mathsf{ker}(f) \to S)$ is a kernel pair in Coalg_F. In particular, the projections π_1 and π_2 are F-homomorphisms:

$$\begin{tikzcd}
S \arrow[d, "\alpha"'] & \mathsf{ker}(f) \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] \arrow[d, dashed, "\exists ! \gamma"] & S \arrow[d, "\alpha"] \\
F(S) & F(\mathsf{ker}(f)) \arrow[l, "F(\pi_1)"] \arrow[r, "F(\pi_2)"'] & F(S)
\end{tikzcd}$$

which is precisely the definition of an F-bisimulation given in Definition 33. □

Exercise 423. Compare the proof of Corollary 422 with that of Proposition 71. □

Corollary 422 can now be used to prove that homomorphisms into a final coalgebra map every state to – a canonical representation of – its bisimilarity equivalence class. The theorem below is formulated for functors on the category of sets, which allows us to speak of states as *elements* of a coalgebra, but the theorem applies to functors on arbitrary categories as well.

Theorem 424 (finality and F-bisimilarity). *Let $F\colon \mathsf{Set} \to \mathsf{Set}$ be a functor that preserves kernel pairs, (S, α) an F-coalgebra, and (B, β) a final F-coalgebra. By finality, there exists a unique F-homomorphism*

$$\begin{tikzcd} S \arrow[r, dashed, "\exists!\, [\![-]\!]"] \arrow[d, "\alpha"'] & B \arrow[d, "\beta"] \\ F(S) \arrow[r, dashed, "F([\![-]\!])"'] & F(B) \end{tikzcd}$$

It satisfies, for all $s, t \in S$,

$$s \sim t \iff [\![s]\!] = [\![t]\!]$$

where we write $s \sim t$ if there exists an F-bisimulation relation containing (s, t).

Proof: Consider an F-bisimulation relation $R \subseteq S \times S$, with projections $\pi_1, \pi_2\colon R \to S$. By the definition of bisimulation, the projections are homomorphisms. Composition with $[\![-]\!]$ gives *two* homomorphisms from R to B. By finality, they must be equal, which proves the implication from left to right. For the converse, assume that $[\![s]\!] = [\![t]\!]$, that is, $(s, t) \in \mathsf{ker}([\![-]\!])$. Since F preserves kernel pairs, it follows from Corollary 422 that $\mathsf{ker}([\![-]\!])$ is an F-bisimulation, which proves $s \sim t$. □

We observe that Theorem 84 on streams and Theorem 156 on languages are both special instances of Theorem 424. The following coinduction proof principle for F-coalgebras is a special case of Theorem 424. It generalises the earlier versions for streams (Theorem 96) and languages (Theorem 160).

Theorem 425 (coinduction proof principle for F-coalgebras). *Let $F\colon \mathsf{Set} \to \mathsf{Set}$ be a functor preserving kernel pairs and let (B, β) be a final F-coalgebra. Then:*

$$s \sim t \iff s = t \qquad (s, t \in B)$$

Proof: This follows from Theorem 424 applied to $(S, \alpha) = (B, \beta)$ and the observation that the identity map is the only homomorphism from a final coalgebra to itself. □

Exercise 426. Compare Theorem 425 with the coinduction proof principle that we saw in Chapter 4, that is, Theorem 34. □

15.5 Enhanced coinduction

We have seen many examples of *enhanced coinduction*: the variation of coinduction based on the combination of algebraic and coalgebraic structure, where one constructs bisimulation relations *up-to* the closure under a given set of algebraic operators:

- for streams: bisimulation up-to-sum, Definition 105, and coinduction -sum, Proposition 106
- for automata: bisimulation up-to-equivalence, Definition 138, and coinduction up-to-equivalence, Proposition 139
- for languages: bisimulation up-to-congruence, Definition 171, and coinduction up-to-congruence, Proposition 172
- for non-deterministic automata: bisimulation up-to-union, Definition 228, and coinduction up-to-union, Proposition 229
- for systems of stream differential equations: bisimulation up-to-Σ, Definition 254, and coinduction up-to-Σ, Proposition 255
- for linear stream systems: bisimulation up-to-linearity, Definition 386, and coinduction up-to-linearity, Proposition 387

Ultimately, all these variations are instances of one and the same, general principle. For a universal coalgebraic account of enhanced coinduction, see [RBR13a], [Rot16], [RBB$^+$17] and [BPPR17]. Coalgebraic coinduction up-to techniques already occur in the work of Lenisa [Len99] and Bartels [Bar04].

15.6 Discussion

The present chapter is based on [Rut96, Rut00c], where many more general properties of coalgebras may be found. Several results presented there apply only to coalgebras of which the functor type satisfies certain well-behavedness conditions (for instance, the functor should preserve weak pullbacks). These conditions were studied in more detail, and in some cases relaxed, in [Gum98]. There exists by now much literature on universal coalgebra. Here is an incomplete selection for further reading, in chronological order: [Tur96], [TP97], [MD97], [Mos99], [Acz00], [Kur01b], [Kur01a] [Gum01], [Pat03] [Bar04], [Adá05], [AMV05], [KR05], [Kup06], [Ven06], [Han09], [Sil10], [JR11], [Sil15], [Rot16], [Jac16], [Kur16], [AMM18].

Chapter 16

Notation and preliminaries

16.1 Numbers

We define the following sets of numbers:

- $\mathbb{N}$: the set of natural numbers $0, 1, 2, \ldots$
- $\mathbb{Z}$: the set of integer numbers $0, 1, -1, 2, -2, \ldots$
- $\mathbb{Q}$: the set of rational numbers n/m, with $n, m \in \mathbb{Z}$ and $m \neq 0$
- $\mathbb{Q}_{odd}$: the set of rational numbers n/m, with $n, m \in \mathbb{Z}$ and m odd
- $\mathbb{R}$: the set of real numbers

16.2 Functions

Functions are sometimes also called *maps*. For a set X, the *identity function* is given by

$$1_X \colon X \to X \qquad\qquad 1_X(x) \;=\; x$$

For functions $f \colon X \to Y$ and $g \colon Y \to Z$, we define the *composition* of f and g by

$$g \circ f \colon X \to Z \qquad\qquad (g \circ f)(x) \;=\; g(f(x))$$

We pronounce $g \circ f$ as: g after f. For a function $f \colon X \to X$, we define

$$f^0 \;=\; 1_X \qquad\qquad f^{n+1} \;=\; f \circ f^n$$

The *image* of $f \colon X \to Y$ is defined by

$$\mathsf{im}(f) \;=\; \{y \in Y \mid \exists\, x \in X,\ y = f(x)\,\}$$

For a subset $V \subseteq X$, the *image of* V *under* f is defined by

$$f(V) \;=\; \{y \in Y \mid \exists\, v \in V,\; y = f(v)\,\}$$

It follows that $\mathsf{im}(f) = f(X)$. The *graph* of $f\colon X \to Y$ is defined by

$$\mathsf{graph}(f) \;=\; \{(x,y) \in X \times Y \mid\; y = f(x)\,\}$$

We define the *kernel* of a function $f\colon X \to Y$ by

$$\mathsf{ker}(f) \;=\; \{(x,x') \in X \times X \mid\; f(x) = f(x')\,\}$$

For a linear map $L\colon V \to W$ between vector spaces V and W, we will also use

$$\mathsf{ker}(L) \;=\; \{\, v \in V \mid\; L(v) = 0\,\}$$

For sets X and A, we define

$$X^A \;=\; \{\, f \mid f\colon A \to X\,\}$$

For a function $g\colon X \to Y$, we define

$$g^A\colon X^A \to Y^A \qquad\qquad g^A(f) \;=\; g \circ f\colon A \to Y \qquad\qquad (f\colon A \to X)$$

16.3 Products and sums

The *(Cartesian) product* of two sets X and Y is

$$X \times Y \;=\; \{(x,y) \mid x \in X,\; y \in Y\}$$

The *projections* of $X \times Y$ are

$$X \xleftarrow{\pi_1} X \times Y \xrightarrow{\pi_2} Y \qquad\qquad \pi_1((x,y)) \;=\; x \qquad\qquad \pi_2((x,y)) \;=\; y$$

The *product* of two functions $f_1\colon X_1 \to Y_1$ and $f_2\colon X_2 \to Y_2$ is

$$f_1 \times f_2\colon\; (X_1 \times X_2) \;\to\; (Y_1 \times Y_2) \qquad\qquad (f_1 \times f_2)(x_1,x_2) \;=\; (f_1(x_1), f_2(x_2))$$

The *pairing* of two functions $f\colon Z \to X$ and $g\colon Z \to Y$ is

$$\langle f,g\rangle\colon Z \to X \times Y \qquad\qquad \langle f,g\rangle(z) \;=\; (f(z), g(z))$$

The *sum* of two sets X and Y is

$$X + Y \;=\; X \,\dot{\cup}\, Y$$

where the *disjoint union* is defined by

$$X \dot{\cup} Y \;=\; \{(x,0) \mid x \in X\} \cup \{(y,1) \mid y \in Y\}$$

The *embeddings* of $X+Y$ are

$$X \xrightarrow{\kappa_1} X+Y \xleftarrow{\kappa_2} Y \qquad \kappa_1(x) \;=\; (x,0) \qquad \kappa_2(y) \;=\; (y,1)$$

The *sum*

$$f_1 + f_2 \colon (X_1 + X_2) \;\to\; (Y_1 + Y_2)$$

of two functions $f_1 \colon X_1 \to Y_1$ and $f_2 \colon X_2 \to Y_2$ is defined by

$$(f_1+f_2)(x_1,0) \;=\; (f_1(x_1),0) \qquad (f_1+f_2)(x_2,1) \;=\; (f_2(x_2),1)$$

The *co-pairing* of two functions $f \colon X \to Z$ and $g \colon y \to Z$

$$[f,g] \colon (X+Y) \to Z$$

is defined by

$$[f,g]((x,0)) \;=\; f(x) \qquad [f,g]((y,1)) \;=\; g(y)$$

16.4 Subsets

We denote the set of all *subsets* of a set X by

$$\mathcal{P}(X) \;=\; \{V \mid V \subseteq X\}$$

The set $\mathcal{P}(X)$ is called the *powerset* of the set X. The *empty set* has no elements and is denoted by $\emptyset$. For

$$\mathcal{V} \;\subseteq\; \mathcal{P}(X)$$

we define the *union* and the *intersection* of $\mathcal{V}$ by

$$\bigcup \mathcal{V} \;=\; \{x \in X \mid \exists V \in \mathcal{V},\ x \in V\} \qquad \bigcap \mathcal{V} \;=\; \{x \in X \mid \forall V \in \mathcal{V},\ x \in V\}$$

16.5 Relations

A *relation* between two sets X and Y is a subset

$$R \subseteq X \times Y$$

The *projections* of R are

$$\begin{tikzcd} X & R \arrow[l, "\pi_1"'] \arrow[r, "\pi_2"] & Y \end{tikzcd} \qquad \pi_1((x,y)) = x \qquad \pi_2((x,y)) = y$$

The *inverse* of $R \subseteq X \times Y$ is

$$R^{-1} \subseteq Y \times X \qquad R^{-1} = \{(y,x) \in Y \times X \mid (x,y) \in R\}$$

The *composition* of two relations $R \subseteq X \times Y$ and $S \subseteq Y \times Z$ is

$$R \circ S = \{(x,z) \in X \times Z \mid \exists\, y \in Y,\ (x,y) \in R \text{ and } (y,z) \in S\}$$

which we pronounce as: R followed by S. A relation

$$R \subseteq X \times X$$

is an *equivalence* if it satisfies

$$\begin{aligned} & & x \leqslant x & \qquad \text{(reflexivity)} \\ (x \leqslant y \text{ and } y \leqslant z) & \implies & x \leqslant z & \qquad \text{(transitivity)} \\ x \leqslant y & \implies & y \leqslant x & \qquad \text{(symmetry)} \end{aligned}$$

For an equivalence relation R on X and $x \in X$, the R-*equivalence class* of x is

$$[x]_R = \{\, y \in X \mid (x,y) \in R \,\}$$

The *quotient* of X with respect to the equivalence relation R is

$$X/R = \{\, [x]_R \mid x \in X \,\}$$

that is, the set of R-equivalence classes. The *quotient function* of R is

$$q \colon X \to X/R \qquad q(x) = [x]_R$$

16.6 Words and languages

Let A be a set. A *word* over A is a finite sequence

$$(a_0, a_1, \ldots, a_{n-1}) \qquad (a_i \in A,\ n \geqslant 0)$$

Words are often simply written as strings

$$a_0 a_1 \cdots a_{n-1} \qquad (a_i \in A,\ n \geqslant 0)$$

The case $n = 0$ represents the *empty word*, which we denote by ε. We sometimes call the set A the *alphabet* and we call the elements $a \in A$ *letters*. The length of a word w is denoted by $|w|$ and is defined, for $w = a_0 a_1 \cdots a_{n-1}$, by n. The set of all words over A is denoted by

$$A^* \;=\; \{(a_0, a_1, \ldots, a_{n-1}) \mid a_i \in A,\ n \geqslant 0\}$$

Subsets of A^* are called a *languages*. We denote the empty language, that is, the empty subset, by $\emptyset$. For a letter $a \in A$ and a word $w = (a_0, a_1, \ldots, a_{n-1}) \in A^*$ we define

$$a \cdot w \;=\; (a, a_0, a_1, \ldots, a_n)$$

More generally, the *concatenation* of two words

$$v = (a_0, a_1, \ldots, a_{n-1}) \qquad\qquad w = (b_0, b_1, \ldots, b_{m-1})$$

is defined by

$$v \cdot w \;=\; (a_0, a_1, \ldots, a_{n-1}, b_0, b_1, \ldots, b_{m-1})$$

which is often written as vw. We write w^n for the n-fold concatenation of w with itself:

$$w^0 = \varepsilon \qquad\qquad w^{n+1} = w \cdot w^n \qquad\qquad (n \geqslant 0)$$

For instance, the word a^n consists of n occurrences of the letter a. We call a word v a *prefix* of a word w and write

$$v \;\leqslant\; w$$

if there exists a word u (possibly empty) such that $v \cdot u = w$.

16.7 Monoids, (semi-)rings, integral domains, fields

A *monoid*

$$(M, \cdot, e) \qquad\qquad \cdot : M \times M \to M \qquad\qquad e \in M$$

consists of a set M, a binary operation $\cdot$, and an identity element e, such that

$$(a \cdot b) \cdot c = a \cdot (b \cdot c) \qquad\qquad a \cdot e = e \cdot a = a$$

A monoid is called *commutative* if it moreover satisfies

$$a \cdot b = b \cdot a$$

A *semi-ring*

$$(R, +, \times, 0, 1) \qquad +\colon R \times R \to R \qquad \times\colon R \times R \to R \qquad 0, 1 \in R$$

consists of a set R equipped with two binary operations called addition and multiplication, such that

- $(R, +)$ is a commutative monoid with identity element 0:

$$(a + b) + c = a + (b + c) \qquad a + b = b + a \qquad a + 0 = a$$

- $(R, \times)$ is a monoid with identity element 1:

$$(a \times b) \times c = a \times (b \times c) \qquad\qquad 1 \times a = a \times 1 = a$$

- multiplication left and right distributes over addition:

$$a \times (b + c) = (a \times b) + (a \times c) \qquad (a + b) \times c = (a \times c) + (b \times c)$$

- multiplication by 0 annihilates R:

$$0 \times a = a \times 0 = 0$$

A *ring* is a semi-ring with the additional property that for each $a \in R$ there exists an additive inverse $-a \in R$ such that

$$a + (-a) = 0$$

A ring is *commutative* if multiplication is not only associative but also commutative:

$$a \times b = b \times a$$

An *integral domain* is a commutative ring in which multiplication has no zero-divisors:

$$a \times b = 0 \implies (\, a = 0 \text{ or } b = 0 \,)$$

A *field* is a commutative ring with the additional property that for each $a \in R$ with $a \neq 0$, there exists a multiplicative inverse $a^{-1} \in R$:

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

16.8 Pre-orders and partial orders

A *preorder*

$$(P, \leqslant) \qquad\qquad\qquad \leqslant \subseteq P \times P$$

consists of a set P and a binary relation $\leqslant$ satisfying

$$a \leqslant a \qquad \text{(reflexivity)}$$
$$(a \leqslant b \text{ and } b \leqslant c) \implies a \leqslant c \qquad \text{(transitivity)}$$

If a preorder moreover satisfies

$$(a \leqslant b \text{ and } b \leqslant a) \implies a = b \qquad \text{(anti-symmetry)}$$

then it is a *partial order*. On the other hand, if a preorder satisfies

$$a \leqslant b \implies b \leqslant a \qquad \text{(symmetry)}$$

then $\leqslant$ is an *equivalence relation*.

Let $(P, \leqslant)$ and $(Q, \leqslant)$ be two preorders. If a function $f \colon P \to Q$ satisfies

$$a \leqslant b \implies f(a) \leqslant f(b)$$

then f is *monotone*.

16.9 Vector spaces and linear maps

A *real vector space* (or: vector space over the field $\mathbb{R}$)

$$(V, +, \times, 0) \qquad +\colon V \times V \to V \qquad \times\colon \mathbb{R} \times V \to V \qquad 0 \in V$$

consists of a set V equipped with two binary operations called addition and *scalar multiplication*, such that, for all $u, v, w \in V$, $r, s \in \mathbb{R}$,

$$\begin{array}{lll} (u+v)+w = u+(v+w) & u+v = v+w & u+0 = u \\ r(v+w) = rv+rw & (r+s)v = rv+sv & r(sv) = (rs)v \\ 1v = v & v+(-v) = 0 & \end{array}$$

where we write $rv = r \times v$, and where the additive inverse $-v$ is assumed to exist, for every $v \in V$. A map $f \colon V \to W$ between two vector spaces V and W is *linear* it it satisfies

$$f(u+v) = f(u)+f(v) \qquad\qquad f(rv) = rf(v)$$

Linear maps between *finite dimensional* vector spaces can be represented by *matrices*, which are occasionally used the present book. We refer the reader to the literature on elementary linear algebra for the basic definitions and results, including the notion of a *basis* of a vector space, and the operations of *matrix multiplication* and *inversion*.

16.10 Functors on sets

The category Set consists of sets and functions. The category $\mathsf{Set} \times \mathsf{Set}$ has pairs of sets X_1 and X_2 as objects and pairs of functions $f_1 \colon X_1 \to Y_1$ and $f_2 \colon X_2 \to Y_2$ as arrows:

$$\text{objects: } (X_1, X_2) \qquad\qquad \text{arrows: } (f_1,\, f_2) \colon (X_1,\, X_2) \to (Y_1,\, Y_2)$$

The functors

$$F_\times \colon \mathsf{Set} \times \mathsf{Set} \to \mathsf{Set} \qquad\qquad F_+ \colon \mathsf{Set} \times \mathsf{Set} \to \mathsf{Set}$$

are defined by

$$\begin{array}{rclrcl} F_\times((X_1, X_2)) & = & X_1 \times X_2 & F_\times((f_1, f_2)) & = & (f_1 \times f_2) \colon (X_1 \times X_2) \to (Y_1 \times Y_2) \\ F_+((X_1, X_2)) & = & X_1 + X_2 & F_+((f_1, f_2)) & = & (f_1 + f_2) \colon (X_1 + X_2) \to (Y_1 + Y_2) \end{array}$$

with the product and sum of sets and functions as defined in Section 16.3.

The identity functor is

$$\mathsf{Id} \colon \mathsf{Set} \to \mathsf{Set} \qquad \mathsf{Id}(X) \;=\; X \qquad \mathsf{Id}(f) \;=\; f \colon X \to Y \qquad (f : X \to Y)$$

For every set A, we have the constant functor

$$F_A \colon \mathsf{Set} \to \mathsf{Set} \qquad F_A(X) \;=\; A \qquad F_A(f) \;=\; 1_A \colon A \to A \qquad (f : X \to Y)$$

and the exponentiation functor

$$\begin{array}{ll} (-)^A \colon \mathsf{Set} \to \mathsf{Set} & (-)^A(X) \;=\; X^A \;=\; \{\, g \mid g \colon A \to X \,\} \\ (-)^A(f) \colon X^A \to Y^A & (-)^A(f)(g) \;=\; f^A(g) \;=\; f \circ g \qquad (f : X \to Y,\, g : A \to X) \end{array}$$

The powerset functor is defined by

$$\begin{array}{lll} \mathcal{P} \colon \mathsf{Set} \to \mathsf{Set} & \mathcal{P}(X) \;=\; \{\, V \mid V \subseteq X \,\} & \\ \mathcal{P}(f) \colon \mathcal{P}(X) \to \mathcal{P}(Y) & \mathcal{P}(f)(V) \;=\; f(V) & (f \colon X \to Y,\, V \subseteq X) \end{array}$$

All the functors that occur in the present book are obtained by composing two or more of the functors above. For instance, the functor for deterministic automata with inputs from a given set S

$$\mathsf{dA} \colon \mathsf{Set} \to \mathsf{Set} \qquad\qquad \mathsf{dA}(S) \;=\; 2 \times S^A$$

is obtained by composing the constant functor F_2 and the exponentiation functor $(-)^A$ with the product functor $F_\times$.

16.11 On-line Encyclopedia of Integer Sequences

We will occasionally refer to OEIS: the On-line Encyclopedia of Integer Sequences (streams, in our terminology), founded by N.J.A. Sloane [OEI]. This is a rather wonderful database, containing descriptions of an incredible number of streams, presenting of each stream its history, various ways in which it can be constructed and characterised, streams to which it is related, etc. One can search the encyclopedia on-line, by simply typing in the first few initial values of a stream. Streams are numbered, and we will use, for instance, [OEI, A000045] to refer to the stream with number A000045 in OEIS, which is the stream $(0, 1, 1, 2, 3, 5, 8, \ldots)$ of Fibonacci numbers. Each such stream can be directly accessed on-line; for instance, one finds the stream of Fibonacci numbers at `https://oeis.org/A000045`.

Bibliography

[Acz88] P. Aczel. *Non-well-founded sets*. Number 14 in CSLI Lecture Notes. Center for the Study of Languages and Information, Stanford, 1988.

[Acz00] Peter Aczel. Algebras and coalgebras. In *Algebraic and Coalgebraic Methods in the Mathematics of Program Construction, International Summer School and Workshop, Oxford, UK, April 10-14, 2000, Revised Lectures*, pages 79–88, 2000.

[Adá05] Jirí Adámek. Introduction to coalgebra. *Theory and Applications of Categories*, 14(8):157–199, 2005.

[AFV01] L. Aceto, W.J. Fokkink, and C. Verhoef. Structural operational semantics. In J.A. Bergstra, A. Ponse, and S.A. Smolka, editors, *Handbook of Process Algebra*, pages 197–292. Elsevier, 2001.

[AM75] M.A. Arbib and E.G. Manes. Adjoint machines, state-behaviour machines, and duality. *Journal of Pure and Applied Algebra*, 6:313–344, 1975.

[AM82] M.A. Arbib and E.G. Manes. Parametrized data types do not need highly constrained parameters. *Information and Control*, 52(2):139–158, 1982.

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

[AMM18] Jirí Adámek, Stefan Milius, and Lawrence S. Moss. Fixed points of functors. *Journal of Logical and Algebraic Methods in Programming*, 95:41 – 81, 2018.

[AMV05] Jirí Adámek, Stefan Milius, and Jiri Velebil. A general final coalgebra theorem. *Mathematical Structures in Computer Science*, 15(3):409–432, 2005.

[AT11] Samson Abramsky and Nikos Tzevelekos. Introduction to categories and categorical logic. *CoRR*, abs/1102.1313, 2011.

[Awo10] S. Awodey. *Category theory*, volume 52 of *Oxford Logic Guides*. Oxford University Press, 2nd edition, 2010.

[Bar93] M. Barr. Terminal coalgebras in well-founded set theory. *Theoretical Computer Science*, 114(2):299–315, June 1993. Some additions and corrections were published in TCS Vol. 124, 1994, pp. 189–192.

[Bar03] Falk Bartels. Generalised coinduction. *Mathematical Structures in Computer Science*, 13(2):321–348, 2003.

[Bar04] F. Bartels. *On generalised coinduction and probabilistic specification formats.* PhD thesis, Vrije Universiteit, Amsterdam, 2004.

[Bas18] H. Basold. *Mixed Inductive-Coinductive Reasoning.* PhD thesis, Radboud Universiteit Nijmegen, 2018.

[BBB$^+$12] F. Bonchi, M. Bonsangue, M. Boreale, J. Rutten, and A. Silva. A coalgebraic perspective on linear weighted automata. *Inf. Comput.*, 211:77–105, 2012.

[BHKR15] M. Bonsangue, H. Hansen, A. Kurz, and J. Rot. Presenting distributive laws. *Logical Methods in Computer Science*, 11(3), 2015.

[BHPR17] H. Basold, H. Hansen, J.E. Pin, and J. Rutten. Newton series, coinductively: A comparative study of composition. *Mathematical Structures in Computer Science*, pages 1–29, 2017.

[BIM95] B. Bloom, S. Istrail, and A. Meyer. Bisimulation can't be traced. *J. ACM*, 42(1):232–268, 1995.

[BKK17] F. Bonchi, B. König, and S. Küpper. Up-to techniques for weighted systems. In *TACAS 2017*, volume 10205 of *LNCS*, pages 535–552, 2017.

[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.

[Bor17] M. Boreale. Algebra, coalgebra, and minimization in polynomial differential equations. In J. Esparza and A.S. Murawski, editors, *FOSSACS 2017, Proceedings*, volume 10203 of *LNCS*, pages 71–87, 2017.

[BP13] F. Bonchi and D. Pous. Checking NFA equivalence with bisimulations up to congruence. In *POPL '13*, pages 457–468, 2013.

[BP15] F. Bonchi and D. Pous. Hacking nondeterminism with induction and coinduction. *Commun. ACM*, 58(2):87–95, 2015.

[BPPR17] F. Bonchi, D. Petrisan, D. Pous, and J. Rot. A general account of coinduction up-to. *Acta Informatica*, 54(2):127–190, 2017.

[BR78] G. Birkhoff and G.-C. Rota. *Ordinary differential equations (third edition).* John Wiley and Sons, 1978.

[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.

[BvBL$^+$11] J. Baeten, D. van Beek, B. Luttik, J. Markovski, and J. Rooda. A process-theoretic approach to supervisory control theory. In *Proceedings of ACC 2011*, pages 4496–4501. IEEE, 2011.

[Cal13] G. Caltais. *Coalgebraic Tools for Bisimilarity and Decorated Trace Semantics*. PhD thesis, Radboud Universiteit, Nijmegen, 2013.

[Com74] L. Comtet. *Advanced combinatorics*. D. Reidel Publishing Company, 1974.

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

[Cos10] D. Costa. *Formal models for component connectors*. PhD thesis, Vrije Universiteit, Amsterdam, 2010.

[DKV09] Manfred Droste, Werner Kuich, and Heiko Vogler, editors. *Handbook of weighted automata*. Springer-Verlag, 2009.

[Eil74] S. Eilenberg. *Automata, languages and machines (Volume A)*. Pure and applied mathematics. Academic Press, 1974.

[Eil76] S. Eilenberg. *Automata, languages and machines (Volume B)*. Pure and applied mathematics. Academic Press, 1976.

[Gou93] F.Q. Gouvêa. *p-adic Numbers: An Introduction*. Springer, 1993.

[Gum98] H. Peter Gumm. *Elements of the general theory of coalgebra*. Course Notes for LUATCS. 1998.

[Gum01] H. Peter Gumm. Functors for coalgebras. *Algebra Universalis*, 45:135–147, 2001.

[Han09] H. Hansen. *Coalgebraic Modelling: Applications in Automata Theory and Modal Logic*. PhD thesis, Vrije Universiteit, Amsterdam, 2009.

[HCR06] H. Hansen, D. Costa, and J. Rutten. Synthesis of Mealy machines using derivatives. In *Proceedings of CMCS 2006*, volume 164(1) of *ENTCS*, pages 27–45. Elsevier Science Publishers, 2006.

[HH79] E.C.R. Hehner and R.N. Horspool. A new representation of the rational numbers for fast easy arithmetic. *SIAM Journal on Computing*, 8:124–134, 1979.

[Hin08] Ralf Hinze. Scans and convolutions - a calculational proof of Moessner's theorem. In *Proceedings of the 20th International Symposium on the Implementation and Application of Functional Languages (IFL '08)*, 2008.

[Hin11] Ralf Hinze. Concrete stream calculus: an extended study. *Journal of Functional Programming*, pages 1–70, 2011.

[HK71] J. Hopcroft and R. Karp. A linear algorithm for testing equivalence of finite automata. Technical Report 114, Dept. of Computer Science, Cornell University, December 1971.

[HKR17] H.H. Hansen, C. Kupke, and J. Rutten. Stream differential equations: specification formats and solution methods. *Logical Methods in Computer Science*, 12, 2017.

[HMU07] J.E. Hopcroft, R. Motwani, and J.D. Ullman. *Introduction to Automata Theory, Languages, and Computation, 3rd Edition.* Pearson, 2007.

[Hop71] John Hopcroft. An $n \log n$ algorithm for minimizing states in a finite automaton. In Z. Kohavi and A. Paz, editors, *Theory of Machines and Computations*, pages 189–196. Academic Press, New York, 1971.

[HR10] H.H. Hansen and J. Rutten. Symbolic synthesis of Mealy machines from arithmetic bitstream functions. *Scientific Annals of Computer Science*, XX:97–130, 2010.

[Jac16] Bart Jacobs. *Introduction to coalgebra. Towards mathematics of states and observations.* Cambridge University Press, 2016.

[JR11] B. Jacobs and J. Rutten. An introduction to (co)algebras and (co)induction. In Davide Sangiorgi and Jan Rutten, editors, *Advanced topics in bisimulation and coinduction*, volume 52 of *Cambridge Tracts in Theoretical Computer Science*, pages 38–99. Cambridge University Press, 2011.

[Kle56] S.C. Kleene. Representation of events in nerve nets and finite automata. In Shannon and McCarthy, editors, *Automata Studies*, pages 3–41. Princeton Univ. Press, 1956.

[Kli11] B. Klin. Bialgebras for structural operational semantics: An introduction. *Theoretical Computer Science*, 412:5043–5069, 2011.

[Koz97] D.C. Kozen. *Automata and computability.* Undergraduate Texts in Computer Science. Springer-Verlag, 1997.

[KPS16] R. Krebbers, L. Parlant, and A. Silva. Moessner's theorem: An exercise in coinductive reasoning in coq. In *Theory and Practice of Formal Methods*, volume 9660 of *Lecture Notes in Computer Science*, pages 309–324. Springer, 2016.

[KR05] Alexander Kurz and Jirí Rosický. Operations and equations for coalgebras. *Mathematical Structures in Computer Science*, 15(1):149–166, 2005.

[KS13] Dexter Kozen and Alexandra Silva. On Moessner's theorem. *The American Mathematical Monthly*, 120(2):131–139, February 2013.

[Kup06] C. Kupke. *Finitary Coalgebraic Logics*. PhD thesis, Universiteit van Amsterdam, 2006.

[Kur01a] A. Kurz. *Coalgebras and modal logic*. Course Notes for ESSLLI. 2001.

[Kur01b] Alexander Kurz. Specifying coalgebras with modal logic. *Theor. Comput. Sci.*, 260(1-2):119–138, 2001.

[Kur16] A. Kurz. Coalgebras and their logics. *ACM SIGACT News – Logic column*, 15, 2016.

[Lam68] J. Lambek. A fixed point theorem for complete categories. *Math. Zeitschr.*, 103:151–161, 1968.

[Lan02] S. Lang. *Algebra*. Graduate Texts in Mathematics. Springer, 2002.

[Len99] M. Lenisa. From set-theoretic coinduction to coalgebraic coinduction. In B. Jacobs and J. Rutten, editors, *Proceedings of CMCS'99*, volume 19 of *ENTCS*. Elsevier, 1999.

[LPW04] M. Lenisa, J. Power, and H. Watanabe. Category theory for operational semantics. *Theoretical Computer Science*, 327(1-2):135–154, 2004.

[LS97] F.W. Lawvere and S.H. Schanuel. *Conceptual mathematics*. Cambridge University Press, 1997.

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

[Mar16] Jasen Markovski. A process-theoretic approach to supervisory coordination under partial observation. *Sci. Comput. Program.*, 115-116:127–145, 2016.

[MD97] L.S. Moss and N. Danner. On the foundations of corecursion. *Logic Journal of the IGPL*, 5(2):231–257, 1997.

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

[Mil89] R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.

[ML71] S. Mac Lane. *Categories for the Working Mathematician*, volume 5 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1971.

[Moe51] A. Moessner. Eine Bemerkung über die Potenzen der natürlichen Zahlen. *Aus den Sitzungsberichten der Bayerische Akademie der Wissenschaften, Mathematisch-naturwissenschaftliche Klasse 1951 Nr.3*, 1951.

[Mos99] Lawrence S. Moss. Coalgebraic logic. *Ann. Pure Appl. Logic*, 96(1-3):277–317, 1999.

[NR11] Milad Niqui and Jan Rutten. A proof of Moessner's theorem by coinduction. *Higher-Order Symbolic Computing*, 24(3):191–206, 2011.

[NR13] M. Niqui and J. Rutten. Stream processing coalgebraically. *Sci. Comput. Program.*, 78(11):2192–2215, 2013.

[OEI] OEIS Foundation inc. (2018), The On-Line Encyclopedia of Integer Sequences. `https://oeis.org`.

[Paa52] I. Paasche. Ein neuer Beweis des Moessnerschen Satz. *Aus den Sitzungsberichten der Bayerische Akademie der Wissenschaften, Mathematisch-naturwissenschaftliche Klasse 1952 Nr.1*, 1952.

[Par81] D.M.R. Park. Concurrency and automata on infinite sequences. In P. Deussen, editor, *Proceedings 5th GI conference*, volume 104 of *Lecture Notes in Computer Science*, pages 167–183. Springer-Verlag, 1981.

[Pat03] D. Pattinson. *An introduction to the theory of coalgebra.* Course Notes for NASSLLI. 2003.

[PE98] D. Pavlović and M. 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.

[Per51] O. Perron. Beweis des Moessnerschen Satz. *Aus den Sitzungsberichten der Bayerische Akademie der Wissenschaften, Mathematisch-naturwissenschaftliche Klasse 1951 Nr.4*, 1951.

[Plo81] G.D. Plotkin. A structural approach to operational semantics. Report DAIMI FN-19, Aarhus University, Aarhus, September 1981.

[Ran58] G.N. Raney. Sequential functions. *J. Assoc. Comput. Mach.*, 3:177–180, 1958.

[RBB$^+$17] J. Rot, F. Bonchi, M. Bonsangue, D. Pous, J. Rutten, and A. Silva. Enhanced coalgebraic bisimulation. *Mathematical Structures in Computer Science*, 27(7):1236–1264, 2017.

[RBR13a] J. Rot, M. Bonsangue, and J. Rutten. Coalgebraic bisimulation-up-to. In *SOFSEM 2013: Theory and Practice of Computer Science, 39th International Conference on Current Trends in Theory and Practice of Computer Science,* *Špindlerův Mlýn, Czech Republic, January 26-31, 2013. Proceedings*, pages 369–381, 2013.

[RBR13b] J. Rot, M. Bonsangue, and J. Rutten. Coinductive proof techniques for language equivalence. In A. H. Dediu, C. Martín-Vide, and B. Truthe, editors, *Proc. LATA 2013*, volume 7810 of *LNCS*, pages 480–492. Springer, 2013.

[RBR16] J. Rot, M. Bonsangue, and J. Rutten. Proving language inclusion and equivalence by coinduction. *Information and Computation*, 246:62–76, 2016.

[Rot16] J. Rot. *Enhanced coinduction.* PhD thesis, Universiteit Leiden, 2016.

[RT93] J. Rutten and D. Turi. On the foundations of final semantics: non-standard sets, metric spaces, partial orders. In J.W. de Bakker, W.-P. de Roever, and G. Rozenberg, editors, *Proceedings of the REX Workshop on Semantics*, volume 666 of *LNCS*, pages 477–530. Springer-Verlag, 1993.

[Rut96] J. Rutten. Universal coalgebra: a theory of systems. Report CS-R9652, CWI, 1996. A slightly modified version of this report appeared in 2000 in Theoretical Computer Science Vol. 249(1).

[Rut98] J. Rutten. Automata and coinduction (an exercise in coalgebra). In D. Sangiorgi and R. de Simone, editors, *Proceedings of CONCUR'98*, volume 1466 of *LNCS*, pages 194–218, 1998.

[Rut99a] J. Rutten. Automata, power series, and coinduction: taking input derivatives seriously (extended abstract). In J. Wiedermann, P. van Emde Boas, and M. Nielsen, editors, *Proceedings of ICALP'99*, volume 1644 of *LNCS*, pages 645–654, 1999.

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

[Rut99c] J. Rutten. A note on coinduction and weak bisimilarity for while programs. *Theoretical Informatics and Applications (RAIRO)*, 33:393–400, 1999.

[Rut00a] J. Rutten. Behavioural differential equations: a coinductive calculus of streams, automata, and power series. Report SEN-R0023, CWI, 2000.

[Rut00b] J. Rutten. Coalgebra, concurrency, and control. In R. Boel and G. Stremersch, editors, *Discrete Event Systems*, pages 31–38. Kluwer, 2000.

[Rut00c] J. Rutten. Universal coalgebra: a theory of systems. *Theoretical Computer Science*, 249(1):3–80, 2000. Fundamental Study.

[Rut01] J. Rutten. Elements of stream calculus (an extensive exercise in coinduction). In S. Brooks and M. Mislove, editors, *Proceedings of MFPS 2001*, volume 45 of *ENTCS*, pages 1–66. Elsevier Science Publishers, 2001.

[Rut03a] J. Rutten. Behavioural differential equations: a coinductive calculus of streams, automata, and power series. *Theoretical Computer Science*, 308(1):1–53, 2003. Fundamental Study.

[Rut03b] J. Rutten. Coinductive counting with weighted automata. *Journal of Automata, Languages and Combinatorics*, 8(2):319–352, 2003.

[Rut05a] J. Rutten. A coinductive calculus of streams. *Mathematical Structures in Computer Science*, 15:93–147, 2005.

[Rut05b] J. Rutten. A tutorial on coinductive stream calculus and signal flow graphs. *Theoretical Computer Science*, 343(3):443–481, 2005.

[Rut06] J. Rutten. Algebraic specification and coalgebraic synthesis of Mealy automata. In *Proceedings of FACS 2005*, volume 160 of *ENTCS*, pages 305–319. Elsevier Science Publishers, 2006.

[Rut08] J. Rutten. Rational streams coalgebraically. *Logical Methods in Computer Science*, 4(3), 2008.

[RW87] P.J. Ramadge and W.M. Wonham. Supervisory control of a class of discrete event processes. *SIAM J. Control and Optimization*, 25:206–230, 1987.

[Sal52] H. Salié. Bemerkung zum einen Satz von Moessner. *Aus den Sitzungsberichten der Bayerische Akademie der Wissenschaften, Mathematisch-naturwissenschaftliche Klasse 1952 Nr.2*, 1952.

[Sal18] J. Salamanca. *Coequations and Eilenberg-type Correspondences.* PhD thesis, Radboud Universiteit Nijmegen, 2018.

[San12] Davide Sangiorgi. *Introduction to Bisimulation and Coinduction.* Cambridge University Press, 2012.

[SBBR13] A. Silva, F. Bonchi, M. Bonsangue, and J. Rutten. Generalizing determinization from automata to coalgebras. *Logical Methods in Computer Science*, 9(1), 2013.

[Sch61] M.P. Schützenberger. On the definition of a family of automata. *Information and Control*, 4(2-3):245–270, 1961.

[Sch68] F. Scheid. *Theory and problems of numerical analysis (Schaum's outline series).* McGraw-Hill, 1968.

[Sil10] A. Silva. *Kleene coalgebra.* PhD thesis, Radboud Universiteit, Nijmegen, 2010.

[Sil15] A. Silva. A short introduction to the coalgebraic method. *SIGLOG News*, 2(2):16–27, 2015.

[SR12] Davide Sangiorgi and Jan Rutten. *Advanced Topics in Bisimulation and Coinduction.* Cambridge University Press, 2012.

[TP97] D. Turi and G.D. Plotkin. Towards a mathematical operational semantics. In *Proc. 12*$^{\text{th}}$ *LICS Conf.*, pages 280–291. IEEE, Computer Society Press, 1997.

[Tur96] D. Turi. *Functorial operational semantics and its denotational dual.* PhD thesis, Vrije Universiteit, Amsterdam, 1996.

[Ven06] Y. Venema. Algebras and coalgebras. In J. van Benthem, P. Blackburn, and F. Wolter, editors, *Handbook of Modal Logic*, pages 331–426. Elsevier, 2006.

[WBR11] J. Winter, M. Bonsangue, and J. Rutten. Context-free languages, coalgebraically. In *Proceedings of CALCO 2011*, volume 6859 of *LNCS*, pages 359–376, 2011.

[WBR13] J. Winter, M. Bonsangue, and J. Rutten. Coalgebraic characterizations of context-free languages. *Logical Methods in Computer Science*, 9(3), 2013.

[WBR15] J. Winter, M. Bonsangue, and J. Rutten. Context-free coalgebras. *J. Comput. Syst. Sci.*, 81(5):911–939, 2015.

[Wil94] H.S. Wilf. *Generatingfunctionology.* Academic Press, 1994.

[Win14] J. Winter. *Coalgebraic characterizations of automata-theoretic classes.* PhD thesis, Radboud Universiteit, Nijmegen, 2014.

[Win15] Joost Winter. A completeness result for finite λ-bisimulations. In *FoSSaCS 2015*, pages 117–132, 2015.

Index

F-algebra, 19
F-bisimulation, 30
F-coalgebra, 21
F-congruence, 28
R-equivalence class, 240
X, 155
$\emptyset$, 239
even, 27, 56, 133
$\mathbb{N}$, 237
$\mathbb{Q}$, 237
$\mathbb{R}$, 237
$\mathbb{Z}$, 237
nat, 67, 70, 132
odd, 27, 56
ones, 131, 146
$\mathcal{P}(X)$, 239
$\sigma^{(n)}$, 53
$\sigma^{-\underline{n}}$, 167
$\sigma^{\langle n \rangle}$, 68, 180
$\sigma^{\underline{n}}$, 165
ε, 241
zip, 23, 57
a-derivative, 89
f^n, 237

algebra, 19
alphabet, 241
analytic
  differential equation, 171
  function, 171
  stream derivative, 168
anti-symmetry, 243
Arden's rule, 99
arrow, 11
automaton
  deterministic, 79
  infinite, 82
  Mealy, 189
  minimal, 100
  minimal for a language, 102
  Moore, 194
  non-deterministic, 119
  partial, 115
  weighted stream, 205

BDE, 147
Bell numbers, 223
bisimilarity
  of automata, 84
  of stream systems, 49
bisimulation
  functional, 51
  of deterministic automata, 83, 84
  of dynamical systems, 54
  of partial systems, 110
  of stream systems, 48, 49
  on F-coalgebras, 30
  proof method, 60, 61
  up-to, 8, 234
  up-to-Σ, 145
  up-to-congruence, 98
  up-to-equivalence, 85
  up-to-linearity, 215
  up-to-sum, 66
  up-to-union, 124
bitstream, 44, 195

carrier
  of a coalgebra, 21
  of an algebra, 19
Catalan numbers, 187, 207, 223
category, 11

opposite, 16
causal stream function, 146, 190, 191
circularity, 73
coalgebra, 21
coinduction, 7, 8
defining streams by, 55, 129
definitions by, 25, 27, 55
enhanced, 8, 234
fixed point, 31
proof principle, 30
for ext. natural numbers, 112
for automata, 91
for languages, 93
for streams, 60
up-to, 8, 107, 234
up-to-Σ, 145
up-to-congruence, 99
up-to-equivalence, 86
up-to-linearity, 215
up-to-sum, 66
up-to-union, 124
commuting diagram, 15
composition
of functions, 237
of streams, 162
concatenation
of languages, 93
of words, 241
congruence
bisimilarity is a, 138
closure, 98
on F-algebras, 28
context-free
stream, 187
convolution product, 151, 152
inverse, 159
coproduct
categorical, 16
of coalgebras, 230

derivative
analytic stream, 168
higher-order, 53
of a language, 89
of a regular operator, 94
of a stream, 52
of a stream function, 191
difference operator, 175
differential equation
analytic, 171
behavioural, 147
stream, 129
disjoint union, 239
distributive law, 149
duality, 15

embeddings, 16
empty
set, 239
word, 241
enhanced coinduction, 234
epic, 12
epimorphism, 39, 83
equivalence
class, 240
closure, 85
Myhill-Nerode, 102
relation, 240, 243
Euler expansion, 179
exponentiation of streams, 167, 210
extended natural numbers, 111

factorial numbers, 75, 133, 178, 207
Fibonacci numbers, 132, 177, 209
field, 242
final
F-coalgebra, 26
automaton of languages, 90
dynamical system, 54
Mealy automaton, 191, 195
Moore automaton, 195
object, 15
partial automaton, 117
partial system, 111
stream system, 53
fixed point

of a dynamical system, 38
of a monotone function, 32
formal power series, 148, 188, 195
functor, 14
fundamental theorem
of bitstream calculus, 198, 202
of calculus, 156
of difference calculus, 175
of stream calculus, 156

generating function, 157
graph, 238
GSOS format, 140, 143, 149

Hamming numbers, 134
higher-order
derivative, 53
SDE, 132
homomorphism
of algebras, 20
of coalgebras, 22
of deterministic automata, 82
of dynamical systems, 39
of Mealy automata, 192
of partial systems, 110
of stream systems, 46

image
of a function, 237
of a homomorphism, 48
induction
and coinduction, 25
definitions by, 25, 26
fixed point, 31
mathematical, 28
proof principle, 28
initial
F-algebra, 25
object, 15
value of
a language, 89
a regular operator, 94
a stream, 52
injective, 12
input derivative, 89
integral domain, 159, 242
inverse
additive, 243
convolution, 158, 159
image, 38
left, 40
multiplicative, 242
of a relation, 240
of an arrow, 229
shuffle, 167
isomorphic, 13
isomorphism, 83

kernel
of a function, 238
kernel pair
of F-homomorphism, 232
of an arrow, 231
preservation of, 232
Kleene
star, 93
Theorem, 104

Lambek's lemma, 228
language, 89, 241
accepted, 80
inclusion, 100
partial, 116
regular, 93
linear
map, 243
stream system, 210
system of SDEs, 181

matrix, 243
minimisation
and finality, 55
of automata, 100, 101
of linear stream systems, 220
of Mealy automata, 192
of stream systems, 51, 55
Moessner's theorem, 67, 69
monic, 12

monoid, 241
  as category, 12
  commutative, 242
monomorphism, 39, 83
monotone function, 243
morphism, 11
Myhill-Nerode equivalence, 102

numbers
  integer, 237
  natural, 237
  rational, 237
  real, 237

object, 11
OEIS, 245

Paasche's Theorem, 75
partial
  automata, 115
  order, 243
  system, 110
  system with outputs, 113
periodic
  state, 38
polynomial stream, 162
powerset, 239
  construction, 121
  functor, 14, 244
  generalised, 212
preorder, 243
  as category, 12
product
  Cartesian, 13, 238
  categorical, 13
  convolution, 151, 152
  Hadamard, 68, 132, 178
  infiltration, 178
  of stream systems, 50
  of two functions, 238
  shuffle, 60, 133, 164
projections, 13, 238, 240

quotient
  of stream system, 51
  w.r.t. equivalence relation, 240

rational
  bitstream function, 201
  stream, 162, 223
recurrence relation, 132
reflexivity, 240
regular
  language, 94
  operator, 93
relation, 240
  equivalence, 240, 243
ring, 242
  commutative, 242
  semi, 242

Schröder numbers, 187, 223
SDE, 129
  higher-order, 132
  system of, 131
shuffle
  inverse, 167
  product, 60, 133, 164
simulation relation, 100
splitting stream derivatives, 74, 207
state, 37
  accepting, 80
  non-accepting, 80
stream, 45
  calculus, 151
    fundamental theorem of, 156
  constant X, 155
  context-free, 187
  derivative, 52
  differential equation, 129
  eventually periodic, 58, 196, 199
  exponentiation, 167, 210
  initial value of, 52
  polynomial, 162
  rational, 162, 223
  system, 45
structure map

of a coalgebra, 21
of an algebra, 19
subautomaton, 87
subsystem
of dynamical system, 42
of stream system, 48
sum
infinite, 156
of two sets, 238
of two streams, 58, 64
support of a function, 205
surjective, 12
symmetry, 240
syntactic solution method, 134
system
dynamical, 37
linear stream, 210
partial, 110
stream, 45

Taylor
coefficients, 156
series, 156, 157, 166
transform, 172
transform
Laplace, 168
Newton, 175
Taylor, 172
transitivity, 240
type
of a coalgebra, 21, 34
of an algebra, 19
of an arrow, 12

universal properties, 15

vector space, 243

weight, 205
weighted
language, 195, 224
stream automaton, 205
while programs, 113
word, 241
empty, 241