Centrum voor Wiskunde en Informatica

Automata and coinduction (an exercise in coalgebra)

J.J.M.M. Rutten

Software Engineering (SEN)

SEN-R9803 May 31, 1998

Report SEN-R9803
ISSN 1386-369X

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

Automata and Coinduction (an exercise in coalgebra)

J.J.M.M. Rutten
CWI
*P.O. Box 94079, 1090 GB Amsterdam, The Netherlands**

Abstract
The classical theory of deterministic automata is presented in terms of the notions of *homomorphism* and *bisimulation*, which are the cornerstones of the theory of (universal) coalgebra. This leads to a transparent and uniform presentation of automata theory and yields some new insights, amongst which coinduction proof methods for language equality and language inclusion. At the same time, the present treatment of automata theory may serve as an introduction to coalgebra.

1991 Mathematics Subject Classification: 68Q10, 68Q55
1991 Computing Reviews Classification System: D.3, F.1, F.3
Keywords & Phrases: Automaton, coalgebra, homomorphism, bisimulation, coinduction, regular expression
Note: This report will appear in the proceedings of CONCUR '98.

*Email: janr@cwi.nl, URL: www.cwi.nl/~janr.

Contents

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

1 Introduction

The classical theory of deterministic automata is presented in terms of the notions of *homomorphism* and *bisimulation*, which are the cornerstones of the theory of (universal) coalgebra. This coalgebraic perspective leads to a transparent and uniform theory, in which the observation that the set $\mathcal{L}$ of all languages is a *final* automaton, plays a central role. The automaton structure on $\mathcal{L}$ is determined by the notion of (input) *derivative*, and gives rise to two new proof principles: 1. a coinduction proof method in terms of bisimulations for demonstrating the equality of languages, which is complete and, for regular languages, effective; and 2. a coinduction proof method in terms of *simulations* for proving language inclusion.

The paper is intended to be self-contained, and no prior knowledge of coalgebra is presupposed. Although the development of our theory has been entirely dictated by a coalgebraic perspective, no explicit reference to coalgebraic notions or results will be made (apart from Section 12). In this way, we hope that this paper may also serve as an introduction to coalgebra.

Sections 2 through 11 deal with (complete) deterministic automata, regular languages, minimization, and Kleene's theorem. Only after these sections, the connection between automata theory and coalgebra is discussed in detail, in Section 12. (For readers that do have some background in category theory and coalgebra, it may be instructive to read Section 12 immediately after having read Section 2.) In the remaining Sections 13 through 15, the coalgebraic approach is further illustrated by the treatment of so-called partial automata, which have transition functions that may be partial. Of special interest is an automaton of languages with infinite words. References to the literature have been collected in Section 16.

2 Deterministic automata

Let A be a (possibly infinite) set of input symbols. A *(deterministic) automaton* with input alphabet A is a triple $S = \langle S, o, t\rangle$ consisting of a set S of *states*, an *output function* $o : S \to 2$, and a *transition function* $t : S \to S^A$. Here 2 denotes the set $\{0,1\}$, and S^A is the set of all functions from A to S. The output function o indicates whether a state s in S is *terminating*[1] ($o(s) = 1$) or not ($o(s) = 0$). The transition function t assigns to a state s a function $t(s) : A \to S$, which specifies the state $t(s)(a)$ that is reached after an input symbol a has been consumed. We shall sometimes write $s{\downarrow}$ for $o(s) = 1$, $s{\uparrow}$ for $o(s) = 0$, and $s \xrightarrow{a} s'$ for $t(s)(a) = s'$.

Contrary to the standard definition, in the present setting both the state space S of an automaton and the set A of input symbols may be infinite. If both S and A are finite then we speak of a *finite automaton*. Another difference with the standard approach is that our automata do not have an initial state. (See Section 12 for a detailed motivation of the present definition of automaton.)

A *bisimulation* between two automata $S = \langle S, o, t\rangle$ and $S' = \langle S', o', t'\rangle$ is a relation $R \subseteq S \times S'$ with, for all s in S, s' in S', and a in A:

$$\text{if } s\,R\,s' \text{ then } \begin{cases} o(s) = o'(s') & \text{and} \\ t(s)(a)\;R\;t'(s')(a). \end{cases}$$

A bisimulation between S and itself is called a bisimulation *on* S. Unions and (relational) compositions of bisimulations are bisimulations again. We write $s \sim s'$ whenever there exists a bisimulation R with $s\,R\,s'$. This relation $\sim$ is the union of all bisimulations and, therewith, the greatest bisimulation. The greatest bisimulation on one and the same automaton, again denoted by $\sim$, is called the *bisimilarity* relation. It is an equivalence relation.

[1] Sometimes also called *accepting* or *final*.

The only thing one can 'observe' about a state of an automaton is whether it is terminating or not. One can also perform 'experiments', by offering an input symbol which then leads to a new state. Of this new state, we can of course observe again whether it is terminating or not. Two states that are related by a bisimulation relation are *observationally indistinguishable* in the sense that 1. they give rise to the same observations, and 2. performing on both states the same experiment will lead to two new states that are indistinguishable again.

A *homomorphism* between S and S' is any function $f : S \to S'$ with, for all s in S, $o(s) = o'(f(s))$ and, for all a in A, $f(t(s)(a)) = t'(f(s))(a)$.

An automaton $S' = \langle S', o', t' \rangle$ is a *subautomaton* of $S = \langle S, o, t \rangle$ if $S' \subseteq S$ and the inclusion function $i : S' \to S$ is a homomorphism. Given $\langle S, o, t \rangle$ and S', the functions o' and t' in that case are uniquely determined. For a state s in S, $\langle s \rangle$ denotes the subautomaton *generated* by s: it is the smallest subautomaton of S containing s, and can be obtained by including all states from S that are reachable via a finite number of transitions from s.

Homomorphisms map subautomata to subautomata: for a homomorphism $f : S \to T$ and subautomaton $S' \subseteq S$, $f(S')$ is a subautomaton of T. For s in S, moreover, $f(\langle s \rangle) = \langle f(s) \rangle$.

The notions of automaton, homomorphism and bisimulation are closely related: a function $f : S \to S'$ is a homomorphism if and only if its graph relation $\{\langle s, f(s) \rangle \mid s \in S\}$ is a bisimulation. And bisimulations are themselves automata: if R is a bisimulation between S and S', then $o_R : R \to 2$ and $t_R : R \to R^A$, given for $\langle s, s' \rangle$ in R and a in A by $o_R(\langle s, s' \rangle) = o(s) = o'(s')$ and $t_R(\langle s, s' \rangle)(a) = \langle t(s)(a), t'(s')(a) \rangle$, define an automaton $\langle R, o_R, t_R \rangle$.

For an example, let $A = \{a, b\}$ and consider the automata $S = \{s_1, s_2, s_3\}$ and $T = \{t_1, t_2\}$, with transitions and termination as specified by the following tables:

	a	b	
s_1	s_2	s_3	$\uparrow$
s_2	s_2	s_3	$\downarrow$
s_3	s_2	s_3	$\downarrow$

	a	b	
t_1	t_2	t_2	$\uparrow$
t_2	t_2	t_2	$\downarrow$

where, for instance, the second row of the first table denotes $s_2 \xrightarrow{a} s_2$, $s_2 \xrightarrow{b} s_3$, and $s_2{\downarrow}$. Then $\{\langle s_1, s_1 \rangle, \langle s_2, s_2 \rangle, \langle s_3, s_3 \rangle\}$ and $\{\langle s_2, s_3 \rangle, \langle s_2, s_2 \rangle, \langle s_3, s_3 \rangle\}$ are bisimulations on S; $\{s_2, s_3\} = \langle s_2 \rangle = \langle s_3 \rangle$ is a subautomaton of S; and $f : S \to T$ mapping s_1 to t_1, and s_2 and s_3 to t_2 is a homomorphism.

3 Languages

Let A^* be the set of all finite words over A. Prefixing a word w in A^* with an input symbol a in A is denoted by aw. Concatenation of words w and w' is denoted by ww'. Let ε denote the empty word. A *language* is any subset of A^*. The language *accepted* by a state s of an automaton $S = \langle S, o, t \rangle$ is $l_S(s) = \{a_1 \cdots a_n \mid s \xrightarrow{a_1} s_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} s_n{\downarrow}\}$, where $s_1 = t(s)(a_1)$ and $s_{i+1} = t(s_i)(a_{i+1})$, for $1 < i < n$.

Let $\mathcal{L} = \{L \mid L \subseteq A^*\}$ be the set of all languages. For a word w in A^*, the *w-derivative* of a language L is $L_w = \{v \in A^* \mid wv \in L\}$. A special case is the *a-derivative* $L_a = \{v \in A^* \mid av \in L\}$, for a in A, which can be used to turn the set $\mathcal{L}$ of languages into an automaton $\langle \mathcal{L}, o_{\mathcal{L}}, t_{\mathcal{L}} \rangle$, defined, for $L \in \mathcal{L}$ and $a \in A$, by

$$o_{\mathcal{L}}(L) = \begin{cases} 1 & \text{if } \varepsilon \in L \\ 0 & \text{if } \varepsilon \notin L \end{cases} \quad \text{and:} \quad t_{\mathcal{L}}(L)(a) = L_a.$$

That is,

$$L{\downarrow} \text{ iff } \varepsilon \in L, \quad \text{and:} \quad L \xrightarrow{a} L' \text{ iff } L' = L_a.$$

This automaton has the pleasing property that the language accepted by a state L in $\mathcal{L}$ is precisely L itself. This will be proved in Section 7, but is already illustrated by the following example. For $L = \{a, ab, ac\}$, there are the following transitions:

$$\{a, ab, ac\} \xrightarrow{a} \{\varepsilon, b, c\}\downarrow \xrightarrow{b,c} \{\varepsilon\}\downarrow,$$

where $\xrightarrow{b,c}$ means that there is both a b and a c transition, and where we have omitted transitions leading to the empty set, such as $\{a, ab, ac\} \xrightarrow{b} \emptyset$. It follows that $l_{\mathcal{L}}(L) = L$.

If the *behaviour* of a state is the language it accepts, then states in $\mathcal{L}$ could be said to 'do as they are'. For them, in other words, 'being is doing'.

4 Coinduction

The automaton $\mathcal{L} = \langle \mathcal{L}, o_{\mathcal{L}}, t_{\mathcal{L}} \rangle$ of languages satisfies, for all languages K and L,

$$\text{if } \; K \sim L \; \text{ then } \; K = L.$$

(The converse trivially holds.) This gives rise to the following *coinduction proof principle*: in order to prove the equality of languages K and L, it is sufficient to establish the existence of a bisimulation relation on $\mathcal{L}$ that includes the pair $\langle K, L \rangle$.

The above implication follows from the fact that for all words w in A^* of length n and for all languages K and L with $K \sim L$: if $w \in K$ then $w \in L$, which we show next by *induction* on n. First note that a bisimulation on $\mathcal{L}$ is any relation R such that for all K and L with $K \, R \, L$, $K\downarrow$ iff $L\downarrow$, and for any a in A, $K_a \, R \, L_a$. Now consider K and L with $K \sim L$. Because $\sim$ is a bisimulation, $\varepsilon \in K$ implies $\varepsilon \in L$. Next consider a word $w = aw'$, of length $n+1$, in K. Because $K \sim L$ also $K_a \sim L_a$. Because $w' \in K_a$ and the length of w' is n, it follows from the inductive hypothesis that $w' \in L_a$. Thus $w \in L$. This shows that $K \sim L$ implies $K \subseteq L$. Since $K \sim L$ implies $L \sim K$, also $L \subseteq K$.

5 Regular expressions

Let the set $\mathcal{R}$ of *regular expressions* be given by the following syntax:

$$E ::= \; 0 \mid \; 1 \mid \; a \in A \mid \; E + F \mid \; EF \mid \; E^*$$

Let the funcion $\lambda : \mathcal{R} \to \mathcal{L}$, which assigns to an expression E the language $\lambda(E)$ it represents, be defined by induction on the structure of E:

$$\begin{aligned}
\lambda(0) &= \emptyset \\
\lambda(1) &= \{\varepsilon\} \\
\lambda(a) &= \{a\} \\
\lambda(E+F) &= \lambda(E) + \lambda(F) \\
\lambda(EF) &= \lambda(E)\lambda(F) \\
\lambda(E^*) &= \lambda(E)^*,
\end{aligned}$$

where on the right hand side of these equations the following so-called *regular operators* are used: for languages K and L,

$$\begin{aligned}
K + L &= K \cup L \\
KL &= \{vw \mid v \in K \;\text{ and }\; w \in L\} \\
K^* &= \bigcup_{n \geq 0} K^n,
\end{aligned}$$

with $K^0 = \{\varepsilon\}$ and $K^{n+1} = KK^n$. Languages $L = \lambda(E)$ are called *regular languages*. Whenever convenient and harmless, we shall simply write E for $\lambda(E)$. Notably, 0, 1, and a will then denote the sets $\emptyset$, $\{\varepsilon\}$, and $\{a\}$, respectively.

The following rules for calculating the a-derivative L_a of a language L are easily verified:

$$\begin{aligned}
0_a &= 0 \\
1_a &= 0 \\
b_a &= \begin{cases} 1 & \text{if } b = a \\ 0 & \text{if } b \neq a \end{cases} \\
(K+L)_a &= K_a + L_a \\
(KL)_a &= \begin{cases} K_a\, L & \text{if } K\!\uparrow \\ K_a\, L + L_a & \text{if } K\!\downarrow \end{cases} \\
(K^*)_a &= K_a\, K^*
\end{aligned}$$

There are also the following rules for termination: $0\!\uparrow$, $1\!\downarrow$, $a\!\uparrow$, $K+L\!\downarrow$ iff $K\!\downarrow$ or $L\!\downarrow$, $KL\!\downarrow$ iff $K\!\downarrow$ and $L\!\downarrow$, $K^*\!\downarrow$. All these rules will be of great help when proving the equality of languages by means of coinduction, as we shall see in Section 6.

6 Proofs by coinduction

The use of coinduction is illustrated by first proving some of the familiar laws for the regular operators, and next some equalities of concrete expressions. We emphasize that the algebraic completeness of these laws in not the issue here. They merely serve as examples, and some of them will be used as lemma's in subsequent proofs.

The strength of the coinduction proof principle is that it works for any valid equality, and that it works in a uniform way: first define a relation consisting of the pair(s) of languages that you want to prove equal; then look at all possible transitions and continue to add pairs of resulting languages if they were not present yet. The original equality holds if and only if this process yields a bisimulation. For regular languages, the coinduction proof principle is effective: If the languages with which one starts are regular, then the construction of a bisimulation relation terminates in finitely many steps. This will be proved in Section 8.

Some laws

All the familiar laws for the regular operators can be proved by coinduction. Some of them are easily proved directly on the basis of the definitions of the regular operators, others are less straightforward. Below some of the following will be proved by coinduction:

$$\begin{aligned}
K + 0 &= K && (1)\\
K + K &= K && (2)\\
K + L &= L + K && (3)\\
(K + L) + M &= K + (L + M) && (4)\\
1K &= K && (5)\\
K1 &= K && (6)\\
K0 &= 0 && (7)\\
0K &= 0 && (8)\\
(KL)M &= K(LM) && (9)\\
1 + LL^* &= L^* && (10)\\
K(L + M) &= KL + KM && (11)\\
(L + M)K &= LK + MK && (12)
\end{aligned}$$

$$L\!\uparrow \wedge (K = LK + M) \;\Rightarrow\; K = L^*M \tag{13}$$
$$(K + L)^* = K^*(LK^*)^* \tag{14}$$
$$(K + L)^* = (K^*L)^*K^* \tag{15}$$

As a consequence of (4) and (9), brackets can often be omitted.

Although all of (1)–(9) are immediate from the definitions, we prove as an example equation (1) by coinduction. We show that

$$R = \{\langle K + 0,\, K\rangle \mid K \in \mathcal{L}\}$$

is a bisimulation; then (1) follows by coinduction. First note that $(K+0)\!\downarrow$ if and only if $K\!\downarrow$. And for any a in A,

$$\begin{array}{lll} & (K+0)_a & \\ = & K_a + 0_a & \\ = & K_a + 0 & \\ R & K_a. & \end{array}$$

Laws (2)-(9) can be proved similarly. Equality (10) follows by coinduction from the fact that

$$\{\langle 1 + LL^*,\, L^*\rangle \mid L \in \mathcal{L}\} \cup \{\langle L, L\rangle \mid L \in \mathcal{L}\}$$

is a bisimulation. For (11), one could try to prove that the relation $\{\langle K(L + M),\, KL + KM\rangle \mid K, L, M \in \mathcal{L}\}$ is a bisimulation. It turns out to be convenient to consider the (by (1)) larger set

$$R = \{\langle K(L + M) + N,\, KL + KM + N\rangle \mid K, L, M, N \in \mathcal{L}\}$$

instead. (Cf. the strengthening of the inductive hypothesis in an inductive argument.) We show that R is a bisimulation. Consider a in A and a pair $\langle K(L+M)+N,\, KL+KM+N\rangle$ in R. First note that $K(L + M) + N$ terminates if and only if $KL + KM + N$ does. Suppose that $K\!\downarrow$ (the case that $K\!\uparrow$ is similar and a little easier). Then

$$\begin{array}{lll} & (K(L + M) + N)_a & \\ = & K_a\,(L + M) + L_a + M_a + N_a & \\ R & K_aL + K_aM + L_a + M_a + N_a & \\ = & K_aL + L_a + K_aM + M_a + N_a & \text{[by (3) and (4)]} \\ = & (KL)_a + (KM)_a + N_a & \\ = & (KL + KM + N)_a, & \end{array}$$

which concludes the proof that R is a bisimulation. Now (11) follows by coinduction. Similarly for (12). For (13), let K, L, and M be expressions with $L\!\uparrow$ and $K = LK + M$. Then $K = L^*M$ follows by coinduction from the fact that $\{\langle UK + V,\, UL^*M + V\rangle \mid U, V \in \mathcal{L}\}$ is a bisimulation on $\mathcal{L}$. Equations (14) and (15) follow from the fact that $\{\langle M(K + L)^*,\, MK^*(LK^*)^*\rangle \mid K, L, M \in \mathcal{L}\}$ and $\{\langle M(K + L)^*,\, M(K^*L)^*K^*\rangle \mid K, L, M \in \mathcal{L}\}$ are bisimulations.

Some regular languages

Below the language $\lambda(E)$ of a regular expression E will be simply denoted by E itself. Similarly, E_a denotes $\lambda(E)_a$. Let $A = \{a, b\}$. As an example, we want to show

$$[(b^*a)^*ab^*]^* = 1 + a(a + b)^* + (a + b)^*aa(a + b)^*. \tag{16}$$

Let $E_1 = [(b^*a)^*ab^*]^*$ and $F_1 = 1 + a(a + b)^* + (a + b)^*aa(a + b)^*$. Using the calculation rules for a-derivatives of Section 5, the following tables are easily computed:

	a	b	
E_1	E_2	E_4	$\downarrow$
E_2	E_2	E_3	$\downarrow$
E_3	E_2	E_3	$\downarrow$
E_4	E_5	E_4	$\uparrow$
E_5	E_2	E_4	$\uparrow$

	a	b	
F_1	F_2	F_4	$\downarrow$
F_2	F_2	F_3	$\downarrow$
F_3	F_2	F_3	$\downarrow$
F_4	F_5	F_4	$\uparrow$
F_5	F_2	F_4	$\uparrow$

where

$$\begin{aligned}
E_2 &= [(b^*a)^*ab^* + b^*]E_1,\\
E_3 &= [(b^*a)(b^*a)^*ab^* + b^*]E_1,\\
E_4 &= [(b^*a)(b^*a)^*ab^*]E_1,\\
E_5 &= [(b^*a)^*ab^*]E_1,\\
F_2 &= (a+b)^* + (a+b)^*aa(a+b)^* + a(a+b)^*,\\
F_3 &= (a+b)^* + (a+b)^*aa(a+b)^*,\\
F_4 &= (a+b)^*aa(a+b)^*,\\
F_5 &= (a+b)^*aa(a+b)^* + a(a+b)^*.
\end{aligned}$$

As a consequence, $T = \{\langle E_i, F_i\rangle \mid 1 \leq i \leq 5\}$ is a bisimulation. Hence $E_i = F_i$, by coinduction, for $1 \leq i \leq 5$. This proves (16).

It follows from the tables above that $\{\langle F_2, F_3\rangle, \langle F_2, F_2\rangle, \langle F_3, F_3\rangle\}$ is a bisimulation as well. Thus $E_2 = E_3$, by coinduction, and similarly $F_2 = F_3$. There is, therefore, some redundancy in the representation of the bisimulation T, which turns out to consist of only 4 different pairs. The interesting point of this observation is that this knowledge was not needed for the conclusion above that T is a bisimulation.

Because $((a+b)^*)_a = (a+b)^*$ and $((a+b)^*)_b = (a+b)^*$ imply that $\{\langle F_2, (a+b)^*\rangle, \langle F_3, (a+b)^*\rangle\}$ is a bisimulation, we also have, as another example, the following equalities:

$$E_2 = E_3 = F_2 = F_3 = (a+b)^*.$$

Inequalities

The coinduction proof method is clearly also of help in proving that two languages are different. In order to prove $E_1 \neq E_2$ in the example above, it is sufficient to show that there is no bisimulation relation containing $\langle E_1, E_2\rangle$. Now the assumption that $\langle E_1, E_2\rangle$ is in some bisimulation leads to a contradiction, since $(E_1)_b = E_4$ and $(E_2)_b = E_3$, but $(E_4)\uparrow$ and $(E_3)\downarrow$.

7 Finality and minimization

We can use coinduction to prove that the automaton $\mathcal{L}$ is *final* among all automata, i.e., for any automaton $S = \langle S, o, t\rangle$ there exists a unique homomorphism from S to $\mathcal{L}$: the *existence* follows from the observation that the function $l_S : S \to \mathcal{L}$ (which assigns to a state the language it accepts) is a homomorphism. For *uniqueness*, suppose f and g are homomorphisms from S to $\mathcal{L}$. The equality of f and g follows by coinduction from the fact that $R = \{\langle f(s), g(s)\rangle \mid s \in S\}$ is a bisimulation on $\mathcal{L}$, which is proved next. Because f and g are homomorphisms, we have, for any s in S, $f(s)\downarrow$ iff $s\downarrow$ iff $g(s)\downarrow$. For any a in A, $f(s) \xrightarrow{a} L$ iff $L = f(s')$, where $s' = t_S(s)(a)$, and similarly $g(s) \xrightarrow{a} g(s')$. Because $\langle f(s'), g(s')\rangle$ is in R, this shows that R is a bisimulation.

The unique homomorphism $l_S : S \to \mathcal{L}$ has the property that it identifies two states in S precisely when they are bisimilar: for all s and s' in S, $s \sim s'$ if and only if $l_S(s) = l_S(s')$. From left to right, this follows by coinduction from the general property of homomorphisms that for any bisimulation R on S the set $\{\langle l_S(s), l_S(s')\rangle \mid s\,R\,s'\}$ is a bisimulation on $\mathcal{L}$. For the converse, note that $\{\langle s, s'\rangle \mid l_S(s) = l_S(s')\}$ is a bisimulation on S.

By the finality of $\mathcal{L}$, the identity function is the only homomorphism from $\mathcal{L}$ to itself. It follows that the language accepted by a state L in $\mathcal{L}$ is L itself, as was announced in Section 3.

The subautomaton $\langle L\rangle \subseteq \mathcal{L}$ generated by L, which is given by

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

is moreover a *minimal* automaton for L in the following sense. Let S be any automaton and s a state in S such that the language accepted by s is L. That is, $l_S(s) = L$, where $l_S : S \to \mathcal{L}$ is the (unique) homomorphism from S to $\mathcal{L}$ that assigns to each state the language it accepts. Because l_S is a homomorphism, $l_S(\langle s\rangle) = \langle l_S(s)\rangle$, whence $l_S(\langle s\rangle) = \langle L\rangle$. Therefore the size of $\langle L\rangle$ is at most that of S. Since S and s were arbitrary, $\langle L\rangle$ is of minimal size.

It follows that for any automaton S and state s in S, the minimization of the automaton $\langle s\rangle$ is $\langle l_S(s)\rangle$. Another consequence is that

$$\begin{array}{l} L \text{ is accepted by a finite automaton iff} \\ \quad \langle L\rangle \text{ is a finite subautomaton of } \mathcal{L}. \end{array} \tag{17}$$

This is in fact equivalent to the following classical theorem by Nerode and Myhill. Let R_L be an equivalence relation on A^* defined, for v and w in A^*, by

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

The *index* of R_L is defined as the number of its equivalence classes. The theorem of Nerode and Myhill now says that

$$\begin{array}{l} L \text{ is accepted by a finite automaton iff} \\ \quad R_L \text{ is of finite index.} \end{array} \tag{18}$$

The equivalence of (17) and (18) follows from the observation that the correspondence between equivalence classes of R_L and elements of $\langle L\rangle$, given for w in A^* by $[w]_{R_L} \mapsto L_w$, is bijective: for v and w in A^*,

$$\begin{array}{ll} & [v]_{R_L} = [w]_{R_L} \\ \text{iff} & v \mathrel{R_L} w \\ \text{iff} & \forall u \in A^*,\ vu \in L \iff wu \in L \\ \text{iff} & \forall u \in A^*,\ u \in L_v \iff u \in L_w \\ \text{iff} & L_v = L_w. \end{array}$$

8 Kleene's theorem

Kleene's celebrated theorem states that a language is regular if and only if it is accepted by a finite automaton. In view of (17), Kleene's theorem can be expressed in terms of subautomata of the automaton $\mathcal{L}$ of languages, as follows. Let A be finite. For any language $L \subseteq A^*$,

$$L \text{ is regular iff } \langle L\rangle \text{ is a finite subautomaton of } \mathcal{L}. \tag{19}$$

As a corollary of (19), it will be shown below that the coinduction proof principle is effective for regular languages (as was announced in Section 6).

In order to prove (19) from left to right, consider $\lambda(E)$, for some regular expression E. One can show by induction on the syntactic structure of E that $\langle\lambda(E)\rangle$ is finite. Consider, for instance, EF and assume that $\langle\lambda(E)\rangle$ and $\langle\lambda(F)\rangle$ are finite. It follows from the rules for a-derivatives that the general format of a state reachable from $\lambda(EF)$ is $K'M + M' + \cdots + M''$, for K' in $\langle\lambda(E)\rangle$ and $M', \ldots, M''$ in $\langle\lambda(F)\rangle$. Using (some of) the laws (1)–(8), it follows from the inductive hypothesis that $\langle\lambda(EF)\rangle \subseteq \{K'M + M' + \cdots + M'' \mid K' \in \langle\lambda(E)\rangle\}$ is finite. The other cases are dealt with similarly.

Conversely, we have to show that for a language L for which $\langle L\rangle$ is finite, there exists a regular expression E with $\lambda(E) = L$. Rather than proving this part of the theorem for arbitrary languages, we consider an example that can be easily generalized to the general case. The following law, which can be readily proved by coinduction, will be helpful: If $A = \{a, \ldots, b\}$ then for all languages L,

$$L = \begin{cases} aL_a + \cdots + bL_b + 1 & \text{if } L{\downarrow} \\ aL_a + \cdots + bL_b & \text{if } L{\uparrow}. \end{cases} \tag{20}$$

For an example, let $A = \{a, b\}$ and K in $\mathcal{L}$ with $\langle K\rangle = \{K, L, M, N\}$, for languages L, M, and N, with transitions and termination as specified by the following table:

	a	b	
K	L	M	$\uparrow$
L	L	M	$\downarrow$
M	M	N	$\downarrow$
N	N	N	$\uparrow$

By (20), there are the following equations:

$$\begin{aligned} K &= aL + bM \\ L &= aL + bM + 1 \\ M &= aM + bN + 1 \\ N &= aN + bN \end{aligned}$$

Because $N = aN + bN = (a + b)N + 0$, law (13) implies $N = (a + b)^*0 = 0$. Thus $M = aM + 1$ which, again by (13) gives $M = a^*$. Similarly it follows that $L = a^*(ba^* + 1)$ and $K = aa^*(ba^* + 1) + ba^*$, which proves that K is regular, indeed. This completes the proof of (19).

A consequence of (19) is that the coinduction proof principle is effective for regular languages $\lambda(E)$ and $\lambda(F)$: In order to construct a bisimulation relation that includes the pair $\langle\lambda(E), \lambda(F)\rangle$, one has to add all pairs of states that are (pair-wise) reachable from $\lambda(E)$ and $\lambda(F)$. Since both $\langle\lambda(E)\rangle$ and $\langle\lambda(F)\rangle$ are finite, by (19), it follows that in finitely many steps, either such a bisimulation is constructed (whence $\lambda(E) = \lambda(F)$) or the conclusion is reached that no bisimulation for $\lambda(E)$ and $\lambda(F)$ exists (whence $\lambda(E) \neq \lambda(F)$).

Note that the use of the simplification laws (1)–(8) is crucial for termination; for instance, they are needed to conclude that all languages occurring in the sequence

$$\lambda(a^*) \xrightarrow{a} 1\lambda(a^*) \xrightarrow{a} 0\lambda(a^*) + 1\lambda(a^*) \xrightarrow{a} 0\lambda(a^*) + 0\lambda(a^*) + 1\lambda(a^*) \xrightarrow{a} \cdots$$

are equal, and hence that $\langle\lambda(a^*)\rangle$ consists of only one state.

9 Nonregular languages

An immediate consequence of Kleene's theorem in the formulation of (19) above is that in order to show that a language L is nonregular, it is sufficient to prove that $\langle L\rangle$ is *not* finite. This method is equivalent, by the equivalence of (17) and (18), to the traditional approach of showing that R_L is of infinite index. Here are three classical examples, in which the following shorthand will be used. For a language K and $k \geq 0$, let the language K_k be the resulting state after k a-steps: $K_k = K_{a^k}$.

Let $L = \{a^n b^n \mid n \geq 0\}$, where as usual $a^0 = 1$ and $a^{n+1} = aa^n$. Clearly, $L_k = \{a^{n-k}b^n \mid n \geq k\}$ and thus L_k and $L_{k'}$ are different whenever k and k' are. This shows that $\langle L\rangle$ is infinite, hence L is nonregular.

For a second example, consider $M = \{w \in A^* \mid \sharp_a(w) = \sharp_b(w)\}$ consisting of all words with an equal number of a's and b's. All languages M_k are different because for any n and k, the word b^n is in M_k iff $k = n$. Thus $\langle M\rangle$ is infinite and M is nonregular.

Finally, let $N = \{a^{n^2} \mid n \geq 0\}$. Note that for any n the length of the shortest word in N_{n^2+1} is $|a^{(n+1)^2-n^2-1}| = |a^{2n}| = 2n$. Therefore N_{n^2} and N_{m^2} are different whenever n and m are. Thus $\langle N\rangle$ is infinite and N is nonregular.

10 Definitions by coinduction

The fact that $\mathcal{L}$ is final gives rise to the following *coinductive definition principle*: in order to define a function from a given *set* S to $\mathcal{L}$, we can turn S into an *automaton* by defining an output function o and a transition function t on S. A function $l_S : S \to \mathcal{L}$ is then obtained by the finality of $\mathcal{L}$ as the unique homomorphism between the automata S and $\mathcal{L}$, which assigns to each element, that is, *state* s in S the language it accepts.

As an example, we shall apply the above principle to obtain a coinductive definition of the *shuffle* of two languages. To this end, let the set $\mathcal{E}$ of expressions be given by the following syntax:

$$E ::= \underline{L} \ (\text{for } \ L \in \mathcal{L}) \mid E + F \mid E \,||\, F$$

Note that $\mathcal{E}$ contains a *symbol* $\underline{L}$ for any language L in $\mathcal{L}$. The set $\mathcal{E}$ can be turned into an automaton $\langle \mathcal{E}, o_\mathcal{E}, t_\mathcal{E} \rangle$, defined by the following axioms and rules (using the arrow notation introduced in Section 2):

$$\underline{L}\!\downarrow \Leftrightarrow \varepsilon \in L, \quad (E+F)\!\downarrow \Leftrightarrow E\!\downarrow \text{ or } F\!\downarrow, \quad (E \,||\, F)\!\downarrow \Leftrightarrow E\!\downarrow \text{ and } F\!\downarrow$$

$$\underline{L} \xrightarrow{a} \underline{L_a} \qquad \frac{E \xrightarrow{a} E' \quad F \xrightarrow{a} F'}{E + F \xrightarrow{a} E' + F'} \qquad \frac{E \xrightarrow{a} E' \quad F \xrightarrow{a} F'}{E \,||\, F \xrightarrow{a} E' \,||\, F + E \,||\, F'}$$

Note that the above axioms and rules uniquely determine two functions $o_\mathcal{E} : \mathcal{E} \to 2$ and $t_\mathcal{E} : \mathcal{E} \to \mathcal{E}^A$. By the coinduction definition principle, there exists a unique homomorphism $l : \mathcal{E} \to \mathcal{L}$, giving for each expression E, that is, state of the automaton $\mathcal{E}$, the language $l(E)$ it accepts. One readily proves (by coinduction) that $l(\underline{L}) = L$ and $l(E + F) = l(E) + l(F)$.

The shuffle of two languages K and L can now be defined as $K||L = l(\underline{K} \,||\, \underline{L})$. Its a-derivative, for a in A, can be computed as follows:

$$\begin{aligned}
&(K||L)_a \\
&\quad = \ (l(\underline{K} \,||\, \underline{L}))_a \\
&\quad = \ t_\mathcal{L}(l(\underline{K} \,||\, \underline{L}))(a) \\
&\quad = \ l(t_\mathcal{E}(\underline{K} \,||\, \underline{L})(a)) \quad [l \text{ is a homomorphism}] \\
&\quad = \ l(\underline{K_a} \,||\, \underline{L} + \underline{K} \,||\, \underline{L_a}) \quad [\text{definition } t_\varepsilon] \\
&\quad = \ l(\underline{K_a} \,||\, \underline{L}) + l(\underline{K} \,||\, \underline{L_a}) \\
&\quad = \ K_a||L + K||L_a. \qquad (21)
\end{aligned}$$

This characterization is useful for proving properties by coinduction, such as $K||L = L||K$, $K||(L + M) = K||L + K||M$, and $(K||L)||M = K||(L||M)$. For instance, the latter equality follows by coinduction from the fact that

$$\{\langle (K||L)||M + \cdots + (K'||L')||M', \ K||(L||M) + \cdots + K'||(L'||M') \rangle \mid$$

$$K, L, M, K', L', M' \in \mathcal{L}\}$$

is readily shown to be a bisimulation.

Let us, once more, make a case for the importance of coinduction by inviting the reader to prove the associativity of the shuffle operator by induction, using the following inductive definition:

$$K||L = \bigcup\{v||w \mid v \in K, w \in L\}, \quad \text{with}$$

$$v||w = v \perp\!\!\!\perp w + w \perp\!\!\!\perp v, \quad \varepsilon \perp\!\!\!\perp v = \{v\}, \quad (av) \perp\!\!\!\perp w = a(v||w),$$

and to compare the inductive proof to the coinductive one above.

11 Simulation

The notion of bisimulation is a special case of the more general notion of *simulation*, which will be introduced below. Simulation is used in the formulation of yet another coinduction principle on $\mathcal{L}$ which generalizes that of Section 4.

A *simulation* between two automata $S = \langle S, o, t\rangle$ and $S' = \langle S', o', t'\rangle$ is any relation $R \subseteq S \times S'$ with, for all s in S, s' in S', and a in A:

$$\text{if } \; s\,R\,s' \; \text{ then } \; \begin{cases} o(s) \leq o'(s') & \text{and} \\ t(s)(a)\; R\; t'(s')(a). \end{cases}$$

Thus if $s\,R\,s'$ then $s{\downarrow}$ implies $s'{\downarrow}$. A simulation between S and itself is called a simulation *on* S. Unions and (relational) compositions of simulations are simulations again. We write $s \leq s'$ whenever there exists a simulation R with $s\,R\,s'$. This relation $\leq$ is the union of all simulations and, therewith, the greatest simulation. The greatest simulation on one and the same automaton S, denoted by $\leq$ (or $\leq_S$, if the name of the automaton is relevant), is called the *similarity* relation. It is a preorder: $s \leq s$ and if $s \leq t$ and $t \leq u$ then $s \leq u$.

Clearly every bisimulation is a simulation. The converse does not hold but $s \leq t$ and $t \leq s$ imply $s \sim t$: if $s\,R\,t$ and $t\,T\,s$ for two simulations R and T then $R \cap T^{-1}$ is a bisimulation with $s(R \cap T^{-1})t$. It follows that $\sim = \leq \cap \leq^{-1}$. (The fact that we are dealing with *deterministic* automata is crucial here.)

The automaton $\mathcal{L} = \langle \mathcal{L}, o_{\mathcal{L}}, t_{\mathcal{L}}\rangle$ satisfies the following proof principle, which is again called coinduction: for all languages K and L,

$$\text{if } \; K \leq L \; \text{ then } \; K \subseteq L.$$

(The converse trivially holds.) The proof principle says that in order to prove the inclusion of a language K in a language L, it is sufficient to establish the existence of a simulation relation R on $\mathcal{L}$ with $K\,R\,L$. Inspecting the proof of the previous coinduction principle in Section 4, we see that it contains a proof of the statement above.

The regular operations on languages can be easily shown to be monotonic with respect to $\subseteq$. For instance, if $K \subseteq K'$ and $L \subseteq L'$ then $KL \subseteq K'L'$. Also $K \subseteq L$ implies $K_a \subseteq L_a$.

The above coinduction principle is often best applied in combination with the following weakening of the notion of simulation. A simulation *up-to-similarity* on automata $S = \langle S, o, t\rangle$ and $S' = \langle S', o', t'\rangle$ is any relation $R \subseteq S \times S'$ with, for all s in S, s' in S', and a in A:

$$\text{if } \; sRs' \; \text{ then } \; \begin{cases} o(s) \leq o'(s') & \text{and} \\ t(s)(a)\; R_{\leq}\; t'(s')(a), \end{cases}$$

where $R_{\leq} = {\leq_S} \circ R \circ {\leq_{S'}}$ ($\circ$ denotes composition of relations). Interestingly, if sRt for a simulation up-to-similarity R then $s \leq t$, since in that case $R_{\leq}$ is a simulation and $R \subseteq R_{\leq}$. Thus in order to prove $K \subseteq L$ it suffices to point to a simulation up-to-similarity R with $K\,R\,L$.

We treat a few examples. The following inclusions and equational implications can all be proved by coinduction:

$$KL \;\subseteq\; K||L \tag{22}$$

$$KL \subseteq L \;\Rightarrow\; K^*L \subseteq L \tag{23}$$

$$LK + M \subseteq K \;\Rightarrow\; L^*M \subseteq K \tag{24}$$

$$KL \subseteq LM \;\Rightarrow\; K^*L \subseteq LM^* \tag{25}$$

For (22), we show that

$$R = \{\langle\, KL + \cdots + K'L',\; K||L + \cdots + K'||L' \,\rangle \mid K, L, K', L' \in \mathcal{L}\}$$

is a simulation up-to-similarity. Consider $\langle KL,\, K||L\rangle$ in R (the other cases of pairs of longer sums are similar). Suppose $K{\downarrow}$ (the case of $K{\uparrow}$ being simpler). If $(KL){\downarrow}$ then $(K||L){\downarrow}$. And for a in A,

$$(KL)_a$$

$$
\begin{array}{ll}
= & K_aL + L_a \\
= & K_aL + 1L_a \\
\subseteq & K_aL + KL_a \quad [1 \subseteq K \text{ since } K{\downarrow}] \\
R & K_a||L + K||L_a \\
= & (K||L)_a \quad [\text{by (21)}],
\end{array}
$$

which shows that R is a simulation up-to-similarity. Now (22) follows by coinduction. For (23) consider K and L with $KL \subseteq L$. Then

$$S = \{\langle\, MK^*L + N,\ ML + N \,\rangle \mid M, N \in \mathcal{L}\}$$

is a simulation up-to-similarity: if $(MK^*L + N){\downarrow}$ then $(ML + N){\downarrow}$. And for a in A,

$$
\begin{array}{ll}
& (MK^*L + N)_a \\
= & M_aK^*L + K_aK^*L + L_a + N_a \quad [\text{supposing that } M{\downarrow}] \\
= & (M_a + K_a)K^*L + L_a + N_a \\
S & (M_a + K_a)L + L_a + N_a \\
= & M_aL + K_aL + L_a + N_a \\
\subseteq & M_aL + L_a + N_a \quad [KL \subseteq L \text{ implies } (KL)_a \subseteq L_a \text{ whence } K_aL + L_a \subseteq L_a] \\
= & (ML + N)_a.
\end{array}
$$

Thus (23) follows by coinduction. Law (24), which refines equation (13) in Section 6, and law (25) are proved similarly.

As another example, we prove the inclusion of the following regular languages:

$$[(b^*a)^*ab^*]^* \subseteq [(b^*a)^*ab^* + b^*][(b^*a)^*ab^*]^*,$$

which we recognize as E_1 and E_2 from Section 6. The inclusion follows by coinduction from the fact that we have a simulation

$$\{\langle E_1, E_2\rangle, \langle E_2, E_2\rangle, \langle E_3, E_3\rangle, \langle E_4, E_3\rangle, \langle E_5, E_2\rangle\}.$$

12 Automata are coalgebras

Classically, an automaton over a (finite) fixed input alphabet A is defined as a 4-tuple

$$\langle S,\ s_0 \in S,\ F \subseteq S,\ \delta : S \times A \to S \,\rangle,$$

consisting of a finite set S of states, an initial state s_0, a set F of terminating (or accepting) states, and a transition function δ. Below our definition of automaton, as given in Section 2, is compared to the one above. It is explained that our definition in essence is *coalgebraic*, and that the notions of homomorphism, bisimulation, and coinduction as introduced in the preceding sections, are special instances of general coalgebraic definitions.

First of all, there is no reason to restrict oneself to *finite* sets A and S. On the contrary, allowing an infinite set of states makes it possible to consider, for instance, the set $\mathcal{L}$ of languages as an automaton. Secondly, we have not included an initial state in our definition, simply because there is no reason to focus attention to one particular state. In the classical theory of automata, initial states play a role, for instance, in the definition of the sequential composition of two automata, where all the terminating states of the first automaton are connected to the initial state of the second automaton (usually by an ϵ-transition). As we have seen, there is no need for such a construction in the present theory.

Allowing infinite sets and forgetting about the initial state, the classical definition of course becomes equivalent to the definition of Section 2, because of the existence of bijections

$$\mathcal{P}(S) \cong (S \to 2) \quad \text{and} \quad (S \times A \to S) \cong (S \to S^A).$$

Thus there is a one-to-one correspondence between triples $\langle S, F, \delta\rangle$ and triples $\langle S, o, t\rangle$. The choice of working with the latter representation is motivated by the observation that in this way, automata can be viewed as coalgebras: Let $F : \mathit{Set} \to \mathit{Set}$ be a functor on the category of sets and functions. An *F-coalgebra* is a pair (S, α_S) consisting of a set S and a function $\alpha_S : S \to F(S)$. Automata are coalgebras of the following functor $D : \mathit{Set} \to \mathit{Set}$, which is defined on sets S by $D(S) = 2 \times S^A$ (below we shall define how D acts on functions). Now for an automaton $\langle S, o, t\rangle$, the functions $o : S \to 2$ and $t : S \to S^A$ can be combined into one function $\langle o, t\rangle : S \to 2 \times S^A$, which sends s in S to the pair $\langle o(s), t(s)\rangle$. In this way, the automaton $\langle S, o, t\rangle$ has been represented as a D-coalgebra

$$\langle o, t\rangle : S \to D(S).$$

The reason to be interested in this coalgebraic representation of automata is that there exist a number of notions and results on coalgebras in general, which can now be applied to automata.

Notably there is the following definition. Consider again an arbitrary functor $F : \mathit{Set} \to \mathit{Set}$ and let (S, α) and (S', α') be two F-coalgebras. A function $f : S \to S'$ is a *homomorphism of F-coalgebras*, or F-homomorphism, if $F(f) \circ \alpha = \alpha' \circ f$. In order to apply this definition to the case of automata, we still have to give the definition of the functor D on functions, which is as follows. For a function $f : S \to S'$, the function $D(f) : (2 \times S^A) \to (2 \times S'^A)$ is defined, for any x in 2 and h in S^A by $D(f)(\langle x, h\rangle) = \langle x,\, f \circ h\rangle$. Now consider two automata, i.e., D-coalgebras, $(S, \langle o, t\rangle)$ and $(S', \langle o', t'\rangle)$, where $\langle o, t\rangle : S \to D(S)$ and $\langle o', t'\rangle : S' \to D(S')$. According to the definition, a function $f : S \to S'$ is a homomorphism of D-coalgebras if $D(f) \circ \langle o, t\rangle = \langle o', t'\rangle \circ f$, which is equivalent to $o(s) = o'(f(s))$ and $f(t(s)(a)) = t'(f(s))(a)$, for all s and a. Note that this is precisely the definition of homomorphism given in Section 2. Indeed, even if we did not mention this before, the general coalgebraic definition of homomorphism has been our starting point.

Also the notion of bisimulation introduced in Section 2 is an instance of a general coalgebraic definition: A relation $R \subseteq S \times S'$ is called an *F-bisimulation* between F-coalgebras (S, α) and (S', α') if there exists an F-coalgebra structure $\alpha_R : R \to F(R)$ on R such that the projections $\pi_1 : R \to S$ and $\pi_2 : R \to S'$ are F-homomorphisms. It is left to the reader to verify that applying this definition to the functor D yields our original definition of bisimulation of automata.

For a functor $F : \mathit{Set} \to \mathit{Set}$, the family of F-coalgebras together with the F-homomorphisms between them, forms a category (indentity functions are homomorphisms, and the composition of homomorphisms is again a homomorphism). In this category, *final* coalgebras are of special interest (if they exist at all): a coalgebra (P, π) is final if there exists from any coalgebra precisely one homomorphism into (P, π). The interest of final coalgebras lies in the fact that they satisfy the following coinduction proof principle: if there exists an F-bisimulation between p and p' in P then p and p' are equal. This is immediate by the finality of (P, π).

Many functors have a final coalgebra (final coalgebras are unique up to isomorphism), and for many functors it can be constructed in a canonical way. For our functor D, this construction yields the set $A^* \to 2$, which is isomorphic to the set $\mathcal{L}$ of all languages. Indeed, we have seen in Sections 7 and 4 that $\mathcal{L}$ is a final automaton and satisfies the coinduction proof principle[2].

Summarizing the above, we hope to have explained the subtitle of the present paper. The treatment of automata in the preceding sections has been coalgebraic: the definitions of automaton, homomorphism, and bisimulation, as well as the focus on finality and coinduction, all have been derived from or motivated by very general definitions and observations from coalgebra.

As such, this coalgebraic story of automata is just one out of many, in principle as many as there are functors (on *Set* but also on other categories). Many other examples have been studied in considerable detail already, including transition systems, data types (such as streams and trees), dynamical systems, probabilistic systems, object-based systems, and many more. And many more are still to follow. It is to be expected that the theory of several other kinds of automata may benefit from a coalgebraic treatment.

In the remaining sections of the present paper, the coalgebraic approach is further illustrated by the treatment of automata with partial transition functions. These *partial* automata are coalgebras

[2]We have proved that $\mathcal{L}$ satisfies the coinduction proof principle *before* proving its finality for didactical reasons.

of a functor $D' : \mathit{Set} \to \mathit{Set}$, which is defined as a minor variation of the functor D: for a set S, $D'(S) = 2 \times (1 + S)^A$. As before, our presentation will make no explicit reference to coalgebra.

13 Partial automata

A *partial automaton* with input alphabet A is a triple $S = \langle S, o, t\rangle$ consisting, as before, of a set S of states and an output function $o : S \to 2$, but now with a transition function t that assigns to each state a *partial* function. That is, $t : S \to (1 + S)^A$, where $1 = \{\Uparrow\}$, and where for a function f in $(1 + S)^A$ and input symbol a in A, $f(a) = \Uparrow$ means that f is undefined in a, sometimes simply denoted by $f(a)\Uparrow$. Dually, $f(a)\Downarrow$ denotes that $f(a)$ *is* defined. (These conventions will more generally be used for functions from X to $1 + Y$, for arbitrary sets X and Y.)

As before, we shall sometimes write $s\downarrow$ for $o(s) = 1$, $s\uparrow$ for $o(s) = 0$, and $s \xrightarrow{a} s'$ for $t(s)(a) = s'$. In addition, $s \xrightarrow{a}\!\!\!\!\!/\;\;$ denotes $t(s)(a)\Uparrow$.

A bisimulation between partial automata $S = \langle S, o, t\rangle$ and $S' = \langle S', o', t'\rangle$ is now a relation $R \subseteq S \times S'$ with, for all s in S, s' in S', and a in A:

$$\text{if } \; s \,R\, s' \;\text{ then }\; \begin{cases} o(s) = o'(s') & \text{and} \\ t(s)(a)\ (1 + R)\ t'(s')(a), \end{cases}$$

where $t(s)(a)\ (1 + R)\ t'(s')(a)$ holds iff either both sides are undefined or both sides are defined and related by R. Note that as a consequence, $s \,R\, s'$ implies $s \xrightarrow{a}\!\!\!\!\!/\;\;$ iff $s' \xrightarrow{a}\!\!\!\!\!/\;\;$.

The notions of bisimilarity, homomorphism and subautomaton are defined as before, and the various properties given in Section 2 again apply.

Due to the possibility of refusing certain input symbols, the language $l_S(s)$ accepted by a state s of a partial automaton $S = \langle S, o, t\rangle$ may now consist of three different kinds of words:

1. If $s \xrightarrow{a_1} s_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} s_n\downarrow$ then $a_1 \cdots a_n \in l_S(s)$, as before.
2. If $s \xrightarrow{a_1} s_1 \xrightarrow{a_2} \cdots \xrightarrow{a_n} s_n\uparrow$ and for all a in A, $s_n \xrightarrow{a}\!\!\!\!\!/\;\;$, then $a_1 \cdots a_n \cdot \delta \in l_S(s)$. Here the postfix δ (which is supposed not to be an element of A) is used to register the fact that after the last input symbol (a_n), a so-called *deadlock* occurs: the automaton has reached a state (s_n) which is not terminating, and from which no further steps are possible.
3. If $s \xrightarrow{a_1} s_1 \xrightarrow{a_2} s_2 \xrightarrow{a_3} \cdots$ then the infinite word $a_1 a_2 a_3 \cdots \in l_S(s)$.

In order to define the collection of all acceptable languages, let

$$A^\infty_\delta = A^* \cup\ A^\omega \cup\ A^* \cdot \delta,$$

where A^* is as before, A^ω is the set of all infinite words over A, and $A^* \cdot \delta = \{w \cdot \delta \mid w \in A^*\}$. Sometimes A^∞ is used as a shorthand for $A^* \cup\ A^\omega$. For an infinite word $w = a_1 a_2 a_3 \cdots$ in A^ω and natural number $n \geq 1$, the n-th truncation of w is given by $w[n] = a_1 \cdots a_n$.

We shall again need the notion of derivative. For a word w in A^* and a subset $L \subseteq A^\infty_\delta$, let the w-derivative of L be defined by

$$L_w = \{v \in A^\infty_\delta \mid wv \in L\},$$

where concatenation of words is extended to A^∞_δ in the obvious way.

A set $L \subseteq A^\infty_\delta$ is *closed*[3] if for all infinite words w in A^ω,

$$\forall n \geq 1,\ L_{w[n]} \neq \emptyset \;\Rightarrow\; w \in L.$$

Typically, a^∞ is closed, whereas a^* is not. A set $L \subseteq A^\infty_\delta$ is *consistent* if for all words w in A^∞_δ,

$$\delta \in L_w \iff L_w = \{\delta\}.$$

[3]The terminology is explained by the fact that this definition is equivalent to being closed with respect to the metric topology on A^∞_δ induced by the Baire metric.

For instance, $\{ab, ac, b\delta\}$ is consistent whereas $\{ab, a\delta\}$ is not.

A *language* (of partial automata) is next defined as a non-empty, closed, and consistent subset of A_δ^∞. Let $\mathcal{L}_p$ denote the set of all languages (of partial automata):

$$\mathcal{L}_p = \{L \mid L \subseteq A_\delta^\infty,\ L \text{ is non-empty, closed, and consistent}\}.$$

It is not difficult to verify that the set $l_S(s)$ above indeed belongs to $\mathcal{L}_p$. We shall see that, conversely, any language in $\mathcal{L}_p$ is accepted by some partial automaton.

The set $\mathcal{L}_p$ can be turned into a partial automaton $\mathcal{L}_p = \langle \mathcal{L}_p, o_{\mathcal{L}_p}, t_{\mathcal{L}_p} \rangle$ by defining, for L in $\mathcal{L}_p$ and a in A,

$$o_{\mathcal{L}_p}(L) = \begin{cases} 1 & \text{if } \varepsilon \in L \\ 0 & \text{if } \varepsilon \notin L \end{cases} \quad \text{and:} \quad t_{\mathcal{L}_p}(L)(a) = \begin{cases} L_a & \text{if } L_a \neq \emptyset \\ \Uparrow & \text{if } L_a = \emptyset. \end{cases}$$

That is,

$$L\!\downarrow \text{ iff } \varepsilon \in L, \quad L \xrightarrow{a} L_a \text{ iff } L_a \neq \emptyset, \quad L \not\xrightarrow{a} \text{ iff } L_a = \emptyset.$$

Again the coinduction principle holds: for all languages K and L in $\mathcal{L}_p$,

$$\text{if } K \sim L \text{ then } K = L.$$

It is identical in shape to the principle of Section 4, but note that the languages under consideration are now living in $\mathcal{L}_p$ instead of $\mathcal{L}$, and that a different notion of bisimilarity is involved. A new proof of the principle is therefore required but nevertheless omitted. It is not very difficult, and one needs to use the fact that the languages in $\mathcal{L}_p$ are both closed and consistent.

As before, it follows by coinduction that the automaton $\mathcal{L}_p$ is final among the collection of all partial automata: the unique homomorphism from a partial automaton S to the automaton $\mathcal{L}_p$ is given by the function $l_S : S \to \mathcal{L}_p$ described above. Because $\mathcal{L}_p$ is final, the coinduction definition principle (Section 10) holds again. It will be used in the next section.

14 Regular expressions for partial automata

In order to formulate a Kleene theorem for partial automata, which will be proved in the next section, a notion of regular expression for partial automata is introduced, as a minor variation on the classical definition (given in Section 5). Next regular languages and regular operators are defined by coinduction, in the same style as the definitions given in Section 10.

The set $\mathcal{R}_p$ of regular expressions (for partial automata) is defined by the following syntax:

$$E ::= 0 \mid 1 \mid a \in A \mid E + F \mid EF \mid E^\infty$$

The only difference with the previous definition is the absence of E^*, which has been replaced by E^∞.

Both the language $l(E)$ of a regular expression E in $\mathcal{R}_p$ and the regular operators will be defined by coinduction. To this end, a class $\mathcal{E}_p$ of expressions (for partial automata) is introduced, given by the following syntax:

$$E ::= \underline{L} \text{ (for } L \in \mathcal{L}_p) \mid E + F \mid EF \mid E^\infty$$

where an underscore is used to distinguish between the syntactic symbol $\underline{L}$ and the language L. However, the underscore will be omitted whenever possible without creating confusion.

The set $\mathcal{R}_p$ of regular expressions can be viewed as a subset of $\mathcal{E}_p$ by making the following identifications: $0 = \underline{\{\delta\}}$, $1 = \underline{\{\epsilon\}}$, and $a = \underline{\{a\}}$.

In order to apply the coinduction definition principle, the set $\mathcal{E}_p$ is turned into a partial automaton $\mathcal{E}_p = \langle \mathcal{E}_p, o_{\mathcal{E}_p}, t_{\mathcal{E}_p} \rangle$, where the functions $o_{\mathcal{E}_p}$ and $t_{\mathcal{E}_p}$ are defined by the following axioms and rules:

$$\underline{L}\!\downarrow \text{ iff } \varepsilon \in L, \quad (E^\infty)\!\downarrow, \quad (E+F)\!\downarrow \Leftrightarrow E\!\downarrow \text{ or } F\!\downarrow, \quad (EF)\!\downarrow \Leftrightarrow E\!\downarrow \text{ and } F\!\downarrow$$

$\underline{L} \xrightarrow{a} \underline{L_a}$ iff $L_a \neq \emptyset$

$$\frac{E \xrightarrow{a} E' \quad F \xrightarrow{a} F'}{E+F \xrightarrow{a} E'+F'} \qquad \frac{E \xrightarrow{a} E' \quad F \not\xrightarrow{a}}{E+F \xrightarrow{a} E'} \qquad \frac{E \not\xrightarrow{a} \quad F \xrightarrow{a} F'}{E+F \xrightarrow{a} F'}$$

$$\frac{E \xrightarrow{a} E' \quad E\!\uparrow}{EF \xrightarrow{a} E'F} \qquad \frac{E \xrightarrow{a} E' \quad F \not\xrightarrow{a} \quad E\!\downarrow}{EF \xrightarrow{a} E'F}$$

$$\frac{E \xrightarrow{a} E' \quad F \xrightarrow{a} F' \quad E\!\downarrow}{EF \xrightarrow{a} E'F+F'} \qquad \frac{E \not\xrightarrow{a} \quad F \xrightarrow{a} F' \quad E\!\downarrow}{EF \xrightarrow{a} F'} \qquad \frac{E \xrightarrow{a} E'}{E^\infty \xrightarrow{a} E'E^\infty}$$

These axioms and rules uniquely define two functions $o_{\mathcal{E}_p}$ and $t_{\mathcal{E}_p}$, essentially by induction on the syntactic structure of expressions. For instance, $o_{\mathcal{E}_p}(E^\infty) = 1$, and $t_{\mathcal{E}_p}(E^\infty)(a) = (t_{\mathcal{E}_p}(E)(a))E^\infty$.

By the finality of the partial automaton of languages $\mathcal{L}_p$, there exists a unique homomorphism $l : \mathcal{E}_p \to \mathcal{L}_p$, which gives for any expression in $\mathcal{E}_p$, notably for each regular expression E in $\mathcal{R}_p$, the language $l(E)$ it represents. As before, a language L is called regular if it equals $l(E)$, for some E in $\mathcal{R}_p$.

The homomorphism $l : \mathcal{E}_p \to \mathcal{L}_p$ can also be used to define the regular operators: for languages K and L in $\mathcal{L}_p$, let

$$\begin{aligned} K+L &= l(\underline{K}+\underline{L}) \\ KL &= l(\underline{KL}) \\ K^\infty &= l((\underline{K})^\infty). \end{aligned}$$

The bisimilarity relation $\sim$ on $\mathcal{E}_p$ can, with a little bit of patience, be shown to be a congruence with respect to the regular operators: if $E \sim G$ and $F \sim H$ then $E+F \sim G+H$, $EF \sim GH$, and $E^\infty \sim G^\infty$. Combining this with the observations that $E \sim \underline{l(E)}$, and that $l(E) = l(F)$ iff $E \sim F$, the following equalities can be readily proved:

$$\begin{aligned} l(0) &= \{\delta\} \\ l(1) &= \{\varepsilon\} \\ l(a) &= \{a\} \\ l(E+F) &= l(E)+l(F) \\ l(EF) &= l(E)l(F) \\ l(E^\infty) &= l(E)^\infty. \end{aligned}$$

For instance, $l(E+F) = l(\underline{l(E)}+\underline{l(F)}) = l(E)+l(F)$. Whenever convenient and harmless, we shall simply write E for $l(E)$. Notably, $\underline{0}$, $\underline{1}$, and $\underline{a}$ will then denote the three singleton sets mentioned above. Note that the language represented by 0 is no longer the empty set, as it is in Section 5, but the singleton set $\{\delta\}$, representing deadlock.

The regular operators could again have been defined 'elementwise', but things would have been slightly more complicated than before. The sum of two languages can no longer be defined as their union, nor does their concatenation consist of the pairwise concatenation of their respective elements. This is illustrated by the following equalities, which are an immediate consequence of the coinductive definitions above:

$$\begin{aligned} \{\delta\}+\{a\} &= \{a\} \\ \{a\delta\}+\{aa\} &= \{aa\} \\ \{a\delta,\epsilon\}\{ab\} &= \{ab\}. \end{aligned}$$

The intuition here is that (a possibly nested occurrence of) the deadlock symbol δ should disappear in the presence of an alternative transition step. Also the definition of K^∞ is essentially more complicated than that of K^*, since the latter could be defined as the union of an inductively defined sequence $(K^n)_n$ of finite powers of K. This is not possible for K^∞, which should include

also infinite words composed of infinitely many finite words from K. Although K^∞ can be defined using, for instance, least upperbounds of chains in K^* with the familiar prefix ordering, the above coinductive definition of K^∞ is simpler in the sense that it is purely set-theoretic.

Equalities of expressions can again be proved by coinduction, by establishing the existence of bisimulation relations. Note that a bisimulation on $\mathcal{L}_p$ is any relation R such that for K and L with $K\,R\,L$, $K{\downarrow}$ iff $L{\downarrow}$, and for any a in A, $t_{\mathcal{L}_p}(K)(a)\,(1+R)\,t_{\mathcal{L}_p}(L)(a)$. It follows from the definitions that the latter formula means that either both K_a and L_a are empty, or both are non-empty and related by R.

The following calculation rules for a-derivatives will again be helpful when proving the existence of bisimulation relations. They follow from the coinductive definition above by exploiting the fact that $l : \mathcal{E}_p \to \mathcal{L}_p$ is a homomorphism:

$$0_a{\Uparrow}, \quad 1_a{\Uparrow}, \quad b_a = \begin{cases} \{\epsilon\} & \text{if } b = a \\ \Uparrow & \text{otherwise} \end{cases}$$

$$\begin{aligned} (K+L)_a &= K_a + L_a \\ (KL)_a &= \begin{cases} K_a\,L & \text{if } K{\uparrow} \\ K_a\,L + L_a & \text{if } K{\downarrow} \end{cases} \\ (K^\infty)_a &= K_a\,K^\infty, \end{aligned}$$

where the latter three equalities are as before (Section 5) but now have to be read with the following conventions in mind: for all languages K and input symbols a,

$$K_a + \emptyset = \emptyset + K_a = K_a, \quad \emptyset K = \emptyset.$$

All the laws (1)–(15) listed in Section 6 are valid for $\mathcal{L}_p$ (replacing, of course, occurrences of $(-)^*$ by $(-)^\infty$, everywhere), but for law (7). The proofs are only slightly more involved due to a greater number of case distinctions. For instance, $K(L+M) = KL + KM$ (11) will now follow from the fact that

$$\{\langle K(L+M)+N,\, KL+KM+N\rangle \mid K, L, M, N \in \mathcal{L}_p\} \cup \{\langle K,\, K\rangle \mid K \in \mathcal{L}_p\}$$

is a bisimulation. Interestingly, the following equation

$$L{\uparrow} \wedge (K = LK + M) \quad \Rightarrow \quad K = L^\infty M \tag{26}$$

is proved in essentially the same way as law (13). Law number (7) is no longer valid: with the present interpretation of 0, $K0$ is generally different from 0. For instance, $a\,0 = \{a\}\{\delta\} = \{a\delta\}$. More interestingly, there is the following equation:

$$K^\infty 0 \quad = \quad K^\omega \tag{27}$$

which can be taken either as a definition of K^ω, or as a theorem once K^ω has been defined first. A coinductive definition of K^ω could be given by extending the set $\mathcal{E}_p$ of expressions with E^ω, and by specifying the following transitions and termination condition:

$$\frac{E \xrightarrow{a} E'}{E^\omega \xrightarrow{a} E'E^\omega}, \qquad (E^\omega){\uparrow}.$$

(Note that this definition is the same as for E^∞, but for the fact that $(E^\infty){\downarrow}$.)

15 Kleene's theorem for partial automata

Kleene's theorem, as formulated in Section 8, also holds for partial automata: For all languages L in $\mathcal{L}_p$,

$$L \text{ is regular iff } \langle L\rangle \text{ is a finite subautomaton of } \mathcal{L}_p. \tag{28}$$

It can be proved in almost exactly the same way as before, now using law (26) and the following variant of law (20). Let A be a finite alphabet and consider L in $\mathcal{L}_p$. If $B = \{a, \ldots, b\}$ is defined as the subset of A containing all input symbols c in A for which $L_c \neq \emptyset$, then:

$$L = \begin{cases} aL_a + \cdots + bL_b + 1 & \text{if } L{\downarrow} \\ aL_a + \cdots + bL_b + 0 & \text{if } L{\uparrow}. \end{cases} \tag{29}$$

(Note that if the set B is empty then the second expression is equal to 0.)

16 Notes and discussion

As we have seen in Section 12, most notions and observations of the present paper are instances of far more general ones, belonging to a theory called *(universal) coalgebra*. See [Rut96, JR97] and the references therein for more information on coalgebra. In [JMRR98], many recent developments in coalgebra are described.

The coalgebraic definition of bisimulation is a categorical generalization, due to Aczel and Mendler [AM89], of Park's [Par81] and Milner's [Mil80] notion of bisimulation for concurrent branching processes. This general categorical definition applies to many different examples, including nondeterministic (possibly probabilistic) transition systems, object-based systems, infinite data structures, various other types of automata, and dynamical systems. See [Rut96, JR97] for many examples and pointers to the literature.

The notions of homomorphism and (generated) subautomaton occur at various places in the literature (usually inspired by universal algebra), for instance in [Géc86].

The coinduction principle of Section 4 for the final automaton $\mathcal{L}$, together with the corresponding 'being is doing' characterization, applies more generally to any final coalgebra. Coinduction as a proof principle for greatest fixed points of monotone operators is already around for some time. For final coalgebras of the powerset functor, it has been introduced in [Acz88]. In [RT93], the principle is stated in its generality for arbitrary functors.

The word coinduction suggests a duality between induction and coinduction. This is explained by the observation that induction principles apply to *initial algebras*. Somewhat more concretely, the duality can be understood as follows. It is not difficult to prove that coinduction on $\mathcal{L}$ is equivalent to the statement that $\mathcal{L}$ has no *proper quotients*, that is, if $f : \mathcal{L} \to S$ is a surjective homomorphism then $\mathcal{L} \cong S$. This property is dual to the principle of mathematical induction on the algebra of natural numbers, which essentially states that the algebra of natural numbers has no *proper subalgebras*. See [Rut96, Sec.13] for a more detailed explanation.

The use of coinduction, both as a proof and as a definition method, is by now widespread (see for instance [BM96], which is a recent textbook on nonwellfounded set theory, and [JR97], for an introductory overview). Its application to languages and regular expressions, in Sections 6 and 10, is to the best of our knowledge new.

The calculation rules for a-derivatives (Section 5) of regular combinations of languages are well-known, have been reinvented several times, and are originally due to Brzozowski [Brz64] (see also [Con71] and [BS86]). Both Brzozowski's paper [Brz64] and Conway's book [Con71] contain, more generally, many of the ingredients that have been used in the present paper.

A well-known way of proving equality of regular expressions is to use a complete axiom system (of which the laws in Section 6 form a subset), such as given by Salomaa in [Sal66], and apply purely algebraic reasoning. The reader is invited to consult [Gin68, pp.68-69], from which the example $E_1 = F_1$ in Section 6 was taken, and convince himself of the greater complexity of that approach.

The most common and practical way of proving equality of two expressions is firstly, to construct for each expression an automaton that accepts the language it represents, and secondly, to minimize both automata. The two expressions are then equal iff the two resulting automata are isomorphic. For both the construction and the minimization step, many different and efficient algorithms exist (see [Wat95] for an extensive overview and comparison).

This classical approach is related to the coinduction proof method by the observation, in Section 2, that bisimulations are automata themselves. Thus also a proof by coinduction consists of the construction of an automaton. Our way of constructing this 'bisimulation automaton' is essentially based on Brzozowski's algorithm, using a-derivatives, but note that only one automaton is constructed for both expressions at the same time. Another difference is that this automaton need not be minimized in order to conclude that the two expressions are equal (this was illustrated by the bisimulation T used for the proof of $E_1 = F_1$ at the end of Section 6). The question whether this can lead to (more) efficient algorithms is yet to be addressed.

The connection between finality and minimality in Section 7 can already be found in [Gog73]. Our formulation of Kleene's theorem in Section 8 and its use as a criterion for nonregularity in Section 9 may be new, though the proofs involved are of course built from well-known ingredients.

Classically, the minimization of an automaton is obtained by identifying all states that are observationally equivalent. Referring to the notation of Section 12, two states s and s' are equivalent iff for all words w in A^*,

$$\hat{\delta}(s, w) \in F \iff \hat{\delta}(s', w) \in F,$$

where $\hat{\delta}(s, \epsilon) = s$ and $\hat{\delta}(s, wa) = \delta(\hat{\delta}(s, w), a)$. This notion of equivalence corresponds to our notion of *greatest* bisimulation relation (bisimilarity). Note that in the present theory, bisimulation relations are considered that generally are not maximal. This is yet another and maybe the most important difference with the classical approach.

Simulation relations have been studied in several forms and ways. We believe the present definition in Section 11, as well as the coinduction principle based on it, to be new. The definition of simulation *up-to-similarity* is a straightforward variation of Milner's notion of bisimulation *up-to-bisimilarity* [Mil80]. Some of the laws of Section 11 have been taken from [Koz94], where a complete axiom system for equality of regular expressions is presented in terms of equational implications.

The treatment of partial automata, which are coalgebras of the set functor $D'(S) = 2\times(1+S)^A$, has been inspired by a recent paper [Bre98] of Franck van Breugel, in which a related functor (on metric spaces) is studied. It comes somewhat as a surprise that the set $\mathcal{L}_p$, which is a final coalgebra of the *set* functor D', consists of *metrically closed* subsets. Such sets have been used at various places in the work of the French and Dutch schools of Nivat and De Bakker on metric semantics (cf. the recent textbook [BV96]). The notion of consistent language corresponds to the notion of *reduced* set in [dB91].

Automata theory has been and still is commonly understood as essentially algebraic. Cf. Ginzburg's *Algebraic theory of automata* [Gin68], Conway's *Regular algebra and finite machines* [Con71], and Kozen's recent textbook *Automata and computability*, from which the following quotation is taken [Koz97, p. 112]: "It should be pretty apparent by now that much of automata theory is just algebra." We hope to have shown that the coalgebraic treatment of automata theory offers, at least, an interesting alternative.

Acknowledgements

Many thanks to Dora Giammarresi for pointers to the literature, and to Jaco de Bakker, Marcello Bonsangue, Franck van Breugel, and Bart Jacobs for discussions and detailed comments.

References

[Acz88] P. Aczel. *Non-well-founded sets.* Number 14 in CSLI Lecture Notes. Center for the Study of Languages and Information, Stanford, 1988.

[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*, Lecture Notes in Computer Science, pages 357–365, 1989.

[BM96] J. Barwise and L.S. Moss. *Vicious Circles, On the Mathematics of Non-wellfounded Phenomena*. CSLI Lecture Notes. Center for the Study of Language and Information, Stanford, 1996.

[Bre98] F. van Breugel. Terminal metric spaces of finitely branching and image finite linear processes. *Theoretical Computer Science*, 202(1-2):223–230, 1998.

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

[BS86] G. Berry and R. Sethi. From regular expressions to deterministic automata. *Theoretical Computer Science*, 48:117–126, 1986.

[BV96] J.W. de Bakker and E. de Vink. *Control Flow Semantics*. Foundations of Computing Series. The MIT Press, 1996.

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

[dB91] J.W. de Bakker. Comparative semantics for flow of control in logic programming without logic. *Information and Computation*, 94(2):123–179, October 1991.

[Géc86] F. Gécseg. *Products of automata*, volume 7 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, 1986.

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

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

[JMRR98] B. Jacobs, L. Moss, H. Reichel, and J.J.M.M. Rutten, editors. *Proceedings of the first international workshop on Coalgebraic Methods in Computer Science (CMCS '98)*, volume 11 of *Electronic Notes in Theoretical Computer Science*. Elsevier Science B.V., 1998. Available at URL: www.elsevier.nl/locate/entcs.

[JR97] Bart Jacobs and Jan Rutten. A tutorial on (co)algebras and (co)induction. *Bulletin of EATCS*, 62:222–259, 1997.

[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.

[Koz94] D. Kozen. A completeness theorem for Kleene algebras and the algebra of regular events. *Information and Computation*, 110:366–390, 1994.

[Koz97] D.C. Kozen. *Automata and computability*. Undergraduate Texts in Computer Science. Springer-Verlag, 1997.

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

[Ner58] A. Nerode. Linear automaton transformations. *Proc. Amer. Math. Soc.*, 9:541–544, 1958.

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

[RS59] M.O. Rabin and D. Scott. Finite automata and their decision problems. *IBM J. Res. Develop.*, 3(2):114–125, 1959.

[RT93] J.J.M.M. 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 *Lecture Notes in Computer Science*, pages 477–530. Springer-Verlag, 1993.

[Rut96] J.J.M.M. Rutten. Universal coalgebra: a theory of systems. Report CS-R9652, CWI, 1996. FTP-available at `ftp.cwi.nl` as pub/CWIreports/AP/CS-R9652.ps.Z. To appear in TCS.

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

[Wat95] B.W. Watson. *Taxonomies and toolkits of regular language algorithms.* PhD thesis, Eindhoven University of Technology, 1995.