An introduction to finite automata and their connection to logic

Howard Straubing, Pascal Weil

Chapter 1

An Introduction to Finite Automata and their Connection to Logic

Howard Straubing*
Computer Science Department, Boston College, Chestnut Hill, Massachusetts, USA

Pascal Weil†
LaBRI, Université de Bordeaux and CNRS, Bordeaux, France

This introductory chapter is a tutorial on finite automata. We present the standard material on determinization and minimization, as well as an account of the equivalence of finite automata and monadic second-order logic. We conclude with an introduction to the syntactic monoid, and as an application give a proof of the equivalence of first-order definability and aperiodicity.

1.1. Introduction

1.1.1. *Motivation*

The word *automaton* (plural: *automata*) was originally used to refer to devices like clocks and watches, as well as mechanical marvels built to resemble moving humans and animals, whose internal mechanisms are hidden and which thus appear to operate spontaneously. In theoretical computer science, the *finite automaton* is among the simplest models of computation: A device that can be in one of finitely many *states,* and that receives a discrete sequence of inputs from the outside world, changing its state accordingly. This is in marked contrast to more general and powerful models of computation, such as Turing machines, in which the set of global states of the device—the so-called *instantaneous descriptions*—is infinite. A finite automaton is more akin to the control unit of the Turing machine (or, for that matter, the control unit of a modern computer processor), in which the present state of the unit and the input symbol under the reading head determine the next state of the unit, as well as signals to move the reading head left or right and to write a symbol on the machine's tape. The crucial distinction is that while the Turing machine can record and consult its entire computation history, all the

*Work partially supported by NSF Grant CCF-0915065
†Work partially supported by ANR 2010 BLAN 0202 01 FREC

information that a finite automaton can use about the sequence of inputs it has seen is represented in its current state.

But as rudimentary as this computational model may appear, it has a rich theory, and many applications. In this introductory chapter, we will present the core theory: that of a finite automaton reading a finite *word,* that is, a finite string of inputs, and using the resulting state to decide whether to accept or reject the word. The central question motivating our presentation is to determine what properties of words can be decided by finite automata. Subsequent chapters will present both generalizations of the basic model (to devices that read infinite words, labeled trees, *etc.*) and to applications. An important theme in this chapter, as well as throughout the volume, is the close connection between automata and formal logic.

1.1.2. *Plan of the chapter*

In Section 1.2, we introduce finite automata as devices for recognizing formal languages, and show the equivalence of several variants of the basic model, most notably the equivalence of deterministic and nondeterministic automata. Section 1.3 describes Büchi's sequential calculus, the framework in predicate logic for describing properties of words that are recognizable by finite automata. In Section 1.4 we prove what might well be described as the two fundamental theorems of finite automata: that the languages recognized by finite automata are exactly those definable by sentences of the sequential calculus, and also exactly those definable by rational expressions (also called regular expressions). Section 1.5 presents methods that can be used to show certain languages cannot be recognized by finite automata. The last sections, 1.6 and 1.7, have a more algebraic flavor: we introduce both the minimal automaton and the syntactic monoid of a language, and prove the important McNaughton-Schützenberger theorem describing the languages definable in the first-order fragment of the sequential calculus.

1.1.3. *Notation*

Throughout this chapter, A denotes a finite *alphabet*, that is, a finite non-empty set. Elements of A are called *letters*, and a finite sequence of letters is called a *word*. We denote words simply by concatenating the letters, so, for example, if $A = \{a, b, c\}$, then *aabacba* is a word over A. The *empty sequence* is considered a word, and we use ε to denote this sequence. The set of all words over A is denoted A^*, and the set of all nonempty words is denoted A^+. The *length* of the word w, that is, the number of letters in w, is denoted $|w|$.

If $u, v \in A^*$ then we can form a new word uv by concatenating the two sequences. Concatenation of words is obviously an associative and (unless A has a single element) noncommutative operation on A^*. We have

$$|uv| = |u| + |v|, \text{ and}$$
$$u\,\varepsilon = \varepsilon\,u = u.$$

(Other texts frequently use Λ or 1 to denote the empty word. The latter choice is justified by the second equation above.)

A subset of A^* is called a *language* over A.

1.1.4. *Historical note and references*

This chapter contains a modern presentation of material that goes back more than fifty years. The reader can find other accounts in classic papers and texts: The equivalence of finite automata and rational expressions given in Section 1.4 was first described by Kleene in [9]. The connection with monadic second-order logic was found independently by Trakhtenbrot [24] and Büchi [1].

Nondeterministic automata were introduced by Rabin and Scott [17], who showed their equivalence to deterministic automata. Minimization of finite-state devices (framed in the language of switching circuits built from relays) is due to Huffman [8]. The simple congruential account of minimization that we give originates with Myhill [13] and Nerode [14].

The equivalence of aperiodicity of the syntactic monoid with star-freeness is due to Schützenberger [19], and the connection with first-order logic is from McNaughton and Papert [11]. Our account of these results relies heavily on an argument given in Wilke [23].

Rational expressions, determinization and minimization have become part of the basic course of study in theoretical computer science, and as such are described in a number of undergraduate textbooks. Hopcroft and Ullman [7], Lewis and Papadimitriou [10] and the more recent Sipser [20] are notable examples. A more technical and algebraically-oriented account is given in the monograph by Eilenberg [4, 5]. An algebraic view of automata is developed by Sakarovitch [18]. Detailed accounts of the connection between automata, logic and algebra can be found in Straubing [21] and Thomas [22]. The state of the art, especially concerning the algebraic classification of automata, will appear in the forthcoming handbook [16].

1.2. Automata and rational expressions

1.2.1. *Operations on languages*

We describe here a collection of basic operations on languages, which will be building blocks in the characterization of the expressive power of automata.

Since languages over A are subsets of A^*, we may of course consider the boolean operations: union, intersection and complement. The product operation on words can be naturally extended to languages: if K and L are languages over A, we define their *concatenation product* KL to be the set of all products of a word in K followed by a word in L:

$$KL = \{uv \mid u \in K \text{ and } v \in L\}.$$

We also use the power notation for languages: if $n > 0$, L^n is the product $LL\cdots L$ of n copies of L. We let $L^0 = \{\varepsilon\}$. Note that if $n > 1$, L^n differs from the set of n-th powers of the elements of L. The *iteration* (or *Kleene star*) of a language L is the language $L^* = \bigcup_{n\geq 0} L^n$.

Finally, we introduce a simple rewriting operation, based on the use of morphisms. If A and B are alphabets, a *morphism* from A^* to B^* is a mapping $\varphi\colon A^* \to B^*$ such that

(1) $\varphi(\varepsilon) = \varepsilon$,
(2) for all $u, v \in A^*$, $\varphi(uv) = \varphi(u)\varphi(v)$.

To specify such a morphism, it suffices to give the images of the letters of A. Then the image of a word $u \in A^*$, say $u = a_1 \cdots a_n$, is obtained by taking the concatenation of the images of the letters, $\varphi(u) = \varphi(a_1)\cdots\varphi(a_n)$. That is, $\varphi(a_1 \cdots a_n)$ is obtained from $a_1 \cdots a_n$ by substituting for each letter a_i the word $\varphi(a_i)$. This operation naturally extends from words to languages: if $L \subseteq A^*$, then $\varphi(L) = \{\varphi(u) \mid u \in L\}$.

The consideration of these operations leads to the classical definition of *rational* languages (also called *regular* languages). The operations of union, concatenation and iteration are called the *rational operations*. A language over alphabet A is called rational if it can be obtained from the letters of A by applying (a finite number of) rational operations.

More formally, the class of rational languages over the alphabet A, denoted $\mathsf{Rat}A^*$, is the least class of languages such that

(1) the languages $\emptyset$ and $\{a\}$ are rational for each letter $a \in A$;
(2) if K and L are rational languages, then $K \cup L$, KL and L^* are also rational.

Example 1.1. The language $\left(\left(a^*(ab)^*A^* \cap A^*(ba)^*\right)^2\right)^*$ is rational. (Note that in order to lighten the notation, we write a, b, etc., instead of $\{a\}$, $\{b\}$.)

The language $\{\varepsilon\}$, containing just the empty word, is rational. Indeed, it is equal to $\emptyset^*$.

Any finite language (that is, containing only finitely many words) is rational.

Let $a, b \in A$ be distinct letters. It is instructive to show that the following languages are rational: (a) the set of all words which do not contain two consecutive a; (b) the set of all words which contain the factor ab but not the factor ba.

We also consider the *extended rational operations*: these are the rational operations, and the operations of intersection, complement and morphic image. A language is said to be *extended rational* if it can be obtained from the letters of A by applying (a finite number of) extended rational operations. The class of extended rational languages over A is written $\mathsf{X\text{-}Rat}A^*$.

Of course, all rational languages are extended rational. The definition of extended rational languages offers more expressive possibilities but as we will see,

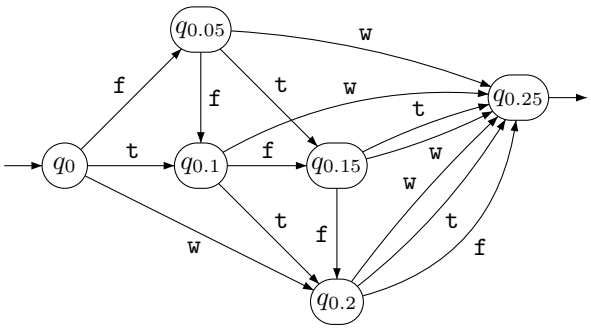

Fig. 1.1. The automaton of a (simplified) coffee machine

they are not properly more expressive than rational languages.

1.2.2. *Automata*

Let us start with a couple of examples.

Example 1.2. A coffee machine delivers a cup of coffee for €.25. It accepts only coins of €.20, €.10 and €.05. While determining whether it has received a sufficient sum, the machine is in one of six states, q_0, $q_{0.05}$, $q_{0.1}$, $q_{0.15}$, $q_{0.2}$ and $q_{0.25}$. The names of the states correspond to the sum already received. The machine changes state after a new coin is inserted, and the new state it assumes is a function of the value of the new coin inserted and of the sum already received. The latter information is encoded in the current state of the machine.

Here, the input word is the sequence of coins inserted, and the alphabet consists of three letters, `w`, `t` and `f`, standing respectively for t`w`enty cents, `t`en cents and `f`ive cents. The machine is represented in Figure 1.1.

The incoming arrow indicates the initial state of the machine (q_0), and the outgoing arrow indicates the only accepting state ($q_{0.25}$), that is, the state in which the machine will indeed prepare a cup of coffee for you. Notice that the machine does not return change, but that it will accept sums up to €.40.

Example 1.3. Our second example (Figure 1.2) reads an integer, given by its binary expansion and read from right to left, that is, starting with the bit of least weight. Upon reading this word on alphabet $\{0, 1\}$, the automaton decides whether the given integer is divisible by 3 or not.

For instance, consider the integer 19, in binary expansion `10011`: our input word is `11001`. It is read letter by letter, starting from the initial state (the state indicated by an incoming arrow, state r_0). After each new letter is read, we follow the corresponding edge starting at the current state. Thus, starting in state r_0, we visit successively the states r'_1, r_0, r'_0, r_0 again, and finally r'_1. This state is not accepting (it is not marked with an outgoing edge), so the word `11001` is not

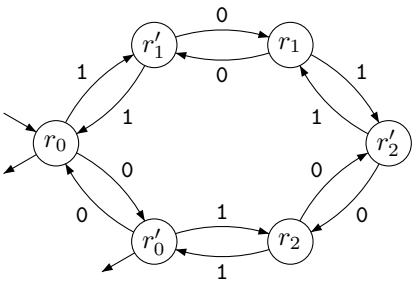

Fig. 1.2. An automaton to compute mod 3 remainders

accepted by the automaton. And indeed, 19 is not divisible by 3.

In contrast, 93 is divisible by 3, which is confirmed by running its binary expansion, namely `1011101`, read from right to left, through the automaton: starting in state r_0, we end in state r'_0.

The reader will quickly see that this automaton is constructed in such a way that, if n is an integer and w_n is the binary expansion of n, then the state reached when reading w_n from right to left, starting in state r_0, is r_k (resp. r'_k) if n is congruent to k (mod 3) and w_n has even (resp. odd) length.

We now turn to a formal definition. A (*finite state*) *automaton* on alphabet A is a 4-tuple $\mathcal{A} = (Q, T, I, F)$ where Q is a finite set, called the set of *states*, T is a subset of $Q \times A \times Q$, called the set of *transitions*, and I and F are subsets of Q, called respectively the sets of *initial states* and *final states*. Final states are also called *accepting states*.

For instance, the automaton of Example 1.2 uses a 3-letter alphabet, $A = \{\mathtt{f}, \mathtt{t}, \mathtt{w}\}$. Formally, it is the automaton $\mathcal{A} = (Q, T, I, F)$ given by $Q = \{q_0, q_{0.05}, q_{0.1}, q_{0.15}, q_{0.2}, q_{0.25}\}$, $I = \{q_0\}$, $F = \{q_{0.25}\}$ and T is a 15-element subset of $Q \times A \times Q$ containing such triples as $(q_0, \mathtt{f}, q_{0.05})$, $(q_{0.1}, \mathtt{t}, q_{0.2})$ or $(q_{0.2}, \mathtt{w}, q_{0.25})$.

As in our first examples, it is often convenient to represent an automaton $\mathcal{A} = (Q, T, I, F)$ by a labeled graph, whose vertices are the elements of Q (the states) and whoses edges are of the form $q \xrightarrow{a} q'$ if (q, a, q') is a transition, that is, if $(q, a, q') \in T$. The initial states are specified by an incoming arrow, and the final states are specified by an outgoing edge.

From now on, we will most often specify our automata by their graphical representations.

Example 1.4. Here, the alphabet is $A = \{a, b\}$. Figure 1.3 represents the automaton $\mathcal{A} = (Q, T, I, F)$ where $Q = \{1, 2, 3\}$, $I = \{1\}$, $F = \{3\}$ and

$$T = \{(1, a, 1), (1, b, 1), (1, a, 2), (2, b, 3), (3, a, 3), (3, b, 3)\}.$$

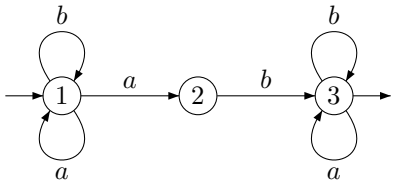

Fig. 1.3. An automaton accepting A^*abA^*

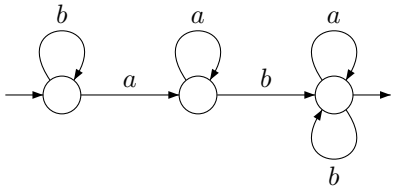

Fig. 1.4. Another automaton accepting A^*abA^*

1.2.2.1. *The language accepted by an automaton*

A *path* in automaton $\mathcal{A}$ is a sequence of consecutive edges,

$$p = (q_0, a_1, q_1)(q_1, a_2, q_2) \ \cdots \ (q_{n-1}, a_n, q_n),$$

also drawn as

$$p = q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \ \cdots \ \xrightarrow{a_n} q_n.$$

Then we say that p is a path of *length* n from q_0 to q_n, *labeled* by the word $u = a_1a_2\cdots a_n$. By convention, for each state q, there exists an *empty path* from q to q labeled by the empty word.

For instance, in the automaton of Figure 1.3, the word a^3ba labels exactly four paths: from 1 to 1, from 1 to 2, from 1 to 3 and from 3 to 3.

A path p is *successful* if its initial state is in I and its final state is in F. A word w is *accepted* (or *recognized*) by $\mathcal{A}$ if there exists a successful path in the automaton with label w. And the *language accepted* (or *recognized*) by $\mathcal{A}$ is the set of labels of successful paths in $\mathcal{A}$. It is denoted by $L(\mathcal{A})$. We say that $\mathcal{A}$ *accepts* (or *recognizes*) $L(\mathcal{A})$.

For instance, the language of the automaton of Figure 1.1 is finite, with exactly 27 words. The automaton of Figure 1.3 accepts the set of words in which at least one occurrence of a is followed immediately by a b, namely A^*abA^*, where $A = \{a, b\}$.

Different automata may recognize the same language: if $\mathcal{A}$ and $\mathcal{B}$ are automata such that $L(\mathcal{A}) = L(\mathcal{B})$, we say that $\mathcal{A}$ and $\mathcal{B}$ are *equivalent*.

Example 1.5. The language A^*abA^*, accepted by the automaton in Figure 1.3, is also recognized by the automaton in Figure 1.4

A language L is said to be *recognizable* if it is recognized by an automaton.

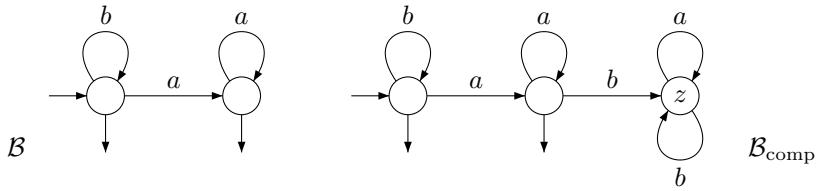

Fig. 1.5. Two automata accepting b^*a^*

1.2.2.2. *Complete automata*

An automaton $\mathcal{A} = (Q, T, I, F)$ on alphabet A is said to be *complete* if, for each state $q \in Q$ and each letter $a \in A$, there exists at least one transition of the form (q, a, q'): in graphical representation, this means that, for each letter of the alphabet, there is an edge labeled by that letter starting from each state. Naturally, this easily implies that, for each state q and each word $w \in A^*$, there exists at least one path labeled w starting at q.

Every automaton can easily be turned into an equivalent complete automaton. If $\mathcal{A} = (Q, T, I, F)$ is not complete, the *completion* of $\mathcal{A}$ is the automaton $\mathcal{A}_{\text{comp}} = (Q', T', I, F)$ given by $Q' = Q \cup \{z\}$, where z is a new state not in Q, and T' is obtained by adding to T all triples (z, a, z) $(a \in A)$ and all triples (q, a, z) $(q \in Q,$ $a \in A)$ such that there is no element of the form (q, a, q') in T.

If $\mathcal{A}$ is complete, we let $\mathcal{A}_{\text{comp}} = \mathcal{A}$. It is immediate that, in every case, $\mathcal{A}_{\text{comp}}$ is complete and $L(\mathcal{A}_{\text{comp}}) = L(\mathcal{A})$.

Example 1.6. Let $A = \{a, b\}$. The automaton $\mathcal{B}$ in Figure 1.5, which accepts the language b^*a^*, is evidently not complete. The automaton $\mathcal{B}_{\text{comp}}$ is represented next to it.

1.2.2.3. *Trim automata*

A complete automaton reads its entire input before deciding to accept or reject it: whatever input it receives, there is a transition that can be followed. However, we have seen that in the completion $\mathcal{A}_{\text{comp}}$ of a non-complete automaton $\mathcal{A}$, state z does not participate in any successful path: it is in a way a useless state. *Trimming* an automaton removes such useless states; it is, in a sense, the opposite of completing an automaton, and aims at producing a more concise device.

A state q of an automaton $\mathcal{A}$ is said to be *accessible* if there exists a path in $\mathcal{A}$ starting from some initial state and ending at q. State q is *co-accessible* if there exists a path in $\mathcal{A}$ starting from q and ending at some final state. Observe that a state is both accessible and co-accessible if and only if it is visited by at least one successful path.

The automaton $\mathcal{A}$ itself is *trim* if all its states are both accessible and co-accessible: in a trim automaton, each state is useful, in the sense that it is used in accepting some word of the language $L(\mathcal{A})$.

Of course, every automaton $\mathcal{A}$ is equivalent to a trim one, written $\mathcal{A}_{\text{trim}}$, obtained by restricting $\mathcal{A}$ to its accessible and co-accessible states and to the transitions between them.

Interestingly, $\mathcal{A}_{\text{trim}}$ can be constructed efficiently, using breadth-first search. One first computes the accessible states of $\mathcal{A}$, by letting $Q_0 = I$ (the initial states are certainly accessible) and by computing iteratively

$$Q_{n+1} \;=\; Q_n \cup \bigcup_{q \in Q_n, a \in A} \{q' \in Q \mid (q, a, q') \in T\}.$$

One verifies that the elements of Q_n are the states that can be reached from an initial state, reading a word of length at most n; and that if two consecutive sets Q_n and Q_{n+1} are equal, then $Q_n = Q_m$ for all $m \geq n$, and Q_n is the set of accessible states of $\mathcal{A}$. In particular, the set of accessible states is computed in at most $|Q|$ steps.

A similar procedure, starting from the final states instead of the initial states, and working in reverse, produces in at most $|Q|$ steps the set of co-accessible states of $\mathcal{A}$. The automaton $\mathcal{A}_{\text{trim}}$ is then immediately constructed.

Remark 1.1. The construction of $\mathcal{A}_{\text{trim}}$, or indeed, just of the set of accessible states of $\mathcal{A}$ provides an efficient solution of the *emptiness problem*: given an automaton $\mathcal{A}$, is the language $L(\mathcal{A})$ empty? that is, does $\mathcal{A}$ accept at least one word?

Indeed, $\mathcal{A}$ recognizes the empty set if and only if no final state is accessible: in order to decide the emptiness problem for automaton $\mathcal{A}$, it suffices to construct the set of accessible states of $\mathcal{A}$ and verify whether it contains a final state. This yields an $\mathcal{O}(|Q|^2|A|)$ algorithm.

1.2.2.4. *Epsilon-automata*

It is sometimes convenient to extend the notion of automata to the so-called ε-*automata*: the difference from ordinary automata is that we also allow ε-labeled transitions, of the form (p, ε, q) with $p, q \in Q$.

Proposition 1.1. *Every ε-automaton is equivalent to an ordinary automaton.*

Sketch of proof. Let $\mathcal{A} = (Q, T, I, F)$ be an ε-automaton, and let $\mathcal{R}$ be the relation on Q given by $p \,\mathcal{R}\, q$ if there exists a path from p to q consisting only of ε-labeled transitions (that is: $\mathcal{R}$ is the reflexive transitive closure of the relation defined by the ε-labeled transitions of $\mathcal{A}$).

Let $\mathcal{A}'$ be the (ordinary) automaton given by the tuple (Q, T', I', F) with

$$T' = \big\{(p, a, q) \mid (p, a, q') \in T \text{ and } q' \,\mathcal{R}\, q \text{ for some } q' \in Q\big\}$$
$$I' = \big\{q \mid p \,\mathcal{R}\, q \text{ for some } p \in I\big\}.$$

Then $\mathcal{A}'$ is equivalent to $\mathcal{A}$. □

1.2.3. *Deterministic automata*

Example 1.7. Consider the automaton of Figure 1.3, say $\mathcal{A}$, and the automaton $\mathcal{B}$ of Figure 1.4. Both recognize the language, $L = A^*abA^*$, but there is an important, qualitative difference beween them.

We have defined automata as *nondeterministic* computing devices: given a state and an input letter, there may be several possible choices for the next state. Thus an input word might be associated with many different computation paths, and the word is accepted if one of these paths ends at an accepting state. In contrast, $\mathcal{B}$ has the convenient property that each input word labels at most one computation path.

These remarks are formalized in the following definition. An automaton $\mathcal{A} = (Q, T, I, F)$ is said to be *deterministic* if it has exactly one initial state, and if, for each letter a and for all states q, q', q'',

$$(q, a, q'),\ (q, a, q'') \in T \qquad \Longrightarrow \qquad q' = q''.$$

Thus, of the automata in Figures 1.3 and 1.4, the second one is deterministic, and the first is non-deterministic.

This definition imposes a certain condition of uniqueness on transitions, that is, on paths of length 1. This property is then extended to longer paths by a simple induction.

Proposition 1.2. *Let $\mathcal{A}$ be a deterministic automaton and let w be a word.*

(1) *For each state q of $\mathcal{A}$, there exists at most one path labeled w starting at q.*
(2) *If $w \in L(\mathcal{A})$, then w labels exactly one successful path.*

In particular, we can represent the set of transitions of a deterministic automaton $\mathcal{A} = (Q, T, I, F)$ by a *transition function*: the (possibly partial) function $\delta\colon Q \times A \to Q$ which maps each pair $(q, a) \in Q \times A$ to the state q' such that $(q, a, q') \in T$ (if it exists). This function is then naturally extended to the set $Q \times A^*$: if $q \in Q$ and $w \in A^*$, $\delta(q, w)$ is the state q' such that there exists a path from q to q' labeled by w in $\mathcal{A}$ (if such a state exists). In the sequel, deterministic automata will be specified as 4-tuples (Q, δ, i, F) instead of the corresponding $(Q, T, \{i\}, F)$. We note the following elementary characterization of δ.

Proposition 1.3. *Let $\mathcal{A} = (Q, \delta, i, F)$ be a deterministic automaton. Then we have*

$$\delta(q, \varepsilon) = q;$$

$$\delta(q, ua) = \begin{cases} \delta(\delta(q, u), a) & \text{if both } \delta(q, u) \text{ and } \delta(\delta(q, u), a) \text{ exist,} \\ \text{undefined} & \text{otherwise;} \end{cases}$$

$$u \in L(\mathcal{A}) \text{ if and only if } \delta(i, u) \in F.$$

for each state q, each word $u \in A^*$ and each letter $a \in A$.

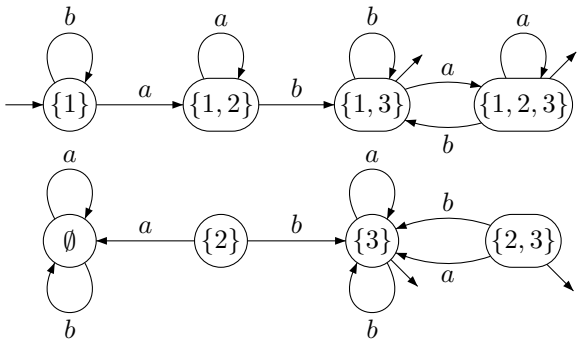

Fig. 1.6. The subset automaton of the automaton in Figure 1.3

Again, it turns out that every automaton is equivalent to a deterministic automaton. This deterministic automaton can be effectively constructed, although the algorithm – the so-called *subset construction* – is more complicated than those used to construct complete or trim automata.

Let $\mathcal{A} = (Q, T, I, F)$ be an automaton. The *subset transition function* of $\mathcal{A}$ is the function $\delta\colon \mathcal{P}(Q) \times A \to \mathcal{P}(Q)$ defined, for each $P \subseteq Q$ and each $a \in A$ by

$$\delta(P, a) = \{q \in Q \mid \exists p \in P,\ (p, a, q) \in T\}.$$

Thus, $\delta(P, a)$ is the set of states of $\mathcal{A}$ which can be reached by an a-labeled transition, starting from an element of P. The *subset automaton* of $\mathcal{A}$ is $\mathcal{A}_{\text{sub}} = (\mathcal{P}(Q), \delta, I, F_{\text{sub}})$ where $F_{\text{sub}} = \{P \subseteq Q \mid P \cap F \neq \emptyset\}$.

The automaton $\mathcal{A}_{\text{sub}}$ is deterministic and complete by construction, and the subset transition function of $\mathcal{A}$ is the transition function of $\mathcal{A}_{\text{sub}}$. Moreover, if $\mathcal{A}$ has n states, then $\mathcal{A}_{\text{sub}}$ has 2^n states.

Example 1.8. The subset automaton of the non-deterministic automaton of Figure 1.3 is given in Figure 1.6. Notice that the states of the second row are not accessible.

Proposition 1.4. *The automata $\mathcal{A}$ and $\mathcal{A}_{\text{sub}}$ are equivalent.*

Sketch of proof. Let $\mathcal{A} = (Q, T, I, F)$. One shows by induction on $|w|$ that for all $P \subseteq Q$ and $w \in A^*$, $\delta(P, w)$ is the set of all states $q \in Q$ such that w labels a path in $\mathcal{A}$ starting at some state in P and ending at q.

Therefore, a word w is accepted by $\mathcal{A}$ if and only if at least one final state lies in the set $\delta(I, w)$, if and only if $\delta(I, w) \in F_{\text{sub}}$, if and only if w is accepted by $\mathcal{A}_{\text{sub}}$. This concludes the proof. □

In general, the subset automaton is not trim (see Example 1.8) and we can find a deterministic automaton smaller than $\mathcal{A}_{\text{sub}}$, which still recognizes the same language as $\mathcal{A}$, namely by trimming $\mathcal{A}_{\text{sub}}$. Observe that in the proof of Proposition 1.4, the

only useful states of $\mathcal{A}_{\rm sub}$ are those of the form $\delta(I, w)$, that is, the accessible states of $\mathcal{A}_{\rm sub}$.

We define the *determinized automaton* of $\mathcal{A}$ to be $\mathcal{A}_{\rm det} = (\mathcal{A}_{\rm sub})_{\rm trim}$. This automaton is equivalent to $\mathcal{A}$.

Example 1.9. The determinized automaton of the non-deterministic automaton of Figure 1.3 consists of the first row of states in Figure 1.6 (see Example 1.8).

An obstacle in the computation of $\mathcal{A}_{\rm det}$ is the explosion in the number of states: if $\mathcal{A}$ has n states, then $\mathcal{A}_{\rm sub}$ has 2^n states. The determinized automaton $\mathcal{A}_{\rm det}$ may well have exponentially many states as well, but it sometimes has fewer. Therefore, it makes sense to try and compute $\mathcal{A}_{\rm det}$ directly, in time proportional to its actual number of states, rather than first constructing the exponentially large automaton $\mathcal{A}_{\rm sub}$ and then trimming it.

This can be done using the same ideas as in the construction of $\mathcal{A}_{\rm trim}$ in Section 1.2.2.3. One first constructs $\mathcal{B}$, the accessible part of $\mathcal{A}_{\rm sub}$, starting with the initial state of $\mathcal{A}_{\rm sub}$, namely I. Then for each constructed state P and each letter a, we construct $\delta(P, a)$ and the transition $(P, a, \delta(P, a))$. And we stop when no new state arises this way.

The second step consists in finding the co-accessible part of $\mathcal{B}$, using the method in Section 1.2.2.3.

Example 1.10. Let $A = \{a, b\}$, let $n \geq 2$, and let $L = A^*aA^{n-2}$. Then L is accepted by a non-deterministic automaton $\mathcal{A}$ with n states. However, any deterministic automaton accepting L must have at least 2^{n-1} states. To see this, suppose that (Q, δ, i, F) is such a deterministic automaton. Let u, v be distinct words of length $n-1$. Then one of the words (let us say u) contains an a in a position in which v contains the letter b. Thus $u = u'ax$, $v = v'by$, where $|x| = |y|$. Let w be any word of length $n - 2 - |x|$. Then $uw \in L$, $vw \notin L$. It follows that $\delta(i, u) \neq \delta(i, v)$ and thus there are at least as many states as there are words of length $n-1$. This shows that the exponential blowup in the number of states in the subset construction cannot in general be reduced.

1.3. Logic: Büchi's sequential calculus

Let us start with an example.

Example 1.11. Recall that $\wedge$ is the logical conjunction, which reads "AND". And $\vee$ is the logical disjunction, which reads "OR". We will consider formulas such as

$$\exists x \exists y \ (x < y) \wedge R_a x \wedge R_b y.$$

This formula has the following interpretation on a word u: there exist two natural numbers $x < y$ such that, in u, the letter in position x is an a and the letter in position y is a b. Thus this formula specifies a language: the set of all words u in which this formula holds, namely $A^*aA^*bA^*$.

1.3.1. *First-order formulas*

Let us now formalize this point of view on languages.

1.3.1.1. *Syntax*

The formulas of Büchi's sequential calculus use the usual logical symbols ($\wedge$, $\vee$, $\neg$ for the negation), the equality symbol $=$, the constant symbol **true**, the quantifiers $\exists$ and $\forall$, variable symbols $(x, y, z, \ldots)$ and parentheses. They also use specific, non-logical symbols: binary relation symbols $<$ and S, and unary relation symbols R_a (one for each letter $a \in A$).

For convenience, we may assume that the variables are drawn from a fixed, countable, set of variables.

The *atomic formulas* are the formulas of the form **true**, $x = y$, $x < y$, $S(x, y)$, and $R_a x$, where x and y are variables and $a \in A$.

The *first-order formulas* are defined as follows:

- Atomic formulas are first-order formulas,
- If φ and ψ are first-order formulas, then $(\neg\varphi)$, $(\varphi \wedge \psi)$ and $(\varphi \vee \psi)$ are first-order formulas,
- If φ is a first-order formula and if x is a variable, then $(\exists x\ \varphi)$ and $(\forall x\ \varphi)$ are first-order formulas.

Remark 1.2. As is usual in logic, we will limit the usage of parentheses in our notation of formulas, to what is necessary for their proper parsing, writing for instance $\forall x\ R_a x$ instead of $(\forall x\ (R_a x))$.

Certain variables appear after a quantifier (existential or universal): occurrences of these variables within the scope of the quantifier are said to be *bound*. Other occurrences are said to be *free*. A precise, recursive, definition of the set $FV(\varphi)$ of the free variables of a formula φ is as follows:

- If φ is atomic, then $FV(\varphi)$ is the set of all variables occurring in φ,
- $FV(\neg\varphi) = FV(\varphi)$,
- $FV(\varphi \wedge \psi) = FV(\varphi \vee \psi) = FV(\varphi) \cup FV(\psi)$,
- $FV(\exists x\ \varphi) = FV(\forall x\ \varphi) = FV(\varphi) \setminus \{x\}$.

A formula without free variables is called a *sentence*.

1.3.1.2. *Interpretation of formulas*

In Büchi's sequential calculus, formulas are interpreted in words: each word u of length $n \geq 0$ determines a *structure* (which we abusively denote by u) with *domain* $\mathsf{Dom}(u) = \{0, \ldots, n-1\}$ ($\mathsf{Dom}(u) = \emptyset$ if $u = \varepsilon$). $\mathsf{Dom}(u)$ is viewed as the set of positions in the word u (numbered from 0).

The symbol $<$ is interpreted in $\mathsf{Dom}(u)$ as the usual order (as in $(2 < 4)$ and $\neg(3 < 2)$). The symbol S is interpreted as the *successor* symbol: if $x, y \in \mathsf{Dom}(u)$, then $S(x, y)$ if and only if $y = x + 1$. Finally, for each letter $a \in A$, the unary relation symbol R_a is interpreted as the set of positions in u that carry an a (a subset of $\mathsf{Dom}(u)$).

Example 1.12. If $u = abbaab$, then $\mathsf{Dom}(u) = \{0, 1, \ldots, 5\}$, $R_a = \{0, 3, 4\}$ and $R_b = \{1, 2, 5\}$.

A *valuation* on u is a mapping ν from a set of variables into the domain $\mathsf{Dom}(u)$. It will be useful to have a notation for small modifications of a valuation: if ν is a valuation and d is an element of $\mathsf{Dom}(u)$, we let $\nu[x \mapsto d]$ be the valuation ν' defined by extending the domain of ν to include the variable x and setting

$$\nu'(y) = \begin{cases} \nu(y) & \text{if } y \neq x, \\ d & \text{if } y = x. \end{cases}$$

If φ is a formula, $u \in A^*$ and ν is a valuation on u whose domain includes the free variables of φ, then we define $u, \nu \models \varphi$ (and say that the valuation ν *satisfies* φ in u, or equivalently u, ν satisfies φ) as follows:

- $u, \nu \models (x = y)$ (resp. $(x < y)$, $S(x, y)$, $R_a x$) if and only if $\nu(x) = \nu(y)$ (resp. $\nu(x) < \nu(y)$, $S(\nu(x), \nu(y))$, $R_a \nu(x)$) in $\mathsf{Dom}(u)$;
- $u, \nu \models \neg\varphi$ if and only if it is not true that $u, \nu \models \varphi$;
- $u, \nu \models (\varphi \vee \psi)$ (resp. $(\varphi \wedge \psi)$) if and only if at least one (resp. both) of $u, \nu \models \varphi$ and $u, \nu \models \psi$ holds (resp. hold);
- $u, \nu \models (\exists x\, \varphi)$ if and only if there exists $d \in \mathsf{Dom}(u)$ such that $u, \nu[x \mapsto d] \models \varphi$;
- $u, \nu \models (\forall x\, \varphi)$ if and only if, for each $d \in \mathsf{Dom}(u)$, $u, \nu[x \mapsto d] \models \varphi$.

Note that the truth value of $u, \nu \models \varphi$ depends only on the values assigned by ν to the free variables of φ. In particular, if φ is a sentence, then there is a valuation μ with an empty domain. We say that φ *is satisfied by* u (or u *satisfies* φ), and we write $u \models \varphi$ for $u, \mu \models \varphi$. Thus each sentence φ defines a language: the set $L(\varphi)$ of all words such that $u \models \varphi$. Note that this interpretation makes sense even if u is the empty word, for then the valuation μ is still defined: Every sentence beginning with a universal quantifier is satisfied by ε, and no sentence beginning with an existential quantifier is satisfied by ε. An early example was given in Example 1.11,

Remark 1.3. Two sentences φ and ψ are said to be *logically equivalent* if they are satisfied by the same structures. We will use freely the classical logical equivalence results, such as the logical equivalence of $\varphi \wedge \psi$ and $\neg(\neg\varphi \vee \neg\psi)$, or the logical equivalence of $\forall x\ \varphi$ and $\neg(\exists x\ \neg\varphi)$. We will also use the implication and bi-implication notation: $\varphi \to \psi$ stands for $\neg\varphi \vee \psi$ and $\varphi \leftrightarrow \psi$ stands for $(\varphi \to \psi) \wedge (\psi \to \varphi)$.

Example 1.13. Let φ and ψ be the following formulas.

$$\varphi = \exists x \left(\big(\forall y \ \neg(y < x)\big) \wedge R_a x \right)$$
$$\psi = \forall x \left(\big(\forall y \ \neg(y < x)\big) \rightarrow R_a x \right).$$

The sentence φ states that there exists a position with no strict predecessor, containing an a, while ψ states that every such position contains an a. The latter sentence, like all universally quantified first-order sentences, is vacuously satisfied by the empty string. Thus $L(\varphi) = aA^*$ and $L(\psi) = aA^* \cup \{\varepsilon\}$.

The *first-order logic* of the *linear order* (resp. of the *successor*), written $\mathsf{FO}(<)$ (resp. $\mathsf{FO}(S)$) is the fragment of the first-order logic described so far, where formulas do not use the symbol S (resp. $<$).

1.3.2. *Monadic second-order formulas*

In *monadic second-order logic*, we add a new type of variable to first-order logic, called *set variables* and usually denoted by upper case letters, e.g. $X, Y, \ldots$ The atomic formulas of monadic second-order are the atomic formulas of first-order logic, and the formulas of the form (Xy), where X is a set variable and y is an ordinary variable.

The recursive definition of *monadic second-order formulas*, starting from the atomic formulas, closely resembles that of first-order formulas: it uses the same rules given in Section 1.3.1, and the additional rule:

- If φ is a monadic second-order formula and X is a set variable, then $(\exists X \varphi)$ and $(\forall X \varphi)$ are monadic second-order formulas.

The notion of free variables is extended in the same fashion.

The interpretation of monadic second-order formulas also requires an extension of the definition of a valuation on a word u: a *monadic second-order valuation* is a mapping ν which associates with each first-order variable an element of the domain $\mathsf{Dom}(u)$, and with each set variable, a subset of $\mathsf{Dom}(u)$.

If ν is a valuation, X is a set variable, and R is a subset of $\mathsf{Dom}(u)$, we denote by $\nu[X \mapsto R]$ the valuation obtained from ν by mapping X to R (see Section 1.3.1.2).

With these definitions, we can recursively give a meaning to the notion that a valuation ν satisfies a formula φ in a word u ($u, \nu \models \varphi$): we use again the rules given in Section 1.3.1.2, to which we add the following:

- $u, \nu \models (Xy)$ if and only if $\nu(y) \in \nu(X)$;
- $u, \nu \models (\exists X \varphi)$ (resp. $(\forall X \varphi)$) if and only if there exists $R \subseteq \mathsf{Dom}(u)$ such that (resp. for each $R \subseteq \mathsf{Dom}(u)$) $u, \nu[X \mapsto R] \models \varphi$.

Note that the empty set is a valid assignment for a set variable: the empty word may satisfy monadic second order variables even if they start with an existential set quantifier.

Büchi's sequential calculus (see Section 1.3.1.2) is thus extended to include monadic second-order formulas. We denote by $\mathsf{MSO}(<)$ (resp. $\mathsf{MSO}(S)$) the fragment of monadic second-order logic, where formulas do not use the symbol S (resp. $<$). Of course, $\mathsf{FO}(<)$ and $\mathsf{FO}(S)$ are subsets of $\mathsf{MSO}(<)$ and $\mathsf{MSO}(S)$, respectively.

Example 1.14. Inspecting the following $\mathsf{MSO}(<)$ sentence,

$$\varphi = \exists X \quad \big[\forall x\ (Xx \leftrightarrow ((\forall y\ \neg(x < y)) \vee (\forall y\ \neg(y < x))))$$
$$\wedge\ \forall x\ (Xx \to R_a x)\ \wedge\ \exists x\ Xx\big].$$

one can see that the elements of X must be the first and last positions of the word in which we interpret φ, so $L(\varphi) = aA^* \cap A^*a$. This language can also be described by a first order sentence, see Example 1.13, that is: this formula is equivalent to a first-order formula.

Example 1.15. We now consider the more complex formula

$$\varphi = \exists X \quad \big((\forall x\ \forall y\ ((x < y) \wedge (\forall z\ \neg((x < z) \wedge (z < y)))) \to (Xx \leftrightarrow \neg Xy))$$
$$\wedge\ (\forall x\ (\forall y\ \neg(y < x)) \to Xx)$$
$$\wedge\ (\forall x\ (\forall y\ \neg(x < y)) \to \neg Xx)\big).$$

The formula φ states that there exists a set X of positions in the word, such that a position is in X if and only if the next position is not in X (so X has every other position), and the first position is in X, and the last position is not in X. Thus $L(\varphi)$ is the set of words of even length. It is an easy consequence of the results of Section 1.7 that this language cannot be described by a first-order formula.

The successor relation can be expressed in $\mathsf{FO}(<)$: $S(x, y)$ is logically equivalent to the following formula:

$$(x < y)\ \wedge\ \forall z\ ((x < z) \to ((y = z) \vee (y < z))).$$

In a weak converse, the order relation $<$ can be expressed in $\mathsf{MSO}(S)$: the formula $x < y$ is equivalent to:

$$\exists X\ \big(Xy \wedge \neg Xx\ \wedge [\forall z\ \forall t\ ((Xz \wedge S(z, t)) \to Xt)]\big).$$

It follows that $\mathsf{MSO}(<)$ and $\mathsf{MSO}(S)$ have the same expressive power.

Proposition 1.5. *A language can be defined by a sentence in $\mathsf{MSO}(S)$, if and only if it can be defined by a sentence in $\mathsf{MSO}(<)$.*

However, the order relation $<$ cannot be expressed in $\mathsf{FO}(S)$. This is a non-trivial result; for a proof, see [21].

Proposition 1.6. *If a language can be defined by a sentence in $\mathsf{FO}(S)$, then it can be defined by a sentence in $\mathsf{FO}(<)$. The converse does not hold.*

1.4. The Kleene-Büchi theorem

In this section, we prove the following theorem, a combination of the classical Kleene and Büchi theorems.

Theorem 1.1. *Let L be a language in A^*. The following conditions are equivalent:*

(1) L is defined by a sentence in $\mathsf{MSO}(<)$;
(2) L is accepted by an automaton;
(3) L is extended rational;
(4) L is rational.

1.4.1. *From automata to monadic second-order formulas*

Let $\mathcal{A} = (Q, i, \delta, F)$ be a deterministic automaton. The idea is to associate with each state $q \in Q$ a second order variable X_q, to encode the set of positions in which a given path visits state q. What we need to express about the sets X_q is the following:

- the sets X_q form a partition of the set of all positions (at each point in time, the automaton must be in one and exactly one state);
- if a path visits state q at time x, state q' at time $x+1$ and if the letter in position $x+1$ is an a, then $\delta(q,a) = q'$;

This analysis leads to the following formula. For convenience, let Q be the set $\{q_0, q_1, \ldots, q_n\}$, with initial state $i = q_0$. We also use the shorthand min and max to designate the first and last positions: this is acceptable as these positions can be expressed by $\mathsf{FO}(S)$-formulas. For instance, $R_a\,\mathrm{min}$ stands for $\forall x\,(\forall y\ \neg S(y,x) \to R_a x)$; and $X\,\mathrm{max}$ stands for $\forall x\,(\forall y\ \neg S(x,y) \to X x)$.

$$\begin{aligned}
&\exists X_{q_0}\ \exists X_{q_1}\ \cdots\ \exists X_{q_n} \\
&\qquad \Bigg(\bigwedge_{q \neq q'} \neg\exists x\ (X_q x \wedge X_{q'} x) \qquad \wedge \qquad \forall x \bigvee_q X_q x \\
&\ \wedge\ \ \forall x\ \forall y\ \Big[S(x,y) \to \bigvee_{q \in Q,\ a \in A} \big(X_q x \wedge R_a y \wedge X_{\delta(q,a)} y \big) \Big] \\
&\ \wedge\ \ \bigwedge_{a \in A} \big(R_a\,\mathrm{min} \to X_{\delta(q_0,a)}\,\mathrm{min} \big)\ \wedge\ \Big(\bigvee_{q \in F} X_q\,\mathrm{max} \Big) \Bigg).
\end{aligned}$$

This sentence is actually verified by the empty word, so the language it defines coincides with $L(\mathcal{A})$ on A^+. If $q_0 \in F$, it accurately defines $L(\mathcal{A})$. But if $q_0 \notin F$, we must consider the conjunction of this sentence with $\exists x$ **true**.

This is a sentence in $\mathsf{MSO}(S, <)$ but as we know, it is logically equivalent to one in $\mathsf{MSO}(<)$. Note that it is in fact an existential monadic second order sentence, that is, the second-order quantifications are all existential.

1.4.2. *From formulas to extended rational expressions*

The proof that an $\mathsf{MSO}(<)$-definable language can be described by an extended rational expression, is more complex. The reasoning is by induction on the recursive definition of formulas. Instead of associating a language only with sentences (formulas without free variables), we will associate languages with all formulas but these languages will be over larger alphabets, which allow us to encode valuations.

1.4.2.1. *The auxiliary alphabets $B_{p,q}$*

Let $p, q \geq 0$ and let $B_{p,q} = A \times \{0,1\}^p \times \{0,1\}^q$. A word over the alphabet $B_{p,q}$ can be identified with a sequence $(u_0, u_1, \ldots, u_p, u_{p+1}, \ldots, u_{p+q})$ where $u_0 \in A^*$, $u_1, \ldots, u_p, u_{p+1}, \ldots, u_{p+q} \in \{0,1\}^*$ and all the u_i have the same length.

Let $K_{p,q}$ consist of the empty word and the words in $B_{p,q}^+$ such that each of the components $u_1, \ldots, u_p$ contains exactly one occurrence of 1. Thus each of these components really designates *one* position in the word u_0, and each of the components $u_{p+1}, \ldots, u_{p+q}$ designates a set of positions in u_0.

Example 1.16. If $A = \{a, b\}$, the following is a word in $K_{2,1}$:

$$\begin{array}{ll} u_0 & a\ b\ a\ a\ b\ a\ b \\ \\ u_1 & 0\ 0\ 0\ 0\ 1\ 0\ 0 \\ u_2 & 0\ 0\ 1\ 0\ 0\ 0\ 0 \\ \\ u_3 & 0\ 1\ 1\ 0\ 0\ 1\ 1 \end{array}$$

Its components u_1 and u_2 designate positions 4 and 2, respectively, and its component u_3 designates the set $\{1, 2, 5, 6\}$.

The languages $K_{p,q}$ are extended rational. Indeed, for $1 \leq i \leq p$, let C_i be the set of elements $(b_0, b_1, \ldots, b_{p+q}) \in B_{p,q}$ such that $b_i = 1$. Then $K_{p,q}$ is the set of words in $B_{p,q}^*$ which contain at most one letter in each C_i:

$$K_{p,q} = \{\varepsilon\} \cup \bigcap_{1 \leq i \leq p} (B_{p,q} \setminus C_i)^* C_i (B_{p,q} \setminus C_i)^* = B_{p,q}^* \setminus \bigcup_{1 \leq i \leq p} B_{p,q}^* C_i B_{p,q}^* C_i B_{p,q}^*.$$

1.4.2.2. *The language associated with a formula*

Let now $\varphi(x_1, \ldots, x_r, X_1, \ldots, X_s)$ be a formula in which the free first order (resp. set) variables are $x_1, \ldots, x_r$ (resp. $X_1, \ldots, X_s$), with $r \leq p$ and $s \leq q$.

We interpret

- R_a as $R_a = \{i \in \mathsf{Dom}(u) \mid u_0(i) = a\}$;
- x_i as the unique position of 1 in u_i (if $u_i \neq \varepsilon$);
- X_j as the set of positions of 1 in u_{p+j}.

Note that if $p = q = 0$, then φ is a sentence and this is the usual notion of interpretation.

More formally, let $(u_0, u_1, \ldots, u_{p+q})$ be a non-empty word in $K_{p,q}$. Let n_i be the position of the unique 1 in the word u_i and let N_j be the set of the positions of the 1's in the word u_{p+j}. We say that $u = (u_0, u_1, \ldots, u_{p+q}) \in K_{p,q}$ satisfies φ if u_0, ν satisfy φ where ν is the valuation defined by

$$\nu(x_i) = n_i \text{ for } 1 \leq i \leq r \quad \text{and} \quad \nu(X_j) = N_j \text{ for } 1 \leq j \leq s.$$

We also say that the empty word (in $K_{p,q}$) satisfies φ if $\varepsilon \models \varphi$. We let $L_{p,q}(\varphi) = \{u \in K_{p,q} \mid u \text{ satisfies } \varphi\}$. Thus each *formula* φ defines a subset of $K_{p,q}$, and hence a language in $B^*_{p,q}$.

Example 1.17. Let $\varphi = \exists x\ (x < y \wedge R_a y)$. Then $FV(\varphi) = \{y\}$. And $L_{1,0}(\varphi)$ is the set of pairs of words (u_0, u_1) such that $u_0 \in A^*$, $u_1 \in \{0,1\}^*$, u_0 and u_1 have the same length, u_1 has a single 1, which is not the first position, and u_0 has an a in that position.

Let $\varphi = \forall x\ ((Xx \wedge x < y \wedge R_b y) \to R_a x)$. Then $L_{1,1}(\varphi)$ is the set of triples of words (u_0, u_1, u_2) with $u_0 \in A^*$, $u_1, u_2 \in \{0,1\}^*$, all three words have the same length, and either this length is zero, or u_1 has a single 1 such that:

> Let n be the position in u_1 which has a 1. If u_0 has a b in position n, then u_0 has an a in each position before n in which u_2 has a 1. If u_0 does not have a b in position n, then there is no constraint.

1.4.2.3. *The MSO(<)-definable languages are extended rational*

We first consider the languages associated with an atomic formula. Let $1 \leq i, j \leq p + q$ and let $a \in A$. Let

$$\begin{aligned} C_{j,a} &= \{b \in B_{p,q} \mid b_j = 1 \text{ and } b_0 = a\}, \\ C_{i,j} &= \{b \in B_{p,q} \mid b_i = b_j = 1\}, \\ \text{and } C_i &= \{b \in B_{p,q} \mid b_i = 1\}. \end{aligned}$$

Then we have

$$\begin{aligned} L_{p,q}(R_a x_i) &= K_{p,q} \cap B^*_{p,q} C_{i,a} B^*_{p,q} \\ L_{p,q}(x_i = x_j) &= K_{p,q} \cap B^*_{p,q} C_{i,j} B^*_{p,q} \\ L_{p,q}(x_i < x_j) &= K_{p,q} \cap B^*_{p,q} C_i B^*_{p,q} C_j B^*_{p,q} \\ L_{p,q}(X_i x_j) &= K_{p,q} \cap B^*_{p,q} C_{i+p,j} B^*_{p,q}. \end{aligned}$$

Thus, the languages defined by the atomic formulas, namely $L_{p,q}(R_a x)$, $L_{p,q}(x = y)$, $L_{p,q}(x < y)$ and $L_{p,q}(Xy)$, are extended rational.

Now let φ and ψ be formulas and let us assume that $L_{p,q}(\varphi)$ and $L_{p,q}(\psi)$ are extended rational. Then we have

$$\begin{aligned} L_{p,q}(\varphi \vee \psi) &= L_{p,q}(\varphi) \cup L_{p,q}(\psi) \\ L_{p,q}(\varphi \wedge \psi) &= L_{p,q}(\varphi) \cap L_{p,q}(\psi) \\ L_{p,q}(\neg\varphi) &= K_{p,q} \setminus L_{p,q}(\varphi), \end{aligned}$$

and hence these three languages are extended rational as well. We still need to handle existential quantification.

Let π_i be the morphism which deletes the i-th component in a word of $B^*_{p,q}$; that is: if $1 \leq i \leq p$, then $\pi_i \colon B^*_{p,q} \to B^*_{p-1,q}$, and if $p < i \leq p+q$, then $\pi_i \colon B^*_{p,q} \to B^*_{p,q-1}$. In either case, we have $\pi_i(b_0, b_1, \ldots, b_{p+q}) = (b_0, b_1, \ldots, b_{i-1}, b_{i+1}, \ldots, b_{p+q})$.

Now, observe that, for any formula $\varphi(x_1, \ldots, x_r, X_1, \ldots, X_s)$, and for $p \geq r$, $q \geq s$, $1 \leq i \leq p$ and $1 \leq j \leq q$ we have

$$L_{p-1,q}(\exists x_i \varphi) = \pi_i(L_{p,q}(\varphi)) \quad \text{and} \quad L_{p,q-1}(\exists X_j \varphi) = \pi_{p+j}(L_{p,q}(\varphi)).$$

This concludes the proof that $L_{p,q}(\varphi)$ is extended rational for any $p \geq r$, $q \geq s$.

In particular, if φ is a sentence in $\mathsf{MSO}(<)$ (that is, φ has no free variables), we may take $p = q = 0$. Then $L_{0,0}(\varphi)$ is extended rational – and we already noted that $L(\varphi) = L_{0,0}(\varphi)$.

1.4.3. *From extended rational expressions to automata*

It is immediately verified that the languages $\emptyset$, $\{\varepsilon\}$, $\{a\}$ $(a \in A)$ are accepted by finite automata. We now need to show that if $K, L \subseteq A^*$ are recognizable and if $\pi \colon A^* \to B^*$ is a morphism, then $\overline{L}$, $K \cup L$, $K \cap L$, KL, K^* and $\pi(L)$ are recognizable.

Proposition 1.7. *If $L \subseteq A^*$ is recognizable, then the complement $\overline{L}$ of L is recognizable as well.*

Proof. Let $\mathcal{A} = (Q, \delta, i, F)$ be a deterministic complete automaton recognizing L. Then $\overline{\mathcal{A}} = (Q, \delta, i, \overline{F})$ recognizes $\overline{L}$ by Proposition 1.3. □

Example 1.18. The deterministic automata in Examples 1.5 and 1.6 confirm that, if $A = \{a, b\}$, then b^*a^* is the complement of A^*abA^*.

Note that the resulting procedure yields a deterministic automaton for $\overline{L}$. It is very efficient if L is given by a deterministic automaton, but may lead to an exponential growth in the number of states if L is given by a non-deterministic automaton.

Proposition 1.8. *If $K, L \subseteq A^*$ are recognizable, then $K \cup L$ and $K \cap L$ are recognizable as well.*

Proof. Let $\mathcal{A} = (Q, T, I, F)$ and $\mathcal{A}' = (Q', T', I', F')$ be automata recognizing L and L', respectively. We assume that the state sets Q and Q' are disjoint. Then it is readily verified that the automaton

$$\mathcal{A} \cup \mathcal{A}' = (Q \cup Q', T \cup T', I \cup I', F \cup F')$$

accepts $L \cup L'$. Thus $L \cup L'$ is recognizable, and hence so is $L \cap L' = \overline{\overline{L} \cup \overline{L'}}$, by Proposition 1.7. □

The construction in the above proof always yields a non-deterministic automaton for $L \cup L'$, even if we start from deterministic automata for L and L'. The product of automata provides an alternative construction which preserves determinism, avoids any exponentiation of the number of states, and works for both the union and the intersection.

Let $\mathcal{A} = (Q, T, I, F)$ and $\mathcal{A}' = (Q', T', I', F')$ be automata recognizing the languages L and L'. Their *cartesian product* is the automaton $\mathcal{A}'' = (Q \times Q', T'', I \times I', F \times F')$ where

$$T'' = \{((p, p'), a, (q, q')) \mid (p, a, q) \in T \text{ and } (p', a, q') \in T'\}.$$

Note that if $\mathcal{A}$ and $\mathcal{A}'$ are deterministic, then $\mathcal{A}''$ is deterministic as well. The main property of $\mathcal{A}''$ is the following: there exists a path $(p, p') \xrightarrow{u} (q, q')$ in $\mathcal{A}''$ if and only if there exist paths $p \xrightarrow{u} q$ and $p' \xrightarrow{u} q'$, in $\mathcal{A}$ and $\mathcal{A}'$ respectively. Therefore $\mathcal{A}''$ recognizes $L \cap L'$.

If we take $(F \times Q') \cup (Q \times F')$ as the set of final states, instead of $F \times F'$, and if the automata $\mathcal{A}$ and $\mathcal{A}'$ are complete, then the product automaton recognizes $L \cup L'$.

In practice, the cartesian product of $\mathcal{A}$ and $\mathcal{A}'$ may not be trim, and one may want to use the procedure in Section 1.2.2.3 to produce more concise automata for $L \cap L'$ and $L \cup L'$.

Remark 1.4. Let us record here an algorithmic consequence of Propositions 1.7 and 1.8: given two automata $\mathcal{A}$ and $\mathcal{B}$, it is decidable whether $L(\mathcal{A}) \subseteq L(\mathcal{B})$ and whether $L(\mathcal{A}) = L(\mathcal{B})$. Indeed, we can compute automata accepting $L(\mathcal{A}) \setminus L(\mathcal{B}) = L(\mathcal{A}) \cap \overline{L(\mathcal{B})}$ and $L(\mathcal{B}) \setminus L(\mathcal{A})$, and decide whether these languages are empty (see Remark 1.1).

Proposition 1.9. *If $L, L' \subseteq A^*$ are recognizable, then LL' and L^* are recognizable as well.*

Sketch of proof. Let $\mathcal{A} = (Q, T, I, F)$ and let $\mathcal{A}' = (Q', T', I', F')$ be automata accepting L and L', respectively, and let us assume that their state sets are disjoint.

It is easily verified that the ε-automaton

$$\big(Q \cup Q', T \cup T' \cup (F \times \{\varepsilon\} \times I'), I, F'\big)$$

accepts LL' (see Section 1.2.2.4). Similarly, if j is a state not in Q, the ε-automaton

$$\big(Q \cup \{j\}, T \cup (F \times \{\varepsilon\} \times I), I \cup \{j\}, F \cup \{j\}\big)$$

accepts L^*. □

Proposition 1.10. *If $L \subseteq A^*$ is recognizable and $\varphi\colon A^* \to B^*$ is a morphism, then $\varphi(L)$ is recognizable as well.*

Sketch of proof. Let $\mathcal{A} = (Q, T, I, F)$ be an automaton recognizing L. We let $\mathcal{A}'$ be the ε-automaton $\mathcal{A}' = (Q \cup Q', T', I, F)$, where the set T' consists of

- the transitions of the form (p, ε, q) such that $(p, a, q) \in T$ for some letter a with $\varphi(a) = \varepsilon$,
- the transitions occurring in the paths of the form

$$p \xrightarrow{b_1} q'_1 \xrightarrow{b_2} \cdots \; q'_{k-1} \xrightarrow{b_k} q$$

such that $(p, a, q) \in T$, $\varphi(a) = b_1 \cdots b_k \neq \varepsilon$ and $q'_1, \ldots, q'_{k-1}$ are new states that we adjoin for each such triple (p, a, q).

The set Q' contains all the new states that occur in the latter paths. It is elementary to verify that $\mathcal{A}'$ recognizes $\varphi(L)$. □

So far, we have shown that a language is recognizable, if and only if it is defined by a sentence in $\mathsf{MSO}(<)$, if and only if it is extended rational.

Remark 1.5. Note that the proofs of this logical equivalence are constructive, in the sense that given a sentence φ in $\mathsf{MSO}(<)$, we can construct an automaton $\mathcal{A}$ such that $L(\varphi) = L(\mathcal{A})$. It follows that $\mathsf{MSO}(<)$ is decidable: given an MSO sentence φ, we can decide whether φ always holds. Indeed, this is the case if and only if $L(\neg\varphi) = \emptyset$, which can be tested as discussed in Remark 1.1.

1.4.4. *From automata to rational expressions*

To complete the proof of the Kleene-Büchi theorem, it suffices to prove that every recognizable language is rational. For this, we use the *McNaughton-Yamada construction*.

Let $\mathcal{A} = (Q, T, I, F)$ be an automaton. For each pair of states $p, q \in Q$ and for each subset $P \subseteq Q$, let $L_{p,q}(P)$ be the set of all words $u \in A^*$ which label a path from state p to state q, such that the states visited internally by that path are all in P:

$$L_{p,q}(P) = \{a_1 a_2 \ldots a_n \in A^* \mid \text{there exists a path in } \mathcal{A} \quad p \xrightarrow{a_1} q_1 \xrightarrow{a_2} \ldots q_{n-1} \xrightarrow{a_n} q \text{ with } q_1, \ldots, q_{n-1} \in P\}.$$

Recall that, by convention, there always exists an empty path, labeled by the empty word, from any state q to itself. So $\varepsilon \in L_{p,q}(P)$ if and only if $p = q$.

We show by induction on the cardinality of P that each language $L_{p,q}(P)$ is rational. This will prove that $L(\mathcal{A})$ is rational, since $L(\mathcal{A}) = \bigcup_{i\in I,\ f\in F} L_{i,f}(Q)$.

If $P = \emptyset$, then $L_{p,q}(\emptyset) = \{a \in A \mid (p, a, q) \in T\}$ if $p \neq q$, and $L_{q,q}(\emptyset) = \{a \in A \mid (q, a, q) \in T\} \cup \{\varepsilon\}$. Thus $L_{p,q}(\emptyset)$ is always finite, and hence rational.

Now let $n > 0$ and let us assume that, for any $p, q \in Q$ and $P \subseteq Q$ containing at most $n - 1$ states, the language $L_{p,q}(P)$ is rational. Let now $P \subseteq Q$ be a subset with n elements and let $r \in P$. Considering the first and the last visit to state r of a path from p to q, we find that

$$L_{p,q}(P) = L_{p,q}(P \setminus \{r\}) \ \cup \ L_{p,r}(P \setminus \{r\})L_{r,r}(P \setminus \{r\})^* L_{r,q}(P \setminus \{r\}).$$

Since $P \setminus \{r\}$ has cardinality $n - 1$, it follows from the induction hypothesis that $L_{p,q}(P)$ is rational.

This concludes the proof of the Kleene-Büchi theorem.

1.4.5. *Closure properties*

Rational languages enjoy many additional closure properties.

Proposition 1.11. *Let $\varphi\colon A^* \to B^*$ be a morphism and let $L \subseteq B^*$. If L is rational, then $\varphi^{-1}(L)$ is rational as well.*

Sketch of proof. Let $\mathcal{A} = (Q, T, I, F)$ be an automaton over B, recognizing L, and let $\mathcal{A}' = (Q, T', I, F)$ be the automaton over A where

$$T' = \{(p, a, q) \mid p \xrightarrow{\varphi(a)} q \text{ is a path in } \mathcal{A}\}.$$

It is readily verified that $\mathcal{A}'$ recognizes $\varphi^{-1}(L)$. □

Let $u \in A^*$ and $L \subseteq A^*$. The *left* and *right quotients* of L by u are defined as follows:

$$u^{-1}L = \{v \in A^* \mid uv \in L\};$$
$$Lu^{-1} = \{v \in A^* \mid vu \in L\}.$$

These notions are generalized to languages: if K and L are languages, the *left* and *right quotients* of L by K are defined as follows:

$$K^{-1}L = \{v \in A^* \mid \exists u \in K \text{ such that } uv \in L\} = \bigcup_{u\in K} u^{-1}L,$$
$$LK^{-1} = \{v \in A^* \mid \exists u \in K \text{ such that } vu \in L\} = \bigcup_{u\in K} Lu^{-1}.$$

Proposition 1.12. *If $L \subseteq A^*$ is rational and $K \subseteq A^*$ is any language (possibly not rational), then $K^{-1}L$ and LK^{-1} are rational as well.*

Sketch of proof. If $\mathcal{A} = (Q, T, I, F)$ is an automaton recognizing L. Let I' be the set of states of $\mathcal{A}$ which are accessible from an initial state of $\mathcal{A}$ following a path labeled by a word of K,

$$I' = \{q \in Q \mid \exists i \in I, \exists u \in K \text{ such that } i \xrightarrow{u} q\}.$$

Then one shows that $\mathcal{A}' = (Q, T, I', F)$ recognizes $K^{-1}L$. The proof for LK^{-1} is similar. □

Remark 1.6. The proof of Proposition 1.12 is not effective: we may not be able to construct the set of states I' associated with K. However, if K is rational too, then I' is effectively constructible.

Recall that a word u is a *prefix* of the word v if there exists a word $v' \in A^*$ such that $v = uv'$ (that is: v "starts" with u). Similarly, u is a *suffix* of v if there exists a word $v' \in A^*$ such that $v = v'u$. Finally u is a *factor* of v if there exist words $v', v'' \in A^*$ such that $v = v'uv''$.

If L is a language, we let $\mathrm{Pref}(L)$ (resp. $\mathrm{Suff}(L)$, $\mathrm{Fact}(L)$) be the set of all prefixes (resp. suffixes, factors) of the words in L.

Proposition 1.13. *If $L \subseteq A^*$ is rational, then $\mathrm{Pref}(L)$, $\mathrm{Suff}(L)$ and $\mathrm{Fact}(L)$ are rational as well.*

Proof. The result follows from Proposition 1.12, since $\mathrm{Pref}(L) = L(A^*)^{-1}$, $\mathrm{Suff}(L) = (A^*)^{-1}L$ and $\mathrm{Fact}(L) = (A^*)^{-1}L(A^*)^{-1}$. □

We leave it to the reader to verify that the following operations also preserve rationality.

The *mirror image* of a word $u = a_1 \dots a_n \in A^*$ is the word $\tilde{u} = a_n \dots a_1$. The corresponding language operation is given by $\tilde{L} = \{\tilde{u} \mid u \in L\}$ for each $L \subseteq A^*$.

A word $u = a_1 \dots a_n \in A^*$ is a *subword* of a word $v \in A^*$ if there exist words $u_0, \dots, u_n \in A^*$ such that $v = u_0 a_1 u_1 \dots a_n u_n$. If $L \subseteq A^*$, we let $\mathrm{SW}(L)$ be the set of all subwords of the words of L.

The *shuffle* of the words u and v is the set

$$u \sqcup\!\sqcup v = \{w \in A^* \mid \exists u_1, v_1, \dots, u_n, v_n \in A^* \text{ such that } u = u_1 \cdots u_n,\ v = v_1 \cdots v_n \text{ and } w = u_1 v_1 \cdots u_n v_n\}.$$

If K and L are languages, we let $K \sqcup\!\sqcup L = \bigcup_{u \in K,\ v \in L} u \sqcup\!\sqcup v$.

Proposition 1.14. *Let $K,\ L \subseteq A^*$ be rational languages. Then $\tilde{L}$, $SW(L)$ and $K \sqcup\!\sqcup L$ are rational as well.*

1.5. Pumping lemmas

The characterizations summarized in the Kleene-Büchi theorem are sufficient most of the time to show that a language is rational. Showing that a language is *not*

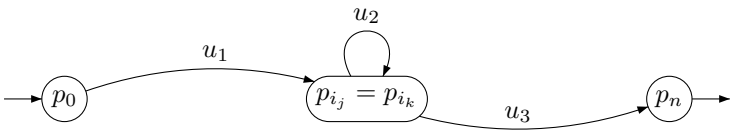

Fig. 1.7. Proof of the pumping lemma

rational is a trickier problem. This short section presents the main tool for that purpose, namely the *pumping lemma*. We actually first present a rather abstract version of this statement, and then its more classical corollaries.

Theorem 1.2. *Let L be a rational language. There exists an integer $N > 0$ with the following property. For each word $w \in L$ and for each sequence of integers $0 \leq i_0 < i_1 < \ldots < i_N \leq |w|$, there exist $0 \leq j < k \leq N$ such that, if $w = u_1u_2u_3$ with $|u_1| = i_j$ and $|u_1u_2| = i_k$, then $u_1u_2^*u_3 \subseteq L$.*

Proof. Let $\mathcal{A}$ be an automaton recognizing L, and let N be the number of states of $\mathcal{A}$. Let $w = a_1a_2 \cdots a_n \in L$ and let

$$p_0 \xrightarrow{a_1} p_1 \xrightarrow{a_2} p_2 \cdots \xrightarrow{a_n} p_n$$

be a successful path in $\mathcal{A}$ labeled w. Let $0 \leq i_0 < i_1 < \cdots < i_N \leq n$ be a sequence of integers. Then two of the states $p_{i_0}, p_{i_1}, \ldots, p_{i_N}$ are equal, that is, there exist $0 \leq j < k \leq N$ such that $p_{i_j} = p_{i_k}$.

Let $u_1 = a_1 \cdots a_{i_j}$, $u_2 = a_{1+i_j} \cdots a_{i_k}$ and $u_3 = a_{1+i_k} \cdots a_n$. Of course, $w = u_1u_2u_3$, $|u_1| = i_j$, $|u_1u_2| = i_k$. The situation is summarized by Figure 1.7: we may iterate or skip the loop labeled u_2 and still retain a successful path, so $u_1u_2^*u_3 \subseteq L$. □

Corollary 1.1. *Let L be a rational language. There exists an integer $N > 0$ such that, for each word $w \in L$ with length $|w| \geq N$, we can factor w in three parts, $w = u_1u_2u_3$, with $u_2 \neq \varepsilon$ and $u_1u_2^*u_3 \subseteq L$.*

Corollary 1.2. *Let L be a rational language. There exists an integer $N > 0$ such that, for each word $w \in L$ with length $|w| \geq N$, we can factor w in three parts, $w = u_1u_2u_3$, with $u_2 \neq \varepsilon$, $|u_1u_2| \leq N$ (resp. $|u_2u_3| \leq N$) and $u_1u_2^*u_3 \subseteq L$.*

Sketch of proof. To prove Corollary 1.2, we apply Theorem 1.2 with $i_j = j$ (resp. $i_j = n-N+j$) for $0 \leq j \leq N$. And to prove Corollary 1.1, we take any sequence. □

Example 1.19. It is a classical application of Corollary 1.1 that $\{a^nb^n \mid n \geq 0\}$ is not rational: for each $N > 0$, the word a^Nb^N cannot be factored as $w = u_1u_2u_3$ with $u_2 \neq \varepsilon$ and $u_1u_2^*u_3 \subseteq \{a^nb^n \mid n \geq 0\}$.

Corollary 1.2 can be used to show that $\{u \in \{a,b\}^* \mid |u|_a = |u|_b\}$ is not rational (take again a^Nb^N); however, this language satisfies the necessary condition for rationality in Corollary 1.1, with $N = 2$.

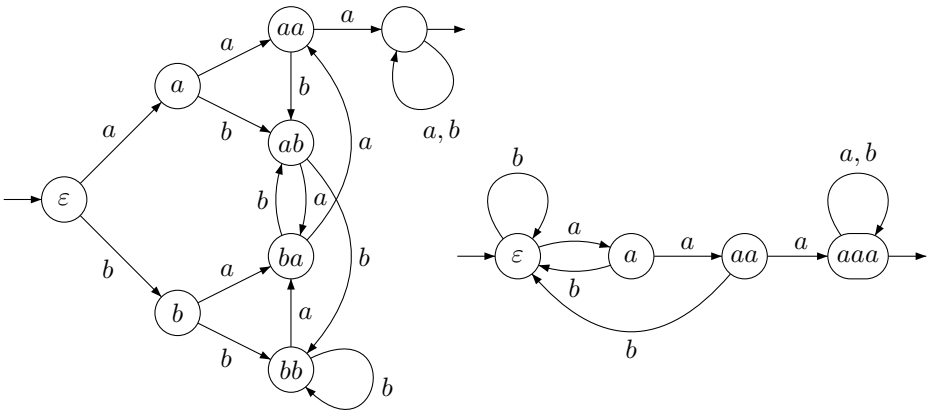

Fig. 1.8. Two different automata for A^*aaaA^*

Consider now the following language over the alphabet $\{a, b, c, d\}$

$$\{(ab)^n(cd)^n \mid n \geq 0\} \cup A^*\{aa, bb, cc, dd, ac\}A^*$$

It satisfies the necessary condition for rationality in Corollary 1.2, but it is not rational, as can be proved using Theorem 1.2.

However, the pumping lemma as stated here may not be enough to prove that a given language is not rational. Let us say that a word *contains a square* if it can be written in the form $uvvw$ with $v \neq \varepsilon$. Then the language

$$\{udv \mid u, v \in \{a, b, c\}^* \text{ and either } u \neq v, \text{ or one of } u \text{ and } v \text{ contains a square}\}$$

satisfies the necessary condition for rationality in Theorem 1.2 (for $N = 4$). Yet it is not rational (the proof of that fact uses the existence of arbitrarily long words on the alphabet $\{a, b, c\}$ containing no square).

Ehrenfeucht, Parikh, Rozenberg gave a necessary and sufficient condition for rationality in the same style as the pumping lemma (see *e.g.* [18, Theorem I.3.3]).

1.6. Minimal automaton and syntactic monoid

Consider the two automata in Figure 1.8. Both are complete and deterministic, and both recognize the set of words over $A = \{a, b\}$ that contain some occurrence of the word aaa as a factor — that is, the language A^*aaaA^*. The two automata were designed using different intuitions about how to go about this task: In the first instance, the underlying algorithm is "keep track of the last two letters read from the input", as indicated by the state labels, while in the second automaton the algorithm is, "keep track of the length of the longest suffix of a's in the input". Thus the second automaton achieves the same result with a smaller number of states. It is easy to see that the second example is also optimal—no complete deterministic automaton recognizing this language can have a smaller number of states.

In this section we will see that for every rational language L there is a unique minimal complete deterministic automaton accepting L. We will also describe an efficient algorithm that takes as input an arbitrary complete deterministic automaton $\mathcal{A}$, and produces as output the minimal automaton for $L(\mathcal{A})$.

1.6.1. *Myhill-Nerode equivalence and the minimal automaton*

One way to see that there is something inefficient about the first automaton in the example above is to observe its behavior on the two input words $u = bab$ and $v = abb$. These words lead from the initial state to two different states. However, for purposes of recognizing words in L, there is no point in distinguishing between u and v, for no matter what the subsequent input w is, the result will be the same: either uw and vw are both in L or both outside of L.

To formalize this notion of inputs that are indistinguishable with respect to L, we make the following definitions: If $u, v \in A^*$ we define $u \equiv_L v$ if and only if $u^{-1}L = v^{-1}L$ (see Section 1.4.5). Obviously, $\equiv_L$ is an equivalence relation on A^*. We also note that if $u \equiv_L v$, and $w \in A^*$, then $uw \equiv_L vw$, since $(uw)^{-1}L = w^{-1}(u^{-1}L)$. An equivalence relation with this multiplicative property is said to be a *right congruence*. Further, L itself is a union of $\equiv_L$-classes, since $w \in L$ if and only if $\varepsilon \in w^{-1}L$.

We can accordingly define a complete deterministic automaton $\mathcal{A}_{\min}(L)$ by making the states these classes of equivalent words: We set $\mathcal{A}_{\min}(L) = (Q_L, \delta_L, i_L, F_L)$, where $Q_L = A^*/\equiv_L$, $i_L = [\varepsilon]_{\equiv_L}$, and F_L and $\delta_L \colon Q_L \times A \to Q_L$ are defined by

$$F_L = \{[v]_{\equiv_L} \mid v \in L\} \qquad \text{and} \qquad \delta([v]_{\equiv_L}, a) = [va]_{\equiv_L}.$$

We need to show that this is well-defined, since a state will in general have many different representations of the form $[v]_{\equiv_L}$. But well-definedness is an immediate consequence of our observation that $\equiv_L$ is a right congruence. We have the following result.

Theorem 1.3. *Let $L \subseteq A^*$.*

(1) $\mathcal{A}_{\min}(L)$ accepts L.
(2) L is rational if and only if $\equiv_L$ has finite index.

Proof. It follows at once by induction on $|w|$ that for all $w \in A^*$,

$$\delta_L([\varepsilon]_{\equiv_L}, w) = [w]_{\equiv_L}.$$

Since, as observed above, L itself is a union of $\equiv_L$-classes, it follows that w is accepted if and only if $w \in L$. This proves the first claim.

To prove the second claim in the theorem, note that if $\equiv_L$ has finite index, then $\mathcal{A}_{\min}$ is a finite automaton, and therefore by *(1)*, L is rational. Conversely, if L is rational, then it is accepted by some complete deterministic automaton (Q, δ, i, F)

with Q finite. Now suppose $u, v \in A^*$ and $\delta(i,u) = \delta(i,v)$. Then if $w \in A^*$ and $uw \in L$, we have

$$\delta(i, vw) = \delta(\delta(i,v), w) = \delta(\delta(i,u), w) = \delta(i, uw) \in F,$$

so $vw \in L$. Similarly, $vw \in L$ implies $uw \in L$, so $u \equiv_L v$. Thus the number of classes of $\equiv_L$ cannot be more than $|Q|$, so $\equiv_L$ has finite index. □

The proof of Theorem 1.3 shows that $\mathcal{A}_{\min}(L)$ has the least number of states among the complete deterministic automata accepting L. The automaton $\mathcal{A}_{\min}(L)$ is called the *minimal automaton* of L. We now give another, more algebraic justification for this terminology.

1.6.2. *Uniqueness and minimality of* $\mathcal{A}_{\min}(L)$

Let $\mathcal{A} = (Q, \delta, i, F)$ be a complete deterministic automaton over A, and let $L = L(\mathcal{A})$. We say that $p, q \in Q$ are *equivalent* states, and write $p \equiv q$, if

$$\{v \in A^* \mid \delta(p,v) \in F\} = \{v \in A^* \mid \delta(q,v) \in F\}.$$

Intuitively, this means that for purposes of recognizing words in L, p and q do the same job, and we might as well merge them into a single state.

We now repeat, in a somewhat different form, an observation made in the proof of Theorem 1.3: If $\delta(i,u) \equiv \delta(i,v)$, then

$$uw \in L \Longleftrightarrow \delta(\delta(i,u), w) \in L \Longleftrightarrow \delta(\delta(i,v), w) \in L \Longleftrightarrow vw \in L,$$

so that $u \equiv_L v$. In particular, if $\delta(i,u) = \delta(i,v)$, then $u \equiv_L v$, so we have a well-defined mapping $\delta(i,w) \mapsto [w]_{\equiv_L}$, from the set of *accessible* states of $\mathcal{A}$ onto the states of $\mathcal{A}_{\min}(L)$. Note that this mapping sends the initial state $i = \delta(i,\varepsilon)$ to $[\varepsilon]_{\equiv_L}$, final states of $\mathcal{A}$ to final states of $\mathcal{A}_{\min}(L)$, and respects the next-state function. We summarize these observations as follows.

Theorem 1.4. *Let $\mathcal{A} = (Q, \delta, i, F)$ be a complete deterministic automaton over A, and let $L = L(\mathcal{A})$. Then there is a map f from the set of accessible states in Q onto Q_L such that*

- *for all $a \in A$ and accessible $q \in Q$, $f(\delta(q,a)) = \delta_L(f(q), a)$,*
- *$f(i) = i_L$,*
- *$f(F) = F_L$.*

Moveover, $f(p) = f(q)$ if and only if $p \equiv q$.

In particular, if $\mathcal{A}$ has the same number of states as $\mathcal{A}_{\min}(L)$, then since f is onto, the two automata are isomorphic by Theorem 1.4.

1.6.3. *An algorithm for computing the minimal automaton*

Theorem 1.4 says that in principle we can compute the minimal automaton of a rational language L starting from any complete deterministic automaton (Q, δ, i, F) accepting L, first by removing the inaccessible states and then merging equivalent states. We have already seen how to compute the accessible states. How do we determine if two states are equivalent? If p, q are inequivalent states then there is a word $v \in A^*$ that distinguishes between these states in the sense that $\delta(p, v) \in F$ and $\delta(q, v) \notin F$, or vice-versa. It follows from a simple pumping argument that if such a distinguishing word exists, then it can be chosen to have length no more than $|Q|^2$. Thus we can effectively determine whether two states are equivalent by calculating $\delta(p, v)$ and $\delta(q, v)$ for all words up to this length.

Of course, this is a terrible algorithm, since there are $|A|^{|Q|^2}$ different words to check! In practice, we can proceed as follows: Let $m \geq 0$. We say $p \equiv_m q$ if for all $v \in A^*$ of length no more than m, $\delta(p, v) \in F$ if and only if $\delta(q, v) \in F$. This is clearly an equivalence relation on A^*, and $\equiv_{m+1}$ refines $\equiv_m$ for all m. The following lemma improves the $|Q|^2$ bound on the length of distinguishing words.

Lemma 1.1. *Let $p, q \in Q$. Then $p \equiv q$ if and only if $p \equiv_m q$ for $m = |Q| - 2$.*

Proof. First suppose that for some m, the equivalence relations $\equiv_m$ and $\equiv_{m+1}$ coincide. We claim that $\equiv_m$ and $\equiv$ coincide. To see this, suppose that p and q are inequivalent, and that w is a word of minimal length distinguishing them. If $|w| > m$, then we can write $w = uv$, where $|v| = m + 1$, so that $p' = \delta(p, u)$ and $q' = \delta(q, u)$ are inequivalent modulo $\equiv_{m+1}$. But this means that they are also inequivalent modulo $\equiv_m$, and thus distinguished by a word v' of length no more than m, and thus p and q are distinguished by the word uv' of length strictly less than that of w, a contradiction. Thus the minimal distinguishing word has length no more than m, so that $\equiv_m$ coincides with $\equiv$.

Now if $\equiv_{m+1}$ does not coincide with $\equiv_m$, then $\equiv_{m+1}$ has a larger number of classes. Since the number of classes can never exceed $|Q|$, and since $\equiv_0$ has two classes, the sequence $\{\equiv_m\}_{m \geq 0}$ will stabilize by the time m reaches $|Q| - 2$. □

Lemma 1.1 leads to the following practical algorithm for minimization. We begin with a list of all the pairs $\{p, q\}$ of distinct accessible states, and mark the pair if $p \in F$ and $q \notin F$, or vice-versa. In each phase of the algorithm, we visit each unmarked pair $\{p, q\}$ and each $a \in A$, we compute $\{p', q'\} = \{\delta(p, a), \delta(q, a)\}$, and we mark $\{p, q\}$ if $\{p', q'\}$ is marked. An easy induction shows that if a pair $\{p, q\}$ is distinguished by a word of length m, then it will be marked by the m^{th} phase of the algorithm. Thus after no more than $|Q| - 2$ phases, the algorithm will not mark any new pairs, with the result that the algorithm terminates, and the unmarked pairs are exactly the pairs of equivalent states.

Example 1.20. Consider the first automaton in Figure 1.9. Initially we mark the pairs $\{i, j\}$, where $i \in \{1, 2, 3\}$ and $j \in \{4, 5, 6\}$. On the next pass, the pairs

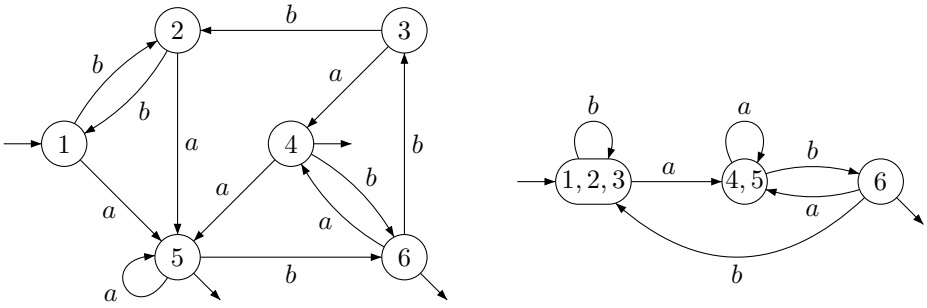

Fig. 1.9. The minimization algorithm

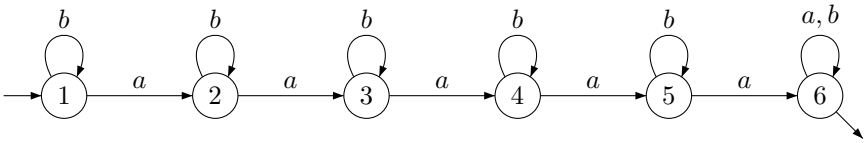

Fig. 1.10. A minimal automaton

$\{4,6\}$ and $\{5,6\}$ are marked since applying b to these pairs gives the marked pair $\{3,6\}$. No further pairs are marked on the next pass, so the algorithm terminates. Since the pairs $\{1,2\}$ and $\{2,3\}$ are unmarked, $\{1,2,3\}$ is an equivalence class, and since $\{4,5\}$ is unmarked, it forms a second class. The remaining class is $\{6\}$. The resulting minimal automaton is pictured on the right-hand side of Figure 1.9.

Example 1.21. We now apply the algorithm to the automaton in Figure 1.10. Initially, the pairs $\{i,6\}$ with $i<6$ are marked. On the next pass the pairs $\{i,5\}$ with $i<5$ are marked, *etc.,* until on the fifth pass the pair $\{1,2\}$ is marked. The result is that every pair of distinct states is marked: the automaton is already minimal.

The pair-marking implementation of the algorithm just illustrated is suitable for small examples worked by hand. In the worst case, shown in the last example, we check $\mathcal{O}(|Q|^2)$ unmarked pairs on each pass, and make $\mathcal{O}(|Q|)$ passes, with $|A|$ consultations of the state-transition table for each pair we inspect. Thus, the overall time complexity of the algorithm is $\mathcal{O}(|A|\cdot|Q|^3)$. More astute bookkeeping, in which we partition equivalence classes at each step, rather than marking pairs of inequivalent states, leads to a $\mathcal{O}(|A|\cdot|Q|^2)$ algorithm (Moore [12]). This can be further improved to $\mathcal{O}(|A|\cdot|Q|\cdot\log|Q|)$ (Hopcroft [6]).

1.6.4. *The transition monoid of an automaton*

Let $\mathcal{A}=(Q,\delta,i,F)$ be a complete deterministic automaton over an alphabet A. Let $w\in A^*$. We study the maps

$$f_w^{\mathcal{A}}\colon q\longmapsto\delta(q,w)$$

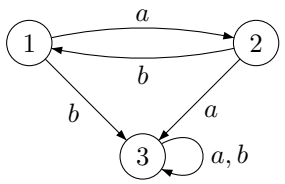

Fig. 1.11. The automaton $\mathcal{A}_1$, with no indication of initial or terminal states

from Q into itself. We will write the image of a state q under $f_w^{\mathcal{A}}$ as $qf_w^{\mathcal{A}}$ rather than the more traditional $f_w^{\mathcal{A}}(q)$. We then have, for $v, w \in A^*$,

$$f_{vw}^{\mathcal{A}} = f_v^{\mathcal{A}} f_w^{\mathcal{A}},$$

where the product in the right-hand side of the equation is left-to-right composition of functions — that is, $q(f_v^{\mathcal{A}} f_w^{\mathcal{A}}) = (qf_v^{\mathcal{A}})f_w^{\mathcal{A}}$.

We will henceforth drop the superscript $\mathcal{A}$, except in situations where several different automata are involved. Observe that f_ε is the identity map on Q. Thus the set of maps

$$M(\mathcal{A}) = \{f_w \mid w \in A^*\}$$

forms an algebraic structure with an associative product and an identity element (usually denoted 1). Such a structure is called a *monoid*, and we call $M(\mathcal{A})$ the *transition monoid* of $\mathcal{A}$. Observe that if Q is finite, then $M(\mathcal{A})$ is finite, and that the structure of $M(\mathcal{A})$ depends only on the next-state function δ, and not at all on the initial or final states.

A^* is, of course, itself a monoid, with concatenation of words as the operation and the empty word ε as the identity. The map

$$\varphi\colon w \longmapsto f_w$$

is consequently a monoid *morphism* from A^* into $M(\mathcal{A})$; that is, it satisfies

$$\varphi(w_1 w_2) = \varphi(w_1)\varphi(w_2)$$

for all w_1, w_2 in A^*, and it maps the identity element of A^* to the identity element of $M(\mathcal{A})$.

Example 1.22. In the diagrams in this example and in Examples 1.23 and 1.24, we indicate only the transitions between states, since, as we have observed, the initial and final states do not enter into the computation of the transition monoid of an automaton.

First, consider the automaton $\mathcal{A}_1$ in Figure 1.11. We will write an element f_w of $M(\mathcal{A}_1)$ as a vector $f_w = (\,1f_w\ 2f_w\ 3f_w\,)$. We can then begin enumerating the

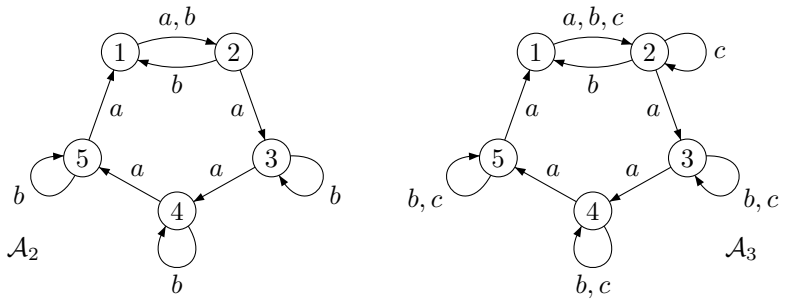

Fig. 1.12. The automata $\mathcal{A}_2$ and $\mathcal{A}_3$

elements of $M(\mathcal{A}_1)$:

$$f_\varepsilon = (\,1\;2\;3\,)$$
$$f_a = (\,2\;3\;3\,) \qquad f_b = (\,3\;1\;3\,)$$
$$f_{aa} = (\,3\;3\;3\,) \qquad f_{ab} = (\,1\;3\;3\,) \qquad f_{ba} = (\,3\;2\;3\,) \qquad f_{bb} = (\,3\;3\;3\,)$$

We could continue enumerating like this, but instead we note that $f_{aba} = f_a$, $f_{bab} = f_b$, and for all other $w \in A^*$ of length 3, $f_w = (\,3\;3\;3\,)$. Thus the inventory above is the entire transition monoid, since any transition induced by a word of length greater than 2 is equal to one induced by a shorter word. Thus $M(\mathcal{A})$ has 6 elements $1, \alpha = f_a, \beta = f_b, \alpha\beta, \beta\alpha$, and 0. The multiplication is then determined by the laws $\alpha\alpha = \beta\beta = 0$, $\alpha = \alpha\beta\alpha$, and $\beta = \beta\alpha\beta$. The complete multiplication table is shown below:

$\cdot$	1	α	β	$\alpha\beta$	$\beta\alpha$	0
1	1	α	β	$\alpha\beta$	$\beta\alpha$	0
α	α	0	$\alpha\beta$	0	α	0
β	β	$\beta\alpha$	0	β	0	0
$\alpha\beta$	$\alpha\beta$	α	0	$\alpha\beta$	0	0
$\beta\alpha$	$\beta\alpha$	0	β	0	$\beta\alpha$	0
0	0	0	0	0	0	0

This example illustrates an important general point: There is an effective procedure for computing the multiplication table of the transition monoid of a complete deterministic finite automaton. We enumerate the maps f_w until we find that all words of some length induce the same maps as shorter words.

Example 1.23. Consider the automaton $\mathcal{A}_2$ in Figure 1.12. The transition monoid is generated by the two permutations f_a and f_b, both of which are permutations of the set of states: f_a cycles the five states and f_b transposes a pair of adjacent states. It is well known from elementary group theory that we can obtain all transpositions t of adjacent elements by repeated conjugation with the cycle (the map $t \mapsto f_a^4\, t\, f_a$), and that all permutations of the states can be obtained by composing transpositions of pairs of adjacent elements. So $M(\mathcal{A})$ consists of all the permutations of

$\{1, 2, 3, 4, 5\}$, and is consequently the symmetric group of degree 5, with $5! = 120$ elements. Of course, we can do likewise with any finite set of states.

Example 1.24. Now consider the effect of adding a third input letter to the preceding example, obtaining the automaton $\mathcal{A}_3$ in Figure 1.12. It is not hard to show that *every* map from $\{1, 2, 3, 4, 5\}$ into itself can be obtained by repeatedly composing f_c with permutations. Thus $M(\mathcal{A}_3)$ is the *full transformation monoid* on 5 states, which has $5^5 = 3125$ elements. We can similarly generate a transition monoid with n^n elements using an n-state automaton.

1.6.5. *The syntactic monoid*

Now let $L \subseteq A^*$, and consider the transition monoid of the minimal automaton $\mathcal{A}_{\min}(L) = (Q_L, \delta_L, i_L, F_L)$. Let $u, v \in A^*$. When are the two elements f_u, f_v of this monoid the same? If they are different, then there is some state q such that $qf_u \neq qf_v$. Since the automaton is minimal, there is a word $y \in A^*$ distinguishing these two states, so that $qf_uf_y \in F_L$ and $qf_vf_y \notin F_L$, or vice-versa. Since every state is accessible, there is also a word x such that $q = if_x$, so that either $xuy \in L$ and $xvy \notin L$, or vice-versa. Conversely, if such a pair of words x, y exists, then f_u and f_v cannot be equal. We thus have:

Theorem 1.5. *Let $L \subseteq A^*$, and let $u, v \in A^*$. Let $\mathcal{A} = \mathcal{A}_{\min}(L)$. Then $f_u^{\mathcal{A}} = f_v^{\mathcal{A}}$ if and only if for all $x, y \in A^*$*

$$xuy \in L \iff xvy \in L.$$

If the conditions in this theorem are satisfied, then we write $u \cong_L v$. The equivalence relation $\cong_L$ is called the *syntactic congruence* of L, and the transition monoid of $\mathcal{A}_{\min}(L)$ is called the *syntactic monoid* of L. We denote the syntactic monoid of L by $M(L)$. In algebraic terms, $M(L) = M(\mathcal{A}_{\min}(L)) = A^*/\cong_L$, that is $M(L)$ is the quotient monoid of A^* by the syntactic congruence. The morphism mapping each $w \in A^*$ to its $\cong_L$-class is called the *syntactic morphism* of L, and is denoted μ_L.

The syntactic congruence is a two-sided congruence on A^*; that is, if $u \cong_L v$ and $u' \cong_L v'$, then $uu' \cong_L vv'$. Compare this to the Myhill-Nerode congruence $\equiv_L$, which, as we noted, is a right congruence. The equivalence $\cong_L$ refines $\equiv_L$.

Transition monoids, and, in particular, the syntactic monoid, allow us to place many questions about the behavior of automata in a purely algebraic setting. For instance, we have the following algebraic characterization of rationality: Let M be a monoid and $\varphi\colon A^* \to M$ a morphism. We say that φ *recognizes* $L \subseteq A^*$ if and only if there is a subset X of A^* such that $L = \varphi^{-1}(X)$. We also say in this situation that M recognizes L.

Theorem 1.6. *Let $L \subseteq A^*$. The following are equivalent:*

(1) L is rational.

(2) $M(L)$ is finite.
(3) L is recognized by a finite monoid.

Proof. To show *(1)* implies *(2)*, note that if L is rational, then $\mathcal{A}_{\min}(L)$ has a finite set of states, and thus its transition monoid, $M(L)$, is finite. For *(2)* implies *(3)*, if $u \in L$ and $u \cong_L v$, then $v = \varepsilon\, v\, \varepsilon$ is also in L. Thus L is a union of equivalence classes of $\cong_L$, so that $L = \mu_L^{-1}(X)$, where $X = \{f_w \in M(\mathcal{A}_{\min}(L)) \mid w \in L\}$. Finally, to show *(3)* implies *(1)*, suppose $\varphi\colon A^* \to M$, where M is finite, and that $L = \varphi^{-1}(X)$. Then L is accepted by the complete deterministic automaton $\mathcal{A}(M) = (M, \delta, 1, X)$, where for $m \in M$ and $a \in A$,

$$\delta(m, a) = m\, \varphi(a).$$

Since M is finite, L is rational. □

Remark 1.7. Observe that if M is a finite monoid and $\mathcal{A}(M)$ is the automaton defined in the proof of Theorem 1.6, then the transition monoid of $\mathcal{A}(M)$ is M itself.

The syntactic monoid plays the same role in this algebraic view of rational languages that the minimal automaton plays in the automaton-theoretic view. Here we make this precise: We say that a monoid N *divides* a monoid M, and write $N \prec M$, if there is a submonoid M' of M and a surjective morphism $\varphi\colon M' \to N$. It is easy to see that $\prec$ is a transitive relation on monoids.

Theorem 1.7. *Let $L \subseteq A^*$. Then M recognizes L if and only if $M(L) \prec M$.*

Proof. First suppose M recognizes L, so that there is a morphism $\varphi\colon A^* \to M$ such that $L = \varphi^{-1}(X)$ for some $X \subseteq M$. We claim that if $\varphi(u) = \varphi(v)$, then $u \cong_L v$. To see this, suppose that $xuy \in L$ for some $x, y \in A^*$. Then $\varphi(xuy) \in X$, and since $\varphi(u) = \varphi(v)$, we have $\varphi(xvy) \in X$, so that $xvy \in L$. By the same argument, if $xvy \in L$ then $xuy \in L$, so that $u \cong_L v$.

Now let $M' = \varphi(A^*)$. We define a map $\psi\colon M' \to M(L)$ by $\psi(\varphi(u)) = \mu_L(u)$. By the remark just made, ψ is well-defined, since the value of ψ only depends on $\varphi(u)$ and not on u. Moreover ψ is clearly a morphism, and it is surjective because μ_L is, so $M(L) \prec M$.

Conversely, suppose $M(L) \prec M$, so that there is a morphism ψ from a submonoid M' of M onto $M(L)$. For $a \in A$, we set $\varphi(a)$ to be any $m \in M'$ for which $\psi(m) = \mu_L(a)$. This can be extended to a unique morphism $\varphi\colon A^* \to M$ such that $\mu_L = \psi \circ \varphi$. Let $X = \varphi(L)$. If $\varphi(u) \in X$ then $\varphi(u) = \varphi(v)$ for some $v \in L$, and thus $\mu_L(u) = \mu_L(v)$, so $u \cong_L v$. Since, as noted in the proof of Theorem 1.6, L is a union of $\cong_L$-classes, this implies $u \in L$, so that $L = \varphi^{-1}(X)$, and thus M recognizes L.□

Example 1.25. Consider the transition monoid of the automaton $\mathcal{A}$ in Figure 1.13. We can fairly easily determine the elements of this monoid without doing an exhaustive tabulation: First, if a word w has even length, then it maps $\{1, 3\}$ into $\{1, 3\}$, and $\{2, 4\}$ into $\{2, 4\}$, while if w has odd length, then it interchanges these

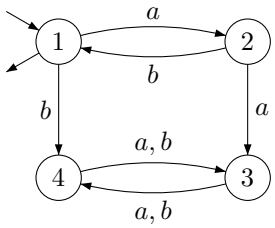

Fig. 1.13. An automaton, whose transition monoid contains a non-trivial group

two sets. Second, if w contains aa or bb as a factor, then the image of f_w is contained in $\{3,4\}$. Finally, if the letters of w alternate, then f_w maps either 1 or 2 to $\{1,2\}$, but not both, depending on whether the first letter of w is a or b. We thus get these elements:

$$1 = f_\varepsilon = (1\ 2\ 3\ 4) \qquad \gamma = f_a = (2\ 3\ 4\ 3) \qquad \delta = f_b = (4\ 1\ 4\ 3)$$
$$\gamma\delta = f_{ab} = (1\ 4\ 3\ 4) \qquad \delta\gamma = f_{ba} = (3\ 2\ 3\ 4)$$
$$\gamma^2 = f_{aa} = (3\ 4\ 3\ 4) \qquad \gamma^3 = f_{aaa} = (4\ 3\ 4\ 3)$$

Observe that $\{\gamma^2, \gamma^3\}$ forms a group, permuting the states 3 and 4. This automaton accepts the language $(ab)^*$. In algebraic terms, the morphism $\varphi\colon w \mapsto f_w$ from A^* into $M(\mathcal{A})$ recognizes this language with $(ab)^* = \varphi^{-1}(X)$, where

$$X = \{f \in M(\mathcal{A}) \mid f \text{ maps state 1 to itself }\}.$$

The states 3 and 4 are equivalent, and the minimal automaton of L is obtained by merging these states: it is the automaton examined in Example 1.22 (with 1 as initial and final state), where we computed its transition monoid, namely $M(L)$. According to Theorem 1.7, $M(L) \prec M(\mathcal{A})$, and, indeed, the map sending 1 to 1, $\gamma, \delta, \gamma\delta, \delta\gamma$ to $\alpha, \beta, \alpha\beta, \beta\alpha$, respectively, and γ^2 and γ^3 both to 0, is a morphism from $M(\mathcal{A})$ onto $M(L)$.

Example 1.26. Let us take the automaton of Example 1.24 and specify 1 as both the initial state and the unique accepting state. With these choices, the automaton is the minimal automaton of the language it accepts, since every state is accessible and no two distinct states are equivalent. This shows that the syntactic monoid of a language accepted by an n-state automaton can have as many as n^n elements.

Example 1.27. Not every finite monoid is the syntactic monoid of a rational language. Consider, for instance, the monoid $M = \{1, \alpha, \beta, \gamma\}$ with multiplication $m_1 m_2 = m_2$ for $m_2 \neq 1$. Suppose A is a finite alphabet and $\varphi\colon A^* \to M$ is a morphism. Let $X \subseteq M$. We partition A into three subsets, B, C, and D,

$$B = \{a \in A \mid \varphi(a) = 1\}$$
$$C = \{a \in A \mid \varphi(a) \in X \setminus \{1\}\}$$
$$D = A \setminus (B \cup C)$$

Then $\varphi^{-1}(X) = B^* \cup A^*CB^*$ if $1 \in X$ and $\varphi^{-1}(X) = A^*CB^*$ otherwise. (Observe that B or C might be empty.) But then $L = \varphi^{-1}(X)$ is recognized by the submonoid $\{1, \alpha, \beta\}$, using the morphism that maps B to 1, C to α and D to β. Thus every language recognized by M is recognized by a strictly smaller monoid, so by Theorem 1.7, M cannot be the syntactic monoid of any language.

1.7. First-order definable languages

This section is devoted to proving one of the earliest and most important applications of the syntactic monoid: the characterization of the languages definable in $\mathsf{FO}(<)$.

A finite monoid M can contain a nontrivial group, as for example the group $\{\gamma^2, \gamma^3\}$ in the monoid $M(\mathcal{A})$ of Example 1.25. If there is no nontrivial group in M, we say that M is *aperiodic*.

Lemma 1.2. *Let M be a finite monoid. Then the following are equivalent:*

(1) M is aperiodic.
(2) There is an integer $n > 0$ such that $m^n = m^{n+1}$ for all $m \in M$.

Proof. Suppose M is aperiodic. Let $m \in M$, and consider the sequence $1, m, m^2, \ldots$ Since M is finite, if we take $n = |M|$, we have $m^r = m^n$ for some $r < n$. Take the largest such r, and consider the set $G = \{m^k \mid r \leq k < n\}$. Observe that for all $g \in G$, $gG = Gg = G$, since

$$m^{r+t}m^s = m^{r+[(t+s) \bmod (n-r)]}$$

for all $s, t \geq 0$. This implies that G is a group, so that $|G| = 1$, and thus $r = n - 1$ and $m^r = m^{r+1}$. Conversely, if M is not aperiodic, then M contains a nontrivial group G, and an element $g \in G$ different from the identity element e of G. Then $g^k = e$ for some $k > 1$, so that $g^n \neq g^{n+1}$ for all $n \geq 0$. □

Note that the proof shows that we can choose n in condition *(2)* of Lemma 1.2 to be $|M| - 1$.

We say that a language $L \subseteq A^*$ is *star-free* if it can be defined by an extended rational expression without the use of the $*$ operation or morphic images. The Schützenberger-McNaughton-Papert Theorem offers the following characterization.

Theorem 1.8. *Let $L \subseteq A^*$ be a rational language. Then the following are equivalent.*

(1) L is star-free.
(2) L is definable by a sentence of $\mathsf{FO}(<)$.
(3) L is recognized by an aperiodic finite monoid.
(4) $M(L)$ is aperiodic.

Before we turn to the proof of this theorem, we give an important corollary, and an example.

Corollary 1.3. *It is decidable whether a rational language (given by a rational expression or an accepting automaton) is definable by a sentence of first-order logic.*

Proof. As we have seen, we can compute $\mathcal{A}_{\min}(L)$ from any automaton or expression for L, and thence compute the multiplication table of $M = M(L)$. We can then test for all $m \in M$ whether $m^{|M|-1} = m^{|M|}$, and thus, by Lemma 1.2 determine whether $M(L)$ is aperiodic. By Theorem 1.8, this decides whether L is first-order definable. □

In fact, the proof of Theorem 1.8 will show that if $M(L)$ is aperiodic, then we can effectively construct both a star-free expression and a first-order sentence for L from an automaton that recognizes L.

Example 1.28. Let $L = (ab)^*$. We computed $M(L)$ in Example 1.22. We have $\alpha^2 = \beta^2 = 0 = \alpha^3 = \beta^3$, and $(\alpha\beta)^2 = \alpha\beta$, $(\beta\alpha)^2 = \beta\alpha$, so by Lemma 1.2, $M(L)$ is aperiodic. Theorem 1.8 says that L is definable by a star-free extended rational expression, and also by a sentence of $\mathsf{FO}(<)$. Let us exhibit such expressions.

First, note that membership of a word w in L is equivalent to saying that w contains no occurrence of either aa or bb as a factor, and that the first letter of w (if there is one) is a, and the last letter is b. We thus have

$$L = \{\varepsilon\} \cup (aA^* \cap A^*b \cap \overline{A^*(aa \cup bb)A^*}).$$

witnessing the fact that L is star-free (note that A^* is star-free, since $A^* = \overline{\emptyset}$).

To obtain a first-order sentence defining L, we use the same characterization of words in L. We say there is no occurrence of aa as a factor using the following sentence:

$$\neg\exists x \exists y (R_a x \wedge R_a y \wedge S(x, y)).$$

This uses the successor predicate S, but as we noted earlier, S can be expressed in $\mathsf{FO}(<)$. We can likewise write a sentence saying that there is no occurrence of bb. An FO-sentence stating that the first letter of a word is a was given in Example 1.13. A similar sentence can be formed to say that the last letter is b. Note that all these sentences are satisfied by the empty word as well, so that the conjunction of the four sentences defines the language $(ab)^*$.

This language is also recognized by the first monoid that we exhibited in Example 1.25, which is not aperiodic. This in no way contradicts Theorem 1.8, which only says that *some* aperiodic monoid recognizes L.

Remark 1.8. The decision procedure outlined in the proof of Corollary 1.3 may take exponential time in the size of an automaton accepting L, since it involves computing the syntactic monoid of L (see Example 1.24). While this procedure

may be improved, this decision problem is intrinsically difficult. In fact, it is known to be PSPACE-complete (Cho and Huynh [2]).

We now turn to the proof of Theorem 1.8. We will show (4) $\Leftrightarrow$ (3) $\Rightarrow$ (1) $\Rightarrow$ (2) $\Rightarrow$ (4). By Theorem 1.7, every language is recognized by its syntactic monoid. Also every divisor of an aperiodic monoid is aperiodic, since the property $m^n = m^{n+1}$ for all elements m in a monoid is inherited by morphic images and submonoids. Thus (3) and (4) are equivalent.

The most difficult part of the proof is (3) $\Rightarrow$ (1). To prove this, we suppose $L \subseteq A^*$ is recognized by a finite aperiodic monoid. This is equivalent to L being accepted by a complete deterministic automaton $\mathcal{A} = (Q, \delta, i, F)$ whose transition monoid is aperiodic (see the proof of Theorem 1.6 and Remark 1.7). We will show that for all $q, q' \in Q$, the set $L^{\mathcal{A}}_{q,q'} = \{w \mid qf^{\mathcal{A}}_w = q'\}$ is a star-free language. Since L is a finite union of such languages, L is star-free.

The proof is by induction on the pair $(|Q|, |A|)$: the induction hypothesis is that the claim holds for all automata with a strictly smaller state set, or with the same size state set and a strictly smaller input alphabet. In the case $|Q| = 1$, L is either A^* or $\emptyset$, which are star-free. In the case $|A| = 1$, so that $A = \{a\}$, aperiodicity implies that L is a finite union of singleton sets $\{a^k\}$, possibly together with the language $a^r a^*$, where $r = |Q| - 1$, which is also star-free, since $a^* = \overline{\emptyset}$.

We thus assume both $|Q| > 1$ and $|A| > 1$. First suppose that for every $a \in A$, $Qf^{\mathcal{A}}_a = Q$, so that $f^{\mathcal{A}}_a$ is a permutation of Q. Aperiodicity implies $(f^{\mathcal{A}}_a)^r = (f^{\mathcal{A}}_a)^{r+1}$ for some r, and thus $f^{\mathcal{A}}_a$ is the identity map on Q. Consequently $f^{\mathcal{A}}_w$ is the identity map for all $w \in A^*$, and thus the claim holds trivially. We can therefore assume that there is some $a \in A$ such that

$$Qf^{\mathcal{A}}_a = Q' \subsetneq Q.$$

We now define two new automata $\mathcal{B}$ and $\mathcal{C}$. Automaton $\mathcal{B}$ has state set Q and next-state function $\delta\big|_{Q\times B}$, where $B = A \setminus \{a\}$. We need not define initial and final states for $\mathcal{B}$, because we are only interested in the state transitions $f^{\mathcal{B}}_w$. Automaton $\mathcal{C}$ has state set Q', input alphabet

$$C = \{(f^{\mathcal{B}}_w, a) \mid w \in B^*, a \in A\},$$

and next-state function

$$\delta' \colon (q, (f^{\mathcal{B}}_w, a)) \longmapsto q \cdot f^{\mathcal{A}}_{wa}.$$

This makes sense, because $Qf^{\mathcal{A}}_a = Q'$ and because $f^{\mathcal{B}}_w = f^{\mathcal{B}}_{w'}$ implies that $f^{\mathcal{A}}_{wa} = f^{\mathcal{A}}_{w'a}$. The inductive hypothesis applies to both $\mathcal{B}$ and $\mathcal{C}$. (The transition monoids of these automata inherit the aperiodicity of $\mathcal{A}$, because every transition in them is the restriction of a transition in $\mathcal{A}$.)

A word in $L^{\mathcal{A}}_{q,q'}$ can contain either no occurrences of a, a single occurrence of a, or two or more occurrences of a. We can accordingly write $L^{\mathcal{A}}_{q,q'}$ as a finite union

of sets of the form

$$L^{\mathcal{B}}_{q,q'}, L^{\mathcal{B}}_{q,p} a L^{\mathcal{B}}_{p',q'}, L^{\mathcal{B}}_{q,p} a T_{p',q''} L^{\mathcal{B}}_{q'',q'},$$

where $p \in Q$, $p' = p \cdot f^{\mathcal{A}}_a \in Q'$, and $T_{p,q''} = L^{\mathcal{A}}_{p,q''} \cap A^* a$.

By the inductive hypothesis all the sets of the form $L^{\mathcal{B}}_{s,t}$ are star-free, so it remains to show that $T_{p',q''}$ is a star-free language. We can factor any $w \in A^* a$ uniquely as

$$w = v_1 a \cdots v_k a,$$

where $v_1, \ldots, v_k \in B^*$. Let us associate to w the word

$$w_C = c_1 \cdots c_k \in C^*,$$

where $c_j = (f^{\mathcal{B}}_{v_j}, a) \in C$. By the inductive hypothesis, the language $L^{\mathcal{C}}_{p',q''}$ is star-free. So we need to show that if $R \subseteq C^*$ is star-free, then $\Psi(R) = \{w \in A^* a \mid w_C \in R\}$ is also star-free, since $T_{p'q''} = \Psi(L^{\mathcal{C}}_{p',q''})$. It is thus enough to show

(i) If $c \in C$, then $\Psi(\{c\})$ is star-free.
(ii) If $\Psi(R)$ is star-free, then $\Psi(C^* \setminus R)$ is star-free.
(iii) If $\Psi(R_1), \Psi(R_2)$ are star-free, then $\Psi(R_1 \cup R_2)$ is star-free.
(iv) If $\Psi(R_1), \Psi(R_2)$ are star-free, then $\Psi(R_1 R_2)$ is star-free.

For *(i)*, note that $\Psi(\{c\}) = Sa$, where $S = \{v \in B^* \mid c = (f^{\mathcal{B}}_v, a)\}$. Since S is a boolean combination of languages of the form $L^{\mathcal{B}}_{p,p'}$, Sa is star-free. For the other assertions, we clearly have $\Psi(C^* \setminus R) = A^* a \cap (A^* \setminus \Psi(R))$, $\Psi(R_1 \cup R_2) = \Psi(R_1) \cup \Psi(R_2)$, and $\Psi(R_1 R_2) = \Psi(R_1)\Psi(R_2)$. This completes the proof that *(3)* $\Rightarrow$ *(1)*.

To prove (1) $\Rightarrow$ (2), we need to show that every star-free language is first-order definable. Since the singleton sets $\{a\}$ for $a \in A$ are clearly first-order definable, and since the boolean operations are part of first-order logic, this reduces to showing that if $L_1, L_2 \subseteq A^*$ are first-order definable, then so is $L_1 L_2$. To do this, we introduce the notion of *relativizing* a first-order sentence. Let φ be a sentence of $\mathsf{FO}(<)$ and x a variable symbol that does not occur in φ. We define a formula $\varphi_{<x}$ with one free variable with the following property: Let ν be an interpretation mapping x to $i \in \mathsf{Dom}(u)$, and let v be the prefix v of u with domain $\{0, \ldots, i-1\}$. Then $u, \nu \models \varphi_{<x}$ if and only if $v \models \varphi$. To construct $\varphi_{<x}$, we simply work from the outermost quantifier of φ inward, replacing each quantified subformula $\exists y\, \alpha$ by $\exists y\, ((y < x) \wedge \alpha)$. We define $\varphi_{>x}$ and $\varphi_{\leq x}$ analogously.

Now suppose φ, ψ are first-order sentences defining L_1 and L_2, respectively. Let x be a variable symbol that does not occur in φ or ψ. We have $L_1 L_2$ defined by the sentence

$$\begin{aligned} &\exists x\ (\varphi_{\leq x} \wedge \psi_{>x}) && \text{if } \varepsilon \notin L_1, \\ &\exists x\ (\varphi_{\leq x} \wedge \psi_{>x}) \vee \psi && \text{if } \varepsilon \in L_1. \end{aligned}$$

To prove (2) $\Rightarrow$ (4), we need to show that the syntactic monoid of every first-order definable language in A^* is aperiodic. We will proceed as in Section 1.4.2, and treat a first-order formula with free variables contained in $\{x_1, \ldots, x_p\}$ as defining a language over the extended alphabet $B_p = A \times \{0,1\}^p$. We will show by induction on the quantifier depth that every first-order definable language $L \subseteq B_p^*$ in this extended sense has an aperiodic syntactic monoid. More precisely, we will show that for each such L there exists an integer $q > 0$ such that for all $v \in B_p^*$, $v^q \cong_L v^{q+1}$. By Lemma 1.2, this implies aperiodicity.

First suppose L is defined by one of the atomic formulas $x_1 < x_2$ or $R_a x_1$. Let $u, v, w \in B_p^*$. If v has a letter with a 1 in one of its last p components, then neither uv^2w nor uv^3w can be in L, since only one letter of a word in L can have a 1 in a given component. If v has no such letter, then membership of uvw in L is witnessed by the relative positions and values of letters in u and w, so that $uvw \in L$ if and only if $uv^2w \in L$. Thus in all cases, we have $uv^2w \in L$ if and only if $uv^3w \in L$, so that $v^2 \cong_L v^3$.

Now suppose the claim is true for $L_1, L_2 \subseteq B_p^*$ defined by formulas φ_1, φ_2, and suppose L is defined by $\varphi_1 \vee \varphi_2$. We have, by assumption, $v^q \cong_{L_1} v^{q+1}$, and $v^q \cong_{L_2} v^{q+1}$, for some $q > 0$. (The exponents for these two languages are, *a priori*, different, but we can then choose q to be the maximum of the two exponents.) Now $\varphi_1 \vee \varphi_2$ defines $L_1 \cup L_2$, and we have directly $uv^qw \in L_1 \cup L_2$ if and only if $uv^{q+1}v \in L_1 \cup L_2$.

Care must be taken with the negation operator, since it does not exactly correspond to the boolean complement. We can assume that the exponent q for L_1 is at least 2. Let L_1' be the language defined by $\neg\varphi_1$. Suppose $uv^qw \in L_1'$. Then $uv^qw \notin L_1$, and thus $uv^{q+1}w \notin L_1$. Further v cannot contain a 1 in the last p components of any of its positions, so $uv^{q+1}w$ has exactly one occurrence of 1 in each of the last p positions, and thus is in L_1'. The same argument shows that if $uv^{q+1}w \in L_1'$, then so is uv^qw. Thus $v^q \cong_{L_1'} v^{q+1}$.

So now let $K \subseteq B_{p-1}^*$ be the language defined by $\exists x_p \varphi_1$. Let $v \in B_{p-1}^*$. We will show $v^{2q+1} \cong_K v^{2q+2}$. Suppose $uv^{2q+1}w \in K$. Let us extend each letter in this word by adding a p^{th} component with 0. We will still denote the resulting word as $uv^{2q+1}w$. Since K is defined by $\exists x_p \varphi_1$, we can switch the p^{th} component of some letter to obtain a word $z \in B_p^*$ such that $z \in L_1$. Now, wherever the position in which we switched the p^{th} component is located, at least q consecutive occurrences of v will be left intact. We thus find that z can be written in the form xv^qy, for some $x, y \in B_p^*$. (The extreme case is when the position is within the middle occurrence of v, in which case we get two factors of the form v^q.) Thus $xv^{q+1}y \in L_1$. If we now switch the changed 1 back to 0, we find $uv^{2q+2}w \in K$. The identical argument shows $uv^{2q+2}w \in K$ implies $uv^{2q+1}w \in K$. Thus $v^{2q+1} \cong_K v^{2q+2}$, as claimed.

Remark 1.9. Interesting presentations of proofs of all or part of Theorem 1.8 can be found, for instance, in the work of Perrin [15], Straubing [21] and Diekert and Gastin [3].

References

[1] J. Richard Büchi. Weak second-order arithmetic and finite automata. *Z. Math. Logik Grundlagen Math.*, 6:66–92, 1960.
[2] Sung Cho and Dung T. Huynh. Finite automaton aperiodicity is PSPACE-complete. *Theoretical Computer Science*, 88:99–116, 1991.
[3] Volker Diekert and Paul Gastin. First-order definable languages. In J. Flum, E. Grädel, and Th. Wilke, editors, *Logic and Automata: History and Perspectives*, Texts in Logic and Games, pages 261–306. Amsterdam University Press, 2008.
[4] Samuel Eilenberg. *Automata, Languages, and Machines*, volume A. Academic Press, New York and London, 1974.
[5] Samuel Eilenberg. *Automata, Languages, and Machines*, volume B. Academic Press, New York and London, 1976.
[6] John E. Hopcroft. An $n \log n$ algorithm for minimizing the states in a finite automaton. In Z. Kohavi, editor, *The Theory of Machines and Computations*, pages 189–196. Academic Press, 1971.
[7] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.
[8] David A. Huffman. The synthesis of sequential switching circuits. *J. Franklin Institute*, 257:161–190, 275–303, 1954.
[9] Steven C. Kleene. Representation of events in nerve nets and finite automata. In C. E. Shannon and J. McCarthy, editors, *Automata Studies*, number 34 in Annals of Mathematics Studies, pages 3–40. Princeton University Press, 1956.
[10] Harry R. Lewis and Christos H. Papadimitriou. *Elements of the Theory of Computation*. Prentice-Hall, 1981.
[11] Robert McNaughton and Seymour Papert. *Counter-Free Automata*. The MIT Press, Cambridge, Mass., 1971.
[12] Edward F. Moore. Gedanken experiments on sequential machines. In *Automata Studies*, pages 129–153. Princeton University Press, 1956.
[13] John R. Myhill. Finite automata and the representation of events. Technical Report 57-624, Wright Airport Development Command, 1957.
[14] Anil Nerode. Linear automaton transformations. *Proc. AMS*, 9:541–544, 1958.
[15] Dominique Perrin. Finite automata. In *Handbook of Theoretical Computer Science, Vol. B*, pages 1–57. Elsevier, Amsterdam, 1990.
[16] Jean-Eric Pin, editor. *Automata: from Mathematics to Applications*. European Mathematical Society, 2011.
[17] Michael O. Rabin and Dana Scott. Finite automata and their decision problems. *IBM Journal of Research and Development*, 3:114–125, 1959. Reprinted in E. F. Moore, editor, *Sequential Machines: Selected Papers*, Addison-Wesley, 1964.
[18] Jacques Sakarovitch. *Elements of Automata Theory*. Cambridge University Press, Cambridge, 2009. Translated from the 2003 French original by Reuben Thomas.
[19] Marcel Paul Schützenberger. On finite monoids having only trivial subgroups. *Inform. and Comput.*, 8:190–194, 1965.
[20] Michael Sipser. *Introduction to the Theory of Computation, 2nd Edition*. Course Technology, 2006.
[21] Howard Straubing. *Finite Automata, Formal Logic, and Circuit Complexity*. Birkhäuser, Boston, Basel and Berlin, 1994.
[22] Wolfgang Thomas. Languages, automata and logic. In A. Salomaa and G. Rozenberg, editors, *Handbook of Formal Languages*, volume 3, Beyond Words. Springer, Berlin, 1997.

[23] Thomas Wilke. Classifying discrete temporal properties. In Chr. Meinel and S. Tison, editors, *Proc.16th Annual Symposium on Theoretical Aspects of Computer Science (STACS'99), Trier (Germany), 1999*, number 1443 in Lecture Notes in Computer Science, pages 32–46, Heidelberg, 1999. Springer-Verlag. Invited Lecture.

[24] Boris A. Trakhtenbrot. The synthesis of logical nets whose operators are described in terms of monadic predicates. *Doklady AN SSR*, 118:646–649, 1958.

Index