Finite automata

Jean-Éric Pin[1]

[1]IRIF, Université Paris Diderot and CNRS
Case 7014, F-75205 Paris Cedex 13

2010 Mathematics Subject Classification: 68Q45

Key words: Finite automata, deterministic automata, regular languages, recognizable languages, rational languages, Kleene theorem.

Contents

1 Basic algebraic structures

A *semigroup* is a pair consisting of a set S and an associative binary operation on S, usually denoted multiplicatively. In this case, xy is called the *product* of x and y. Sometimes, the additive notation is preferred, and $x + y$ denotes the *sum* of x and y.

A *monoid* is a triple consisting of a set M, an associative binary operation on M and an identity for this operation. This identity is denoted by 1 in the multiplicative notation and by 0 in the additive notation.

A *semiring* consists of a set k, two binary operations on k, denoted additively and multiplicatively, and two elements 0 and 1, satisfying the following conditions:

(1) k is a commutative monoid for addition with identity 0,
(2) k is a monoid for multiplication with identity 1,
(3) Multiplication is distributive over addition: for all $s, t_1, t_2 \in k$, we have $s(t_1 + t_2) = st_1 + st_2$ and $(t_1 + t_2)s = t_1s + t_2s$,
(4) for all $s \in k$, we have $0s = s0 = 0$.

A *ring* is a semiring in which the monoid $(k, +, 0)$ is a group. A semiring is *commutative* if its multiplication is commutative.

The simplest example of a semiring which is not a ring is the *Boolean semiring* $\mathbb{B} = \{0, 1\}$ whose operations are defined by the following tables

$+$	0	1
0	0	1
1	1	1

$\times$	0	1
0	0	0
1	0	1

2 Words, languages and automata

2.1 Words and languages

Let A be a set called an *alphabet*, whose elements are called *letters*. A finite sequence of elements of A is called a *finite word* on A, or just a *word*. We denote by mere juxtaposition

$$a_1 \cdots a_n$$

the sequence $(a_1, \ldots, a_n)$. The set of words is endowed with the operation of *concatenation product* also called *product*, which associates the word $xy = a_1 \cdots a_p b_1 \cdots b_q$

with two words $x = a_1 \cdots a_p$ and $y = b_1 \cdots b_q$. This operation is associative. It has an identity, the *empty word*, denoted by 1 or ε, which is the empty sequence.

We let A^* denote the set of words on A and A^+ the set of nonempty words. The set A^* $[A^+]$, equipped with the concatenation product is thus a monoid with identity 1 [a semigroup]. The set A^* is called the *free monoid* on A and A^+ the *free semigroup* on A.

Let $u = a_1 \cdots a_n$ be a word in A^* and let a be a letter of A. A nonnegative integer i is said to be an *occurrence* of the letter a in u if $a_i = a$. We denote by $|u|_a$ the number of occurrences of a in u. Thus, if $A = \{a, b\}$ and $u = abaab$, one has $|u|_a = 3$ and $|u|_b = 2$. The sum

$$|u| = \sum_{a \in A} |u|_a$$

is the *length* of the word u. Thus $|abaab| = 5$.

A *language* is a set of words. The empty language is denoted by 0 and each singleton $\{u\}$ is simply denoted u. Several operations can be defined on languages:

(1) *Boolean operations*, which comprise *union* (which we often denote by $+$), *intersection* and *complement* (denoted by $L \to L^c$).

(2) *Quotients*: given a language L and a word u of A^*, $u^{-1}L = \{v \mid uv \in L\}$ and $Lu^{-1} = \{v \mid vu \in L\}$.

(3) *Star and Plus*: if L is a language, L^* $[L^+]$ is the submonoid (subsemigroup) of A^* generated by L. Thus $L^* = \{u_1u_2 \cdots u_n \mid n \geqslant 0, u_1, \ldots, u_n \in L\}$ and $L^* = L^+ + 1$.

(4) *Product*: the product of two languages L_1 and L_2 is the language $L_1L_2 = \{u_1u_2 \mid u_1 \in L_1, u_2 \in L_2\}$.

(5) *Morphisms*. Let A and B be two alphabets, and let φ be a function from A into B^*. Then φ extends in a unique way to a monoid morphism from A^* into B^*. If L is a language of A^*, then $\varphi(L) = \{\varphi(u) \mid u \in L\}$ is a language of B^*.

(6) *Inverses of morphisms*. If $\varphi\colon A^* \to B^*$ is a monoid morphism and L is a language of B^*, then $\varphi^{-1}(L) = \{u \in A^* \mid \varphi(u) \in L\}$ is a language of A^*.

The set of *rational* (or *regular*) languages on A^*, denoted by $\mathrm{Rat}(A^*)$, form the smallest set of languages containing the languages 0, 1 and a for each letter $a \in A$, and closed under finite union, product and star. That is, if L and L' are rational languages, then the languages $L + L'$, LL' and L^* are also rational.

For instance, if $A = \{a, b\}$, the language $(a + ab + ba)^*$ is a rational language. The set A^*uA^* of all words containing a given factor u is rational. The set of words of odd length is rational and can be written as $(A^2)^*A$.

We conclude this section by a standard result: rational languages are closed under morphisms. An extension of this result will be given in Proposition 6.1.

Proposition 2.1. *Let $\varphi : A^* \to B^*$ be a morphism. If L is a rational language of A^*, then $\varphi(L)$ is a rational language of B^*.*

2.2 Finite automata and recognizable languages

A finite *automaton* is a 5-tuple $\mathcal{A} = (Q, A, E, I, F)$, where Q is a finite set called the set of *states*, A is an alphabet, E 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 set of *initial states* and the set of *final states*.

It is convenient to represent an automaton by a labelled graph whose vertices are the states of the automaton and the edges represent the transitions. The initial [final] states are pictured by incoming [outgoing] arrows.

Example 2.1. Let $\mathcal{A} = (Q, A, E, I, F)$ where $Q = \{1, 2\}$, $I = \{1, 2\}$, $F = \{2\}$, $A = \{a, b\}$ and $E = \{(1, a, 1), (2, b, 1), (1, a, 2), (2, b, 2)\}$. This automaton is represented in Figure 1.

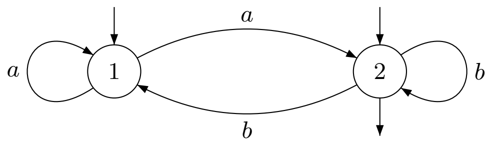


Figure 1. An automaton.

Two transitions (p, a, q) and (p', a', q') are *consecutive* if $q = p'$. A *path* in the automaton $\mathcal{A}$ is a finite sequence of consecutive transitions

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

also denoted by

$$c : q_0 \xrightarrow{a_1} q_1 \cdots q_{n-1} \xrightarrow{a_n} q_n \quad \text{or} \quad q_0 \xrightarrow{a_1 \cdots a_n} q_n.$$

The state q_0 is its *origin*, the state q_n its *end*, the word $a_1 \cdots a_n$ is its *label* and the integer n is its *length*. Is is also convenient to consider that for each state $q \in Q$, there is an empty path $q \xrightarrow{1} q$ from q to q labelled by the empty word.

A path in $\mathcal{A}$ is called *initial* if its origin is an initial state and *final* if its end is a final state. It is *successful* (or *accepting*) if it is initial and final.

A state q is *accessible* if there is an initial path ending in q and it is *coaccessible* if there is a final path starting in q.

Example 2.2. Consider the automaton represented in Figure 1. The path

$$c : 1 \xrightarrow{a} 1 \xrightarrow{a} 2 \xrightarrow{b} 2 \xrightarrow{b} 1 \xrightarrow{a} 2 \xrightarrow{b} 2$$

is successful, since its end is a final state. However the path

$$c : 1 \xrightarrow{a} 1 \xrightarrow{a} 2 \xrightarrow{b} 2 \xrightarrow{b} 1 \xrightarrow{a} 2 \xrightarrow{b} 1$$

has the same label, but is not successful, since its end is 1, a nonfinal state.

A word is *accepted* by the automaton $\mathcal{A}$ if it is the label of at least one successful path (beware that it can be simultaneously the label of a nonsuccessful path). The *language* *recognized* (or *accepted*) by the automaton $\mathcal{A}$ is the set, denoted by $L(\mathcal{A})$, of all the words accepted by $\mathcal{A}$. Two automata are *equivalent* if they recognize the same language.

A language $L \subseteq A^*$ is *recognizable* if it is recognized by a finite automaton, that is, if there is a finite automaton $\mathcal{A}$ such that $L = L(\mathcal{A})$.

Example 2.3. Consider the automaton represented in Figure 2.

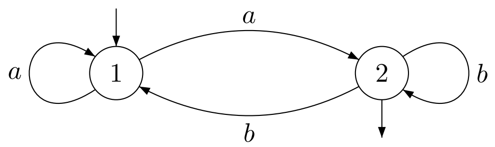


Figure 2. The automaton $\mathcal{A}$.

We let the reader verify that the language accepted by $\mathcal{A}$ is aA^*, the set of all words whose first letter is a.

Example 2.3 is elementary but it already raises some difficulties. In general, deciding whether a given word is accepted or not might be laborious, since a word might be the label of several paths. The notion of deterministic automaton introduced in Section 2.3 permits one to avoid these problems.

A standard property of recognizable languages is known as the *pumping lemma*. Although it is formally true for any recognizable language, it is only interesting for the infinite ones.

Proposition 2.2 (Pumping lemma). *Let L be a recognizable language. Then there is an integer $n > 0$ such that every word u of L of length greater than or equal to n can be factorized as $u = xyz$ with $x, y, z \in A^*$, $|xy| \leqslant n$, $y \neq 1$ and, for all $k \geqslant 0$, $xy^kz \in L$.*

Proof. Let $\mathcal{A} = (Q, A, E, I, F)$ be an n-state automaton recognizing L and let $u = a_1 \cdots a_r$ be a word of L of length $r \geqslant n$. Let $q_0 \xrightarrow{a_1} q_1 \cdots q_{r-1} \xrightarrow{a_r} q_r$ be a successful path labelled by u. As $r \geqslant n$, there are two integers i and j, with $i < j \leqslant n$, such that $q_i = q_j$. Therefore, the word $a_{i+1} \ldots a_j$ is the label of a loop around q_i, represented in Figure 3.

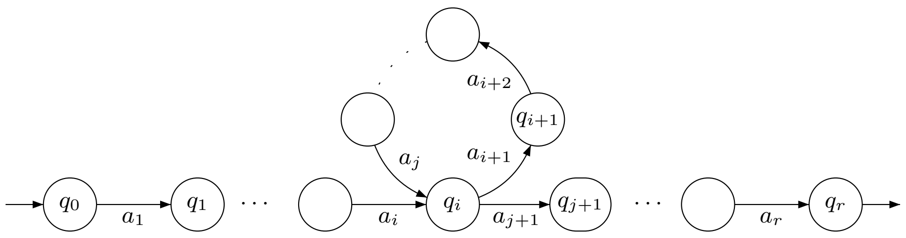


Figure 3. Illustration of the pumping lemma.

Let $x = a_1 \ldots a_i$, $y = a_{i+1} \ldots a_j$ and $z = a_{j+1} \ldots a_r$. Then $|xy| \leqslant n$ and for all $k \geqslant 0$, one gets $xy^kz \in L$, since the word xy^kz is the label of a successful path. $\square$

The pumping lemma permits one to show that a language like $\{a^n b^n \mid n \geqslant 0\}$ is not recognizable. However, it does not characterize the recognizable languages. For instance, if $A = \{a, b, c\}$, the nonrecognizable language $\{(ab)^n c^n \mid n > 0\} \cup A^*bbA^* \cup A^*aaA^*$ satisfies the pumping lemma.

2.3 Deterministic automata

An automaton $\mathcal{A} = (Q, A, E, I, F)$ is *deterministic* if I contains exactly one initial state and if, for every state $q \in Q$ and for every letter $a \in A$, there exists *at most* one state q' such that $q \xrightarrow{a} q'$ is a transition of E. If q_- is the unique initial state, we adopt the notation (Q, A, E, q_-, F) instead of $(Q, A, E, \{q_-\}, F)$.

Example 2.4. The automaton represented in Figure 4 is deterministic.

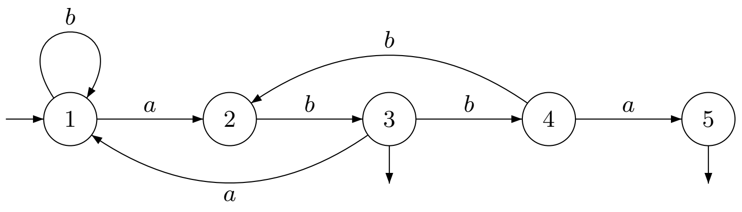


Figure 4. A deterministic automaton.

The following result is one of the cornerstones of automata theory. Its proof is based on the so-called *subset construction*.

Proposition 2.3. *Every finite automaton is equivalent to a deterministic one.*

Proof. Let $\mathcal{A} = (Q, A, E, I, F)$ be an automaton. Consider the deterministic automaton $D(\mathcal{A}) = (\mathcal{P}(Q), A, \cdot, I, \mathcal{F})$ where $\mathcal{F} = \{P \subseteq Q \mid P \cap F \neq \emptyset\}$ and, for each subset P of Q and for each letter $a \in A$,

$$P \cdot a = \{q \in Q \mid \text{there exists } p \in P \text{ such that } (p, a, q) \in E\}.$$

We claim that $D(\mathcal{A})$ is equivalent to $\mathcal{A}$.

If $u = a_1 \cdots a_n$ is accepted by $\mathcal{A}$, there is a successful path

$$c : q_0 \xrightarrow{a_1} q_1 \ \cdots \ q_{n-1} \xrightarrow{a_n} q_n \ .$$

The word u also defines a path

$$I = P_0 \xrightarrow{a_1} P_1 \ \cdots \ P_{n-1} \xrightarrow{a_n} P_n \tag{2.1}$$

in $D(\mathcal{A})$. Let us show by induction on i that, for $0 \leqslant i \leqslant n$, $q_i \in P_i$. Since c is a successful path, one has $q_0 \in I = P_0$. Suppose that $q_{i-1} \in P_{i-1}$. Then since $q_{i-1} \xrightarrow{a_i} q_i$ is a transition, one gets $q_i \in P_{i-1} \cdot a_i = P_i$. For $i = n$, we get $q_n \in P_n$ and since c is a successful path, $q_n \in F$. It follows that P_n meets F and hence $P_n \in \mathcal{F}$. Therefore u is accepted by $D(\mathcal{A})$.

Conversely, let $u = a_1 \cdots a_n$ be a word accepted by $D(\mathcal{A})$ and let (2.1) be the successful path defined by u. Since P_n is a final state, one can choose an element q_n in $P_n \cap F$. We can now select, for $i = n, n-1, \ldots, 1$, an element q_{i-1} of P_{i-1} such that $q_{i-1} \xrightarrow{a_i} q_i$ is a transition of $\mathcal{A}$. Since $q_0 \in I$ and $q_n \in F$, the path $q_0 \xrightarrow{a_1} q_1 \;\cdots\; q_{n-1} \xrightarrow{a_n} q_n$ is successful, and thus u is accepted by $\mathcal{A}$. This proves the claim and the proposition. □

The subset construction converts a nondeterministic n-state automaton into a deterministic automaton with at most 2^n states. One can show that this bound is tight.

2.4 Complete, accessible, coaccessible and trim automata

An automaton $\mathcal{A} = (Q, A, \cdot, q_-, F)$ is *complete* if, for each state $q \in Q$ and for each letter $a \in A$, there is *at least* one state q' such that $q \xrightarrow{a} q'$ is a transition.

Example 2.5. The automaton represented in Figure 5 is neither complete, nor deterministic. It is not deterministic, since the transitions $(1, a, 1)$ and $(1, a, 2)$ have the same label and the same origin. It is not complete, since there is no transition of the form $2 \xrightarrow{a} q$.

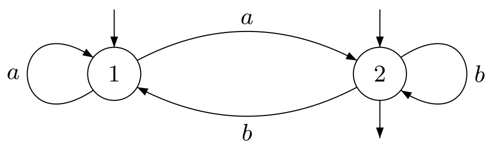


Figure 5. An incomplete, nondeterministic automaton.

On the other hand, the automaton represented in Figure 6 is complete and deterministic.

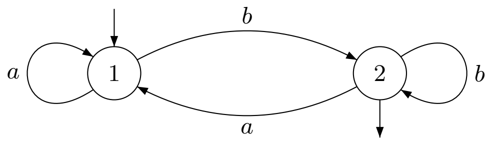


Figure 6. A complete and deterministic automaton.

A finite automaton is *accessible* if all its states are accessible. Similarly, it is *coaccessible* if all its states are coaccessible. Finally, an automaton is *trim* if it is simultaneously accessible and coaccessible. It is not difficult to see that every deterministic automaton is equivalent to a trim one.

Example 2.6. Let $A = \{a, b\}$. Starting from the nondeterministic automaton $\mathcal{A}$ represented in Figure 7, we get the deterministic automaton $D(\mathcal{A})$ drawn in Figure 8. In practice, it suffices to compute the accessible states of $D(\mathcal{A})$, which gives the deterministic automaton shown in Figure 9.

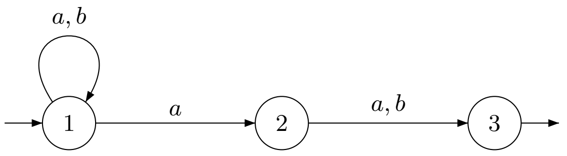


Figure 7. A nondeterministic automaton.

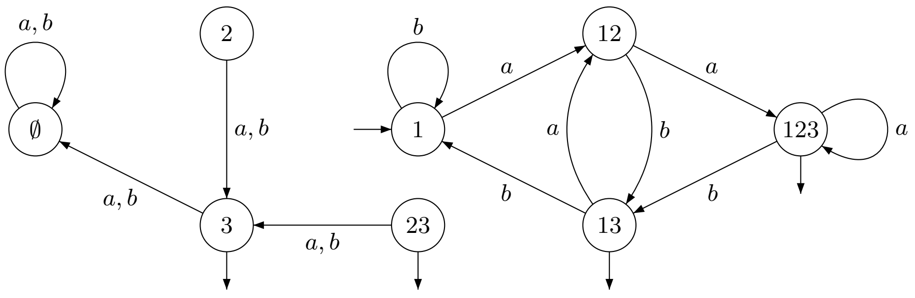


Figure 8. After determinisation...

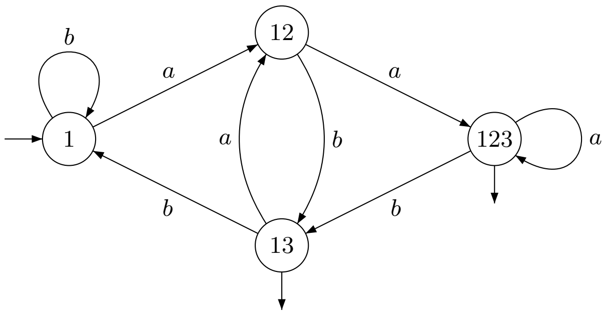


Figure 9. . . . and trimming.

2.5 Standard automata

The construction described in this section might look somewhat artificial, but it will be used in the study of the product and of the star operation.

A deterministic automaton is *standard* if there is no transition ending in the initial state.

Proposition 2.4. *Every deterministic automaton is equivalent to a deterministic standard automaton.*

Proof. Let $\mathcal{A} = (Q, A, E, q_-, F)$ be a deterministic automaton. If $\mathcal{A}$ is not standard, let p be a new state and $\mathcal{A}' = (Q \cup \{p\}, A, E', p, F')$ be the standard automaton defined by

$E' = E \cup \{(p, a, q) \mid (i, a, q) \in E\}$ and

$$F' = \begin{cases} F & \text{if } i \notin F, \\ F \cup \{p\} & \text{if } i \in F. \end{cases}$$

Then the path $q_- \xrightarrow{a_0} q_1 \xrightarrow{a_1} q_2 \ \cdots\ q_{n-1} \xrightarrow{a_{n-1}} q_n$ is successful in $\mathcal{A}$ if and only if the path $p \xrightarrow{a_0} q_1 \xrightarrow{a_1} q_2 \ \cdots\ q_{n-1} \xrightarrow{a_{n-1}} q_n$ is successful in $\mathcal{A}'$. Consequently, $\mathcal{A}$ and $\mathcal{A}'$ are equivalent. ☐

Example 2.7. Standardization is illustrated in Figure 10.

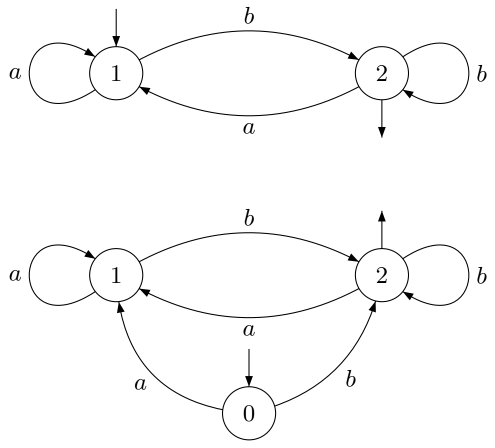


Figure 10. An automaton and its standardized version.

3 Operations on recognizable languages

We review in this section some classical results on finite automaton. We give explicit constructions for the following operations: Boolean operations, product, star, quotients and inverses of morphisms.

3.1 Boolean operations

We give in this section the well known constructions for union, intersection and complement. Complementation is trivial, but requires a deterministic automaton.

Proposition 3.1. *The union of two recognizable languages is recognizable.*

Proof. Let L $[L']$ be a recognizable language of A^* recognized by the automaton $\mathcal{A} = (Q, A, E, I, F)$ $[\mathcal{A}' = (Q', A, E', I', F')]$. We suppose that Q and Q' are disjoint sets and thus one can identify E and E' with subsets of $(Q \cup Q') \times A \times (Q \cup Q')$. Then $L + L'$ is recognized by the automaton $(Q \cup Q', A, E \cup E', I \cup I', F \cup F')$. □

Example 3.1. If L $[L']$ is recognized by the automaton $\mathcal{A}$ $[\mathcal{A}']$ represented in Figure 11, then $L + L'$ is recognized by the automaton represented in Figure 12.

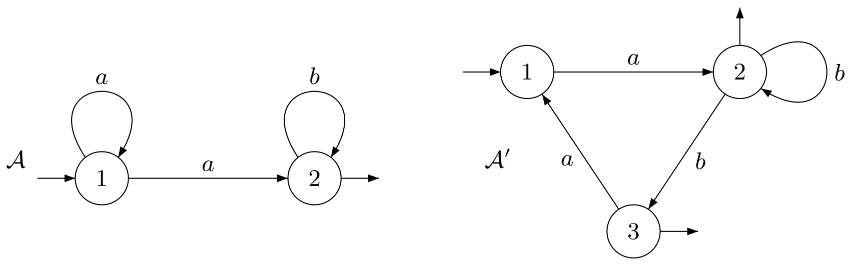


Figure 11. The automata $\mathcal{A}$ and $\mathcal{A}'$.

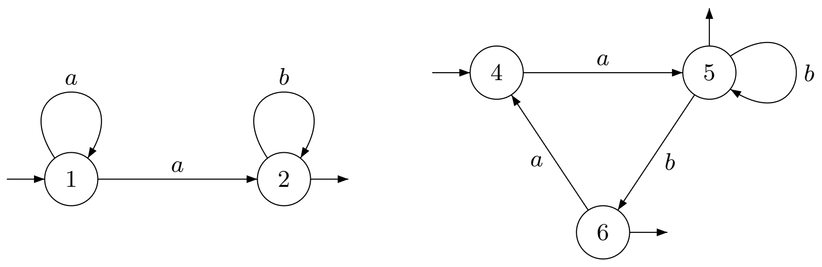


Figure 12. An automaton recognizing $L + L'$.

Corollary 3.2. *Every finite language is recognizable.*

Proof. Since recognizable languages are closed under union, it suffices to verify that the singletons are recognizable. But it is clear that the language $a_1 a_2 \cdots a_n$ is recognized by the automaton represented in Figure 13. □

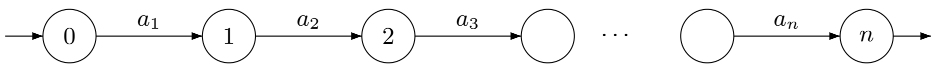


Figure 13. An automaton recognizing $a_1 \cdots a_n$.

Proposition 3.3. *The intersection of two recognizable languages is recognizable.*

Proof. Let L $[L']$ be a recognizable language of A^* recognized by the automaton $\mathcal{A} = (Q, A, E, I, F)$ $[\mathcal{A}' = (Q', A, E', I', F')]$. Consider the automaton $\mathcal{B} = (Q \times Q', A, T, I \times I', F \times F')$ where

$$T = \Big\{ \big((q_1, q_1'), a, (q_2, q_2')\big) \mid (q_1, a, q_2) \in E \text{ and } (q_1', a, q_2') \in E' \Big\}.$$

A word $u = a_1a_2 \cdots a_n$ is the label of a successful path in $\mathcal{B}$

$$(q_0, q'_0) \xrightarrow{a_1} (q_1, q'_1) \xrightarrow{a_2} (q_2, q'_2) \quad \cdots \quad (q_{n-1}, q'_{n-1}) \xrightarrow{a_n} (q_n, q'_n)$$

if and only if the paths

$$q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \cdots q_{n-1} \xrightarrow{a_n} q_n \text{ and}$$
$$q'_0 \xrightarrow{a_1} q'_1 \xrightarrow{a_2} q'_2 \cdots q'_{n-1} \xrightarrow{a_n} q_n$$

are successful paths in $\mathcal{A}$ and $\mathcal{A}'$ respectively. Therefore, $\mathcal{B}$ recognizes $L \cap L'$. $\square$

In practice, one just computes the trim part of $\mathcal{B}$.

Example 3.2. If L [L'] is recognized by the automaton $\mathcal{A}$ [$\mathcal{A}'$] represented in Figure 11, then $L \cap L'$ is recognized by the trim automaton represented in Figure 14.

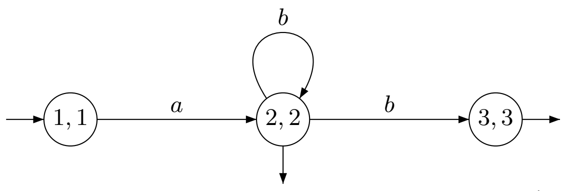


Figure 14. A trim automaton recognizing $L \cap L'$.

Proposition 3.4. *The complement of a recognizable language is recognizable.*

Proof. Let L be a recognizable language of A^* and let $\mathcal{A} = (Q, A, \cdot, q_-, F)$ be a complete deterministic automaton recognizing L. Then the automaton $\mathcal{A}' = (Q, A, \cdot, q_-, Q \setminus F)$ recognizes L^c. Indeed, since $\mathcal{A}$ and $\mathcal{A}'$ are both deterministic and complete, every word u of A^* is the label of exactly one path starting in q_-. Let q be the end of this path. Then u belongs to L if and only if q belongs to F and u belongs to L^c if and only if q belongs to $Q \setminus F$. $\square$

Example 3.3. The language $(ab)^*$ is recognized by the complete deterministic automaton $\mathcal{A}$, and its complement is recognized by the automaton $\mathcal{A}'$ represented in Figure 15.

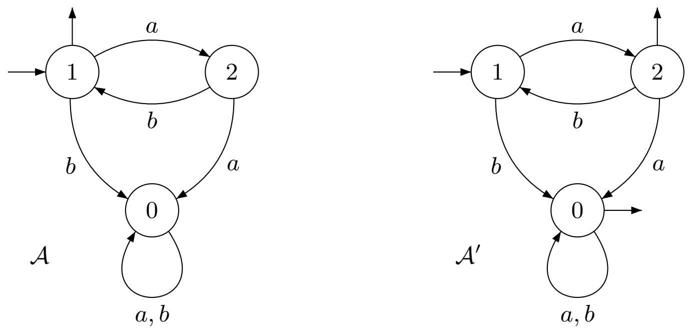


Figure 15. Complementation of a deterministic automaton.

3.2 Product

Proposition 3.5. *The product of two recognizable languages is recognizable.*

Proof. Let L_1 and L_2 be two recognizable languages of A^*, recognized by the automata $\mathcal{A}_1 = (Q_1, A, E_1, I_1, F_1)$ and $\mathcal{A}_2 = (Q_2, A, E_2, I_2, F_2)$, respectively. One may assume, by Propositions 2.3 and 2.4, that $\mathcal{A}_2$ is a standard deterministic automaton and thus in particular that $I_2 = \{i\}$. One can also suppose that Q_1 and Q_2 are disjoint. Let now $\mathcal{A} = (Q, A, E, I, F)$, where

$$\begin{aligned}
&Q = (Q_1 \cup Q_2) \setminus \{i\},\\
&E = E_1 \cup \{(q, a, q') \in E_2 \mid q \neq i\} \cup \{(q_1, a, q') \mid q_1 \in F_1 \text{ and } (i, a, q') \in E_2\},\\
&I = I_1, \text{ and}\\
&F = \begin{cases} F_2 & \text{if } i \notin F_2,\\ F_1 \cup (F_2 \setminus \{i\}) & \text{if } i \in F_2 \text{ (i.e., if } 1 \in L_2). \end{cases}
\end{aligned}$$

We claim that $\mathcal{A}$ recognizes L_1L_2. If u is a word of L_1L_2, then $u = u_1u_2$ for some $u_1 \in L_1$ and $u_2 \in L_2$. Therefore, there is a successful path $c_1 : i_1 \xrightarrow{u_1} q_1$ in $\mathcal{A}_1$ (with $i_1 \in I_1$ and $q_1 \in F_1$) and a successful path $c_2 : i \xrightarrow{u_2} q_2$ in $\mathcal{A}_2$, with $q_2 \in F_2$. If $u_2 = 1$, then L_2 contains the empty word, the path c_1 is a successful path in $\mathcal{A}$ and u is accepted by $\mathcal{A}$. If u_2 is not the empty word, let a be the first letter of u_2 and let $i \xrightarrow{a} q$ be the first transition of c_2. Since $q_1 \in F_1$, $q_1 \xrightarrow{a} q$ is by definition a transition of E. Furthermore, if $q' \xrightarrow{b} q''$ is a transition of c_2 different from the first transition, then q' is the end of a transition of $\mathcal{A}_2$. Since $\mathcal{A}_2$ is standard, this implies $q' \neq i$ and it follows from the definition of E that the transition $q' \xrightarrow{b} q''$ is also a transition of $\mathcal{A}$. Let c'_2 be the path of $\mathcal{A}$ obtained by replacing in c_2 the first transition $i \xrightarrow{a} q$ by $q_1 \xrightarrow{a} q$. The resulting path $c_1c'_2$ is a successful path in $\mathcal{A}$ of label u and hence u is accepted by $\mathcal{A}$.

Conversely, let u be a word accepted by $\mathcal{A}$. Then u is the label of a successful path $c : i_1 \xrightarrow{u} f$ of $\mathcal{A}$. Since the initial states of $\mathcal{A}$ are contained in Q_1, and since there is no transition of $\mathcal{A}$ starting in Q_2 and ending in Q_1, c visits first some states of Q_1 and then possibly some states of Q_2. If all the states visited by c are in Q_1, one has in particular $f \in Q_1$. But this is only possible if $1 \in L_2$, and in this case, c is also a successful path of $\mathcal{A}_1$, and hence $u \in L_1 \subseteq L_1L_2$. If c visits some states of Q_2, then c contains a unique transition of the form $e = (q_1, a, q_2)$ with $q_1 \in F_1$ and $q_2 \in Q_2$. Therefore $c = c_1ec_2$, where c_1 is a path in $\mathcal{A}_1$ and c_2 is a path in $\mathcal{A}_2$. Denoting by u_1 $[u_2]$ the label of c_1 $[c_2]$, we get $u = u_1au_2$. Since c_1 is a successful path in $\mathcal{A}_1$, one has $u_1 \in L_1$. Moreover, by definition of E, $e' = (i, a, q_2)$ is a transition of $\mathcal{A}_2$. Therefore the path $e'c_2$ is a successful path in $\mathcal{A}_2$ of label au_2. It follows that $au_2 \in L_2$ and thus $u \in L_1L_2$, proving the claim and the proposition. $\square$

Example 3.4. If L_1 $[L_2]$ is recognized by the automaton $\mathcal{A}_1$ $[\mathcal{A}_2]$ represented in Figure 16, then L_1L_2 is recognized by the automaton represented in Figure 17.

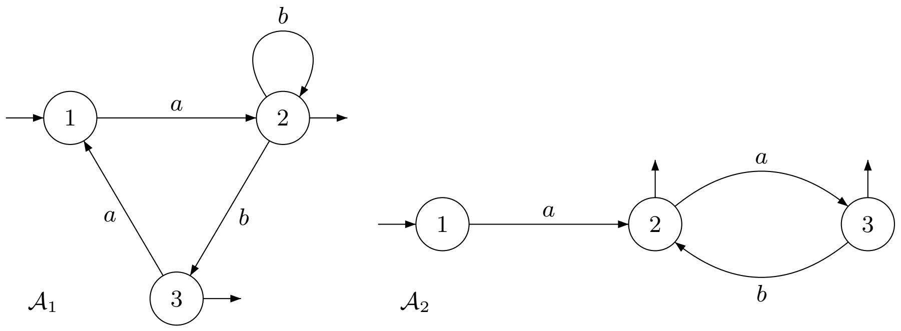


Figure 16. The automata $\mathcal{A}_1$ and $\mathcal{A}_2$.

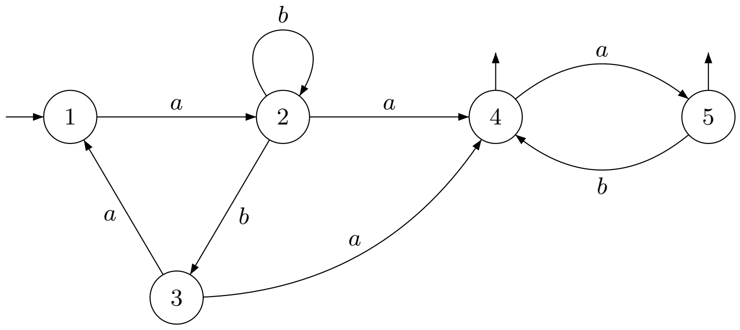


Figure 17. An automaton recognizing L_1L_2.

3.3 Star

Proposition 3.6. *The star of a recognizable language is recognizable.*

Proof. Let L be a recognizable language of A^*, recognized by the deterministic standard automaton $\mathcal{A} = (Q, A, E, q_-, F)$. Let $\mathcal{A}' = (Q, A, E', \{q_-\}, F \cup \{q_-\})$ be the nondeterministic automaton defined by

$$E' = E \cup \{(q, a, q') \mid q \in F \text{ and } (q_-, a, q') \in E\}.$$

Let us show that $\mathcal{A}'$ recognizes L^*. If u is a word of L^*, then either u is the empty word, which is accepted by $\mathcal{A}'$ since q_- is a final state, or $u = u_1u_2 \cdots u_n$ with $u_1, \ldots, u_n \in L\setminus1$. Each u_i is the label of a successful path in $\mathcal{A}$, say $c_i : q_- \xrightarrow{u_i} q_i$ with $q_i \in F$. Let a_i be the first letter of u_i and let $q_- \xrightarrow{a_i} p_i$ be the first transition of c_i. Let $i \in \{2, \ldots, n\}$. As $q_{i-1} \in F$, the definition of E' shows that $q_{i-1} \xrightarrow{a_i} p_i$ is a transition of $\mathcal{A}'$. Denote by c'_i the path obtained by replacing in c_i the first transition $q_- \xrightarrow{a_i} p_i$ by $q_{i-1} \xrightarrow{a_i} p_i$. This defines, for $2 \leqslant i \leqslant n$, a path $c'_i : q_{i-1} \xrightarrow{u_i} q_i$ in $\mathcal{A}'$. Therefore, the path $c_1c'_2 \cdots c'_n$ is a successful path of label u in $\mathcal{A}'$ and hence u is accepted by $\mathcal{A}'$.

Conversely, let u be a word accepted by $\mathcal{A}'$. If $u = 1$, one has $u \in L^*$. Otherwise, u is the label of a nonempty successful path c of $\mathcal{A}'$. This path can be factorized as

$$c = q_- \xrightarrow{u_0} q_1 \xrightarrow{a_1} q'_1 \xrightarrow{u_1} q_2 \xrightarrow{a_2} q'_2 \quad \cdots \quad q_n \xrightarrow{a_n} q'_n \xrightarrow{u_n} q_{n+1}$$

where the transitions $e_1 = q_1 \xrightarrow{a_1} q'_1$, $e_2 = q_2 \xrightarrow{a_2} q'_2$, …, $e_n = q_n \xrightarrow{u_n} q'_{n+1}$ are exactly the transitions of $E' \setminus E$ occurring in c. Thus by definition of E', one gets, for $1 \leqslant i \leqslant n$, $q_i \in F$ and $e'_i = (q_-, a_i, q'_i) \in E$. Furthermore, $q_{n+1} \in F \cup \{q_-\}$ since c is a successful path. Consequently, the paths

$$q_- \xrightarrow{a_i} q'_i \xrightarrow{u_i} q_{i+1}$$

are paths of $\mathcal{A}$. For $1 \leqslant i \leqslant n-1$, these paths are successful, since $q_i \in F$. Moreover, since $\mathcal{A}$ is standard, q_{n+1} is different from q_- and hence $q_{n+1} \in F$. Consequently $a_iu_i \in L$ for $1 \leqslant i \leqslant n$. Since $q_- \xrightarrow{u_0} q_1$ is also a successful path of $\mathcal{A}$, one also has $u_0 \in L$, and hence $u \in L^*$. □

Example 3.5. If L is recognized by the standard deterministic automaton $\mathcal{A}_2$ represented in Figure 16, then L^* is recognized by the nondeterministic automaton represented in Figure 18.

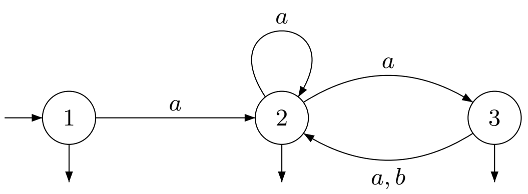


Figure 18. An automaton recognizing L^*.

3.4 Quotients

We first treat the left quotient by a word and then the general case.

Proposition 3.7. *Let $\mathcal{A} = (Q, A, \cdot, q_-, F)$ be a deterministic automaton recognizing a language L of A^*. Then, for each word u of A^*, the language $u^{-1}L$ is recognized by the automaton $\mathcal{A}_u = (Q, A, \cdot, q_- \cdot u, F)$, obtained from $\mathcal{A}$ by changing the initial state. In particular $u^{-1}L$ is recognizable.*

Proof. First the following formulas hold:

$$\begin{aligned} u^{-1}L &= \{v \in A^* \mid uv \in L\} = \{v \in A^* \mid q_- \cdot (uv) \in F\} \\ &= \{v \in A^* \mid (q_- \cdot u) \cdot v \in F\}. \end{aligned}$$

Therefore $u^{-1}L$ is accepted by $\mathcal{A}_u$. □

Proposition 3.8. *Any quotient of a recognizable language is recognizable.*

Proof. Let (Q, A, E, I, F) be an automaton recognizing a language L of A^* and let K be a language of A^*. We do not assume that K is recognizable. Setting

$$I' = \{q \in Q \mid q \text{ is the end of an initial path whose label belongs to } K\},$$

it is easy to see that the automaton $\mathcal{B} = (Q, A, E, I', F)$ recognizes $K^{-1}L$. For the language LK^{-1}, a similar proof works by considering the automaton (Q, A, E, I, F'), where $F' = \{q \in Q \mid q \text{ is the origin of a final path whose label belongs to } K\}$. □

3.5 Inverses of morphisms

We now show that recognizable languages are closed under inverses of morphisms.

Proposition 3.9. *Let $\varphi : A^* \to B^*$ be a morphism. If L is a recognizable language of B^*, then $\varphi^{-1}(L)$ is a recognizable language of A^*.*

Proof. Let $\mathcal{B} = (Q, B, E, I, F)$ be an automaton recognizing L. Let $\mathcal{A} = (Q, A, T, I, F)$, where

$$T = \{(p, a, q) \mid \text{there is a path labelled by } \varphi(a) \text{ from } p \text{ to } q \text{ in } \mathcal{B}\}.$$

We claim that $\mathcal{A}$ recognizes $\varphi^{-1}(L)$. First, if u is accepted by $\mathcal{A}$, there is a successful path of $\mathcal{A}$ labelled by u. Consequently, there is a successful path of $\mathcal{B}$ labelled by $\varphi(u)$. Thus $\varphi(u)$ is accepted by $\mathcal{B}$ and $u \in \varphi^{-1}(L)$.

Let now $u = a_1 \cdots a_n$ be a word of $\varphi^{-1}(L)$. Since the word $\varphi(u)$ is accepted by L, there is a successful path in $\mathcal{B}$ labelled by $\varphi(u)$. Let us factorize this path as

$$q_0 \xrightarrow{\varphi(a_1)} q_1 \ \cdots \ q_{n-1} \xrightarrow{\varphi(a_n)} q_n \ .$$

These paths define in turn a successful path in $\mathcal{A}$ labelled by u:

$$q_0 \xrightarrow{a_1} q_1 \ \cdots \ q_{n-1} \xrightarrow{a_n} q_n,$$

which shows that u is accepted by $\mathcal{A}$. □

4 Minimal automaton and syntactic monoid

4.1 Minimal automaton

Let L be a language of A^*. The *Nerode automaton* of L is the deterministic automaton $\mathcal{A}(L) = (Q, A, \cdot, L, F)$ where $Q = \{u^{-1}L \mid u \in A^*\}$, $F = \{u^{-1}L \mid u \in L\}$ and the transition function is defined, for each $a \in A$, by the formula

$$(u^{-1}L)\cdot a = a^{-1}(u^{-1}L) = (ua)^{-1}L.$$

Beware of this rather abstract definition. Each state of $\mathcal{A}(L)$ is a left quotient of L by a word, and hence is a language of A^*. The initial state is the language L, and the set of final states is the set of all left quotients of L by a word of L.

Proposition 4.1. *A language L is recognizable if and only if the set $\{u^{-1}L \mid u \in A^*\}$ is finite. In this case, L is recognized by its Nerode automaton.*

Proof. Let L be a recognizable language, accepted by the deterministic automaton $\mathcal{A} = (Q, A, \cdot, q_-, F)$. By Proposition 3.7, the language $u^{-1}L$ is accepted by the automaton $\mathcal{A}_u = (Q, A, \cdot, q_- \cdot u, F)$. If n is the number of states of $\mathcal{A}$, there are at most n automata of the form $\mathcal{A}_u$ and hence at most n distinct languages of the form $u^{-1}L$.

Conversely, if the set $\{u^{-1}L \mid u \in A^*\}$ is finite, the Nerode automaton of L is finite and recognizes L. Indeed, a word u is accepted by $\mathcal{A}(L)$ if and only if $L \cdot u = u^{-1}L$ is a final state, that is if $u \in L$. It follows that L is recognizable. □

Let $\mathcal{A} = (Q, A, E, q_-, F)$ and $\mathcal{A}' = (Q', A, E', q'_-, F')$ be two deterministic automata. A *morphism of automata* from $\mathcal{A}$ to $\mathcal{A}'$ is a surjective function $\varphi : Q \to Q'$ such that $\varphi(q_-) = q'_-$, $\varphi^{-1}(F') = F$ and, for every $u \in A^*$ and $q \in Q$, $\varphi(q \cdot u) = \varphi(q) \cdot u$. We write $\mathcal{A} \leqslant \mathcal{A}'$ if there is a morphism from $\mathcal{A}$ to $\mathcal{A}'$.

Let L be a recognizable language. The next proposition shows that, amongst the accessible and complete deterministic automata recognizing L, the Nerode automaton of L is minimal for $\leqslant$. For this reason it is called the *minimal complete automaton* of L.

Proposition 4.2. *Let $\mathcal{A} = (Q, A, \cdot, q_-, F)$ be a accessible and complete deterministic automaton accepting L. For each state q of Q, let L_q be the language recognized by $(Q, A, \cdot, q, F)$. Then $\mathcal{A}(L) = (\{L_q \mid q \in Q\}, A, \cdot, L_{q_-}, \{L_q \mid q \in F\})$, where, for all $a \in A$ and for all $q \in Q$, $L_q \cdot a = L_{q \cdot a}$. Moreover, the map $q \mapsto L_q$ defines a morphism from $\mathcal{A}$ onto $\mathcal{A}(L)$.*

Proof. Let q be a state of Q. Since q is accessible, there is a word u of A^* such that $q_- \cdot u = q$, and by Proposition 3.7, one has $L_q = u^{-1}L$. Conversely, if u is a word, one has $u^{-1}L = L_q$ with $q = q_- \cdot u$. Therefore

$$\{L_q \mid q \in Q\} = \{u^{-1}L \mid u \in A^*\} \quad \text{and} \quad \{L_q \mid q \in F\} = \{u^{-1}L \mid u \in L\},$$

which proves the first part of the statement.

Furthermore, for all $a \in A$, one has $\varphi(q \cdot a) = L_{q \cdot a} = L_q \cdot a = \varphi(q) \cdot a$ which shows that the map $\varphi : q \mapsto L_q$ is a morphism from $\mathcal{A}$ onto $\mathcal{A}(L)$. □

The direct computation of the Nerode automaton is probably the most efficient method for a computation by hand, because it gives directly the minimal automaton. In practice, one starts with the quotient $L = 1^{-1}L$ and one maintains a table of quotients of L. For each quotient R, it suffices to compute the quotients $a^{-1}R$ for each letter a. These quotients are compared to the existing list of quotients and possibly added to this list. But there is a hidden difficulty: the comparison of two rational expressions is not always easy, since a given language might be represented by two very different rational expressions.

Example 4.1. For $L = (a(ab)^*)^* \cup (ba)^*$, we get $1^{-1}L = L = L_1$ and

$$\begin{aligned}
a^{-1}L_1 &= (ab)^*(a(ab)^*)^* = L_2 & b^{-1}L_1 &= a(ba)^* = L_3 \\
a^{-1}L_2 &= bL_2 \cup L_2 = L_4 & b^{-1}L_2 &= \emptyset \\
a^{-1}L_3 &= (ba)^* = L_5 & b^{-1}L_3 &= \emptyset \\
a^{-1}L_4 &= a^{-1}(bL_2 \cup L_2) = L_4 & b^{-1}L_4 &= b^{-1}(bL_2 \cup L_2) = L_2 \\
a^{-1}L_5 &= \emptyset & b^{-1}L_5 &= a(ba)^* = L_3
\end{aligned}$$

which gives the minimal automaton represented in Figure 19.

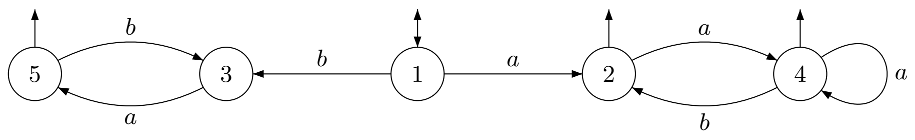


Figure 19. The minimal automaton of L.

There are standard algorithms for minimizing a given accessible deterministic automaton [3] based on the computation of the *Nerode equivalence*. Let $\mathcal{A} = (Q, A, E, q_-, F)$ be a accessible deterministic automaton. The Nerode equivalence $\sim$ on Q is defined by $p \sim q$ if and only if, for every word $u \in A^*$,

$$p\cdot u \in F \iff q\cdot u \in F.$$

One can show that $\sim$ is actually a congruence, in the sense that F is saturated by $\sim$ and that $p \sim q$ implies $p\cdot x \sim q\cdot x$ for all $x \in A^*$. It follows that there is a well-defined quotient automaton $\mathcal{A}/{\sim} = (Q/{\sim}, A, E, \tilde{q}_-, F/{\sim})$, where $\tilde{q}_-$ is the equivalence class of q_-.

Proposition 4.3. *Let $\mathcal{A}$ be an accessible and complete deterministic automaton. Then $\mathcal{A}/{\sim}$ is the minimal automaton of $\mathcal{A}$.*

We shall in particular use the following consequence.

Corollary 4.4. *An accessible and complete deterministic automaton is minimal if and only if its Nerode equivalence is the identity.*

4.2 Automata and monoids

Let $\mathcal{A} = (Q, A, \cdot, q_-, F)$ be a complete deterministic automaton. For each $v \in A^*$, the mapping

$$q \mapsto q\cdot v$$

defines a function $f_v \colon Q \to Q$. If we compose these functions from left to right (in effect, letting them act on Q on the right), we evidently have

$$f_v f_w = f_{vw}$$

for all $v, w \in A^*$. Thus the set

$$M(\mathcal{A}) = \{f_v \mid v \in A^*\}$$

of these functions is a monoid, with composition as the operation, and the map

$$\eta_{\mathcal{A}} \colon v \mapsto f_v$$

is a morphism from A^* onto $M(\mathcal{A})$. We call $M(\mathcal{A})$ the *transition monoid* of the automaton $\mathcal{A}$. Since there are only finitely many maps from Q into itself when Q is a finite set, $M(\mathcal{A})$ is a finite monoid.

Example 4.2. Consider the automaton $\mathcal{A}$ in Example 3.3, which recognizes the language $(ab)^*$. We will represent the mapping f_v by the table

	f_v
0	$0\cdot v$
1	$1\cdot v$
2	$2\cdot v$

We can then tabulate the functions f_v for short words v:

	f_1	f_a	f_b	f_{ab}	f_{ba}	f_{aa}
0	0	0	0	0	0	0
1	2	0	1	0	0	0
2	0	1	0	1	2	0

It is easy to check that for any of the words v for which f_v is tabulated above, the maps $f_{va} = f_v f_a$ and $f_{vb} = f_v f_b$ are already in the list. For example, $f_{aba} = f_a$. These six elements thus constitute the entire transition monoid $M(\mathcal{A})$.

Since f_1 is the identity element of $M(\mathcal{A})$, we will denote it by 1. The element f_{aa} satisfies

$$f_{aa}m = f_{aa} = mf_{aa}$$

for all $m \in M(\mathcal{A})$ — that is, it is a *zero* of the monoid — and so we will write $f_{aa} = 0$. We abbreviate the remaining elements by a, b, ab, ba, and obtain the following multiplication table for $M(\mathcal{A})$.

$\cdot$	1	a	b	ab	ba	0
1	1	a	b	ab	ba	0
a	a	0	ab	0	a	0
b	b	ba	0	b	0	0
ab	ab	a	0	ab	0	0
ba	ba	0	b	0	ba	0
0	0	0	0	0	0	0

Again, let $\mathcal{A} = (Q, A, \cdot, q_-, F)$ be a complete deterministic automaton. Then

$$L(\mathcal{A}) = \eta_{\mathcal{A}}^{-1}(X),$$

where

$$X = \{f_v \mid q_- f_v \in F\} \subseteq M(\mathcal{A}).$$

We say that a monoid M *recognizes* a language $L \subseteq A^*$ if there is a morphism $\varphi\colon A^* \to M$ and a set $X \subseteq M$ such that $L = \varphi^{-1}(X)$. We also say in this instance that the morphism φ recognizes L. Thus our observation above can be restated: for any complete deterministic automaton $\mathcal{A}$, $L(\mathcal{A})$ is recognized both by the monoid $M(\mathcal{A})$ and by the morphism $\eta_{\mathcal{A}}$.

Example 4.3. In Example 4.2 we have $L(\mathcal{A}) = (ab)^* = \eta_{\mathcal{A}}^{-1}(\{1, f_{ab}\})$.

4.3 Syntactic monoid and syntactic morphism

There are two ways to introduce the syntactic monoid of a recognizable language. We first present an algorithm to compute it and then give an abstract definition.

Let $L \subseteq A^*$ be a recognizable language and let $\mathcal{A}(L)$ be its minimal automaton. The monoid $\mathrm{Synt}(L) = M(\mathcal{A}(L))$ and the morphism $\eta_L = \eta_{\mathcal{A}(L)}$ are called, respectively, the *syntactic monoid* and the *syntactic morphism* of L. In other words, the syntactic monoid of L is the transition monoid of the minimal automaton of L.

Example 4.4. In Example 4.2 we have $\mathrm{Synt}((ab)^*) = M(\mathcal{A})$, because $\mathcal{A}$ is the minimal automaton of $(ab)^*$.

Example 4.5. Let A be a finite alphabet and let $a \in A$. Consider the language $L = A^*aA^*$, which consists of all words that contain an occurrence of the letter a. The minimal automaton $\mathcal{A}(L)$ is $(\{0,1\}, A, \cdot, 1, \{0\})$, where $1 \cdot a = 0$ and $q \cdot b = q$ whenever either $q \neq 1$ or $b \neq a$. The syntactic monoid $\mathrm{Synt}(L)$ then contains two transitions: f_1, which is the identity of the monoid, and f_a, which is a zero. We can thus write this monoid as $\{0,1\}$, with the usual multiplication.

Let M and N be monoids. We say that M *divides* N, and write $M \prec N$, if there is a submonoid N' of N and a morphism $\varphi \colon N' \to M$ that maps onto M.

Informally, $M \prec N$ means that M is *simpler* than N. We would naturally consider M to be simpler than N if M is either a submonoid or a quotient of N. We would also expect 'simpler than' to be a transitive relation. It is easy to prove (and left as an exercise for the reader) that division is the least transitive relation that includes both the submonoid and quotient relation. The next proposition says that the syntactic monoid of L is the simplest monoid recognizing L.

Proposition 4.5. *Let L be a recognizable language and M a monoid. Then M recognizes L if and only if* $\mathrm{Synt}(L)$ *divides M.*

Proof. First suppose that $\mathrm{Synt}(L)$ divides M. Then there is a submonoid M' of M and a surjective morphism $\varphi \colon M' \to \mathrm{Synt}(L)$. Since φ is surjective, there exists for each $a \in A$ at least one $m_a \in M'$ such that $\varphi(m_a) = \eta_L(a)$. Set $\psi(a) = m_a$. Then ψ extends to a unique morphism (also denoted ψ) from A^* into M'. Observe that $\varphi \circ \psi = \eta_L$, because $\varphi(\psi(a)) = \eta_L(a)$ for all $a \in A$.

Let $P = \psi(L)$. We claim that for all words $v \in A^*$, $\psi(v) \in P$ if and only if $v \in L$. This will show $L = \psi^{-1}(P)$, and hence M recognizes L. By definition, $v \in L$ implies $\psi(v) \in P$. On the other hand, if $\psi(v) \in P$, then there is some $v' \in L$ such that $\psi(v) = \psi(v')$. We thus have

$$f_v = \eta_L(v) = \varphi \circ \psi(v) = \varphi \circ \psi(v') = \eta_L(v') = f_{v'}.$$

Let $\mathcal{A}(L) = (Q, A, \cdot, q_-, F)$ be the minimal automaton of L. Since $v' \in L$, we have $q_- \cdot v' \in F$ and, since $f_v = f_{v'}$, $q_- \cdot v = q_- \cdot v'$. So $q_- \cdot v \in F$ and thus $v \in L$, as claimed. Note that we have not used the minimality of this automaton — we have actually proved that if N recognizes L and N divides M, then M recognizes L as well.

For the converse, suppose that M recognizes L via a morphism $\psi\colon A^* \to M$ and a subset $P \subseteq M$ such that $\psi^{-1}(P) = L$. As before, let $\mathcal{A}(L) = (Q, A, \cdot, q_-, F)$ be the minimal automaton of L. We will show that if $w, w' \in A^*$ with $\psi(w) = \psi(w')$, then $\eta_L(w) = \eta_L(w')$, in other words, that the syntactic morphism η_L 'factors through' ψ. This implies the existence of a morphism $\varphi\colon \mathrm{Im}(\psi) \to \mathrm{Synt}(L)$ such that $\varphi \circ \psi = \eta_L$. Since η_L maps onto $\mathrm{Synt}(L)$, we conclude that $\mathrm{Synt}(L) \prec M$. Suppose, contrary to what we are trying to prove, that $\eta_L(w) \neq \eta_L(w')$. Thus there is some state q of $\mathcal{A}(L)$ such that $q\cdot w \neq q\cdot w'$. In particular, by Corollary 4.4, there is some $v \in A^*$ such that $q\cdot wv \in F$ and $q\cdot w'v \notin F$, or vice-versa. Since every state of $\mathcal{A}(L)$ is accessible from the initial state, there is accordingly some $u \in A^*$ such that $q_- \cdot u = q$, and thus $uwv \in L$, $uw'v \notin L$. Since $uwv \in L$, we must have $\psi(uwv) \in P$. But $\psi(uw'v) = \psi(uwv)$ is then also in P, so that $uw'v \in L$, a contradiction. □

We have defined the syntactic monoid of a recognizable language as the transition monoid of its minimal automaton. One often sees a different, although equivalent, definition in terms of *congruences*. A congruence on a monoid M (finite or infinite) is an equivalence relation $\sim$ on M that is compatible with multiplication in M: in other words, if $m_i \sim m_i'$ for $i = 1, 2$, then $m_1m_2 \sim m_1'm_2'$. In this case, there is a well-defined multiplication on the quotient set $M/{\sim}$ of equivalence classes of $\sim$ that turns $M/{\sim}$ into a monoid. The map taking $m \in M$ to its equivalence class $[m]_\sim$ is then a morphism from M onto this quotient monoid, called the *projection morphism*.

Let $L \subseteq A^*$ be a recognizable language. We define an equivalence relation $\sim_L$ on A^* as follows: if $u, v \in A^*$, then $u \sim_L v$ if and only if for every $x, y \in A^*$, xuy and xvy are either both in L or both outside of L. We call $\sim_L$ the *syntactic congruence* of L.

To see the connection to the syntactic monoid and syntactic morphism, as we have defined them, first suppose that $u \sim_L v$. Let q be any state of the minimal automaton of L. If $q\cdot u \neq q\cdot v$, then there is some word $y \in A^*$ such that $q\cdot uy$ is an accepting state and $q\cdot vy$ is not, or vice-versa. Without loss of generality, we can assume that $q\cdot uy$ is accepting. Since every state of the minimal automaton is accessible from the initial state q_-, we have $q = q_- \cdot x$ for some $x \in A^*$. Thus $xuy \in L$ and $xvy \notin L$, contradicting $u \sim_L v$. Thus we must have $q\cdot u = q\cdot v$ for every state q. This shows that $\eta_L(u) = \eta_L(v)$. Conversely, suppose $\eta_L(u) = \eta_L(v)$. Then $f_{xuy} = f_x f_u f_y = f_x f_v f_y$. Thus $q_- \cdot xuy$ is accepting if and only if $q_- \cdot xvy$ is. That is, $xuy \in L$ if and only if $xvy \in L$, *i.e.*, $u \sim_L v$. We have shown that $u \sim_L v$ if and only if $\eta_L(u) = \eta_L(v)$, so that equivalence classes of $\sim_L$ are in one-to-one correspondence with elements of the syntactic monoid. This implies immediately that $\sim_L$ is a congruence, and that the correspondence $\eta_L(u) \leftrightarrow [u]_{\sim_L}$ is an isomorphism of monoids. We have thus proved

Proposition 4.6. *Let $L \subseteq A^*$ be a recognizable language. Then $\sim_L$ is a congruence, and the quotient monoid $A^*/{\sim_L}$ is isomorphic to the syntactic monoid* $\mathrm{Synt}(L)$.

Example 4.6. Let us use this alternative definition of the syntactic monoid to recompute $\mathrm{Synt}((ab)^*)$, the syntactic monoid we originally computed in Example 4.2. The syntactic congruence of L identifies two words u and v if the set of pairs of words (x, y) such that $xuy \in L$ is the same as the set of pairs for which $xvy \in L$. If u contains two consecutive occurrences of a or two consecutive occurrences of b, then this set of pairs is empty.

Indeed, these words u are the only ones for which there is no pair (x, y) with $xuy \in L$, so one of the $\sim_L$-classes is $A^*aaA^* \cup A^*bbA^*$.

Nonempty words in which the letters a and b alternate lie in one of the sets

$$(ab)^+,\ (ba)^+,\ b(ab)^*,\ (ab)^*a.$$

For all u in, say $b(ab)^+$, $xuy \in L$ if and only if $x \in (ab)^*a$ and $y \in (ab)^*$. So all the words in $b(ab)^+$ belong to a single congruence class. Likewise, all the words in each of the other three sets belong to a single congruence class. The only word we have not considered is the empty word 1. This is not congruent to any nonempty word: for example if u starts with a we have $a \cdot 1 \cdot b \in L$ and $aub \notin L$. We have thus determined all six congruences classes:

$$1,\ (ab)^+,\ (ab)^*a,\ b(ab)^*,\ (ba)^+,\ A^*aaA^* \cup A^*bbA^*.$$

We can compute the multiplication table of the syntactic monoid by choosing a representative word from each of the two classes whose product we seek, and finding which class the concatenation of the two words belongs to. This gives the same multiplication table we constructed earlier using the minimal automaton.

4.4 Ordered versions

A non-symmetrical variant of the definition of the syntactic congruence turns out to be very useful in studying recognizable languages. Let $u, v \in A^*$: we write $u \preccurlyeq_L v$ if for every $x, y \in A^*$ such that $xuy \in L$, we have also $xvy \in L$. Then $\preccurlyeq_L$ is a preorder and it is immediately verified that $u \sim_L v$ if and only if $u \preccurlyeq_L v$ and $v \preccurlyeq_L u$. Moreover, if $u_1 \preccurlyeq_L v_1$ and $u_2 \preccurlyeq_L v_2$, then we have $u_1u_2 \preccurlyeq_L v_1v_2$. As a result, $\preccurlyeq_L$ defines an order relation $\leqslant_L$ on $\mathrm{Synt}(L) = A^*/{\sim_L}$, the syntactic monoid of L, which is stable under product. This order is called the *syntactic order* of L and the resulting ordered monoid $(\mathrm{Synt}(L), \leqslant_L)$ is called the *ordered syntactic monoid* of L.

Example 4.7. We have already seen that $M = \{1, a, b, ab, ba, 0\}$ is the syntactic monoid of $(ab)^*$. The order of its ordered syntactic monoid is given by the relations $ab \leqslant 1$, $ba \leqslant 1$ and $0 \leqslant x$ for all $x \in M$.

The syntactic order can also be computed from the minimal automaton of L. Let $\mathcal{A} = (Q, A, \cdot, q_-, F)$ be the minimal automaton of L. Define a relation $\leqslant$ on Q by setting $p \leqslant q$ if and only if, for all $u \in A^*$, $p \cdot u \in F$ implies $q \cdot u \in F$. The relation $\leqslant$ is clearly reflexive and transitive. Suppose that $p \leqslant q$ and $q \leqslant p$. Then, for all $u \in A^*$, $p \cdot u \in F$ if and only if $q \cdot u \in F$. Since $\mathcal{A}$ is minimal, this implies $p = q$. Thus $\leqslant$ is an order. Furthermore, if $p \leqslant q$, then for all $a \in A$, $p \cdot a \leqslant q \cdot a$ since, for all $u \in A^*$, $p \cdot au \in F$ implies $q \cdot au \in F$.

We know that the syntactic monoid of L is the transition monoid of its minimal automaton. The syntactic order of L can now be defined directly as follows: $f_u \leqslant f_v$ if and only if, for every $q \in Q$, $q \cdot u \leqslant q \cdot v$.

Example 4.8. Consider the minimal complete automaton of $(ab)^*$, represented in Figure 15. The order on the set of states is $0 < 1$ and $0 < 2$. Indeed, one has $0 \cdot u = 0$ for all

$u \in A^*$ and thus, the formal implication

$$0 \cdot u \in F \Rightarrow q \cdot u \in F$$

holds for any $q \in Q$. One can verify that there is no other relations among the states of Q. For instance, 1 and 2 are incomparable since $1 \cdot ab = 1 \in F$ but $2 \cdot ab = 0 \notin F$ and $1 \cdot b = 0 \notin F$ but $2 \cdot b = 1 \in F$. One also recovers the syntactic order described in Example 4.7. For instance, $ab \leqslant 1$ since $0 \cdot ab = 0 \cdot 1$, $1 \cdot ab = 1 \cdot 1$ and $2 \cdot ab = 0 \leqslant 2 = 2 \cdot 1$.

4.5 Operations on recognizable languages

The *direct product* $M_1 \times M_2$ of two monoids M_1 and M_2 is just the ordinary cartesian product with the operation $(m_1, m_2)(m_1', m_2') = (m_1 m_1', m_2 m_2')$.

Proposition 4.7. *Let* $L, L_1, L_2 \subseteq A^*$ *be recognizable languages. Then*

$$\begin{aligned} \mathrm{Synt}(L) &= \mathrm{Synt}(A^* \setminus L), \\ \mathrm{Synt}(L_1 \cup L_2) &\prec \mathrm{Synt}(L_1) \times \mathrm{Synt}(L_2), \\ \mathrm{Synt}(L_1 \cap L_2) &\prec \mathrm{Synt}(L_1) \times \mathrm{Synt}(L_2). \end{aligned}$$

Proof. The minimal automaton of L is identical to that of $A^* \setminus L$, except for the set of accepting states. In particular, these two automata have the same transition monoid, which gives the first part of the claim. For the second, it is enough to prove that if M_1 recognizes L_1 and M_2 recognizes L_2, then $M_1 \times M_2$ recognizes $L_1 \cup L_2$. The result will then follow from Proposition 4.5. Suppose then that for $i = 1, 2$, M_i recognizes L_i through morphisms φ_i and subsets P_i. Let $\varphi\colon A^* \to M_1 \times M_2$ be the morphism defined by $\varphi(w) = (\varphi_1(w), \varphi_2(w))$, and let

$$P = \{(m_1, m_2) \mid m_1 \in P_1 \text{ or } m_2 \in P_2\}.$$

Then $\varphi(w) \in P$ if and only if either $w \in L_1$ or $w \in L_2$. Thus $M_1 \times M_2$ recognizes $L_1 \cup L_2$, as required.

The last part of the Proposition follows from the first two parts by DeMorgan's Laws. □

Example 4.9. The reason for studying the syntactic monoid is that it enables us to characterize properties of a language in terms of algebraic properties of the syntactic monoid. What follows is a very simple example of this approach, which is the subject of Chapter 16. Suppose that a recognizable language $L \subseteq A^*$ is recognized by a monoid M that is both *idempotent*, that is $m \cdot m = m$ for every $m \in M$, and *commutative*, that is, $m \cdot n = n \cdot m$ for all $m, n \in M$. Let $\varphi\colon A^* \to M$ be a morphism that recognizes L. If $w, w' \in A^*$ contain the same *set* of letters, then $\varphi(w) = \varphi(w')$: because of idempotence and commutativity, we can duplicate letters of a word and rearrange them without changing the image of the word under φ. Thus $w \in L$ if and only if $w' \in L$; in other words, membership of a word in L is determined entirely by the set of letters of L. Conversely, if L has this property, then L can be written as a boolean combination of languages of the form A^*aA^*, where $a \in A$. Thus by Proposition 4.7 and Example 4.5, $\mathrm{Synt}(L)$

divides a direct product of copies of the monoid $\{0, 1\}$. This monoid is idempotent and commutative, and it is straightforward to verify that both the idempotence and commutativity properties are preserved under direct products, quotients and taking submonoids. Thus $\mathrm{Synt}(L)$ is idempotent and commutative. We have shown that membership in L is dependent only on the set of letters in the word if and only if the syntactic monoid of L is idempotent and commutative.

5 Rational versus recognizable

The aim of this section is to show that, if A is a finite alphabet, a language of A^* is recognizable if and only if it is rational.

5.1 Local languages

A language L of A^* is said to be *local* if there exist two subsets P and S of A and a subset N of A^2 such that [1]

$$L \setminus 1 = (PA^* \cap A^*S) \setminus A^*NA^*.$$

For instance, if $A = \{a, b, c\}$, the language

$$(abc)^* = 1 \cup [(aA^* \cap A^*c) \setminus A^*\{aa, ac, ba, bb, cb, cc\}A^*]$$

is local. The terminology can be explained as follows: in order to check whether a nonempty word belongs to L, it suffices to verify that its first letter is in P, its last letter is in S and its factors of length 2 are not in N: all these conditions are local. Conversely, if a language L is local, it is easy to recover the parameters P, S and N. Indeed, P $[S]$ is the set of first [last] letters of the words of L, and N is the set of words of length 2 that are factors of no word of L.

It is easy to compute a deterministic automaton recognizing a local language, given the parameters P, S and N.

Proposition 5.1. *Let $L = (PA^* \cap A^*S) \setminus A^*NA^*$ be a local language. Then L is recognized by the automaton $\mathcal{A}$ in which the set of states is $A \cup \{1\}$, the initial state is 1, the set of final states is S, and the transitions are given by the rules $1 \cdot a = a$ if $a \in P$ and $a \cdot b = b$ if $ab \notin N$.*

Proof. Let $u = a_1 \cdots a_n$ be a word accepted by $\mathcal{A}$ and let

$$1 \xrightarrow{a_1} a_1 \xrightarrow{a_2} a_2 \quad \cdots \quad a_{n-1} \xrightarrow{a_n} a_n$$

be a successful path of label u. Then the state a_n is final and hence $a_n \in S$. Similarly, since $1 \xrightarrow{a_1} a_1$ is a transition, one has necessarily $a_1 \in P$. Finally, since for $1 \leqslant i \leqslant n-1$, $a_i \xrightarrow{a_{i+1}} a_{i+1}$ is a transition, the word a_ia_{i+1} is not in N. Consequently, u belongs to L.

[1] P stands for prefix, S for suffix and N for non-factor.

Conversely, if $u = a_1 \cdots a_n \in L$, one has $a_1 \in P$, $a_n \in S$ and, for $1 \leqslant i \leqslant n$, $a_i a_{i+1} \notin N$. Thus $1 \xrightarrow{a_1} a_1 \xrightarrow{a_2} a_2 \quad \cdots \quad a_{n-1} \xrightarrow{a_n} a_n$ is a successful path of $\mathcal{A}$ and $\mathcal{A}$ accepts u. Therefore, the language accepted by $\mathcal{A}$ is L. □

For a local language containing the empty word, the previous construction can be easily modified by taking $S \cup \{1\}$ as the set of final states.

Example 5.1. Let $A = \{a, b, c\}$, $P = \{a, b\}$, $S = \{a, c\}$ and $N = \{ab, bc, ca\}$. Then the automaton in Figure 20 recognizes the language $L = (PA^* \cap A^*S) \setminus A^*NA^*$.

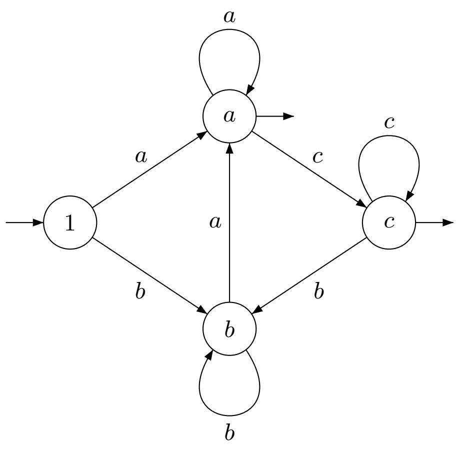


Figure 20. An automaton recognizing a local language.

Note also that the automaton $\mathcal{A}$ described in Proposition 5.1 has a special property: all the transitions of label a have the same end, namely the state a. More generally, we shall say that a deterministic automaton (not necessarily complete) $\mathcal{A} = (Q, A, \cdot\,)$ is *local* if, for each letter a, the set $\{q \cdot a \mid q \in Q\}$ contains at most one element. Local languages have the following characterization:

Proposition 5.2. *A rational language is local if and only if it is recognized by a local automaton.*

Proof. One direction follows from Proposition 5.1. To prove the opposite direction, consider a local automaton $\mathcal{A} = (Q, A, \cdot\,, q_0, F)$ recognizing a language L and let

$$\begin{aligned} P &= \{a \in A \mid q_0 \cdot a \text{ is defined }\},\\ S &= \{a \in A \mid \text{ there exists } q \in Q \text{ such that } q \cdot a \in F\},\\ N &= \{x \in A^2 \mid x \text{ is the label of no path in } \mathcal{A}\,\}\\ K &= (PA^* \cap A^*S) \setminus A^*NA^*. \end{aligned}$$

Let $u = a_1 \cdots a_n$ be a nonempty word of L and let $q_0 \xrightarrow{a_1} q_1 \quad \cdots \quad q_{n-1} \xrightarrow{a_n} q_n$ be a successful path of label u. Necessarily, $a_1 \in P$, $a_n \in S$ and, for $1 \leqslant i \leqslant n-1$, $a_i a_{i+1} \notin N$. Consequently, $u \in K$, which shows that $L \setminus 1$ is contained in K.

Let now $u = a_1 \cdots a_n$ be a nonempty word of K. Then $a_1 \in P$, $a_n \in S$, and, for $1 \leqslant i \leqslant n-1$, $a_i a_{i+1} \notin N$. Since $a_1 \in P$, the state $q_1 = q_0 \cdot a_1$ is well defined. Moreover, since $a_1 a_2 \notin N$, $a_1 a_2$ is the label of some path $p_0 \xrightarrow{a_1} p_1 \xrightarrow{a_2} p_2$ in $\mathcal{A}$. But since $\mathcal{A}$ is a local automaton, $q_0 \cdot a_1 = p_0 \cdot a_1$. It follows that the word $a_1 a_2$ is also the label of the path $q_0 \xrightarrow{a_1} p_1 \xrightarrow{a_2} p_2$. One can show in the same way by induction that there exists a sequence of states p_i ($0 \leqslant i \leqslant n$) such that $a_i a_{i+1}$ is the label of a path $p_{i-1} \xrightarrow{a_i} p_i \xrightarrow{a_{i+1}} p_{i+1}$ of $\mathcal{A}$. Finally, since $a_n \in S$, there is a state q such that $q \cdot a_n \in F$. But since $\mathcal{A}$ is a local automaton, one has $q \cdot a_n = p_{n-1} \cdot a_n = p_n$, whence $p_n \in F$. Therefore $q_0 \xrightarrow{a_1} p_1 \ \cdots \ p_{n-1} \xrightarrow{a_n} p_n$ is a successful path in $\mathcal{A}$ and its label u is accepted by $\mathcal{A}$. Thus $K = L \setminus 1$. □

Local languages are stable under various operations:

Proposition 5.3. *Let A_1 and A_2 be two disjoint subsets of the alphabet A and let $L_1 \subseteq A_1^*$ and $L_2 \subseteq A_2^*$ be two local languages. Then the languages $L_1 + L_2$ and $L_1 L_2$ are local languages.*

Proof. Let $\mathcal{A}_1$ [$\mathcal{A}_2$] be a local automaton recognizing L_1 [L_2]. The proofs of Propositions 3.1 and 3.5 give an automaton recognizing $L_1 + L_2$ and $L_1 L_2$. A simple verification shows that these constructions produce a local automaton when $\mathcal{A}_1$ and $\mathcal{A}_2$ are local. □

Proposition 5.4. *Let L be a local language. Then the language L^* is a local language.*

Proof. Let $\mathcal{A}$ be a local automaton recognizing L. The proof of Proposition 3.6 gives an automaton recognizing L^*. A simple verification shows that this construction produces a local automaton when $\mathcal{A}$ is local. □

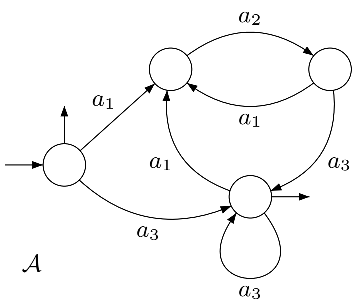
5.2 Glushkov's algorithm

Glushkov's algorithm [2] is an efficient way to convert a rational expression into a nondeterministic automaton.

A rational expression is said to be *linear* if each letter has at most one occurrence in the expression. For instance, the expression

$$[a_1 a_2 (a_3 a_4)^* \cup (a_5 a_6)^* a_7]^* \tag{5.1}$$

is linear. One can linearize a rational expression by replacing each occurrence of a letter by a distinct symbol. For instance, the expression (5.1) is a linearization of the expression $e = [ab(ba)^* \cup (ac)^* b]^*$. Now, given an automaton for e', the linearization of e, it is easy to obtain an automaton for e, simply by replacing the letters of e' by the corresponding letters in e. For instance, starting from the automaton $\mathcal{A}$ which recognizes $[(a_1 a_2)^* a_3]^*$, one gets a nondeterministic automaton $\mathcal{A}'$ which recognizes $[(ab)^* a]^*$ by replacing a_1 and a_3 by a and a_2 by b, as shown in Figure 21.



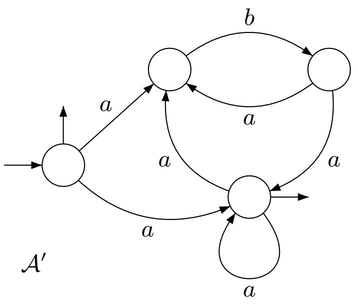


Figure 21. Construction of an automaton recognizing $[(ab)^*a]^*$.

It remains to find an algorithm to compute the automaton of a linear expression.

Proposition 5.5. *Every linear expression represents a local language.*

Proof. The proof works by induction on the formation rules of a linear expression. First, the languages represented by 0, 1 and a, for $a \in A$, are local languages. Next, by Proposition 5.4, if e represents a local language, then so does e^*. Let now e and e' be two linear expressions and suppose that the expression $(e \cup e')$ is still linear. Let B $[B']$ be the set of letters occurring in e $[e']$. Since $(e \cup e')$ is linear, the letters of B $[B']$ do not occur in e' $[e]$. In other words, B and B' are disjoint and the local language represented by e $[e']$ is contained in B^* $[B'^*]$. By Proposition 5.3, the language represented by $(e \cup e')$ is also a local language. A similar argument applies for the language represented by ee'. $\square$

Proposition 5.1 allows one to compute a deterministic automaton recognizing a local language. It suffices to test whether the empty word belongs to L and to compute the sets

$$\begin{aligned}
P(L) &= \{a \in A \mid aA^* \cap L \neq \emptyset\},\\
S(L) &= \{a \in A \mid A^*a \cap L \neq \emptyset\},\\
F(L) &= \{x \in A^2 \mid A^*xA^* \cap L \neq \emptyset\}.
\end{aligned}$$

This can be done by recursion, given a linear rational expression representing the language. We first compute the procedure

EmptyWord(e: linear expression): **boolean**;

which tells whether the empty word belongs to the language represented by e.

```
EmptyWord$(0)$ = **false**;
EmptyWord$(1)$ = **true**;
EmptyWord$(a)$ = **false** for all $a \in A$;
EmptyWord$(e \cup e')$ = EmptyWord($e$) **or** EmptyWord($e'$);
EmptyWord$(e \cdot e')$ = EmptyWord($e$) **and** EmptyWord($e'$);
EmptyWord$(e^*)$ = **true**;
```

Now P, S and F are computed by the following recursive procedures:

```
$\mathrm{P}(0) = \emptyset$;
$\mathrm{P}(1) = \emptyset$;
$\mathrm{P}(a) = \{a\}$ for all $a \in A$;
$\mathrm{P}(e \cup e') = \mathrm{P}(e) \cup \mathrm{P}(e')$;
if EmptyWord$(e)$
  then $\mathrm{P}(e \cdot e') = \mathrm{P}(e) \cup \mathrm{P}(e')$
  else $\mathrm{P}(e \cdot e') = \mathrm{P}(e)$;
$\mathrm{P}(e^*) = \mathrm{P}(e)$;
```

```
$\mathrm{S}(0) = \emptyset$;
$\mathrm{S}(1) = \emptyset$;
$\mathrm{S}(a) = \{a\}$ for all $a \in A$;
$\mathrm{S}(e \cup e') = \mathrm{S}(e) \cup \mathrm{S}(e')$;
if EmptyWord$(e')$
  then $\mathrm{S}(e \cdot e') = \mathrm{S}(e) \cup \mathrm{S}(e')$
  else $\mathrm{S}(e \cdot e') = \mathrm{S}(e')$;
$\mathrm{S}(e^*) = \mathrm{S}(e)$;
```

```
$\mathrm{F}(0) = \emptyset$;
$\mathrm{F}(1) = \emptyset$;
$\mathrm{F}(a) = \emptyset$ for all $a \in A$;
$\mathrm{F}(e \cup e') = \mathrm{F}(e) \cup \mathrm{F}(e')$;
$\mathrm{F}(e \cdot e') = \mathrm{F}(e) \cup \mathrm{F}(e') \cup \mathrm{S}(e)\mathrm{P}(e')$;
$\mathrm{F}(e^*) = \mathrm{F}(e) \cup \mathrm{S}(e)\mathrm{P}(e)$;
```

In summary, Glushkov's algorithm to convert a rational expression e into a nondeterministic automaton works as follows:

(1) Linearize e into e' and memorize the coding of the letters.

(2) Compute recursively the sets $P(e')$, $S(e')$ and $F(e')$. Then compute a deterministic automaton $\mathcal{A}'$ recognizing e'.

(3) Convert $\mathcal{A}'$ into a nondeterministic automaton $\mathcal{A}$ recognizing e.

Example 5.2. Consider the rational expression $e = (a(ab)^*)^* \cup (ba)^*$. We first linearize e into $e' = (a_1(a_2a_3)^*)^* \cup (a_4a_5)^*$. Let $L = L(e)$ and $L' = L(e')$. To compute the sets P, S and F, one can either use the above-mentioned recursive procedures, or proceed to a direct computation (this method is usually preferred in a computation by hand...). Recall that P [S] is the set of first [last] letters of the words of L'. We get

$$P = \{a_1, a_4\} \quad \text{and} \quad S = \{a_1, a_3, a_5\}.$$

Note that a_1 belongs to S since a_1 is a word of L'.

Next we compute the set F of all words of length 2 that are factors of some word of L'. We get $F = \{a_1a_2, a_1a_1, a_2a_3, a_3a_1, a_3a_2, a_4a_5, a_5a_4\}$. For instance, a_3a_1 is a factor of $a_1a_2a_3a_1$ and a_3a_2 is a factor of $a_1a_2a_3a_2a_3$. Since the empty word belongs to L', the state 1 is final and we finally obtain the automaton represented in Figure 22. Since this automaton is local, there is actually no need to write the labels on the transitions. We now convert this automaton into a nondeterministic automaton recognizing L, represented in Figure 23.

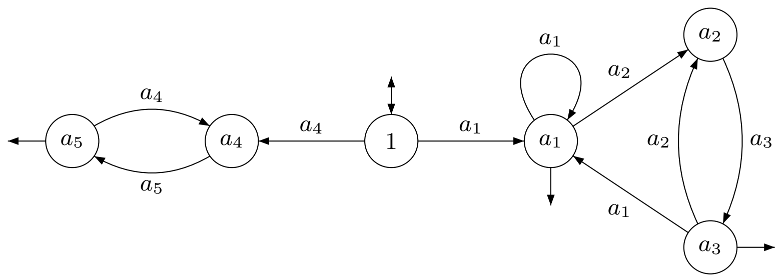


Figure 22. A local automaton recognizing L'.

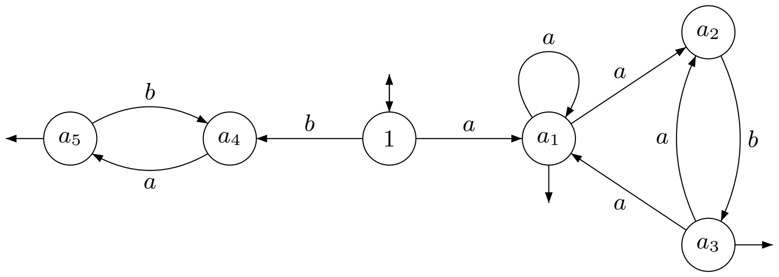


Figure 23. A nondeterministic automaton recognizing L.

To get a deterministic automaton, it remains to apply the algorithm described in Section 2.3.

5.3 Linear equations

In this section, we give an algorithm to convert an automaton into a rational expression. The algorithm amounts to solving a system of linear equations on languages. We first consider an equation of the form

$$X = KX + L, \tag{5.2}$$

where K and L are languages and X is the unknown. When K does not contain the empty word, the equation admits a unique solution.

Proposition 5.6. *If K does not contain the empty word, then $X = K^*L$ is the unique solution of the equation $X = KX + L$.*

Proof. Replacing X by K^*L in the expression $KX + L$, one gets

$$K(K^*L) + L = K^+L + L = (K^+ + 1)L = K^*L,$$

and hence $X = K^*L$ is a solution of (5.2). To prove uniqueness, consider two solutions X_1 and X_2 of (5.2). By symmetry, it suffices to show that each word u of X_1 also belongs to X_2. Let us prove this result by induction on the length of u.

If $|u| = 0$, u is the empty word and if $u \in X_1 = KX_1 + L$, then necessarily $u \in L$ since $1 \notin K$. But in this case, $u \in KX_2 + L = X_2$. For the induction step, consider a word u of X_1 of length $n + 1$. Since $X_1 = KX_1 + L$, u belongs either to L or to KX_1. If $u \in L$, then $u \in KX_2 + L = X_2$. If $u \in KX_1$ then $u = kx$ for some $k \in K$ and $x \in X_1$. Since k is not the empty word, one has necessarily $|x| \leqslant n$ and hence by induction $x \in X_2$. It follows that $u \in KX_2$ and finally $u \in X_2$. This concludes the induction and the proof of the proposition. $\square$

If K contains the empty word, uniqueness is lost.

Proposition 5.7. *If K contains the empty word, the solutions of (5.2) are the languages of the form K^*M with $L \subseteq M$.*

Proof. Since K contains the empty word, one has $K^+ = K^*$. If $L \subseteq M$, one has $L \subseteq M \subseteq K^*M$. It follows that the language K^*M is solution of (5.2) since

$$K(K^*M) + L = K^+M + L = K^*M + L = K^*M.$$

Conversely, let X be a solution of (5.2). Then $L \subseteq X$ and $KX \subseteq X$. Consequently, $K^2X \subseteq KX \subseteq X$ and by induction, $K^nX \subseteq X$ for all n. It follows that $K^*X = \sum_{n \geqslant 0} X^nK \subseteq X$. The language X can thus be written as K^*M with $L \subseteq M$: it suffices to take $M = X$. □

In particular, if K contains the empty word, then A^* is the maximal solution of (5.2) and the minimal solution is K^*L.

Consider now a system of the form

$$\begin{aligned} X_1 &= K_{1,1}X_1 + K_{1,2}X_2 + \cdots + K_{1,n}X_n + L_1 \\ X_2 &= K_{2,1}X_1 + K_{2,2}X_2 + \cdots + K_{2,n}X_n + L_2 \\ &\vdots \qquad\qquad\qquad \vdots \\ X_n &= K_{n,1}X_1 + K_{n,2}X_2 + \cdots + K_{n,n}X_n + L_n. \end{aligned} \tag{5.3}$$

We shall only consider the case when the system admits a unique solution.

Proposition 5.8. *If, for $1 \leqslant i, j \leqslant n$, the languages $K_{i,j}$ do not contain the empty word, the system (5.3) admits a unique solution. Moreover, if the $K_{i,j}$ and the L_i are rational languages, then the solutions X_i of (5.3) are rational languages.*

Proof. The case $n = 1$ is handled by Proposition 5.6. Suppose that $n > 1$. Consider the last equation of the system (5.3), which can be written

$$X_n = K_{n,n}X_n + (K_{n,1}X_1 + \ldots + K_{n,n-1}X_{n-1} + L_n).$$

According to Proposition 5.6, the unique solution of this equation is

$$X_n = K^*_{n,n}(K_{n,1}X_1 + \ldots + K_{n,n-1}X_{n-1} + L_n).$$

Replacing X_n by this expression in the $n - 1$ first equations, we obtain a system of $n - 1$ equations with $n - 1$ unknowns and one can conclude by induction. □

We shall now associate a system of linear equations with every finite automaton $\mathcal{A} = (Q, A, E, I, F)$. Let us set, for $p, q \in Q$,

$$K_{p,q} = \{a \in A \mid (p, a, q) \in E\}$$

$$L_q = \begin{cases} 1 & \text{if } q \in F, \\ 0 & \text{if } q \notin F. \end{cases}$$

The solutions of the system defined by these parameters are the languages recognized by the automata

$$\mathcal{A}_q = (Q, A, E, \{q\}, F).$$

More precisely, we get the following result:

Proposition 5.9. *The system (5.3) admits a unique solution $(R_q)_{q \in Q}$, given by the formula*

$$R_q = \{u \in A^* \mid \text{there is a path of label } u \text{ from } q \text{ to } F\}$$

Furthermore, the language recognized by $\mathcal{A}$ is $\sum_{q \in I} R_q$.

Proof. Since the languages $K_{p,q}$ do not contain the empty word, Proposition 5.8 shows that the system (5.3) admits a unique solution. It remains to verify that the family $(R_q)_{q \in Q}$ is solution of the system, that is, satisfies for all $q \in Q$ the formula

$$R_q = K_{q,1}R_1 + K_{q,2}R_2 + \cdots + K_{q,n}R_n + L_q. \tag{5.4}$$

Let us denote by S_q the right hand side of (5.4). If $u \in R_q$, then u is by definition the label of a path from q to a final state f. If u is the empty word, one has necessarily $q = f$ and hence $L_q = 1$. Thus $u \in S_q$ in this case. Otherwise, let (q, a, q') be the first transition of the path. One has $u = au'$, where u' is the label of a path from q' to f. Then one has $a \in K_{q,q'}$, $u' \in R_{q'}$ and finally $u \in S_q$.

Conversely, let $u \in S_q$. If $u = 1$, one has necessarily $u \in L_q$, whence $q \in F$ and $u \in R_q$. Otherwise there is a state q' such that $u \in K_{q,q'}R_{q'}$. Therefore, $u = au'$ for some $a \in K_{q,q'}$ and $u' \in R_{q'}$. On the one hand, (q, a, q') is a transition of $\mathcal{A}$ by definition of $K_{q,q'}$ and on the other hand u' is the label of a final path starting in q'. The composition of these paths gives a final path of label u starting in q. Therefore $u \in R_q$ and thus $R_q = S_q$. □

For example, if $\mathcal{A}$ is the automaton represented in Figure 24,

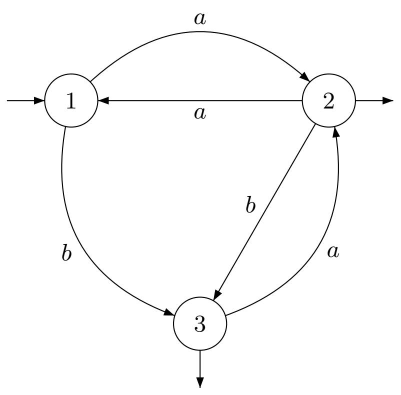


Figure 24. An automaton.

The system can be written

$$\begin{aligned} X_1 &= aX_2 + bX_3 \\ X_2 &= aX_1 + bX_3 + 1 \\ X_3 &= aX_2 + 1 \end{aligned}$$

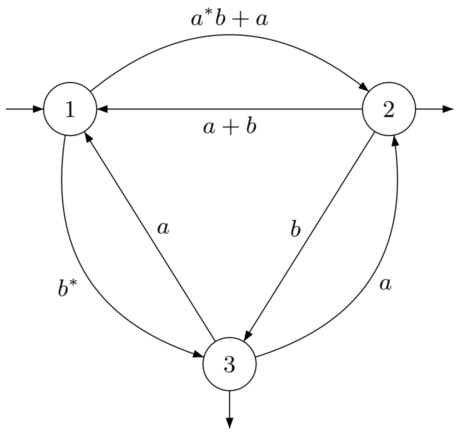


Figure 25. An extended automaton.

and has for solution

$$\begin{aligned}X_1 &= (a+ba)(aa+aba+ba)^*(ab+b+1)+b\\ X_2 &= (aa+aba+ba)^*(ab+b+1)\\ X_3 &= a(aa+aba+ba)^*(ab+b+1)+1.\end{aligned}$$

Since 1 is the unique initial state, the language recognized by the automaton is X_1.

5.4 Extended automata

The use of equations is not limited to deterministic automata. The same technique applies to nondeterministic automata and to more powerful automata, in which the transition labels are not letters, but rational languages.

An *extended automaton* is a quintuple $\mathcal{A} = (Q, A, E, I, F)$, where Q is a set of states, A is an alphabet, E is a subset of $Q \times \mathrm{Rat}(A^*) \times Q$, called the set of transitions, I $[F]$ is the set of initial [final] states. The label of a path

$$c = (q_0, L_1, q_1), (q_1, L_2, q_2), \ldots, (q_{n-1}, L_n, q_n)$$

is the rational language $L_1 L_2 \cdots L_n$. The definition of a successful path is unchanged. A word is accepted by $\mathcal{A}$ if *it belongs to* the label of a successful path. In the example represented in Figure 25, the set of transitions is

$$\{(1, a^*b+a, 2), (1, b^*, 3), (2, a+b, 1), (2, b, 3), (3, a, 1), (3, a, 2)\}.$$

Let $\mathcal{A} = (Q, A, E, I, F)$ be an extended automaton. For all $p, q \in Q$, we let $K_{p,q}$ denote the label of the transition from p to q. Notice that $K_{p,q}$ might possibly be the empty language. We also put

$$L_q = \begin{cases} 1 & \text{if there is a path labelled by } 1 \text{ from } q \text{ to } F\\ 0 & \text{otherwise.}\end{cases}$$

Yet the associated system does not necessarily fulfil the condition $1 \notin K_{i,j}$ and Proposition 5.9 needs to be modified as follows:

Proposition 5.10. *The system (5.3) has a minimal solution $(R_q)_{q \in Q}$, given by the formula*

$$R_q = \{u \in A^* \mid \text{ there is a path labelled by } u \text{ from } q \text{ to } F\}$$

In particular the language recognized by $\mathcal{A}$ is $\sum_{q \in I} R_q$.

Proof. Let us first verify that the family $(R_q)_{q \in Q}$ is indeed a solution of (5.3), i.e., satisfies, for all $q \in Q$:

$$R_q = K_{q,1}R_1 + K_{q,2}R_2 + \cdots + K_{q,n}R_n + L_q. \tag{5.5}$$

Denote by S_q the right hand side of (5.5). If $u \in R_q$, then u is by definition the label of a path from q to F. If $u = 1$, one has $L_q = 1$ and thus $u \in S_q$. Otherwise, let (q, u_1, q') be the first transition of the path. One has $u = u_1u'$, where u' is the label of a path from q' to F. Therefore $u_1 \in K_{q,q'}$, $u' \in R_{q'}$ and finally $u \in S_q$.

Conversely, let $u \in S_q$. If $u = 1$, one has necessarily $u \in L_q$, whence $q \in F$ and $u \in R_q$. Otherwise, there is a state q' such that $u \in K_{q,q'}R_{q'}$. Thus $u = u_1u'$ for some $u_1 \in K_{q,q'}$ and $u' \in R_{q'}$. On the one hand, (q, u_1, q') is a transition of $\mathcal{A}$ by the definition of $K_{q,q'}$ and on the other hand, u' is the label of a path from q' to F. Therefore $u = u_1u'$ is the label of a path from q to F and $u \in R_q$. Consequently $R_q = S_q$.

It remains to verify that if $(X_q)_{q \in Q}$ is a solution of the system, then $R_q \subseteq X_q$ for all $q \in Q$. If $u \in R_q$, there exists a path labelled by u from q to F:

$$(q_0, u_1, q_1)(q_1, u_2, q_2) \cdots (q_{r-1}, u_r, q_r)$$

with $q_0 = q$, $q_r \in F$, $u_i \in K_{q_{i-1},q_i}$ and $u_1u_2 \cdots u_r = u$. Let us show by induction on $r - i$ that $u_{i+1} \cdots u_r$ belongs to X_{q_i}. By hypothesis, the X_q are solutions of

$$X_q = K_{q,1}X_1 + K_{q,2}X_2 + \cdots + K_{q,n}X_n + L_q.$$

In particular, since $q_r \in F$, one has $1 \in L_{q_r}$ and hence $1 \in X_{q_r}$, which gives the result for $r - i = 0$. Moreover, if $u_{i+1} \cdots u_r$ is an element of X_{q_i}, the inclusion $K_{q_{i-1},q_i}X_{q_i} \subseteq X_{q_{i-1}}$ shows that $u_iu_{i+1} \cdots u_r$ is an element of $X_{q_{i-1}}$, which concludes the induction. In particular, $u = u_1 \cdots u_r \in X_q$. □

Example 5.3. For the extended automaton represented in Figure 25, the system can be written

$$\begin{aligned} X_1 &= (a^*b + a)X_2 + b^*X_3 \\ X_2 &= (a + b)X_1 + bX_3 + 1 \\ X_3 &= aX_1 + aX_2 + 1. \end{aligned}$$

Replacing X_3 by $aX_1 + aX_2 + 1$, and observing that $a + b^*a = b^*a$, we obtain the equivalent system

$$\begin{aligned} X_1 &= (a^*b + a)X_2 + b^*(aX_1 + aX_2 + 1) = b^*aX_1 + (a^*b + b^*a)X_2 + b^* \\ X_2 &= (a + b)X_1 + b(aX_1 + aX_2 + 1) + 1 = (a + b + ba)X_1 + baX_2 + b + 1 \\ X_3 &= aX_1 + aX_2 + 1. \end{aligned}$$

We deduce from the second equation

$$X_2 = (ba)^*((a+b+ba)X_1 + b + 1),$$

and replacing X_2 by its value in the first equation, we obtain

$$\begin{aligned} X_1 &= b^*aX_1 + (a^*b + b^*a)(ba)^*((a+b+ba)X_1 + b + 1) + b^* \\ &= (b^*a + (a^*b+b^*a)(ba)^*(a+b+ba))X_1 + (a^*b+b^*a)(ba)^*(b+1) + b^*. \end{aligned}$$

Finally, the language recognized by the automaton is

$$X_1 = \big(b^*a + (a^*b+b^*a)(ba)^*(a+b+ba)\big)^*[(a^*b+b^*a)(ba)^*(b+1) + b^*],$$

since 1 is the unique initial state.

5.5 Kleene's theorem

We are now ready to state the most important result of automata theory.

Theorem 5.11 (Kleene [4]). *A language is rational if and only if it is recognizable.*

Proof. It follows from Proposition 5.9 that every recognizable language is rational. Corollary 3.2 states that every finite language is recognizable. Furthermore, Propositions 3.1, 3.5 and 3.6 show that recognizable languages are closed under union, product and star. Thus every rational language is recognizable. □

The following corollary is now a consequence of Propositions 2.1, 3.1, 3.3, 3.4, 3.5, 3.6, 3.8 and 3.9.

Corollary 5.12. *Recognizable [rational] languages are closed under Boolean operations, product, star, quotients, morphisms and inverses of morphisms.*

We conclude this section by proving some elementary decidability results on recognizable languages. Recall that a property is *decidable* if there is an algorithm to check whether this property holds or not. We shall also often use the expressions "given a recognizable language L" or "given a rational language L". As long as only decidability is concerned, it makes no difference to give a language by a nondeterministic automaton, a deterministic automaton or a regular expression, since there are algorithms to convert one of the forms into the other. However, the chosen representation is important for complexity issues, which will not be discussed here.

Theorem 5.13. *Given a recognizable language L, the following properties are decidable:*

(1) *whether a given word belongs to L,*
(2) *whether L is empty,*
(3) *whether L is finite,*
(4) *whether L is infinite.*

Proof. We may assume that L is given by a trim deterministic automaton $\mathcal{A} = (Q, A, \cdot, q_-, F)$.

(1) To test whether $u \in L$, it suffices to compute $q_- \cdot u$. If $q_- \cdot u \in F$, then $u \in L$; if $q_- \cdot u \notin F$, or if $q_- \cdot u$ is undefined, then $u \notin L$.

(2) Let us show that L is empty if and only if $F = \emptyset$. The condition $F = \emptyset$ is clearly sufficient. Since $\mathcal{A}$ is trim, every state of $\mathcal{A}$ is accessible. Now, if $\mathcal{A}$ has at least one final state q, there is a word u such that $q_- \cdot u = q$. Therefore $u \in L$ and L is nonempty.

(3) and (4). Let us show that L is finite if and only if $\mathcal{A}$ does not contain any loop. If $\mathcal{A}$ contains a loop $q \xrightarrow{u} q$, then L is infinite: indeed, since $\mathcal{A}$ is trim, there exist paths $i \xrightarrow{x} q$ and $q \xrightarrow{y} f$, where f is a final state and thus L contains all the words xu^ny. Conversely, if L is infinite, the proof of the pumping lemma shows that $\mathcal{A}$ contains a loop. Now, checking whether an automaton contains a loop is easy. Consider the directed graph G obtained from $\mathcal{A}$ by removing all the labels. Then $\mathcal{A}$ is loop-free if and only if G is acyclic, a property that can be checked by standard algorithms. One can for instance compute the transitive closure G' of G and check whether G' contains an edge of the form (q, q). □

We leave as an exercise to the reader to prove that the inclusion problem and the equality problem are decidable for two given recognizable languages.

6 Algebraic approach

The notions of rational and recognizable sets can be defined in arbitrary monoids. However, Kleene's theorem does not extend to arbitrary monoids since rational and recognizable sets form in general two incomparable classes.

6.1 Rational subsets of a monoid

Let M be a monoid. The set $\mathcal{P}(M)$ of subsets of M is a semiring with union as addition and product defined by the formula

$$XY = \{xy \mid x \in X \quad \text{and} \quad y \in Y\}.$$

For this reason, we shall adopt the notation we already introduced for languages. Union is denoted by $+$, the empty set by 0 and the singleton $\{m\}$, for $m \in M$ by m. This notation has the advantage that the identity of $\mathcal{P}(M)$ is denoted by 1.

The powers of a subset X of M are defined by induction by setting $X^0 = 1$, $X^1 = X$ and $X^n = X^{n-1}X$ for all $n > 1$. The *star* operation is defined by

$$X^* = \sum_{n \geqslant 0} X^n = 1 + X + X^2 + X^3 + \cdots .$$

In other words, X^* is the submonoid of M generated by X. The set of rational subsets of a monoid M is the smallest set $\mathcal{F}$ of subsets of M satisfying the following conditions:

(1) $\mathcal{F}$ contains 0 and the singletons of $\mathcal{P}(M)$,
(2) $\mathcal{F}$ is closed under union, product and star (in other words, if $X, Y \in \mathcal{F}$, then $X + Y \in \mathcal{F}$, $XY \in \mathcal{F}$ and $X^* \in \mathcal{F}$).

For instance, in a finite monoid, all subsets are rational. The rational subsets of $\mathbb{N}^k$ are the *semilinear* sets, which are finite unions of subsets of the form

$$\{v_0 + n_1 v_1 + \cdots + n_r v_r \mid n_1, \ldots, n_r \in \mathbb{N}\},$$

where $v_0, v_1, \ldots, v_r$ are vectors of $\mathbb{N}^k$.

Rational subsets are also stable under morphisms.

Proposition 6.1. *Let $\varphi : M \to N$ be a monoid morphism. If R is a rational subset of M, then $\varphi(R)$ is a rational subset of N. Moreover, if φ is surjective, then for each rational subset S of N, there exists a rational subset R of M such that $\varphi(R) = S$.*

However, the rational subsets of a monoid are not necessarily closed under intersection, as shown by the following counterexample: Let $M = a^* \times \{b, c\}^*$. Consider the rational subsets

$$(a, b)^*(1, c)^* = \{(a^n, b^n c^m) \mid n, m \geqslant 0\}$$
$$(1, b)^*(a, c)^* = \{(a^n, b^m c^n) \mid n, m \geqslant 0\}.$$

Their intersection is $\{(a^n, b^n c^n) \mid n \geqslant 0\}$, a nonrational subset of M. It follows also that the complement of a rational subset is not necessarily rational. Otherwise, the rational subsets of a monoid would be closed under union and complement and hence under intersection.

Proposition 6.2. *Each rational subset of a monoid M is a rational subset of a finitely generated submonoid of M.*

6.2 Recognizable subsets of a monoid

Let $\varphi : M \to N$ be a monoid morphism. A subset L of M is *recognized* by φ if there exists a subset P of N such that

$$L = \varphi^{-1}(P).$$

If φ is surjective, we say that φ *recognizes* L. Note that in this case, the condition $L = \varphi^{-1}(P)$ implies $P = \varphi(L)$.

A subset of a monoid is *recognizable* if it is recognized by a finite monoid. We let $\mathrm{Rec}(M)$ denote the set of recognizable subsets of M.

Proposition 6.3. *For any monoid M, $\mathrm{Rec}(M)$ is closed under Boolean operations and left and right quotients. Moreover, if $\varphi : N \to M$ is a morphism, $L \in \mathrm{Rec}(M)$ implies $\varphi^{-1}(L) \in \mathrm{Rec}(N)$.*

Although Kleene's theorem does not extend to arbitrary monoids, a weaker property holds for finitely generated monoids.

Theorem 6.4 (McKnight [5]). *Let M be a monoid. The following conditions are equivalent:*

(1) *M is finitely generated,*

(2) *every recognizable subset of M is rational,*
(3) *the set M is a rational subset of M.*

We have seen that the intersection of two rational subsets is not necessarily rational. What about the intersection of a rational subset and a recognizable subset?

Proposition 6.5. *The intersection of a rational subset and of a recognizable subset of a monoid is rational.*

The next theorem gives a description of the recognizable subsets of a finite product of monoids. Eilenberg [1] attributes it to Mezei. Note that this result does not extend to finite products of semigroups.

Theorem 6.6. *Let $M_1, \ldots, M_n$ be monoids and let $M = M_1 \times \cdots \times M_n$. A subset of M is recognizable if and only if it is a finite union of subsets of the form $R_1 \times \cdots \times R_n$, where each R_i is a recognizable subset of M_i.*

One of the most important applications of Theorem 6.6 is the fact that the product of two recognizable relations over finitely generated free monoids is recognizable. Let $A_1, \ldots, A_n$ be finite alphabets. Then the monoid $A_1^* \times \cdots \times A_n^*$ is finitely generated, since it is generated by the finite set

$$\{(1, \ldots, 1, a_i, 1, \ldots, 1) \mid a_i \in A_i, 1 \leqslant i \leqslant n\}.$$

Proposition 6.7. *Let $A_1, \ldots, A_n$ be finite alphabets. The product of two recognizable subsets of $A_1^* \times \cdots \times A_n^*$ is recognizable.*

References

[1] S. Eilenberg. *Automata, languages and machines*, volume A. Academic Press, 1974. 36

[2] V. Glushkov. The abstract theory of automata. *Russ. Math. Surv.*, 16(5):1–53, 1962. 25

[3] J. E. Hopcroft and J. D. Ullman. *Introduction to automata theory, languages and computation*. Addison-Wesley Publishing Co., Reading, Mass., 1979. 17

[4] S. C. Kleene. Representation of events in nerve nets and finite automata. In C. Shannon and J. McCarthy, editors, *Automata Studies*, pages 3–42, Princeton, New Jersey, 1956. Princeton University Press. 33

[5] J. D. McKnight, Jr. Kleene quotient theorems. *Pacific J. Math.*, 14:1343–1352, 1964. 35