CHAPTER 11

RATIONAL LANGUAGES AND FINITE AUTOMATA

Objects used in computer science are always representable as strings of symbols. We examine an algebraic structure on such strings of symbols, the free monoid. We will see how particular sets of such strings of symbols can be defined. We may think of a string of symbols as a representation of a computation. Finally, we will introduce machines, called *finite automata*, that can decide whether a given string belongs to a given set. These automata are simple examples of abstract machines. The entire study of computation theory relies mainly on generalizations and extensions of these simple machines.

This chapter defines rational subsets of a free monoid. It also defines (deterministic and non-deterministic) finite automata. It introduces some basic operations on automata: direct product, sequential product, determinization, minimization; it concludes with the proof of Kleene's theorem that rational languages are exactly those languages recognized by finite automata.

We recommend the following further reading:

Garrett Birkhoff, Thomas Bartee, *Modern Applied Algebra*, McGraw Hill, New York (1970).

Michael Harrison, *Introduction to Formal Languages*, Addison-Wesley, Reading (1978).

John Hopcroft, Jeffrey Ullman, *Introduction to Automata Theory, Languages and Computation*, Addison-Wesley, Reading (1979).

Arto Salomaa, *Formal Languages*, Academic Press, New York (1973).

11.1 The free monoid

We defined monoids and free monoids in Definition 1.12 and Definition 1.15, with slightly different notations. We recall these definitions.

Definition 11.1 *A monoid $\mathcal{M} = (M, ., e)$ is a set M equipped with an associative operation, denoted by '.', that has a unit e.*

The free monoid over A, denoted by A^*, is the set of strings over the alphabet A, equipped with concatenation (see Definition 1.15). The concatenation of two strings $u = u_1u_2\cdots u_n$ and $v = v_1v_2\cdots v_m$ is the string $u \cdot v = u_1u_2\cdots u_nv_1v_2\cdots v_m$.

Exercise 11.1 Are the following objects monoids?

1. The set of mappings $E \to E$ equipped with the composition operation. (The composition $g \circ f$ of f and g is denoted by $g.f$ or gf and is defined by $(gf)(x) = g\big(f(x)\big)$ for $x \in E$.)
2. $\mathbb{N}$ equipped with the 'power' operation : $(x, y) \to x^y$.
3. The set of strings in A^* whose length is even.
4. The set of strings in A^* with as many occurrences of as as occurrences of bs.
5. The set of finite subsets of E, with the union operation.
6. The set of finite subsets of E, with the intersection operation. ♢

Exercise 11.2 Let the matrices

$$A = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \text{ and } B = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}.$$

Consider the set $\mathcal{M}_1$ consisting of the identity matrix $I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$ and the matrices obtained by products of matrices A and B. Consider also the set $\mathcal{M}_2$ of 2×2 matrices with determinant $+1$ whose coefficients are non-negative integers.

1. Show that all the determinants of the matrices in $\mathcal{M}_1$ are equal to $+1$ and that all their coefficients are non-negative integers. Hint: try a proof by induction on the number of factors A and B in the product.

This implies that $\mathcal{M}_1 \subseteq \mathcal{M}_2$; we now establish the reverse inclusion.

2. Show that a matrix with non-negative coefficients, and which is not the identity matrix, has determinant $+1$ *only if* all the coefficients of one of its columns are simultaneously greater than the coefficients of another column. More precisely, $\begin{pmatrix} x & y \\ z & w \end{pmatrix}$ $(x, y, z, w \in \mathbb{N})$ has determinant $+1$ only if either $x \geq y$ and $z \geq w$, or $x \leq y$ and $z \leq w$.
3. Compute the inverses of the matrices A and B. Use the result of 2 to show that any matrix $\begin{pmatrix} x & y \\ z & w \end{pmatrix}$ in $\mathcal{M}_2$ can be uniquely decomposed as a product of matrices A and B. (Use the complete induction principle and a recurrence on $x + y + z + w$.)
4. Deduce the equality $\mathcal{M}_1 = \mathcal{M}_2$. ♢

Let $\mathcal{M} = (M, ., e)$ and let M' be a subset of M. $\mathcal{M}' = (M', ., e)$ is a *submonoid* of $\mathcal{M}$ if it is a monoid, namely, if

- $e \in M'$ and
- $\forall m, m' \in M'$, $m.m' \in M'$.

If I is a set of indices, and if $\forall i \in I$, $\mathcal{M}_i = (M_i, ., e)$ is a submonoid of $\mathcal{M}$, then $(\bigcap_{i \in I} M_i, ., e)$ is a submonoid of $\mathcal{M}$. For a subset X of M, we can thus define the least submonoid of $\mathcal{M}$ containing X, called the submonoid of $\mathcal{M}$ generated by X, as the intersection of all the submonoids of $\mathcal{M}$ containing X.

EXAMPLE 11.2 Let $\mathcal{N} = (\mathbb{N}, +, 0)$. Let A be the set of even numbers and B be the set of odd numbers. $(A, +, 0)$ is the submonoid of $\mathcal{N}$ generated by $\{2\}$ while $(B \cup \{0\}, +, 0)$ is not a submonoid of $\mathcal{N}$ (e.g. $3 + 1$ is even).

If $\mathcal{M} = (M, ., e)$ and $\mathcal{M}' = (M', \times, e')$ are two monoids, a mapping h from M in M' is a *monoid homomorphism* if it satisfies

- $h(e) = e'$ and
- $\forall m, m' \in M$, $h(m.m') = h(m) \times h(m')$.

EXAMPLE 11.3 The mapping associating with each string u its length $|u|$ is a homomorphism from $(A^*, ., \varepsilon)$ to $(\mathbb{N}, +, 0)$.

The mapping associating with each $n \in \mathbb{N}$ the number 2^n is a homomorphism from $(\mathbb{N}, +, 0)$ to $(\mathbb{N}, \times, 1)$, because $2^0 = 1$ and $2^{n+m} = 2^n \times 2^m$.

EXERCISE 11.3 Let M_1 be the monoid defined in Exercise 11.2. Construct a monoid homomorphism $\{a, b\}^* \longrightarrow M_1$. Show that this homomorphism is both injective and surjective. ♢

Theorem 11.4 *Let A be an alphabet. Let $(M, ., e)$ be a monoid and let h be a mapping from A to M. There exists a unique homomorphism $h^*: A^* \longrightarrow M$ such that $\forall a \in A$, $h^*(a) = h(a)$.*

Proof. Existence: Let $h^*(\varepsilon) = e$ and $h^*(a_1 \cdots a_n) = h(a_1).\cdots.h(a_n)$. It is easy to see that h^* indeed is a homomorphism.
Uniqueness: Let g and g' be two homomorphisms from A^* in M such that $\forall a \in A$, $g(a) = g'(a)$. Then $g(\varepsilon) = g'(\varepsilon) = e$, and for any string $u = a_1 \cdots a_n$,

$$g(u) = g(a_1).\cdots.g(a_n) = g'(a_1).\cdots.g'(a_n) = g'(u). \qquad \square$$

Inspired by this theorem, we call the monoid A^* the *free monoid generated by A*.

EXERCISE 11.4 Let A be an alphabet containing at least the letters a and b.
1. Give a homomorphism from A^* to $(A \cdot A)^*$.
2. Give a homomorphism from A^* to $(A \setminus \{b\})^*$.
3. Is there a monoid isomorphism (i.e. a bijective homomorphism whose inverse is a homomorphism) from $(A \setminus \{a\})^*$ to $(A \setminus \{b\})^*$? ♢

EXERCISE 11.5 Levi's lemma: Let $u, v, x, y \in A^*$. Show that $uv = xy$ if and only if exactly one of the three following cases holds:
(i) $|u| = |x|$, $u = x$, and $v = y$.
(ii) $|u| < |x|$ and there exists a $t \in A^*$ such that $ut = x$ and $v = ty$.
(iii) $|u| > |x|$ and there exists a $t \in A^*$ such that $u = xt$ and $tv = y$. ♢

EXERCISE 11.6 Equations on strings: Let $u, v \in A^*$. Show the following equivalences:
1. $uv = vu$ if and only if there exists a $w \in A^*$ and $m, n \in \mathbb{N}$ such that $u = w^m$ and $v = w^n$.
2. $u^p = v^q$ (with $p, q \in \mathbb{N}$) if and only if there exists a $w \in A^*$ and $m, n \in \mathbb{N}$ such that $u = w^m$, $v = w^n$.
3. $uu = u$ if and only if $u = \varepsilon$.
4. $ua = au$ if and only if there exists a $p \in \mathbb{N}$ such that $u = a^p$.
5. $ua = bu$ (with $a \neq b$) if and only if $0 \neq 0$. ♢

11.2 Regular languages

In computer science we often define sets of strings of symbols, or subsets of a free monoid A^*, called languages. There are many ways of defining languages. Here we describe one such way, which also gives a simple way of determining whether a given string u is in the language or not.

We first define some operations on $\mathcal{P}(A^*)$:

- The *union*, denoted by $\cup$ or $+$: if L and L' are two subsets of A^*,
$$L + L' = L \cup L' = \{u \,/\, u \in L \text{ or } u \in L'\}.$$
- The *product*, denoted by $\cdot$: if L and L' are two subsets of A^*,
$$L \cdot L' = \{uv \,/\, u \in L, v \in L'\}.$$
- The *iteration*, denoted by * : if L is a subset of A^*, let
$$\begin{aligned} L^0 &= \{\varepsilon\}, \\ L^1 &= L, \\ &\vdots \\ L^{i+1} &= L \cdot L^i, \\ &\vdots \end{aligned}$$
$$\text{and} \qquad L^* = \bigcup_{i \geq 0} L^i.$$

We will denote by L^+ the set $\bigcup_{i>0} L^i$. We thus have

$$L^+ = L \cdot L^* = L^* \cdot L \text{ and } L^* = \{\varepsilon\} + L^+.$$

EXERCISE 11.7
1. Show that $(\mathcal{P}(A^*), \cdot, \{\varepsilon\})$ is a monoid.
2. Show that if $(L_i)_{i\in I}$ is any family of languages, then

$$\left(\bigcup_{i\in I} L_i\right) \cdot L = \bigcup_{i\in I} L_i \cdot L.$$

3. Show that $L^* = (L + \{\varepsilon\})^*$ and that $L^* = \{\varepsilon\} + L \cdot L^*$.
4. Show that $\emptyset^* = \{\varepsilon\}$. ♢

Definition 11.5 *A subset of A^* is said to be a regular set (over A) if it can be built up from the finite subsets of A^* by using the three operations $\cup, \cdot$, and *. We denote by $Rat(A^*)$ the regular sets over A and by $Fin(A^*)$ the set of finite subsets of A^*.*

In order to specify a regular set over A it is enough to describe a procedure to construct it from finite subsets by the three operations given above. We do this by writing a *regular expression*. Since each non-empty finite subset is a finite union of singleton sets, and since any subset consisting of a single non-empty string is obtained by taking a product of sets consisting of a single one-letter string, the regular subsets can also be obtained by taking the closures under product, union and iteration of the sets consisting of a single one-letter string and of the empty set.

Definition 11.6 *We formally define the regular expressions as strings of symbols:*

- *$\emptyset$ is a regular expression.*
- *If a is a letter, a is a regular expression.*
- *If E is a regular expression , E^* is a regular expression.*
- *If E_1 and E_2 are regular expressions, $(E_1 + E_2)$ and (E_1E_2) are regular expressions.*

The set $I(E)$ specified by the regular expression E is defined by

$$\begin{aligned} I(\emptyset) &= \emptyset, \\ I(a) &= \{a\}, \\ I(E^*) &= I(E)^*, \\ I(E_1E_2) &= I(E_1) \cdot I(E_2) \quad \text{and} \\ I(E_1 + E_2) &= I(E_1) + I(E_2). \end{aligned}$$

Note that the definition of I is an inductive definition (see Definition 3.20). The rules we gave are very strict. For instance, $(a+b+c)\cdot d$ is disallowed unless it is written in the form $(((a+b)+c)\cdot d)$ or $((a+(b+c))\cdot d)$. (Both these expressions denote the same set $\{ad, bd, cd\}$.) Usually, when there is no ambiguity we allow ourselves more flexibility in denoting regular expressions, in a similar manner to the way algebraic expressions are denoted in mathematics. However, if these regular expressions are used in computer systems, we must abide strictly by the rules of the system, which might possibly differ from the rules given above. It is exactly the same situation as for arithmetic expressions, whose denotations in programming languages are also very strictly regulated.

EXERCISE 11.8 Let $X = \{b\}$ and $Y = (A \setminus \{b\}) \cdot \{b\}^*$.

1. Informally describe the elements of X^*, Y, and Y^*.
2. Show that any string of A^* starting with a letter different from b is in Y^*.
3. Show that any string u of A^* can be uniquely written in the form $u = vw$, where $v \in X^*$ and $w \in Y^*$. ♢

11.3 Finite automata

11.3.1 Labelled transition systems

Definition 11.7 *A labelled transition system over an alphabet A is a quintuple $(S, T, \alpha, \beta, \lambda)$ where*

- *(S, T, α, β) is a directed graph ($\alpha: T \longrightarrow S$ (resp. $\beta: T \longrightarrow S$) is called the source (resp. target) mapping): the elements of S are called states instead of vertices, and the elements of T are called transitions instead of edges.*
- *λ is a mapping from T to A: $\lambda(t)$ is called the label of the transition t. The triple (α, λ, β) must define an injective mapping from T to $S \times A \times S$: there can be no two different transitions with the same source, the same target and the same label. This reduces to requiring T to be a subset of $S \times A \times S$ and allows us to denote a transition system more simply by (S, T).*

A *path* is a non-empty sequence $c = t_1 \cdots t_n$ of transitions such that $\beta(t_i) = \alpha(t_{i+1})$ for $1 \leq i \leq n-1$. The source $\alpha(c)$ of path c is $\alpha(t_1)$; the target $\beta(c)$ of path c is $\beta(t_n)$. A path is also called a *computation* of the transition system.

Consider also the empty paths: to each state s is associated an empty path ε_s, whose source and target are s. We might also have considered empty paths in graphs.

If c is a path, the *trace* of this path is the string $\lambda(c)$ of A^* defined by

- $\lambda(\varepsilon_s) = \varepsilon$, $\forall s \in S$,
- $\lambda(t_1 \cdots t_n) = \lambda(t_1) \cdots \lambda(t_n)$.

A labelled transition system is said to be *deterministic* if $\forall s \in S$, $\forall a \in A$, there exists at most one transition t with source s and label a. A labelled transition system is said to be *complete* if $\forall s \in S$, $\forall a \in A$, there exists at least one transition t with source s and label a.

Proposition 11.8 *If a transition system is deterministic, then $\forall u \in A^*$, $\forall s \in S$, there exists at most one path c with source s such that $\lambda(c) = u$. If a transition system is complete, there exists at least one such path.*

Proof. The proof proceeds by induction on the length of u.

If $u = \varepsilon$ then the unique path with source s and with trace ε is the empty path ε_s.

Let $u = av$ and let s be a given state. Consider the case when the transition system is deterministic and assume that there exist two paths c_1 and c_2 with source s. The first transition of both paths is the unique transition $t = (s, a, s')$ with source s and label a. We thus have $c_1 = tc'_1$ and $c_2 = tc'_2$. The paths c'_1 and c'_2 are then two distinct paths with trace v and with source s', and this is impossible by the induction hypothesis. If the transition system is complete, there exists at least one transition $t = (s, a, s')$ and at least one path c with source s' and with trace v. The path tc is then a path with source s and with trace av. □

If s and s' are two states of S, $L_{s,s'}$ is the set of traces of the paths with source s and with target s':

$$L_{s,s'} = \{\lambda(c) \;/\; \alpha(c) = s \text{ and } \beta(c) = s'\}.$$

EXERCISE 11.9 Show that $\varepsilon \in L_{s,s'}$ if and only if $s = s'$. ♢

Let Q and Q' be two subsets of S. $L_{Q,Q'} = \bigcup_{s \in Q, s' \in Q'} L_{s,s'}$.

Definition 11.9 *A finite-state automaton is an eight-tuple $(A, S, T, \alpha, \beta, \lambda, I, F)$, where*

- *$(S, T, \alpha, \beta, \lambda)$ is a labelled transition system with label alphabet A, and S is a finite set,*
- *I and F are two subsets of S whose elements are called initial states and final states respectively.*

When no ambiguity can occur, a finite-state automaton will simply be denoted by (S, T, I, F).

A finite-state automaton is said to be *complete* if (S, T) is a complete transition system. A finite-state automaton is said to be *deterministic* if (S, T) is a deterministic transition system and if there is a single initial state.

Definition 11.10 *A language $L \subseteq A^*$ is said to be recognizable if and only if there exists a finite-state automaton $\mathcal{A} = (A, S, T, \alpha, \beta, \lambda, I, F)$ such that $L = L(\mathcal{A})$, where $L(\mathcal{A})$ is, by definition, $L_{I,F}$.*

Intuitively, to check whether a string u is in $L(\mathcal{A})$, we start from an initial state of the automaton and then read the string u using each successive letter of u, from left to right. Each time it reads a letter, the automaton changes its state by taking a transition labelled by this letter. If we can thus reach a final state when all the letters of u heve been read, then the string is recognized. Note, however, that if a finite-state automaton is non-deterministic, then the same string u can yield several different evolutions of the automaton. Some of these evolutions will lead to a final state and others will not. If there is at least one evolution that leads to a final state, then u is in $L(\mathcal{A})$.

EXAMPLE 11.11 Let $\mathcal{A}$ be the finite-state automaton with two states s_1 and s_2 whose transitions are (s_1, a, s_1), (s_1, b, s_1), and (s_1, b, s_2). See Figure 11.1. The string ab is the label of two different paths: (s_1, a, s_1, b, s_1) and (s_1, a, s_1, b, s_2). If $I = \{s_1\}$ and $F = \{s_2\}$, then $ab \in L(\mathcal{A})$.

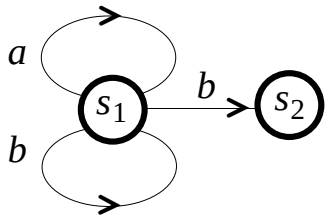

Figure 11.1

EXERCISE 11.10 Let L be a language and let $\mathrm{Pref}(L) = \{u \in A^* \,/\, \exists v \in A^* : uv \in L\}$ be the set of prefixes of the strings in this language L. Show that if a language L is recognizable then the set $\mathrm{Pref}(L)$ is also recognizable. ♢

EXERCISE 11.11 Show that any *finite* language is recognized by a deterministic finite-state automaton. (Explain how such an automaton can be constructed using the prefixes of the strings in the language.) ♢

EXERCISE 11.12 Let $\mathcal{A}$ be a finite-state automaton with a finite alphabet A. Show that the language recognized by $\mathcal{A}$ is finite if and only if the directed graph corresponding to the set of transitions of $\mathcal{A}$ has no circuit containing a vertex that is on a path going from an initial state to a final state. Give the least upper bound on the size of this language in terms of the number of states and of the size of the alphabet; also give this bound in the particular case when the finite-state automaton is deterministic and complete. Describe the finite-state automata for which this bound is reached. ♢

11.3.2 Completion of a finite-state automaton

In all cases, the demand that a finite-state automaton be complete is not a restriction on its language, because it is always possible to transform an incomplete finite-state automaton into a complete finite-state automaton without changing the language recognized.

Thus let $\mathcal{A} = (S, T, I, F)$ be a finite-state automaton assumed to be incomplete. We build a new finite-state automaton $\mathcal{A}' = (S', T', I, F)$ as follows:

- To obtain S', we add to S a new state, usually called the *sink state*, denoted here by p.
- To obtain T' we add to T
  - the transitions (s, a, p) for any state s and for any letter a such that there do not exist in T any transitions with source s and label a,
  - the transitions (p, a, p) for all the letters a.

$\mathcal{A}'$ is complete by construction. Moreover, paths existing in $\mathcal{A}'$ and which did not exist in $\mathcal{A}$ are paths ending in state p, which is not a final state, and thus $L(\mathcal{A}) = L(\mathcal{A}')$.

EXERCISE 11.13 Consider an incomplete finite-state automaton $\mathcal{A}$. Assume that we transform it into a complete finite-state automaton $\mathcal{A}'$ as follows: for all $a \in A$ and $s \in S$ such that there is no transition with source s and label a, we add a loop with label a in s, namely, the transition (s, a, s):

1. If $\mathcal{A}$ is deterministic, is $\mathcal{A}'$ also deterministic?
2. What is the relationship between $L(\mathcal{A})$ and $L(\mathcal{A}')$? ♢

11.3.3 Determinization of a finite-state automaton

The construction of a deterministic finite-state automaton from an arbitrary one is more complex than its completion, but it is also possible. Moreover, this construction is more interesting, because if $\mathcal{A}$ is a finite-state automaton then it is much easier to check whether any given string u is in $L(\mathcal{A})$ when $\mathcal{A}$ is deterministic than when it is not deterministic.

Let $\mathcal{A} = (S, T, I, F)$ be a finite-state automaton. Let us construct the deterministic finite-state automaton $\mathcal{A}_d$ recognizing the same language as $\mathcal{A}$. This automaton $\mathcal{A}_d = (S', T', \{i'\}, F')$ is defined by:

$$
\begin{aligned}
S' &= \mathcal{P}(S)\ , \\
T' &= \left\{(Q, a, Q')\ /\ Q \subseteq S\ ,\ Q' \subseteq S\ ,\ Q' = \{q'\ /\ \exists q \in Q\colon (q, a, q') \in T\}\right\}\ , \\
i' &= I\ , \\
F' &= \{Q \subseteq S\ /\ Q \cap F \neq \emptyset\}\ .
\end{aligned}
$$

Proposition 11.12 *Let c be a path in $\mathcal{A}_d$ with source Q, target Q' and trace u. Then $Q' = \{q' \,/\, \exists q \in Q$: there exists in $\mathcal{A}$ a path with source q, target q' and trace $u\}$.*

Proof. By induction on the length of u:

- If $u = \varepsilon$, then $Q = Q'$ and the result trivially holds.
- If $u = au'$, then there exists in $\mathcal{A}_d$ a path tc with $t = (Q, a, Q'')$. By the definition of $\mathcal{A}_d$, $Q'' = \{q'' \,/\, \exists q \in Q\colon (q, a, q'') \in T\}$, and, by the induction hypothesis,

$$Q' = \{q' \,/\, \exists q'' \in Q''\text{: there exists in } \mathcal{A} \text{ a path with source } q'', \\ \text{with target } q' \text{ and with trace } u'\},$$

and by combining both cases we have the result. □

Corollary 11.13 $L(\mathcal{A}) = L(\mathcal{A}_d)$.

Proof. $u \in L(\mathcal{A})$ if and only if u is the trace of a path with source $q \in I$ and with target $q' \in F$; $u \in L(\mathcal{A}_d)$ if and only if the set

$$\{q' \,/\, \exists q \in I\colon u \text{ is the trace of a path with source } q \text{ and with target } q'\}$$

contains at least an element of F. By Proposition 11.12 it is clear that these two conditions are equivalent. □

Example 11.14 We return to the finite-state automaton of Example 11.11. It is associated with the deterministic finite-state automaton $\mathcal{A}_d$ with states $\big\{\emptyset, \{s_1\}, \{s_2\}, \{s_1, s_2\}\big\}$, whose initial state is $i = \{s_1\}$, whose set of final states is $\big\{\{s_2\}, \{s_1, s_2\}\big\}$ and whose transitions are given by the following table, where the set appearing in line Q ($Q \subseteq \{s_1, s_2\}$) and column x ($x \in \{a, b\}$) is the set $\{q' \,/\, \exists q \in Q : (q, x, q') \in T\}$.

s	a	b
$\emptyset$	$\emptyset$	$\emptyset$
$\{s_1\}$	$\{s_1\}$	$\{s_1, s_2\}$
$\{s_2\}$	$\emptyset$	$\emptyset$
$\{s_1, s_2\}$	$\{s_1\}$	$\{s_1, s_2\}$

The string ab is the trace of a path from the initial state $\{s_1\}$ to the final state $\{s_1, s_2\}$. In the original finite-state automaton, there are indeed two (and exactly two) different paths with trace ab and with source s_1; these lead to s_1 and s_2 respectively (Figure 11.2).

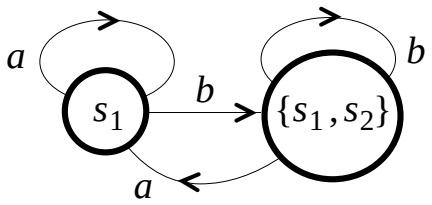

Figure 11.2

EXERCISE 11.14 Make deterministic and complete the following finite-state automaton over the alphabet $\{a, b, c\}$, where the initial state is denoted by i and the final states are denoted by f, and f'.

$$f \xleftarrow{a,c} x \xleftarrow{b} i \xleftrightarrow{b} y \xrightarrow{b,c} z \xleftrightarrow{a,c} f'. \qquad \diamondsuit$$

EXERCISE 11.15 Let $A = \{a, b, c\}$. Give a complete deterministic finite-state automaton to recognize each of the following languages:

1. the set of strings of even length,
2. the set of strings with a number of occurrences of 'b' that is divisible by 3,
3. the set of strings ending with 'b',
4. the set of strings not ending with 'b',
5. the set of non-empty strings not ending with 'b',
6. the set of strings with at least one 'b',
7. the set of strings with at most one 'b'
8. the set of strings with exactly one 'b',
9. the set of strings with no 'b' at all,
10. the set of strings with at least one 'a' and whose first 'a' is not followed by a 'c',
11. the set of strings where at least three letters occur and whose third-from-last letter is an 'a' or a 'c'. $\diamondsuit$

EXERCISE 11.16 (Iteration Lemma)

1. Show that if G is a directed graph with n vertices and c is a path of length $\geq n$ in G, then the subpath of c consisting of the first n edges of c contains a circuit.
2. Consider a transition system with n states labelled by the alphabet A, and let q, q' be two among these states. Show that if $z \in L_{q,q'}$ and $|z| \geq n$, then there exist $u, v, w \in A^*$ such that $z = uvw$, $v \neq \varepsilon$, $|uv| \leq n$ and v is the trace of a circuit, and for any $i \in \mathbb{N}$, $uv^i w \in L_{q,q'}$.
3. Let $\mathcal{A}$ be a finite-state automaton with n states. Then for any $z \in L(\mathcal{A})$ verifying $|z| \geq n$, there exist $u, v, w \in A^*$ such that $z = uvw$, $v \neq \varepsilon$, $|uv| \leq n$, and for any $i \in \mathbb{N}$, $uv^i w \in L(\mathcal{A})$. $\diamondsuit$

EXERCISE 11.17 Application: Show that the following languages are not recognizable because they cannot be equal to $L(\mathcal{A})$ for any finite-state automaton $\mathcal{A}$:

1. $\{a^n b^n \mid n \in \mathbb{N}\}$.
2. $\{a^{n^2} \mid n \in \mathbb{N}\}$.
3. $\{a^p \mid p \text{ is a prime number}\}$.
4. $\{ww \mid w \in A^*\}$, when A has at least two letters.
5. The set of *palindromes*: the strings $w_1 \cdots w_n \in A^*$ $(n \geq 0)$ such that $w_1 \cdots w_n = w_n \cdots w_1$ (i.e. $w_i = w_{n+1-i}$ for $i = 1, \ldots, n$) when A has at least two letters. $\diamondsuit$

11.3.4 Minimal finite-state automata

Let $\mathcal{A}$ be a complete deterministic finite-state automaton whose only initial state is i. With any string u we associate the mapping $\delta_u : S \longrightarrow S$ defined by: $\delta_u(s)$ is the target of the unique path with source s and with trace u. Two strings u and v are said to be equivalent modulo $\mathcal{A}$ (denoted by $u \sim_{\mathcal{A}} v$) if $\delta_u(i) = \delta_v(i)$. It is easy to see that this is indeed an equivalence relation. Moreover, we have

Proposition 11.15
1. *The relation $\sim_{\mathcal{A}}$ has a finite number of equivalence classes.*
2. *If $u \sim_{\mathcal{A}} v$, then $\forall a \in A, \quad ua \sim_{\mathcal{A}} va$.*
3. *If $u \sim_{\mathcal{A}} v$ and $u \in L(\mathcal{A})$, then $v \in L(\mathcal{A})$.*

Proof.
1. With each equivalence class of $\sim_{\mathcal{A}}$ we can associate a state of $\mathcal{A}$; namely, the class of a string u is associated with the state $\delta_u(i)$. The number of classes is thus less than or equal to the number of states of $\mathcal{A}$.
2. If the two paths with respective traces u and v and with source i have the same target state q, the two paths with respective traces ua and va and with source i also have the same target state $q' = \delta_a(q)$.
3. If $u \in L(\mathcal{A})$, the unique path with source i and trace u has a final state as its target. If $u \sim_{\mathcal{A}} v$ then the only path with source i and trace v has the same final state as its target, and thus $v \in L(\mathcal{A})$. □

An equivalence relation $\sim$ on A^* verifying

$$u \sim v \quad \Longrightarrow \quad \forall a \in A,\ ua \sim va$$

is called a *(right) semi-congruence*. If it has only a finite number of equivalence classes, it is said to have a *finite index*. Finally, if a language L is a union of equivalence classes of $\sim$, then $\sim$ is said to *saturate* L.

We have thus shown that if $\mathcal{A}$ is a complete deterministic finite-state automaton, then $\sim_{\mathcal{A}}$ is a right semi-congruence of finite index saturating $L(\mathcal{A})$. Because any language recognized by a finite-state automaton can be recognized by a complete deterministic finite-state automaton, any recognizable language is saturated by a finite index semi-congruence. We will also show the converse of this property which will give a characterization of recognizable languages.

Theorem 11.16 *A language L in A^* is recognizable if and only if it is saturated by a finite index semi-congruence.*

Proof. Let $\sim$ be a finite index semi-congruence saturating the language L. We will construct a complete deterministic finite-state automaton $\mathcal{A} = (S, T, i, F)$ recognizing L:

- S is the set of equivalence classes of A^* modulo $\sim$.
- T is the set of triples $([u]_\sim, a, [ua]_\sim)$ for all $[u]_\sim$ in S and all a in A.
- i is the class of ε.
- F is the set of classes of strings in L.

The fact that $\sim$ has a finite index implies that S is finite. The fact that it is a semi-congruence implies that the definition of T is meaningful since $[u]_\sim = [v]_\sim \Longrightarrow [ua]_\sim = [va]_\sim$. It is clear that $\mathcal{A}$ is deterministic and complete.

We now show that this automaton recognizes L (i.e. $L = L(\mathcal{A})$). We easily show by induction on the length of a string $u \in A^*$ that the target of the unique path with source $i = [\varepsilon]_\sim$ and with trace u is $[u]_\sim$. It follows that if u is in L then this state is final, and hence $L \subseteq L(\mathcal{A})$. Conversely, if $[u]_\sim$ is final, then there exists a string v in L with $u \sim v$ and, since $\sim$ saturates L, u is also in L and thus $L(\mathcal{A}) \subseteq L$. □

Among the finite-index semi-congruences saturating a given recognizable language L, there is a semi-congruence that is less refined than all the others (i.e. it has fewer equivalence classes, and each equivalence class is 'fatter'). It is denoted $\sim_L$ and is defined by $u \sim_L v$ if and only if

$$\forall w \in A^*, \quad uw \in L \Longleftrightarrow vw \in L.$$

It is clear that $\sim_L$ is an equivalence relation. It is also a semi-congruence. Indeed, if $u \sim_L v$ then $\forall w \in A^*$, $uaw \in L \Longleftrightarrow vaw \in L$ and thus $ua \sim_L va$. It saturates L because letting $w = \varepsilon$ and $u \sim_L v$ implies that

$$u \in L \Longleftrightarrow v \in L\,.$$

Finally, the fact that it has finite index follows from the next result.

Theorem 11.17 *Let L be a recognizable language and $\sim$ a finite index semi-congruence saturating L. Then $\sim_L \supseteq \sim$. That is,*

$$\forall u, v \in A^*, \quad u \sim v \Longrightarrow u \sim_L v.$$

Proof. Assume that $u \sim v$ and show that

$$\forall w \in A^*, \quad uw \in L \Longleftrightarrow vw \in L\,.$$

Let w be an arbitrary string in A^*. Because $\sim$ is a semi-congruence, $uw \sim vw$, and because $\sim$ saturates L, $uw \in L \Longleftrightarrow vw \in L$. □

We saw in the proof of Theorem 11.16 that any finite-index semi-congruence $\sim$ saturating L enabled us to define a finite-state automaton recognizing L, with

the equivalence classes of $\sim$ as states. Therefore, the semi-congruence $\sim_L$ enables us to define a deterministic finite-state automaton recognizing L whose number of states is minimal. We will temporarily call this automaton the canonical automaton of L.

There is a more algebraic way of characterizing this canonical automaton recognizing a language L. First we must introduce some definitions.

Definition 11.18 *A complete deterministic finite-state automaton $\mathcal{A} = (S, T, i, F)$ is said to be reachable if $\forall s \in S$, $\exists u \in A^*$: $\delta_u(i) = s$.*

It is easy to see that we can always delete from a finite-state automaton its non-reachable states (namely, those states s such that $\forall u \in A^*, \delta_u(i) \neq s$) without modifying the recognized language and also that the finite-state automaton constructed in the proof of Theorem 11.16 is reachable.

Definition 11.19 *Let $\mathcal{A} = (S, T, i, F)$ and $\mathcal{A}' = (S', T', i', F')$ be two reachable complete deterministic finite-state automata. A mapping $h: S \longrightarrow S'$ is a homomorphism of automata from $\mathcal{A}$ to $\mathcal{A}'$ if*

- *h is surjective,*
- *$\forall s \in S$, $\forall a \in A$, $h(\delta_a(s)) = \delta'_a(h(s))$,*
- *$h(i) = i'$ and*
- *$\forall s \in S$, $h(s) \in F' \Longleftrightarrow s \in F$.*

Proposition 11.20 *If there exists a homomorphism from $\mathcal{A}$ to $\mathcal{A}'$, then $L(\mathcal{A}) \subseteq L(\mathcal{A}')$.*

Proof. We prove, by induction on the length of $u \in A^*$, that, for any path in $\mathcal{A}$ with source s, target s' and trace u, there exists in $\mathcal{A}'$ a path with source $h(s)$, target $h(s')$ and trace u.

It follows that if there exists in $\mathcal{A}$ a path with source i, trace u and target in F, then there exists in $\mathcal{A}'$ a path with source i', trace u and target in F'. □

Theorem 11.21 *For any reachable complete deterministic finite-state automaton $\mathcal{A}$, there exists a homomorphism from $\mathcal{A}$ to the canonical automaton recognizing $L(\mathcal{A})$.*

Proof. Let s be a state of $\mathcal{A}$. Since $\mathcal{A}$ is reachable, there exists a string u such that $s = \delta_u(i)$. Let $h(s) = [u]_{\sim_L}$. This mapping is well defined, because if $\delta_v(i) = s$, then $u \sim_{\mathcal{A}} v$ and, by Theorem 11.17, $u \sim_L v$. This mapping is indeed surjective ($\forall u \in A^*, [u]_{\sim_L} = h(\delta_u(i))$. It is a homomorphism because

- $h(i) = [\varepsilon]_{\sim_L}$,
- $h(\delta_a(\delta_u(i)) = h(\delta_{ua}(i)) = [ua]_{\sim_L}$ and $h(\delta_u(i) = [u]_{\sim L}$ and

- $h(\delta_u(i)) = [u]_{\sim_L} \subseteq L$ if and only if $u \in L$ if and only if $\delta_u(i) \in F$. □

Justified by this theorem, we call the canonical automaton the *minimal* automaton. It is indeed minimal, among complete deterministic finite-state automata recognizing L, for the ordering $\leq$ defined by $\mathcal{A} \leq \mathcal{A}'$ if there exists a homomorphism from $\mathcal{A}'$ to A.

The canonical homomorphism h from $\mathcal{A}$ to the minimal automaton induces an equivalence relation on the states of $\mathcal{A}$ defined by

$$s \approx s' \iff h(s) = h(s').$$

We will show how to compute this equivalence, and this will enable us to find the minimal automaton recognizing L from any (reachable complete) deterministic finite-state automaton recognizing L.

Let $\mathcal{A} = (S, T, i, F)$ be a reachable complete deterministic finite-state automaton. Define the sequence $(\approx_i)_{i \geq 0}$ of equivalence relations on S by:

- $s \approx_0 s'$ if and only if $s \in F \iff s' \in F$ or, equivalently, $\approx_0$ has exactly two classes, F and $S \setminus F$.
- $s \approx_{i+1} s'$ if and only if $s \approx_i s'$ and $\forall a \in A, \delta_a(s) \approx_i \delta_a(s')$.

Proposition 11.22
(a) $\approx_{i+1} \subseteq \approx_i$.
(b) *If* $\approx_{i+1} = \approx_i$, *then* $\forall j \geq i$, $\approx_j = \approx_i$.
(c) $\exists n < |S|$: $\approx_{n+1} = \approx_n$.

Proof. Point (a) is straightforward by the definition of $\approx_{i+1}$.

Point (b) is proved by induction on $n = j - i$. It is straightforward if $n = 0$ and $n = 1$. Assume $\approx_j = \approx_i$. Then,

$$\begin{aligned} s \approx_{j+1} s' &\iff s \approx_j s' \text{ and } \forall a \in A,\ \delta_a(s) \approx_j \delta_a(s') \\ &\iff s \approx_i s' \text{ and } \forall a \in A,\ \delta_a(s) \approx_i \delta_a(s') \\ &\iff s \approx_{i+1} s' \\ &\iff s \approx_i s'. \end{aligned}$$

To show point (c), let m_i be equal to the number of classes of $\approx_i$. It is clear that $m_{i+1} \geq m_i$ and that $m_{i+1} = m_i$ implies $\approx_{i+1} = \approx_i$. Assume that $\forall n < |S|$, $\approx_n \neq \approx_{n+1}$. We thus have $m_0 < m_1 < \cdots < m_{|S|}$; hence $2 + |S| = m_0 + |S| \leq m_{|S|}$, and this is impossible since an equivalence relation on S can have at most $|S|$ classes. □

To minimize a finite-state automaton, we will thus construct such a sequence of equivalence relations. As soon as $\approx_{i+1} = \approx_i$, the mapping h mapping a state to its class modulo $\approx_i$ is the required homomorphism, as shown by the following proposition.

Proposition 11.23 *Let $\mathcal{A}$ be a finite-state automaton and let h be the canonical homomorphism from $\mathcal{A}$ to the minimal automaton recognizing $L(\mathcal{A})$. Let $n < |S|$ be such that $\approx_n = \approx_{n+1}$. Then*

$$\forall s, s' \in S, \qquad h(s) = h(s') \Longleftrightarrow s \approx_n s'.$$

Proof.

1. First show by induction on i that

$$\forall i \geq 0, \forall s, s' \in S, \qquad h(s) = h(s') \Longrightarrow s \approx_n s'.$$

Let u and v be two strings such that $\delta_u(i) = s$ and $\delta_v(i) = s'$. If $h(s) = h(s')$, then $u \sim_L v$, by the definition of h.

- $s \in F \Longleftrightarrow u \in L \Longleftrightarrow v \in L \Longleftrightarrow s' \in F$, and thus $s \approx_0 s'$.
- Assume that $\forall s, s' \in S, \quad h(s) = h(s') \Longleftrightarrow s \approx_i s'$. If $h(s) = h(s')$ then $s \approx_i s'$; but we also have, for any a in A, $h(\delta_a(s)) = h(\delta_a(s'))$, and thus $\delta_a(s) \approx_i \delta_a(s')$. Hence, $s \approx_{i+1} s'$.

2. It is easy to show, by induction on the length $|w|$ of $w \in A^*$, that $s \approx_i s' \Longrightarrow \delta_w(s) \approx_{i+|w|} \delta_w(s')$ and, in particular, since $\approx_n = \approx_{n+|w|}$, $s \approx_n s' \Longrightarrow \delta_w(s) \approx_n \delta_w(s')$. Since $\approx_n \subseteq \approx_0$, we also obtain $s \approx_n s' \Longrightarrow \delta_w(s) \approx_0 \delta_w(s')$. Hence,

$$\begin{aligned}
\delta_u(i) \approx_n \delta_v(i) &\Longrightarrow \forall w \in A^*,\ \delta_{uw}(i) = \delta_w(\delta_u(i)) \approx_0 \delta_w(\delta_v(i)) = \delta_{vw}(i) \\
&\Longleftrightarrow \forall w \in A^*,\ \delta_{uw}(i) \in F \Longleftrightarrow \delta_{vw}(i) \in F \\
&\Longleftrightarrow \forall w \in A^*,\ uw \in L \Longleftrightarrow vw \in L \\
&\Longleftrightarrow u \sim_L v.
\end{aligned}$$

□

Exercise 11.18 Give minimal complete deterministic finite-state automata recognizing the languages over the alphabet $A = \{a, b\}$ specified by the following regular expressions:

1. $(a + b)^* b (a + b)^*$.
2. $ba^* + ab + (a + bb)ab^*$.
3. $(b + ab + aab)^*(\varepsilon + a + aaa)$.
4. $(a + b)^2 + (a + b)^3 + (a + b)^4$.
5. $\left((a + b)^2\right)^* + \left((a + b)^3\right)^*$.
6. $\varepsilon + ab^*a + (ab + ba)^*$.

♢

11.3.5 Operations on finite-state automata

Let $\mathcal{A} = (S, T, \{i\}, F)$ be a complete deterministic finite-state automaton. For any string u in A^*, there exists in $\mathcal{A}$ a unique path c with source i and label u. The target state of this path c is in F if and only if $u \in L(\mathcal{A})$. We immediately obtain a finite-state automaton recognizing the complement of $L(\mathcal{A})$.

Proposition 11.24 *If $\mathcal{A} = (S, T, \{i\}, F)$ is a complete deterministic finite-state automaton, the finite-state automaton $\mathcal{A}' = (S, T, \{i\}, S \setminus F)$ is again a complete deterministic finite-state automaton and $L(\mathcal{A}') = A^* \setminus L(\mathcal{A})$.*

EXERCISE 11.19 Let $\mathcal{A} = (S, T, I, F)$ be a finite-state automaton and let $\mathcal{A}' = (S, T, I, S \setminus F)$. Show, by means of examples, that $L(\mathcal{A}') = A^* \setminus L(\mathcal{A})$ does not necessarily hold if $\mathcal{A}$ is not both deterministic and complete. ♢

Let $\mathcal{A}' = (S', T', I', F')$ and $\mathcal{A}'' = (S'', T'', I'', F'')$ be two finite-state automata over the same alphabet A. The *direct product* of the transition systems (S', T') and (S'', T'') is the transition system (S, T) defined by

$$S = S' \times S'' \text{ and } T = \{((s'_1, s''_1), a, (s'_2, s''_2)) \,/\, (s'_1, a, s'_2) \in T', (s''_1, a, s''_2) \in T''\}.$$

It is easy to see that there is in (S, T) a path c with trace u, source (s'_1, s''_1) and target (s'_2, s''_2) if and only if there is in (S', T') a path c' with trace u, source s'_1 and target s'_2, and in (S'', T'') a path c'' with trace u, source s''_1 and target s''_2.

Now consider the finite-state automata

$$\mathcal{A}_1 = (S, T, I' \times I'', F' \times F'') \text{ and } \mathcal{A}_2 = (S, T, I' \times I'', F' \times S'' \cup S' \times F'').$$

Clearly, using the above remark we have

Proposition 11.25 $L(\mathcal{A}_1) = L(\mathcal{A}) \cap L(\mathcal{A}')$, $L(\mathcal{A}_2) = L(\mathcal{A}) \cup L(\mathcal{A}')$.

11.4 Equation systems

With each labelled transition system $\mathcal{S}$ we associate an *equation system* $\widehat{\mathcal{S}}$ on $\mathcal{P}(A^*)$ such that the least solution of $\widehat{\mathcal{S}}$ is the set of traces of paths of the transition system $\mathcal{S}$.

Let $\mathcal{S} = (S, T, \alpha, \beta, \lambda)$ be a labelled transition system, where S is finite. Let $\mathcal{D}$ be the set of mappings from $S \times S$ to $\mathcal{P}(A^*)$ ordered as follows: $D \leq D'$ if and only if $\forall s, s' \in S$, $D(s, s') \subseteq D'(s, s')$.

With $\mathcal{S}$ we associate the *equation system* $\widehat{\mathcal{S}}$ on $\mathcal{P}(A^*)$ whose variables are $x_{s,s'}$ for all pairs (s, s') of states of S and whose equations are

$$x_{s,s'} = \sum \{a x_{s'',s'} \,/\, \exists t \colon \alpha(t) = s, \lambda(t) = a, \beta(t) = s''\} \cup \{\varepsilon \,/\, s = s'\}.$$

Consider now this equation system as a mapping, also denoted by $\widehat{\mathcal{S}}$, from $\mathcal{D}$ to $\mathcal{D}$ defined as follows: if D is a mapping from $S \times S$ to $\mathcal{P}(A^*)$, then $\widehat{\mathcal{S}}(D)$ is the mapping D' defined by

$$D'(s,s') = \bigcup\{a \cdot D(s'',s') \,/\, \exists t\colon \alpha(t) = s, \lambda(t) = a, \beta(t) = s''\} \cup \{\varepsilon \,/\, s = s'\}.$$

It is easy to see that the mapping $\widehat{\mathcal{S}}$ is continuous. It thus has a least fixpoint D^μ defined by (see Theorem 2.40)

$$D^\mu(s,s') = \bigcup_{i \geq 0} D_i(s,s')$$

with

- $D_0(s,s') = \emptyset$, $\forall s, s' \in S$ and
- $D_{i+1}(s,s') = D_i(s,s') \cup \widehat{\mathcal{S}}(D_i)(s,s')$.

D^μ is also said to be the least solution of the equation system $\widehat{\mathcal{S}}$.

Exercise 11.20 Show that $\widehat{\mathcal{S}}$ is continuous. ♢

Theorem 11.26 $\forall s, s'$, $L_{s,s'} = D^\mu(s,s')$.

Proof. Let $D_L \in \mathcal{D}$ be defined by $D_L(s,s') = L_{s,s'}$.
We will first show that $\widehat{\mathcal{S}}(D_L) \subseteq D_L$, and hence it will follow that $D^\mu \subseteq D_L$. Since

$$\widehat{\mathcal{S}}(D_L)(s,s') = \bigcup\{a \cdot L_{s'',s'} \,/\, \exists t\colon \alpha(t) = s, \lambda(t) = a, \beta(t) = s''\} \cup \{\varepsilon \,/\, s = s'\},$$

we have:

1. $\varepsilon \in \widehat{\mathcal{S}}(D_L)(s,s')$ if and only if $s = s'$ and thus if and only if $\varepsilon \in L_{s,s'}$.
2. If $au \in \widehat{\mathcal{S}}(D_L)(s,s')$, there exists a path c with source s'', target s' and trace u, and there exists a transition t with source s, target s'' and label a. It follows that $t \cdot c$ is a path with source s, target s' and trace au in $L_{s,s'}$.

We now show that $L_{s,s'} \subseteq D^\mu(s,s')$. To this end, let $L^i_{s,s'}$ be the set of strings of $L_{s,s'}$ with length strictly less than i. We will show by induction that

$$L^i_{s,s'} \subseteq D_i(s,s').$$

Hence, we will immediately deduce that $L_{s,s'} \subseteq D^\mu(s,s')$.

1. If $i = 0$ then $D_i(s,s') = \emptyset$ and $L^i_{s,s'} = \{u \in L_{s,s'} \,/\, |u| < 0\} = \emptyset$.
2. Let u be a string in $L_{s,s'}$ with length strictly less than $i+1$.

(2.1) If its length is strictly less than i then it is in $D_i(s, s')$ and hence in $D_{i+1}(s, s')$.

(2.2) If it has length i, then the following holds:

(2.2.1) If $i = 0$ then $u = \varepsilon$ and thus $s = s'$; hence $\varepsilon \in \widehat{\mathcal{S}}(D_i)(s, s')$.

(2.2.2) If $i > 0$ then $u = au'$ with $|u'| = i$. Since u is the trace of a path with source s and with target s', there exists a path $c = t_1 \cdots t_{i+1}$ with $\alpha(c) = \alpha(t_1) = s$, $\beta(c) = \beta(t_{i+1}) = s'$ and $\lambda(t_1) = a$, $\lambda(t_2 \cdots t_{i+1}) = u'$. The transition t_1 has source s, target s'' and label a, and the path $t_2 \cdots t_{i+1}$ has source s'', target s' and trace u'. We thus have $u' \in L^i_{s'',s'} \subseteq D_i(s'', s')$ and $au' \in \widehat{\mathcal{S}}(D_i)(s, s') \subseteq D_{i+1}(s, s')$. □

We will now generalize the equation systems that we have just introduced.

Let $X = \{x_1, \ldots, x_n\}$ be variables and let $K_{i,j}$ for $i \in \{0, \ldots, n\}$ and $j \in \{1, \ldots, n\}$ be subsets of A^*.

We define the mapping $\widehat{K}$ from $\mathcal{P}(A^*)^n$ to $\mathcal{P}(A^*)^n$ by

$$\widehat{K}(D_1, \ldots, D_n) = (D'_1, \ldots, D'_n) \quad \Longleftrightarrow \quad D'_j = K_{0,j} \cup \bigcup_{i=1}^{n} K_{i,j} D_i \,.$$

This mapping is continuous and hence has a least fixpoint, $(D^\mu_1, \ldots, D^\mu_n)$.

It is easy to see that the equation system $\widehat{\mathcal{S}}$ associated with a transition system $\mathcal{S}$ is of this form.

Lemma 11.27 *Let g be the mapping from $\mathcal{P}(A^*)$ to $\mathcal{P}(A^*)$ defined by $g(D) = K_1 + K_2 D$. Its least fixpoint is $K_2^* K_1$.*

Proof. (Cf. also Exercise 3.11 where we showed that if $\varepsilon \notin L$ then L^*M is the only fixpoint of $X = LX + M$.)

1. $g(K_2^* K_1) = K_1 + K_2 K_2^* K_1 = K_2^* K_1$, and thus $K_2^* K_1$ is a fixpoint of g.
2. Let D^μ be the least fixpoint of g. Then $D^\mu = K_1 + K_2 D^\mu$. We thus have

$$K_1 \subseteq D^\mu, \quad K_1 + K_2 K_1 \subseteq D^\mu,$$

and, by induction,

$$K_1 + K_2 K_1 + K_2^2 K_1 + \cdots + K_2^i K_1 \subseteq D^\mu.$$

Hence, $K_2^* K_1 \subseteq D^\mu$, and thus $K_2^* K_1$ is indeed the least fixpoint of g. □

Theorem 11.28 *Let $\widehat{K}$ be defined as previously, and let $(D_1^\mu, \ldots, D_n^\mu)$ be its least fixpoint. If all the $K_{i,j}$s are regular, then each D_i^μ is also regular.*

Proof. The proof is by induction on n. If $n = 1$ then $\widehat{K}(D) = K_1 + K_2 D$. Its least fixpoint is $K_2^* K_1$. If K_1 and K_2 are regular, then $K_2^* K_1$ is also regular.

Deducing the result for $n+1$ from the result for n is analogous to the elimination method for solving linear systems.

The required least fixpoint is the least solution of the following equation system:

$$\begin{aligned} x_1 &= K_{0,1} + \bigcup_{i=1}^{n+1} K_{i,1} x_i \\ &\vdots \\ x_n &= K_{0,n} + \bigcup_{i=1}^{n+1} K_{i,n} x_i \\ x_{n+1} &= K_{0,n+1} + \bigcup_{i=1}^{n+1} K_{i,n+1} x_i \ . \end{aligned}$$

Let us solve the last equation:

$$x_{n+1} = K_{n+1,n+1}^* (K_{0,n+1} + \bigcup_{i=1}^{n} K_{i,n+1} x_i).$$

Substituting the value of x_{n+1} in the other equations:

$$\begin{aligned} x_1 &= K_{0,1} + K_{n+1,n+1}^* K_{0,n+1} + \bigcup_{i=1}^{n} (K_{i,1} + K_{n+1,n+1}^* K_{i,n+1}) x_i \\ &\vdots \\ x_n &= K_{0,n} + K_{n+1,n+1}^* K_{0,n+1} + \bigcup_{i=1}^{n} (K_{i,n} + K_{n+1,n+1}^* K_{i,n+1}) x_i \ . \end{aligned}$$

By the induction hypothesis, all the components of the least solution $(D_1, \ldots, D_n)$ of this system are regular.

Substituting each D_i in the definition of x_{n+1}, we have

$$x_{n+1} = K_{n+1,n+1}^* \left(K_{0,n+1} + \bigcup_{i=1}^{n} K_{i,n+1} D_i \right) ,$$

which is also regular.

We claim that this elimination method indeed computes the least solutions of the considered equations. □

Corollary 11.29 *Let $(S, T, \alpha, \beta, \lambda)$ be a transition system. For any $Q, Q' \subseteq S$, $L_{Q,Q'}$ is a regular language.*

EXAMPLE 11.30 Consider the finite-state automaton with transitions $(1, b, 1)$, $(1, a, 2)$, $(2, b, 2)$, $(2, a, 1)$ shown in Figure 11.3.

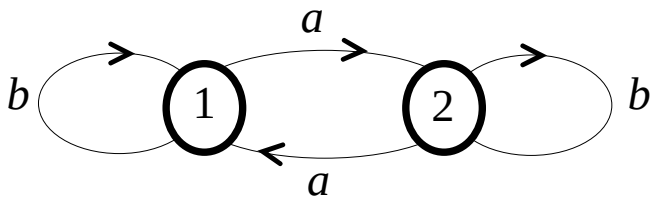

Figure 11.3

Consider the variables $x_{1,1}, x_{1,2}, x_{2,1}, x_{2,2}$. The equation system associated with this transition system is as follows:

$$\begin{aligned} x_{1,1} &= \varepsilon + bx_{1,1} + ax_{2,1} \ , \\ x_{2,1} &= bx_{2,1} + ax_{1,1} \ , \\ x_{1,2} &= bx_{1,2} + ax_{2,2} \ , \\ x_{2,2} &= \varepsilon + bx_{2,2} + ax_{1,2} \ . \end{aligned}$$

Compute $x_{2,2}$:

$$x_{2,2} = b^*(\varepsilon + ax_{1,2}) = b^* + b^*ax_{1,2}.$$

Substitute $x_{2,2}$ in the other equations:

$$\begin{aligned} x_{1,1} &= \varepsilon + bx_{1,1} + ax_{2,1} \ , \\ x_{2,1} &= bx_{2,1} + ax_{1,1} \ , \\ x_{1,2} &= bx_{1,2} + ab^* + ab^*ax_{1,2} = (ab^*a + b)x_{1,2} + ab^*. \end{aligned}$$

Compute $x_{1,2}$:

$$x_{1,2} = (ab^*a + b)^*ab^*.$$

Substitute $x_{1,2}$ in $x_{2,2}$:

$$x_{2,2} = b^* + b^*a(ab^*a + b)^*ab^* = (ab^*a + b)^* \ .$$

EXERCISE 11.21 Let L be the set of strings over the alphabet $\{a, b\}$ such that the number of bs is divisible by 3. There is a finite-state automaton $\mathcal{A}$ with three states such that $L = L(\mathcal{A})$. Give the equation system associated with the transitions, and solve this system for $L_{I,F}$ in order to give a regular expression denoting L. ♢

EXERCISE 11.22 Let L be the language over the alphabet $A = \{a, b\}$ consisting of all the strings without three consecutive as. Give the minimal complete deterministic finite-state automaton recognizing L and solve the corresponding equation system; deduce a regular expression for L. ♢

We will now prove the converse of Corollary 11.29.

Theorem 11.31 *If L is a regular language then there exists a finite-state automaton $\mathcal{A}$ such that $L = L(\mathcal{A})$.*

Proof.
1. If L consists of a single string, the theorem holds as follows. If $u = a_1 \cdots a_n$, consider the incomplete deterministic finite-state automaton whose states are $s_0, s_1, \ldots, s_n$, the unique initial state is s_0, the unique final state is s_n and the transitions are (s_{i-1}, a_i, s_i) for $1 \leq i \leq n$. If u is the empty string, we will let $n = 0$ and the automaton will have no transition.
2. If $L = L_1 + L_2$ then by the induction hypothesis there exist $\mathcal{A}_1$ and $\mathcal{A}_2$, with $\mathcal{A}_i = (S_i, T_i, I_i, F_i)$ such that $L_i = L(\mathcal{A}_i)$. We can assume that the states of $\mathcal{A}_1$ and of $\mathcal{A}_2$ are two disjoint sets. The finite-state automaton $\mathcal{A} = (S, T, I, F)$, with $S = S_1 \cup S_2$, $T = T_1 \cup T_2$, $I = I_1 \cup I_2$, and $F = F_1 \cup F_2$, recognizes the language $L(\mathcal{A}) = L(\mathcal{A}_1) \cup L(\mathcal{A}_2)$.
3. If $L = L_1 L_2$, there exist $\mathcal{A}_1$ and $\mathcal{A}_2$ such that $L_i = L(\mathcal{A}_i)$. Assume again that their states are two disjoint sets and let

$$
\begin{aligned}
S &= S_1 \cup S_2, \\
T &= T_1 \cup T_2 \cup \{(s, a, s') \,/\, s' \in I_2, \exists s'' \in F_1 : (s, a, s'') \in T_1\}, \\
I &= \begin{cases} I_1 & \text{if } I_1 \cap F_1 = \emptyset, \\ I_1 \cup I_2 & \text{otherwise,} \end{cases} \\
F &= F_2.
\end{aligned}
$$

Then $L(\mathcal{A}) = L(\mathcal{A}_1)L(\mathcal{A}_2)$. We show this as follows:

(a) Let $v \in L(\mathcal{A}_1)$ and $w \in L(\mathcal{A}_2)$. Since $w \in L(\mathcal{A}_2)$, there exists in $\mathcal{A}_2$ a path c' with trace w whose source s'_1 is in I_2 and whose target s'_2 is in $F_2 = F$.

(a.1) If $v = \varepsilon$, then $I_2 \cap F_1 \neq \emptyset$, and thus $I_2 \subseteq I$. The path c' is also a path of $\mathcal{A}$, and since $s'_1 \in I_2 \subseteq I$ and $s'_2 \in F_2 = F$, $vw = w \in L(\mathcal{A})$.

(a.2) If $v = v'a$, then there exists in $\mathcal{A}_1$ a path c with trace v', source $s_1 \in I_1$ and target s_2, and a transition $t = (s_2, a, s_3)$ with $s_3 \in F_1$. By the definition of T, there also exists a transition $t' = (s_2, a, s'_1)$. The path $ct'c'$ of $\mathcal{A}$ has trace $v'aw = vw$, source $s_1 \in I_1 \subseteq I$ and target $s'_2 \in F_2 = F$. Hence $vw \in L(\mathcal{A})$.

(b) Let $u \in L(\mathcal{A})$. There thus exists in $\mathcal{A}$ a path c with trace u, source $s_1 \in I$ and target $s'_2 \in F = F_2$.

(b.1) If $s_1 \in I_2$, then,

- on the one hand, c is a path of $\mathcal{A}_2$ and $u \in L(\mathcal{A}_2)$, and
- on the other hand, since $I_2 \cap I \neq \emptyset$, then, by the construction of I, $I_1 \cap F_1 \neq \emptyset$ and thus $\varepsilon \in L(\mathcal{A}_1)$.

We thus have $u = \varepsilon u \in L(\mathcal{A}_1)L(\mathcal{A}_2)$.

(b.2) If $s_1 \notin I_2$, then $s_1 \in I_1$, and the path c must be of the form $c_1(s_2, a, s_1')c_2$ where c_i $(i = 1, 2)$ is a path of $\mathcal{A}_i$. By the construction of T, (s_2, a, s_1') is a transition of T if $s_1' \in I_2$ and $\exists s_3 \in F_1 : (s_2, a, s_3) \in T_1$. It follows that $u = vaw$, where v is the trace of c_1 and w the trace of c_2, and that $va \in L(\mathcal{A}_1)$ and $w \in L(\mathcal{A}_2)$.

4. Lastly, let $\mathcal{A} = (S, T, I, F)$ be a finite-state automaton. Since $L(\mathcal{A})^* = \{\varepsilon\} \cup L(\mathcal{A})^+$, in order to construct a finite-state automaton recognizing $L(\mathcal{A})^*$, it suffices to construct a finite-state automaton $\mathcal{A}'$ recognizing $L(\mathcal{A})^+$. To this end, it is enough to add to T the set of transitions

$$T'' = \{(s, a, s') \,/\, s' \in I \text{ and } \exists s'' \in F : (s, a, s'') \in T\}.$$

The proof is similar to the proof of point 2, in one direction by induction on the least integer i such that $u \in L(\mathcal{A})^i$, and in the other direction by induction on the number of transitions of T'' occurring in a path of $\mathcal{A}'$. □