Proposition 8.2. *Every Muller automaton is equivalent to a transition Muller automaton. Conversely, every transition Muller automaton is equivalent to a Muller automaton.*

Proof. Let $\mathcal{A} = (Q, A, E, i, \mathcal{T})$ be a Muller automaton. We claim that $\mathcal{A}$ is equivalent to the transition Muller automaton $\mathcal{A}' = (Q, A, E, j, \mathcal{T}')$, where j is a new state $Q' = E \times \{j\}$, $\mathcal{T}' = \mathcal{T}$ and

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

Indeed, to any infinite path of $\mathcal{A}$ starting at i

$$p = (i, a_0, q_1)(q_1, a_1, q_2) \cdots$$

corresponds an infinite path of $\mathcal{A}'$ with the same label starting at j

$$p' = (j, a_0, (i, a_0, q_1))((i, a_0, q_1), a_1, (q_1, a_1, q_2)) \cdots$$

and conversely, every infinite path of $\mathcal{A}'$ starting at j arises this way. Furthermore, the transition (q, a, q') occurs in p if and only if p' visits the state (q, a, q'). Therefore $\mathrm{Inf}_{\mathcal{T}}(p) = \mathrm{Inf}(p')$ and $\mathcal{A}$ and $\mathcal{A}'$ are equivalent. □

9 McNaughton's theorem

The aim of this section is to prove the following result, due to R. McNaughton.

Theorem 9.1. *Any recognizable subset of A^ω can be recognized by a Rabin automaton.*

The proof that we are going to present relies on a determinization algorithm due to S. Safra. It computes a Rabin automaton equivalent to a given Büchi automaton. Using Proposition 7.7 allows one to obtain the conclusion. The states of the Rabin automaton are labeled trees.

We first give an informal description of the construction. Consider a Büchi automaton

$$\mathcal{A} = (Q, A, E, I, F)$$

and a successful path p labeled $u \in A^\omega$. There exist states $i \in I$ and $f \in F$ and a factorization $p = p_0 p_1 p_2 \cdots$ where p_0 is a path from i to f labeled u_0 and for every $n > 0$, p_n is a path from f to f labeled u_n.

$$i \xrightarrow{u_0} f \xrightarrow{u_1} f \xrightarrow{u_2} f \cdots$$

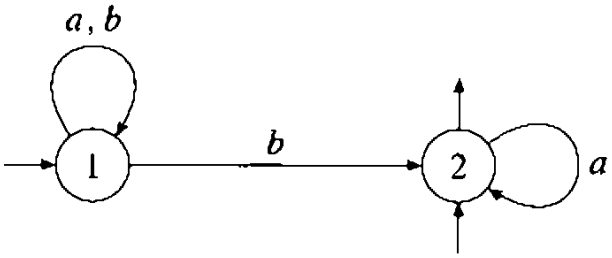
The usual determinization algorithm does not work with infinite words. One obtains indeed a deterministic automaton $\mathcal{B} = (\mathcal{P}(Q), A, \cdot, \{I\}, \mathcal{F})$, where $\mathcal{F} = \{P \mid P \cap F \neq \emptyset\}$ with transitions given by

$$S.a = \delta(S, a)$$

for $S \subset Q$ and $a \in A$. Processing u in $\mathcal{B}$ gives a path p'

$$i \xrightarrow{u_0} S_0 \xrightarrow{u_1} S_1 \xrightarrow{u_2} S_2 \cdots$$

Each S_i contains f, but this is not enough to make sure that such a path is successful, since nothing says that the state f appearing in S_i comes from the state f appearing in S_{i-1}. Thus, one cannot define as a table $\mathcal{T} = \{P \subset Q \mid P \cap F \neq \emptyset\}$. For example, if $\mathcal{A}$ is the automaton of Example 5.2, recognizing the set of infinite words containing a finite number of b's,



the path $\{1\} \xrightarrow{b} \{1,2\} \xrightarrow{b} \{1,2\} \xrightarrow{b} \{1,2\} \cdots$ would be successful in $\mathcal{B}$, although b^ω is not recognized by $\mathcal{A}$. Actually, the automaton obtained by this algorithm, once made trim, contains only one state and thus recognizes A^ω whatever be the acceptance mode (see Figure 9.1).

The idea is to look for a path $I \xrightarrow{u_0} S_0 \xrightarrow{u_1} S_1 \xrightarrow{u_2} S_2 \cdots$ such that the following two conditions are satisfied:

(1) $S_0 \subset \delta(I, u_0)$, and, for every $n \geq 0$, $S_{n+1} \subset \delta(S_n, u_{n+1})$.

(2) For every $n \geq 0$ and every $q \in S_{n+1}$, there is a state $p \in S_n$ and a path $p \xrightarrow{u_{n+1}} q$ in $\mathcal{A}$ *passing through a final state.*

To find such a path, we are going to build an automaton memorizing the occurrences of final states. The states of this automaton are oriented trees whose nodes are labeled by the sets S_i mentioned above. We then apply the usual determinization algorithm, taking care of adding the new final states that appear as the label of a new child of the vertex (see Figure 9.2). When all the states in the label S of a vertex have already visited a final state, that is when they all appear in the children of the node, this node is marked and all its descendants disappear.

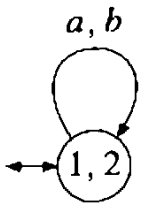


Figure 9.1. The automaton obtained by determinization.

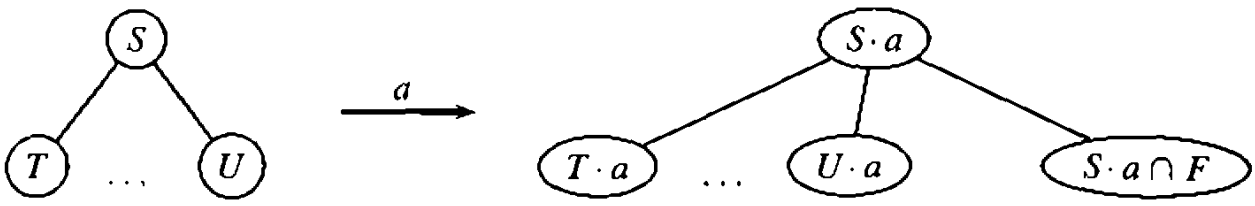


Figure 9.2. The action of letter a.

We now proceed to the formal description of the construction. Let $\mathcal{A} = (Q, A, E, I, F)$ be a finite Büchi automaton with

$$Q = \{1, 2, \ldots, n\} \quad \text{and} \quad V = \{1, 2, \ldots, 2n\}.$$

We build a deterministic Rabin automaton $\mathcal{D}$ as follows. Its *states* are labeled oriented trees with marks on some nodes. Formally the states are tuples (T, f, e, M) where

(1) the set of nodes T is a subset of V,

(2) $f : T \to T^*$ is a function mapping each node on the ordered sequence of its children,

(3) e is a function from T into the set of nonempty subsets of Q, mapping each node to its *label*,

(4) $M \subset T$ is the set of *marked nodes*.

These trees should also satisfy the following conditions:

(5) The root of the tree is 1.

(6) The marked nodes have to be leaves in the tree.

(7) For every node v, the union of the labels of its children is a strict subset of $e(v)$.

(8) If v is not an ancestor of w and if w is not an ancestor of v, then $e(v) \cap e(w) = \emptyset$.

The set $\mathcal{T}_n$ of all trees defined in this way is finite. More precisely, the following result holds:

Proposition 9.2. *A tree in $\mathcal{T}_n$ has at most n nodes.*

Proof. We associate with each node $v \in T$, the set

$$r(v) = e(v) \setminus \bigcup_{w \text{ child of } v} e(w)$$

By condition (7), $r(v)$ is not empty and, if v_1 and v_2 are distinct, we have $r(v_1) \cap r(v_2) = \emptyset$. This follows from condition (7) if v_1 is an ancestor of v_2 and from condition (8) in the other cases, since $r(v) \subset e(v)$. The sets $r(v)$ are thus pairwise distinct and we obtain

$$\mathrm{Card}(T) = \sum_{v \in T} 1 \leq \sum_{v \in T} \mathrm{Card}(r(v)) \leq \mathrm{Card}(Q) = n$$

establishing the proposition. □

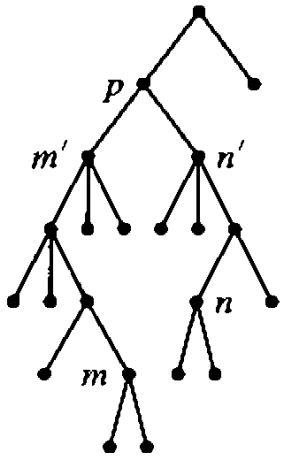


Figure 9.3. The node m is on the left of n.

In an oriented tree, the children of a given node are ordered. These local orders can be extended to a partial order on the set of nodes as follows. Given two nodes m and n which are not ancestor of one another, let p be their least common ancestor and let m' (resp. n') be the child of p which is an ancestor of m (resp. n). We say that m is *on the left* of n if $m' < n'$, as illustrated in Figure 9.3.

We return to the construction of the automaton $\mathcal{D}$. The set of its states is thus $\mathcal{T}_n$ and its transition function Δ is defined as follows. Let $R = (T, f, e, M)$ be a tree in $\mathcal{T}_n$ and let a be a letter from A. The state $\Delta(R, a)$ is obtained by the following steps.

(1) We perform the transition by a on the labels of each node and we erase the marks. For this, we build the tree (T, f, e_1, M_1), with $M_1 = \emptyset$, and, for each $v \in T$,

$$e_1(v) = \delta(e(v), a)$$

(2) We add to each node v a new child placed at the right of all children of v and labeled $e(v) \cap F$. This new node is marked and taken arbitrarily among the available nodes (in practice, we take the smallest available node). Formally, we choose an injection from T into $V \setminus T$ associating with each node $v \in T$ a node denoted $\overline{v}$. This is possible since T has at most n elements. Let $\overline{T} = \{\overline{v} \mid v \in T\}$ and consider the tree (T_2, f_2, e_2, M_2) where

$$T_2 = T \cup \overline{T}, \quad M_2 = \overline{T}$$

and, for every $v \in T$,

$$f_2(v) = f(v)\overline{v}, \qquad f_2(\overline{v}) = \varepsilon$$
$$e_2(v) = e_1(v), \qquad e_2(\overline{v}) = e_1(v) \cap F.$$

(3) In the label of each node v, we suppress the states appearing in the label of a node at the left of v. For this, we build e_3 defined for each node $v \in T_2$ by,

$$e_3(v) = e_2(v) \setminus \bigcup_{w \text{ to the left of } v} e_2(w).$$

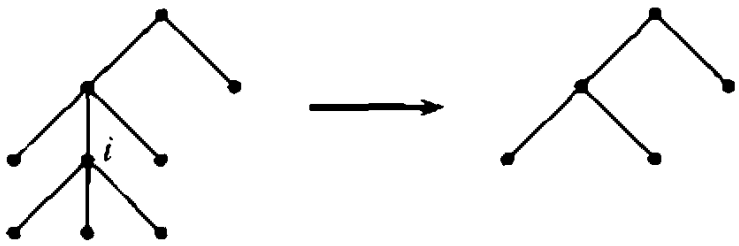


Figure 9.4. Suppressing a node with an empty label.

(4) We suppress the nodes with an empty label and we update the function f and the marks accordingly. This operation is represented in Figure 9.4. Formally, we change to the tree (T_4, f_4, e_4, M_4) where

$$T_4 = \{v \in T_2 \mid e_3(v) \neq \emptyset\},$$

$M_4 = M_2 \cap T_4$, $e_4(v)$ is the restriction of $e_3(v)$ to T_4 and, for each node $v \in T_4$, the word $f_4(v)$ is obtained by erasing the symbols of $T_2 \setminus T_4$.

(5) We mark all nodes with a label equal to the union of the labels of their children, i.e. such that

$$e(v) = \bigcup_{w \text{ child of } v} e(w),$$

and we suppress all their descendants.

We finally obtain a state $(T_5, f_5, e_5, M_5) = \Delta(R, a)$ which is an element of $\mathcal{T}_n$.

The initial state of $\mathcal{D}$ is the tree reduced to an unmarked node labeled I if $I \cap F = \emptyset$, to a marked node labeled I if $I \subset F$ and to a node labeled I with a marked child labeled $I \cap F$ in all other cases.

There remains to specify the set $\mathcal{R}$ defining the acceptance condition. Let

$$\mathcal{R} = \{(L_v, U_v) \mid v \in V\}$$

where

$$L_v = \{R \in \mathcal{T}_n \mid v \text{ is not a node of } R\}$$
$$U_v = \{R \in \mathcal{T}_n \mid v \text{ is a marked node of } R\}.$$

Thus, a path in $\mathcal{D}$ is successful if there exists an element $v \in V$ such that, ultimately, the path uses states in which v is a node and infinitely often states in which v is marked.

Before proving that this Rabin automaton recognizes the same set of infinite words as the automaton we started from, we are going to illustrate the construction by some examples. In these examples, the states are represented by labeled oriented trees and marked nodes are indicated by a double circle. An arrow of the form $\xrightarrow{(i)}$ indicates that step i of the algorithm has been performed.

Example 9.1. Consider the automaton represented in Figure 9.5, which recognizes the set of words having a finite nonzero number of b's. We detail the steps of Safra's algorithm. The initial state is the tree with a single node of Figure 9.6. The action of the letters a and b on the initial state are represented in Figure 9.7 and Figure 9.8, respectively. A new state has now been created. The actions of the letters a and b on this new state are represented in Figure 9.9 and Figure 9.10, respectively. Thus another new state has been created. The action of the letters a and b on this new state are easily derived from the ones represented in Figures 9.9 and 9.10 by exchanging the names 2 and 3 in every place. After renaming the states, we obtain the automaton of Figure 9.11.

We have $L_1 = \emptyset$, $L_2 = \{1, 3\}$, $L_3 = \{1, 2\}$, $U_1 = \emptyset$, $U_2 = \{2\}$, $U_3 = \{3\}$. Thus the accepting pairs are $(\{1, 3\}, \{2\})$ and $(\{1, 2\}, \{3\})$. Note that, by Formula (7.4), these Rabin pairs are equivalent to the table $\mathcal{T} = \{\{2\}, \{3\}\}$.

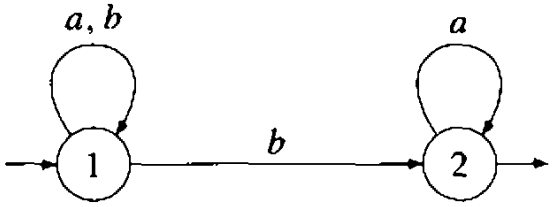


Figure 9.5. A Büchi automaton.

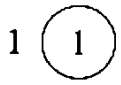


Figure 9.6. The initial state.

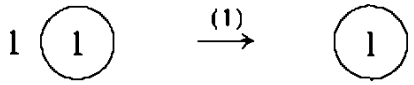


Figure 9.7. The action of a on the initial state.

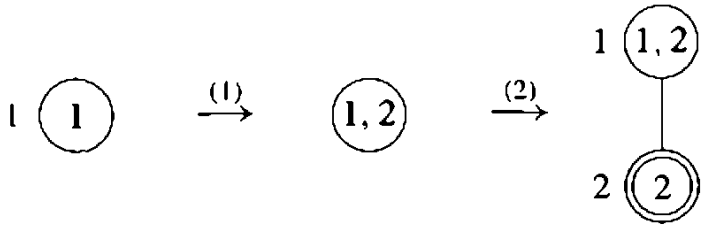


Figure 9.8. The action of b on the initial state.

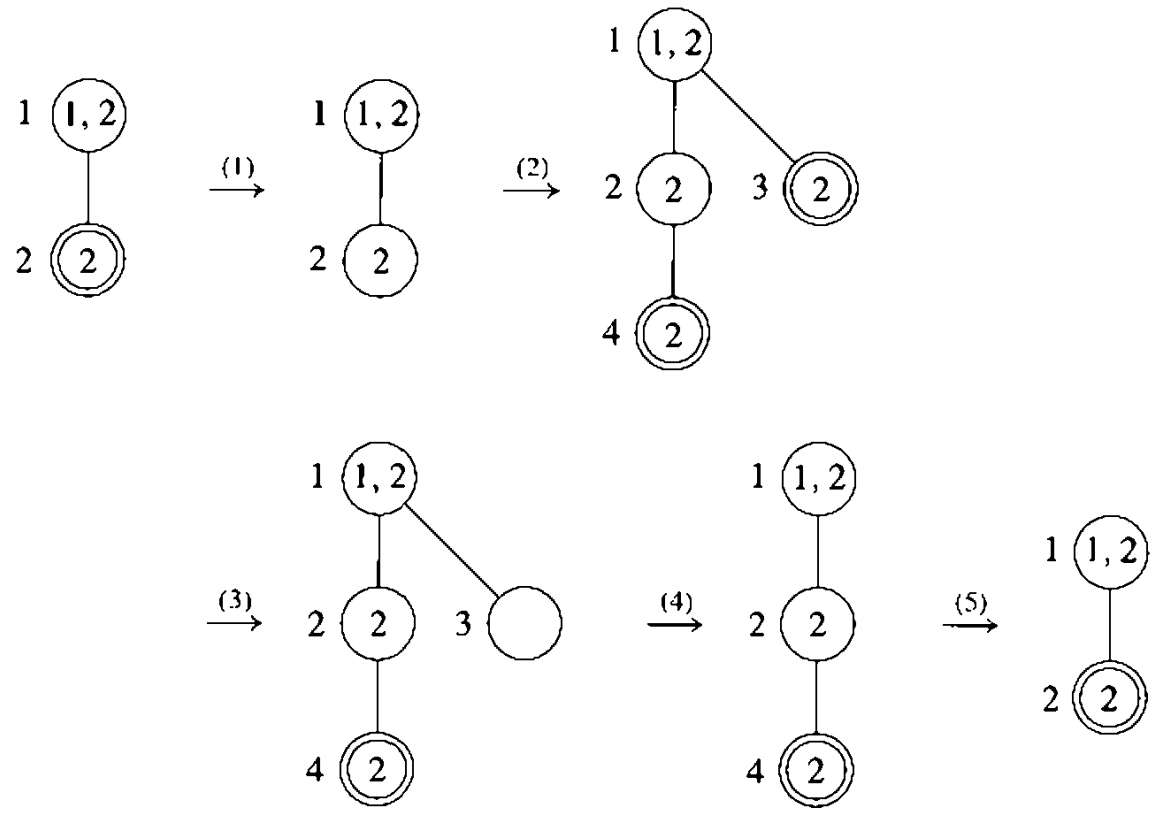


Figure 9.9. The action of a on the new state.

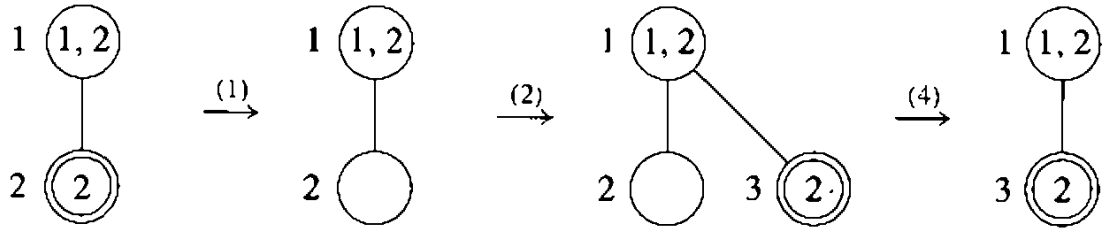


Figure 9.10. The action of b on the new state.

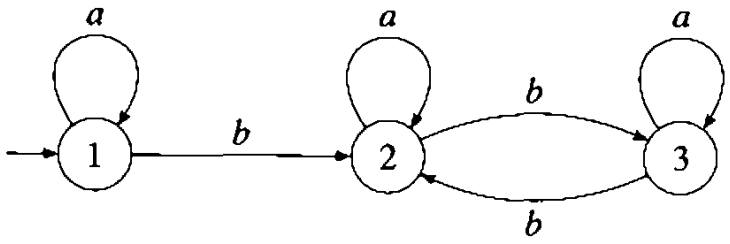


Figure 9.11. The Rabin automaton obtained by Safra's algorithm.

Example 9.2. The Büchi automaton represented in Figure 9.12 recognizes the set of words with a finite number of b. The deterministic automaton obtained by Safra's algorithm is represented in Figure 9.13. We recognize the automaton computing the parity of the number of b. The acceptance conditions are, in Rabin's form, the pairs $\{(\{1\}, \{2\}), (\{2\}, \{1\})\}$, which gives the table $\{\{1\}, \{2\}\}$.

Example 9.3. Consider the set $X = (\{b, c\}^*a \cup b)^\omega$. A Büchi automaton recognizing X is given in Figure 9.15: The application of Safra's algorithm gives the deterministic automaton of Figure 9.16, in which the set of states is $\{I, II, III, IV, V\}$:

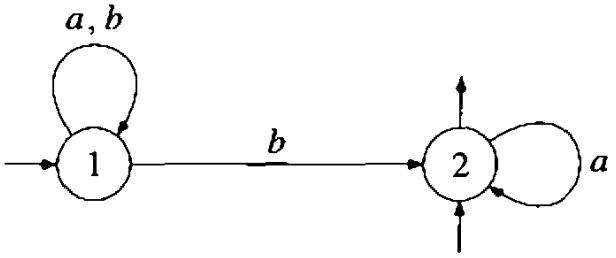


Figure 9.12. A Büchi automaton for the set A^*a^ω.

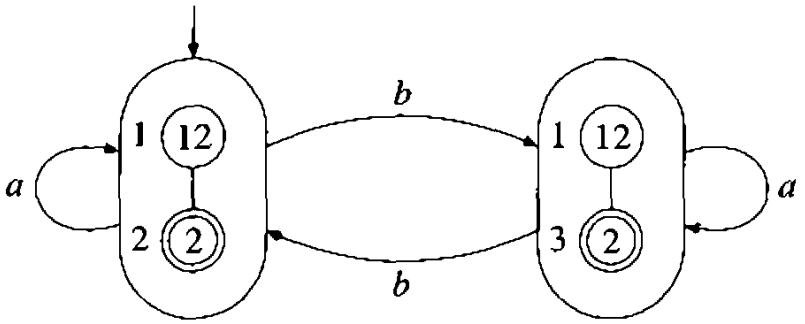


Figure 9.13. The deterministic automaton obtained by Safra's algorithm.

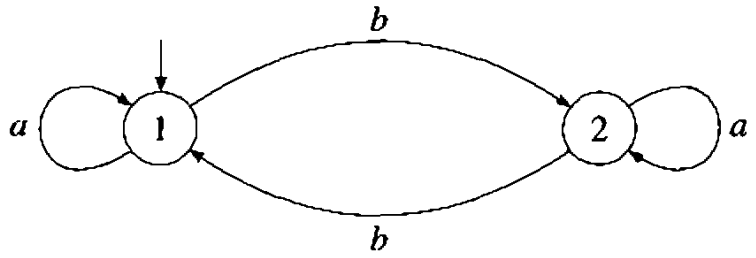


Figure 9.14. The same automaton after renaming the states.

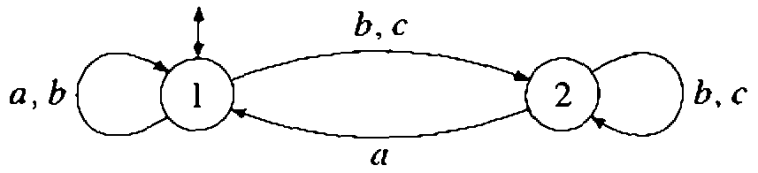


Figure 9.15. A Büchi automaton for $(\{b, c\}^* a \cup b)^\omega$.

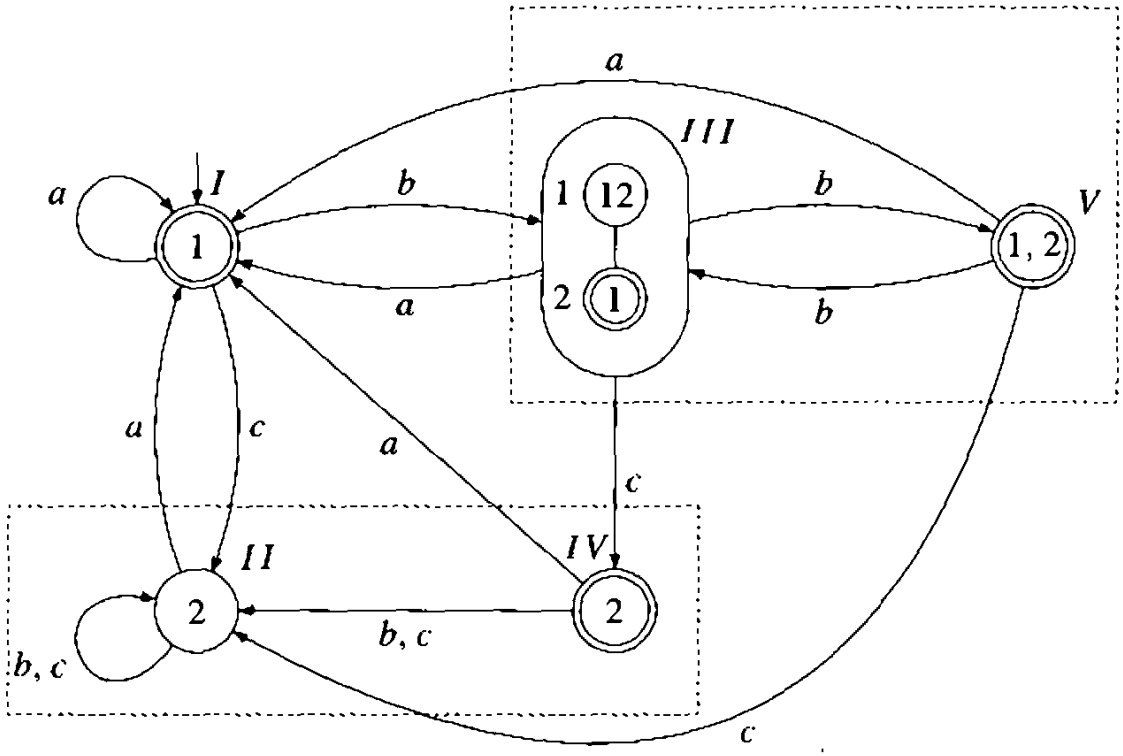


Figure 9.16. A deterministic Muller automaton for $(\{b, c\}^* a \cup b)^\omega$.

The Rabin pairs are $(\emptyset, \{I, IV, V\})$ and $(\{I, II, IV, V\}, \{III\})$. Therefore the table of the corresponding Muller automaton is

$$\mathcal{T} = \{T \subset Q \mid T \text{ contains either } I, IV \text{ or } V\} \cup \{\{III\}\}$$

This table is full. Indeed, if a set T contains I, IV or V, any superset of T has the same property. If $T = \{III\}$, any superset of T contains I, IV or V, or is equal to the set $\{II, III\}$. But this latter set is not admissible.

One can in fact obtain a smaller automaton by merging the states grouped inside each dashed rectangle. After renaming the states, the resulting automaton is represented in Figure 9.17. The table is $\mathcal{T} = \{\{1\}, \{2\}, \{1, 2\}, \{1, 3\}, \{1, 2, 3\}\}$, which is the comple-

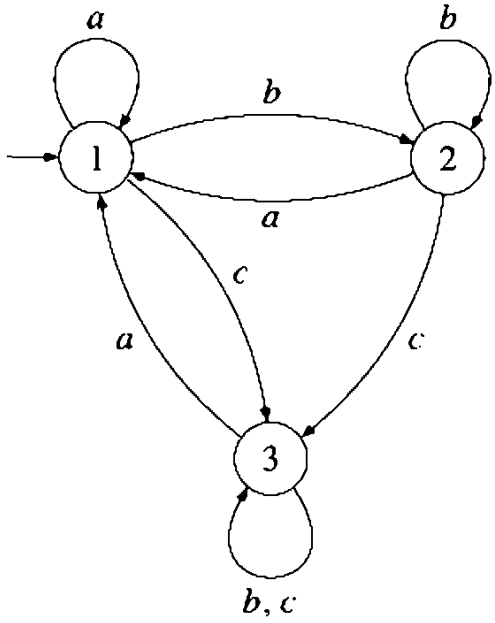


Figure 9.17. Applying Safra's construction.

ment of the table $\{\{3\}\}$. Since the table is full, X is deterministic. We have actually $X = \overrightarrow{\{a, b, c\}^* ab^* \cup b^+}$.

We are now going to prove that the deterministic automaton $\mathcal{D}$ is equivalent to the automaton $\mathcal{A}$ we started from. We shall need a lemma which makes more precise the behavior of $\mathcal{D}$. Let $u = a_1 \cdots a_n$ be a finite word and let R_0 be a state of $\mathcal{D}$ containing a marked node v labeled S_0. We suppose that, for $1 \leq i \leq n$, the states $R_i = \Delta(R_0, a_1 \cdots a_i)$ also contain the node v, with a label S_i, but that this node is marked only for $i = n$. The hypotheses are represented in Figure 9.18.

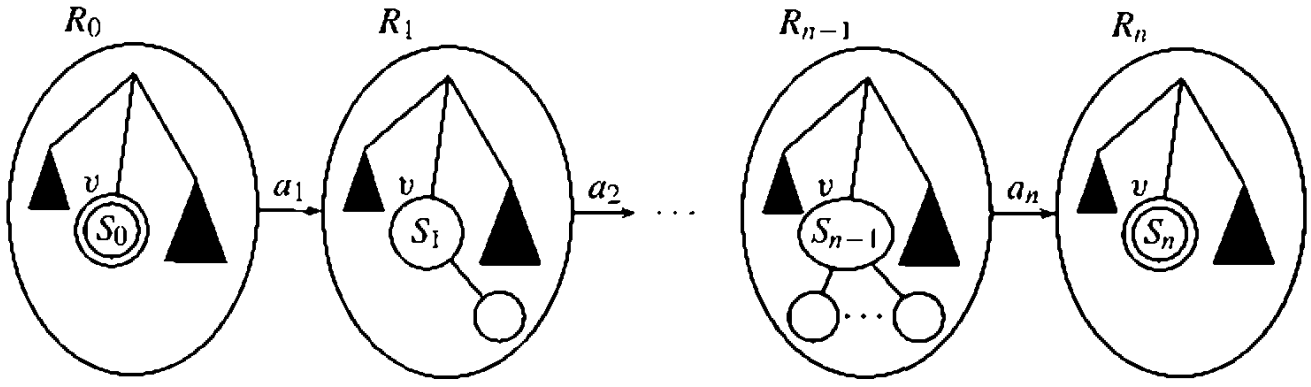


Figure 9.18. The states R_i.

Lemma 9.3. *For $0 \leq i \leq n-1$, S_{i+1} is contained in $\delta(S_i, a_{i+1})$. Moreover, for every $q \in S_n$, there is a path in $\mathcal{A}$ starting in S_0, ending at q, labeled u and visiting at least one final state after its origin.*

Proof. We follow the construction step by step. We first compute at step 1 the set $S_{i+1} = \delta(S_i, a_{i+1})$, then we suppress some states of S_{i+1} during step 3. The first part of the lemma follows.

Let us show by induction on i that, for $0 \leq i \leq n-1$ and for every state q_i appearing in the label of a descendant of v in R_i, there exists a path in $\mathcal{A}$ starting in S_0, ending with q_i, labeled $a_1 \cdots a_i$ and passing at least once more by a final state. The result holds for $i = 0$, since v, which is marked has to be a leaf of R_0 and has no descendants. On the other hand, if q_{i+1} appears in the label of a descendant of v in R_{i+1}, either $q_{i+1} \in \delta(q_i, a_{i+1})$ for some q_i appearing in the label of a strict descendant of v in R_i, and we conclude by induction, or q_{i+1} appears in a label created at step 2 and thus $q_{i+1} \in F$, which also allows one to conclude.

Finally, since v is marked in R_n, it received its mark at step 5. Thus if $q \in S_n$, either $q \in \delta(S_{n-1}, a_n) \cap F$, or q belongs to the union of the $\delta(q_{n-1}, a_n)$ where q_{n-1} appears in the label of a descendant of v in R_{n-1}. In the first case, there exists a path labeled u, starting in S_0 and ending with q, which is a final state. In the second case, we use the conclusion of the above induction: there exists a path in $\mathcal{A}$ starting in S_0, ending with q_{n-1}, labeled $a_1 \cdots a_{n-1}$ and passing at least once more by a final state. The lemma follows immediately. □

Consider now a successful path c in $\mathcal{D}$ and let $u \in A^\omega$ be the label of c. There exists a $v \in V$ such that, ultimately, the path visits only states in which v is a node and infinitely often states in which v is a marked node. Setting $S_0 = I$, there exists by Lemma 9.3 a factorization $u = u_0 u_1 u_2 \cdots$ and subsets S_n of Q, such that

(a) for every $n \geq 0$, $S_{n+1} \subset \delta(S_n, u_n)$,

(b) for every $n > 0$ and for every $q \in S_{n+1}$, there exists a path in $\mathcal{A}$ starting in S_n, ending with q, labeled u_n and visiting at least one final state after its origin.

In order to apply König's lemma, we build a tree (N, r, p) as follows. The set of nodes is

$$N = \{r\} \cup \{(q, n) \mid q \in S_n, n \in \mathbb{N}\}$$

The parent of each node of the form $(q, 0)$ is r and, for $n > 0$, the parent of each node of the form $(q, n+1)$ is chosen among the states (q', n) such that there is a path in $\mathcal{A}$ starting in q', ending in q, labeled u_n and visiting at least one final state after its origin. Conditions (a) and (b) guarantee the possibility of such a construction. Since the tree thus obtained is infinite and since each child has only a finite number of children, it contains an infinite path by König's lemma. This implies the existence of an infinite path in $\mathcal{A}$, labeled u, starting in I and passing infinitely often through a final state. Thus u is accepted by $\mathcal{A}$.

Conversely, let us consider a successful path c of the automaton $\mathcal{A}$

$$c : q_0 \xrightarrow{a_0} q_1 \xrightarrow{a_1} q_2 \cdots$$

There is a unique initial path in $\mathcal{D}$ with the same label u

$$d : I = R_0 \xrightarrow{a_0} R_1 \xrightarrow{a_1} R_2 \cdots$$

Each of the states q_i belongs to the label of the root of the trees R_i. This root is never suppressed and it is thus a fixed element v_0 of V. If v_0 is marked infinitely often in the R_i's, the path is successful in $\mathcal{D}$ and the word u is accepted. Otherwise, there is a largest integer n such that v_0 is marked in R_n. Let n_0 be this integer and let us consider the smallest integer $m > n_0$ such that q_m is an infinitely repeated final state. Since q_m is final, it appears in a child of the root, and from some time $n_1 \geq m$ on, each q_n with $n \geq n_1$ appears in a fixed child v_1 of the root of R_n. Indeed, if q_n occurs in the label of given node v, then q_{n+1} occurs again in the label of v at the next step, unless it occurs on the left of v (step 3). But such a left shift can occur only a finite number of times. If v_1 is marked infinitely often, the path is successful in $\mathcal{D}$. Otherwise, we repeat the same process, replacing v_0 by v_1. Since the tree has a finite height, we always find some node which is marked infinitely often. □

We shall see later other proofs of McNaughton's theorem which cast a different light upon it (see Section II.9). Among its numerous consequences, we begin with the most important one, known as Büchi's theorem.

Theorem 9.4. *The class of recognizable subsets of A^ω is closed under complement.*

Proof. By McNaughton's theorem, any recognizable set can be recognized by a Muller automaton. This automaton can be supposed to be complete by Proposition 7.3. Conversely, by Theorem 7.1 and Theorem 5.4 any set recognized by a Muller automaton is recognizable. The result follows from the fact that, by Proposition 7.6, the class of sets recognized by Muller automata is closed under all boolean operations. □

Büchi's theorem can also be proved directly using congruences (see Chapter II). But the size of the automaton for the complement given by Safra's algorithm is asymptotically optimal, as will be shown in Section 10.6 using the following result.

Theorem 9.5. *For each $n > 0$, there exists a set L_n of infinite words recognized by a Büchi automaton with $n + 2$ states, such that any Büchi automaton recognizing the complement of L_n has at least $n!$ states.*

Proof. Let $A_n = \{0, 1, \ldots, n\}$ and let $\mathcal{A}_n$ be the automaton on the alphabet A_n represented in Figure 9.19 and let $L_n = L^\omega(\mathcal{A}_n)$. One could of course describe precisely

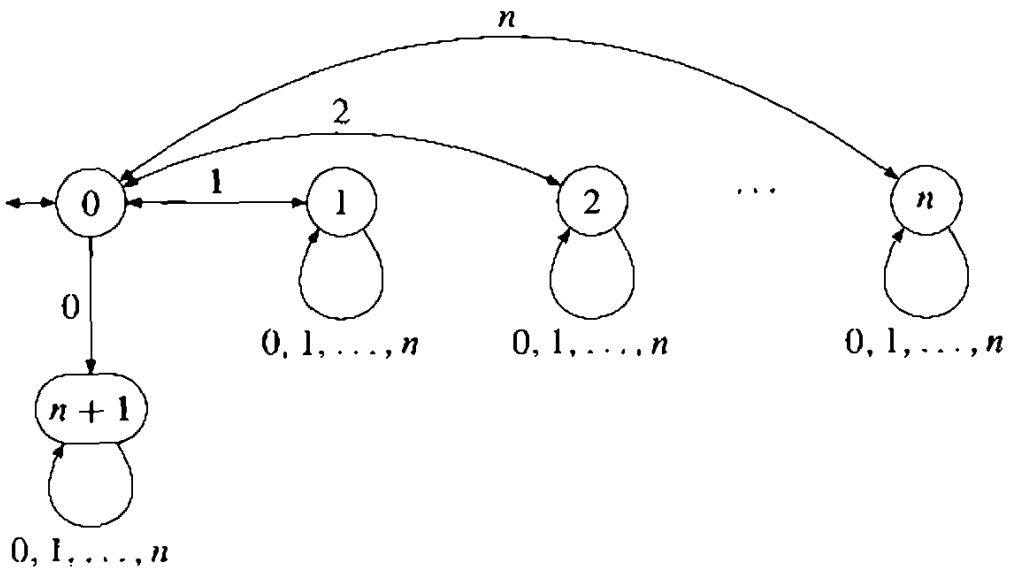


Figure 9.19. A Büchi automaton recognizing L_n.

L_n, but two weaker lemmas will be sufficient for our purpose. We start with a sufficient condition for a word to be in L_n.

Lemma 9.6. *Let $\{i_1, i_2, \ldots, i_k\}$ be a subset of $\{1, 2, \ldots, n\}$. If an infinite word u contains infinitely many occurrences of each of the factors i_1i_2, i_2i_3, ..., i_ki_1, and if in $\mathcal{A}_n$, there is a finite path from 1 to i_1 labeled by a prefix of u, then $u \in L_n$.*

Proof. It suffices to describe a successful path of label u in $\mathcal{A}_n$. By hypothesis, there is a path from 1 to i_1 labeled by a prefix of u. We then stay in state i_1 until the next occurrence of i_1i_2, that is used to produce the transitions $i_1 \xrightarrow{i_1} 0 \xrightarrow{i_2} i_2$. Then we stay in state i_2 until the next occurrence of i_2i_3, that is used to produce the transitions $i_2 \xrightarrow{i_2} 0 \xrightarrow{i_3} i_3$, etc. This process, repeated infinitely often on the cycle $(i_1i_2, i_2i_3, \ldots, i_ki_1)$, produces the desired successful path. □

With each permutation σ of $\{1, 2, \ldots, n\}$, associate the infinite word $u_\sigma = (\sigma(1) \cdots \sigma(n)0)^\omega$.

Lemma 9.7. *For any permutation σ of $\{1, \ldots, n\}$, the infinite word u_σ is not in L_n.*

Proof. Clearly, $L_n \subset K^\omega$, where $K = \bigcup_{1 \le i \le n} iA_n^*i$. Therefore, if $u_\sigma \in L_n$, $u = u_1u_2 \ldots$, where each u_i is in K. It follows that $\sigma(1)$ is the first and last letter of u_1, $\sigma(2)$ is the first and last letter of u_2, and $\sigma(n)$ is the first and last letter of u_n. Consequently, the first letter of u_{n+1} is 0, a contradiction, since $u_{n+1} \in K$. □

Let now $\mathcal{B}$ be a Büchi automaton accepting the complement of L_n. By Lemma 9.7, each word u_σ is accepted by $\mathcal{B}$. Therefore, there is in $\mathcal{B}$ a successful path p_σ of label u_σ.

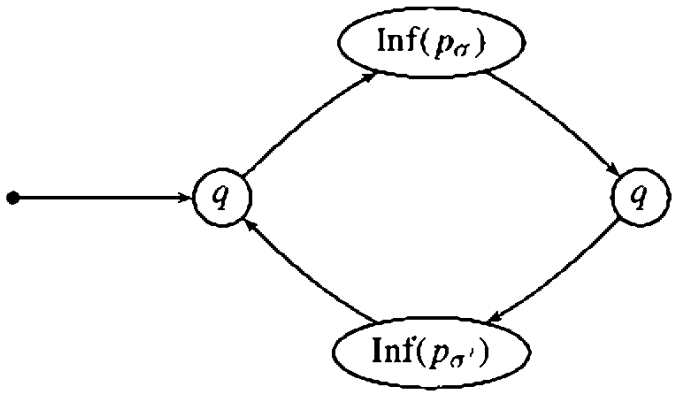


Figure 9.20. The path p.

We claim that if $\sigma \neq \sigma'$, then $\mathrm{Inf}(p_\sigma) \cap \mathrm{Inf}(p_{\sigma'}) = \emptyset$. Assume by contradiction that some state q belongs to both $\mathrm{Inf}(p_\sigma)$ and $\mathrm{Inf}(p_{\sigma'})$. Using the two paths, we build a new path p in $\mathcal{B}$ which, at the beginning, follows a prefix of p_σ of length at least $n(n+1)$ until it reaches q. Then p enters a loop which is repeated infinitely often. This loop consists of two parts that we also take of length at least $n+1$: in the first part, p follows a portion of p_σ to go from q to q after visiting at least once all states of $\mathrm{Inf}(p_\sigma)$ and in the second part, p follows a portion of $p_{\sigma'}$ to go from q to q after visiting at least once all states of $\mathrm{Inf}(p_{\sigma'})$ (see Figure 9.20). Then $\mathrm{Inf}(p)$ contains $\mathrm{Inf}(p_\sigma)$ (and $\mathrm{Inf}(p_{\sigma'})$) and in particular contains a final state, since p_σ is successful. It follows that p is successful and thus its label u is not in L_n. We shall arrive to a contradiction by showing that u satisfies the conditions of Lemma 9.6, and therefore belongs to L_n.

We first verify the existence of a cycle of infinitely repeated factors of length two. Let k be the smallest integer such that $\sigma(k) \neq \sigma'(k)$. Then $\sigma'(k) = \sigma(l)$ for some $l > k$ and $\sigma(k) = \sigma'(m)$ for some $m > k$. Since u is a concatenation of factors of length at least $n+1$ of u_σ and $u_{\sigma'}$, each of the factors $\sigma(k)\sigma(k+1)$, $\sigma(k+1)\sigma(k+2)$, ..., $\sigma(\ell-1)\sigma(\ell)(=\sigma(\ell-1)\sigma'(k))$, $\sigma'(k)\sigma'(k+1)$, ..., $\sigma'(m-1)\sigma'(m)(=\sigma'(m-1)\sigma(k))$ occur infinitely often in u.

It suffices now to verify that the state $\sigma(k)$ is reachable in $\mathcal{A}_n$ by a path labelled by a prefix of u. By construction, the word $(\sigma(1)\cdots\sigma(n)0)^n$ is a prefix of u. Therefore, the path

$$0 \xrightarrow{\sigma(1)} \sigma(1) \xrightarrow{\sigma(2)\cdots\sigma(n)0} \sigma(1) \xrightarrow{\sigma(1)} 0 \xrightarrow{\sigma(2)} \sigma(2) \cdots \sigma(k-1) \xrightarrow{\sigma(k-1)} 0 \xrightarrow{\sigma(k)} \sigma(k)$$

is suitable for our purpose.

This proves the claim, and since there are $n!$ permutations on $\{1, \ldots, n\}$, there are at least $n!$ disjoint sets of the form $\mathrm{Inf}(p_\sigma)$, which clearly implies that $\mathcal{B}$ has at least $n!$ states. □

As announced above, recognizable sets are determined by the ultimately periodic words they contain.

Corollary 9.8. *Let X and Y be two recognizable subsets of A^ω. Let $U \subset A^\omega$ be the set of ultimately periodic words. If $X \cap U \subset Y$, then $X \subset Y$. In particular $X = Y$ if and only if X and Y contain the same ultimately periodic words, i.e. if $X \cap U = Y \cap U$.*

Proof. In fact, if X is not contained in Y, the set $X \setminus Y$ is, by Büchi's theorem, a nonempty recognizable subset of A^ω. By Lemma 5.1, there exists an ultimately periodic word which is in X but not in Y. □

We now turn to another consequence of McNaughton's theorem, which solves a subtle point raised in Section 6.

Theorem 9.9. *A subset of A^ω is recognizable by a finite deterministic Büchi automaton if and only if it is both deterministic and recognizable.*

Proof. Any set recognized by a finite deterministic Büchi automaton satisfies certainly these two conditions. Conversely, let X be a subset of A^ω satisfying the two conditions. By McNaughton's theorem, the set X is recognized by a Muller automaton $\mathcal{A} = (Q, A, E, i, \mathcal{T})$. But since X is deterministic, the table $\mathcal{T}$ is full by Proposition 7.10. Finally, Proposition 7.9 shows that X can be recognized by a finite deterministic Büchi automaton. □

Corollary 9.10. *It is decidable whether a given recognizable subset of A^ω is deterministic or not.*

Proof. Let X be a recognizable subset of A^ω. We may build, using the previously described algorithms, a Muller automaton recognizing X. Proposition 7.10 allows one to conclude. □

Example 9.4. The set $X = (\{b, c\}^* a \cup b)^\omega$ of Example 9.3 is deterministic. On the contrary, the set $Y = (a\{b, c\}^* \cup b)^\omega$ is not deterministic. In fact Y is recognized by the Büchi automaton represented in Figure 9.21. The deterministic automaton obtained

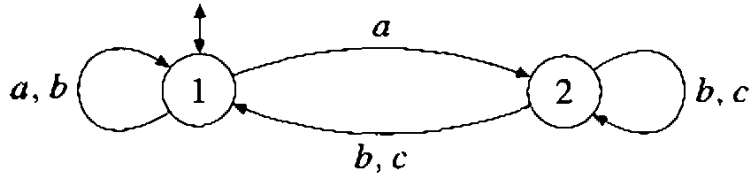


Figure 9.21. A co-deterministic but non deterministic automaton.

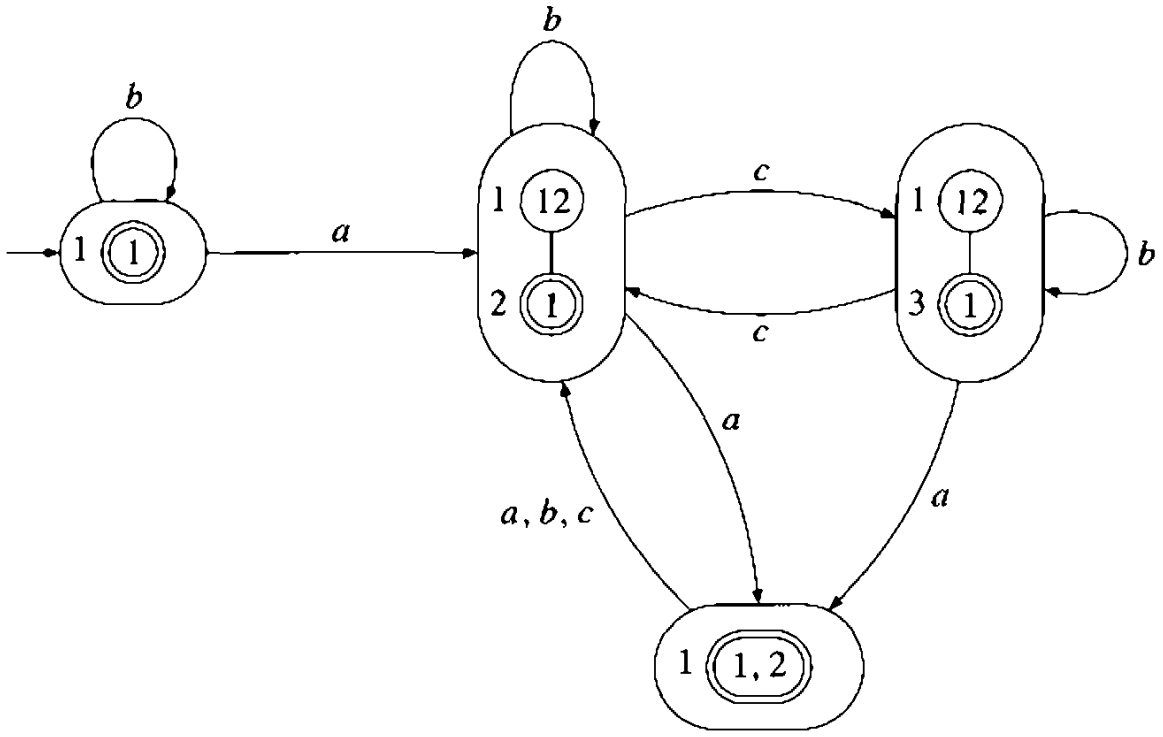


Figure 9.22. The resulting automaton.

by Safra's algorithm is represented in Figure 9.22 One can check directly that Y is not deterministic, by imitating the construction used in Example 6.2. Let us indeed suppose that $Y = \overrightarrow{L}$. Since $acb^\omega \in Y$, there is an integer n_1 such that $acb^{n_1} \in L$. Again, since $acb^{n_1}cb^\omega \in Y$, there is an integer n_2 such that $acb^{n_1}cb^{n_2} \in L$, etc. and the infinite word $u = acb^{n_1}cb^{n_2}cb^{n_3}\cdots$ has an infinite number of prefixes in L. This implies that $u \in Y$, which is impossible since u contains infinitely many c's but a finite number of a's.

10 Computational complexity issues

In this section, we address the problem of the computational complexity of the various transformations introduced in this chapter. The results are summarized in Figure 10.1. The nodes of this graph illustrate various representations of sets of infinite words, such as ω-rational expression, Büchi automaton, etc. An arrow between two nodes indicates an algorithm to convert one representation into another one. The label of the arrow indicates the complexity of the corresponding algorithm. The label P stands for a polynomial time algorithm and *Exp* for an exponential one.

The size of the various objects is defined according to the following conventions. As a general rule, we consider the cardinality of the alphabet as being a constant.

The *size* of an ω-rational expression is the number of symbols that it involves, without the parenthesis but taking the dot into account for the product. Thus size(ε) =