Chapter 4

Automata on Infinite Words

In Chapter 3 we have seen how standard techniques for manipulating automata on finite words can be used to verify (or falsify) safety properties. However, these techniques cannot be used to check liveness properties, because, by definition, such properties cannot be falsified by any finite word. In this section we introduce automata on infinite words, and show how many of the results of Chapter 3, like the closure under boolean operations, still hold for them.

Regular expressions define languages of finite words. We introduce ω-*regular expressions* to define languages of infinite words. An ω-regular expression over an alphabet Σ is one of the following

- r^ω, $r \cdot s_1$, or $s_1 \vee s_2$ for a regular expression r and ω-regular expressions s_1 and s_2.

We use $\mathcal{L}_\omega(s)$ to denote the language defined by an ω-regular expression s. The operators ω, $\vee$, and $\cdot$ stand for infinitary repetition, concatenation, and union, formally defined by

- $\mathcal{L}_\omega(r^\omega) = \{w_1 \cdot w_2 \cdot w_3 \cdots \mid w_i \in \mathcal{L}_\omega(r) \text{ for } i \geq 1\ \}$
- $\mathcal{L}_\omega(r \cdot s) = \{w_1 \cdot w_2 \colon w_1 \in \mathcal{L}(r) \text{ and } w_2 \in \mathcal{L}_\omega(s)\}$
- $\mathcal{L}_\omega(s_1 \vee s_2) = \mathcal{L}_\omega(s_1) \cup \mathcal{L}_\omega(s_2)$

Suppose now that an automaton $\mathcal{A} = (\Sigma, Q, Q^0, \delta, \alpha)$ is given as input an infinite word $w = \sigma_0, \sigma_1, \ldots$ over Σ. A *run* ρ of $\mathcal{A}$ on w is a sequence $q_0, q_1, \ldots$, where $q_0 \in Q^0$ and $q_{i+1} \in \delta(q_i, \sigma_i)$, for all $i \geq 0$. Since the run is infinite, we cannot define acceptance by the type of the final state of the run. Instead we have to consider the *limit* behavior of the run. We define $\inf(\rho)$

to be the set $\{q \mid q = q_i$ for infinitely many i's$\}$, i.e., the set of states that occur in r infinitely often. Since Q is finite, $\inf(\rho)$ is necessarily nonempty. The run ρ is *accepting* if there is some accepting state that repeats in ρ infinitely often, i.e., $\inf(\rho) \cap \alpha \neq \emptyset$. If we picture the automaton as having a green light that is switched on precisely when the automaton is in an accepting state, then the run is accepting if the green light is switched on infinitely many times. The infinite word w is *accepted* by $\mathcal{A}$ if there is an accepting run of $\mathcal{A}$ on w. The infinitary language of $\mathcal{A}$, denoted $\mathcal{L}_\omega(A)$, is the set of infinite words accepted by $\mathcal{A}$.

Thus, $\mathcal{A}$ can be viewed both as an automaton on finite words and as an automaton on infinite words. When viewed as an automaton on infinite words it is called a *Büchi* automaton after J.R. Büchi, who introduced them in [**?**]. A language $L \subseteq \Sigma^\omega$ is *ω-regular* if it is recognized by a Büchi automaton $\mathcal{A}$, i.e., if $\mathcal{L}_\omega(\mathcal{A}) = L$.

Figure 4.1 shows automata for the following infinitary languages over the alphabet $\{a, b\}$: words containing infinitely many occurrences of a; words containing finitely many occurrences of a; words containing infinitely many occurrences of both a and b.

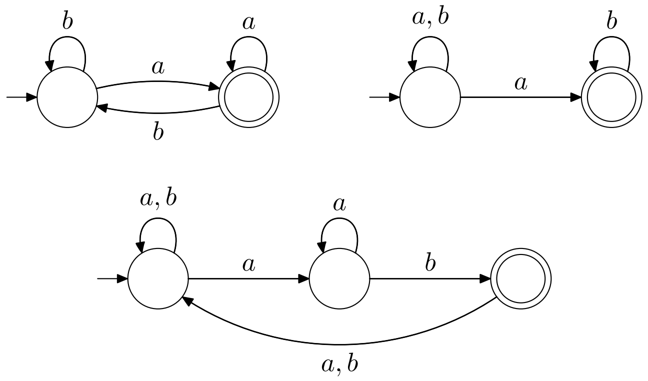


Figure 4.1: Three Büchi automata

4.1 Closure Properties

Do automata on infinite words have closure properties similar to those of automata on finite words? In most cases the answer is positive, but the proofs may be more involved. In this section we consider closure under boolean operators.

4.1.1 Closure under union and intersection

Closure under union is easy to prove, because the construction of Proposition 3.1.1 also works for Büchi automata.

Proposition 4.1.1 [?] *Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be Büchi automata. Then $\mathcal{L}_\omega(\mathcal{A}_1 \cup \mathcal{A}_2) = \mathcal{L}_\omega(\mathcal{A}_1) \cup \mathcal{L}_\omega(\mathcal{A}_2)$.*

One might be tempted to think that similarly we have $\mathcal{L}_\omega(\mathcal{A}_1 \times \mathcal{A}_2) = \mathcal{L}_\omega(\mathcal{A}_1) \cap \mathcal{L}_\omega(\mathcal{A}_2)$, but this is not the case. Consider the two Büchi automata $\mathcal{A}_1$ and $\mathcal{A}_2$ of Figure 4.2. The automaton $\mathcal{A}_1 \times \mathcal{A}_2$ is shown in Figure 4.3.

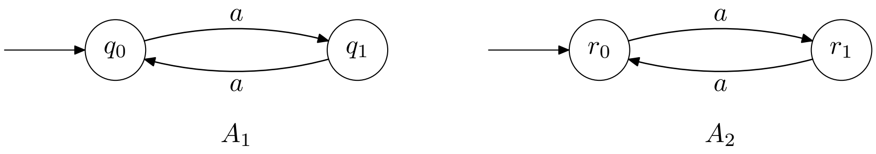


Figure 4.2: Two Büchi automata accepting the language a^ω

Clearly we have that $\mathcal{L}_\omega(\mathcal{A}_1) = \mathcal{L}_\omega(\mathcal{A}_2) = \{a^\omega\}$, but $\mathcal{L}_\omega(\mathcal{A}_1 \times \mathcal{A}_2) = \emptyset$.

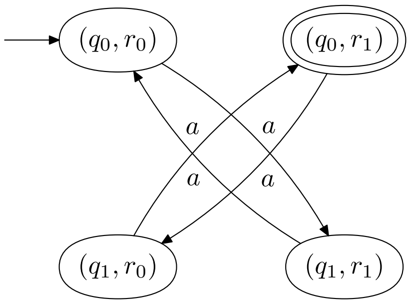


Figure 4.3: The automaton $\mathcal{A}_1 \times \mathcal{A}_2$

What happened? The automaton $\mathcal{A}$ runs both $\mathcal{A}_1$ and $\mathcal{A}_2$ on the input word. Since the accepting set of $\mathcal{A}_1 \times \mathcal{A}_2$ is the product of the accepting sets of $\mathcal{A}_1$ and $\mathcal{A}_2$. $\mathcal{A}_1 \times \mathcal{A}_2$ accepts an infinite word w if there are accepting runs r_1 and r_2 of $\mathcal{A}_1$ and $\mathcal{A}_2$, respectively, on w, where both runs go infinitely often and *simultaneously* through accepting states. This requirement is too strong. As a result, $\mathcal{L}_\omega(\mathcal{A}_1 \times \mathcal{A}_2)$ could be a strict subset of $\mathcal{L}_\omega(A_1) \cap \mathcal{L}_\omega(A_2)$.

Nevertheless, closure under intersection does hold. The key is to observe that the requirement "r_1 and r_2 go infinitely often through accepting states" is equivalent to

- r_1 goes at least once through an accepting state, and
- if r_1 goes through an accepting state then r_2 goes *later* through an accepting state, and
- if r_2 goes through an accepting state then r_1 goes *later* through an accepting state.

The automaton accepting the intersection, described in the next proposition, checks this property for runs r_1 and r_2 on the same word.

Proposition 4.1.2 [**?**] *Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be Büchi automata. Then there is a Büchi automaton $\mathcal{A}$ such that $\mathcal{L}_\omega(\mathcal{A}) = \mathcal{L}_\omega(\mathcal{A}_1) \cap \mathcal{L}_\omega(\mathcal{A}_2)$.*

Proof: Let $\mathcal{A}_1 = (\Sigma, Q_1, Q_1^0, \delta_1, \alpha_1)$ and $\mathcal{A}_2 = (\Sigma, Q_2, Q_2^0, \delta_2, \alpha_2)$. Let $\mathcal{A} = (\Sigma, Q, Q^0, \delta, \alpha)$, where

- $Q = Q_1 \times Q_2 \times \{1, 2\}$,
- $Q^0 = Q_1^0 \times Q_2^0 \times \{1\}$,
- $\delta(\langle s, t, i\rangle, a) = \delta_i(s, a) \times \delta_2(t, a) \times \{j\}$, where $i = j$ unless $i = 1$ and $s \in \alpha_1$, in which case $j = 2$, or $i = 2$ and $t \in \alpha_2$, in which case $j = 1$, and
- $\alpha = \alpha_1 \times Q_2 \times \{1\}$.

The automaton $\mathcal{A}$ for the automata $\mathcal{A}_1$ and $\mathcal{A}_2$ of Figure 4.2 is shown in Figure 4.4. The second track is depicted on top of the first for better readability. It can be viewed has having two "tracks", one for each of $\mathcal{A}_1$ and $\mathcal{A}_2$. When executing a run, if we are in the first track (the second track), and we visit an accepting state of $\mathcal{A}_1$ ($\mathcal{A}_2$), we move to the other track. The accepting condition requires to go infinitely often through $\alpha_1 \times Q_2 \times \{1\}$, i.e., to visit infinitely often accepting states of $\mathcal{A}_1$ in the first track. Since we can only return to the first track after having visited an accepting state of $\mathcal{A}_2$ on the second track, both tracks visit accepting states infinitely often. $\square$

Thus, Büchi automata are closed under both union and intersection, though the construction for intersection is somewhat more involved than the simple product of Proposition 4.1.2. However, there is an important special case in which the simple product still works.

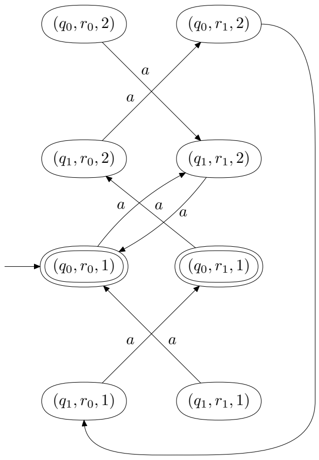


Figure 4.4: The automaton $\mathcal{A}$ for the automata $\mathcal{A}_1$ and $\mathcal{A}_2$ of Figure 4.2

Proposition 4.1.3 [?] *Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be Büchi automata such that all states of $\mathcal{A}_1$ are accepting states. Then $\mathcal{L}_\omega(\mathcal{A}_1 \times \mathcal{A}_2) = \mathcal{L}_\omega(\mathcal{A}_1) \cap \mathcal{L}_\omega(\mathcal{A}_2)$.*

Proof: Let r_1 and r_2 be two runs of $\mathcal{A}_1$ and $\mathcal{A}_2$, respectively, on a word w. Since r_1 only goes through accepting states, r_2 goes through an accepting state if and only if both r_1 and r_2 go simultaneously through accepting states. $\square$

Exercise 20. Given Büchi automata $\mathcal{A}_1, \ldots, \mathcal{A}_n$, the construction of Proposition 4.1.2 can be applied $n-1$ times to yield an automaton $\mathcal{A}$ such that $\mathcal{L}_\omega(\mathcal{A}) = \mathcal{L}_\omega(\mathcal{A}_1) \cap \ldots \cap \mathcal{L}_\omega(\mathcal{A}_n)$. However, if the automaton $\mathcal{A}_i$ has k_i states, the automaton $\mathcal{A}$ has $2^{n-1} \cdot k_1 \cdot \ldots \cdot k_n$ states. Give a better construction such that $\mathcal{A}$ has $n \cdot k_1 \cdot \ldots \cdot k_n$ states.

4.1.2 The problems of complement

So far we have been able to adapt the constructions for automata on finite words to Büchi automata. The situation is considerably more involved with respect to closure under complementation. In the case of finite automata, the complement of an automaton is constructed in two steps. First, an equivalent deterministic automaton is obtained by means of the subset construction of Proposition 3.1.5; second, this automaton is complemented by exchanging final and non-final states, as shown in Proposition 3.1.4. In the case of automata on infinite words, this approach does not work at all, for the following reasons.

(a) The subset construction does not preserve the accepted infinitary language; i.e, a Büchi automaton and the result of applying the subset construction to it do not necessarily accept the same infinitary language.

    Exercise 21. Prove this.

(b) The subset construction cannot be replaced by another determinization procedure, because no such procedure exists: Some languages that are accepted by nondeterministic but not by deterministic Büchi automata, as shown in Proposition 4.1.4 below.

(c) The automaton obtained by exchanging accepting and non-accepting states in deterministic Büchi automata does not necessarily recognize the complementary infinitary language.

    Exercise 22. Prove this.

Proposition 4.1.4 *The language $L = (a \vee b)^* b^\omega$, (i.e., L consists of all infinite words in which a occurs only finitely many times) is recognized by nondeterministic but not by deterministic Büchi automata.*

Proof: L is defined by the nondeterministic Büchi automaton of Figure 4.5. As long as it is in the state q_0, the automaton can read both inputs a and b. At some point, however, it makes a nondeterministic transition to q_1, from which it can read only the input b.

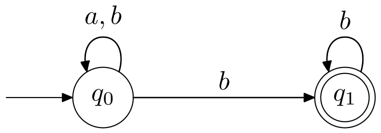


Figure 4.5: A nondeterministic Büchi automaton accepting $(a \vee b)^* b^\omega$

We now show that L is not defined by any deterministic Büchi automaton. Since it suffices to consider automata with one initial state, assume by way of contradiction that $L = \mathcal{L}_\omega(A)$, where $\mathcal{A} = (\{a, b\}, Q, \{q_0\}, \delta, \alpha)$, and $\mathcal{A}$ is deterministic. Recall that δ can be viewed as a partial mapping from $Q \times \{a, b\}^*$ to Q.

Consider the infinite word $w_0 = b^\omega$. Clearly, w_0 is accepted by $\mathcal{A}$, so $\mathcal{A}$ has an accepting run on w_0. Thus, w_0 has a finite prefix u_0 such that $\delta(q_0, u_0) \in \alpha$. Consider now the infinite word $w_1 = u_0 a b^\omega$. Clearly, w_1 is also accepted by $\mathcal{A}$, so $\mathcal{A}$ has an accepting run on w_1. Thus, w_1 has a finite prefix $u_0 b u_1$ such that $\delta(q_0, u_0 a u_1) \in \alpha$. In a similar fashion we can continue to find finite words u_i such that $\delta(q_0, u_0 a u_1 a \dots a u_i) \in \alpha$. Since Q is finite, there are i, j, where $0 \leq i < j$, such that $\delta(q_0, u_0 a u_1 a \dots a u_i) = \delta(q_0, u_0 a u_1 a \dots a u_i a \dots a u_j)$. It follows that $\mathcal{A}$ has an accepting run on

$$u_0 a u_1 a \dots a u_i (a u_{i+1} \dots u_{j-1} a u_j)^\omega.$$

But the latter word has infinitely many occurrences of a, so it is not in L. $\square$

Note that the complementary language $(a \vee b)^\omega \setminus L = ((a \vee b)^* a)^\omega$ (the set of infinite words in which a occurs infinitely often) is accepted by the deterministic Büchi automaton of Figure 4.6. The automaton remembers the last symbol it read (q_0 corresponds to a and q_1 corresponds to b).

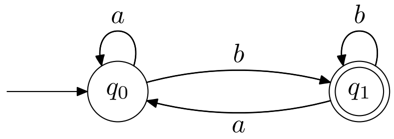


Figure 4.6: A deterministic Büchi automaton accepting $((a \vee b)^* a)^\omega$

Exercise 23. A Büchi automaton is *deterministic in the limit* if all the accepting states and their descendants are deterministic states. Formally, $\mathcal{A} = (\Sigma, Q, Q^0, \delta, \alpha)$ is deterministic in the limit if $|\delta(q, \sigma)| \leq 1$ for every state $q \in Q$ that is reachable from some state of α, and for every $\sigma \in \Sigma$.

Prove that every language recognized by nondeterministic Büchi automata is also accepted by Büchi automata deterministic in the limit.

Despite these discouraging observations, Büchi automata turn out to be closed under complement. The proof of this remarkable result is contained in the next section.

4.1.3 Closure under complement

Fix a Büchi automaton $\mathcal{A} = (\Sigma, Q, Q_0, \delta, \alpha)$ with n states. We prove the existence of a Büchi automaton $comp(\mathcal{A})$ such that $\mathcal{L}_\omega(comp(\mathcal{A})) = comp(\mathcal{L}_\omega(\mathcal{A}))$ in three steps.

1. We define a mapping *dag* which assigns to each word $w \in \Sigma^\omega$ a directed acyclic graph $dag(w)$. We also define an *odd ranking* of $dag(w)$ as a labelling of the nodes of $dag(w)$ by natural numbers satisfying certain properties.

2. We prove that w is rejected by $\mathcal{A}$ if and only if $dag(w)$ admits an odd ranking.

3. We construct a Büchi automaton $comp(\mathcal{A})$ which accepts w if and only if $dag(w)$ admits an odd ranking.

These steps are carried out in the next three sections. We use the automaton of Figure 4.7 as running example.

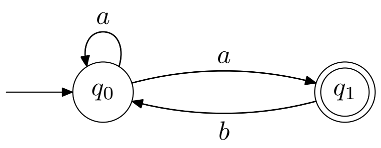


Figure 4.7: Running example for the complementation proof

The mapping *dag*, rankings, and good rankings

Let $w = \sigma_1\sigma_2 \ldots$ be a word of Σ^ω. The directed acyclic graph $dag(w)$ has nodes in $Q \times \mathbb{N}$ and edges labelled by letters of Σ; it is inductively defined as follows:

- $dag(w)$ contains a node $\langle q, 0\rangle$ for every initial state $q \in Q_0$.

- If $dag(w)$ contains a node $\langle q, i\rangle$ and $q' \in \delta(\sigma_{i+1}, q)$, then $dag(w)$ also contains a node $\langle q', i+1\rangle$ and an edge $\langle\langle q, i\rangle, \langle q', i+1\rangle\rangle$ labelled by σ_{i+1}.

- $dag(w)$ contains no other nodes or edges.

Loosely speaking, $dag(w)$ is the result of bundling together all possible runs of $\mathcal{A}$ on the word w. Formally, $q_0q_1q_2\ldots$ is a run of $\mathcal{A}$ if and only if $\langle q_0, 0\rangle\langle q_1, 1\rangle\langle q_2, 2\rangle \ldots$ is a path of $dag(w)$. Figure 4.8 shows the initial parts of $dag(aba^\omega)$ and $dag((ab)^\omega)$ for our running example. Nodes corresponding to the final state q_1 are drawn with a double circle.

We introduce some basic terminology. A node $\langle q, i\rangle$ is *accepting* if q is an accepting state. A path is accepting if it contains infinitely many accepting

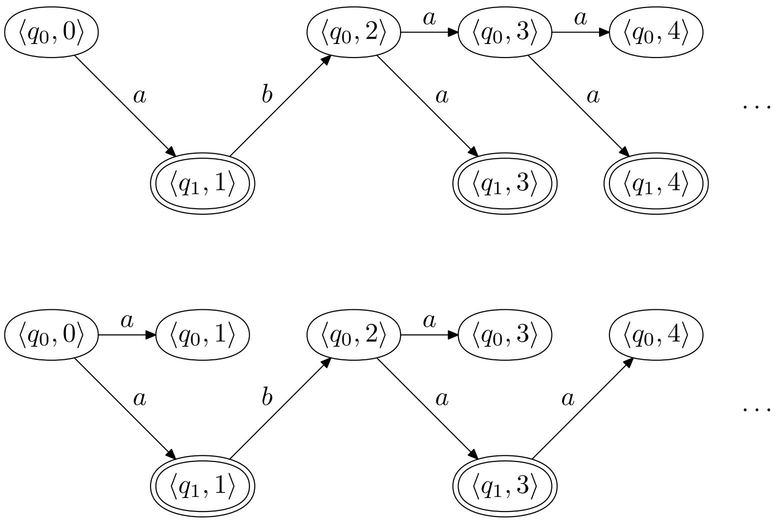


Figure 4.8: The (initial parts of) $dag(aba^\omega)$ and $dag((ab)^\omega)$

nodes. If there is a path from $\langle q, i\rangle$ to $\langle q', j\rangle$, then $\langle q', j\rangle$ is *reachable* from $\langle q, i\rangle$. This path may possibly contain no edges, and so in particular a node is a successor of itself. If the path contains exactly one edge, i.e., if $j = i+1$, then we say that $\langle q', j\rangle$ is a *child* of $\langle q, i\rangle$.

A *ranking* of $dag(w)$ is a mapping that associates to each node a natural number in the range $[0 \ldots 2n]$, called a *rank*, satisfying the following two properties:

- the rank of a node is greater than or equal to the rank of its children, and
- the rank of an accepting node is even.

The ranks of the nodes in an infinite path form a non-increasing sequence, and so there is a node such that all its (infinitely many) successors have the same rank; we call this number the *stable rank* of the path. A ranking is *odd* if all infinite paths have odd stable rank. Figure 4.9 shows rankings for $dag(aba^\omega)$ and $dag((ab)^\omega)$. The ranking for $dag(aba^\omega)$ is odd, since the only infinite path has stable rank 1. The ranking for $dag((ab)^\omega)$ is not odd, because the only infinite path has stable rank 0.

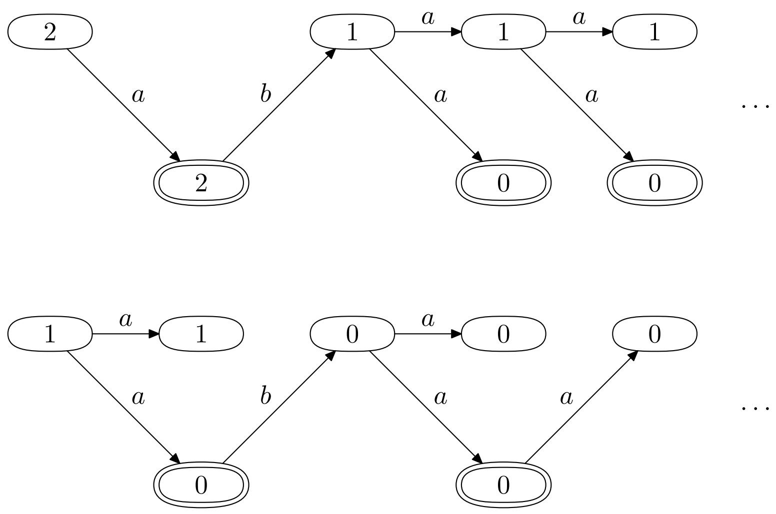


Figure 4.9: Rankings for $dag(aba^\omega)$ and $dag((ab)^\omega)$

Acceptance and good rankings

We prove that w is rejected by $\mathcal{A}$ if and only if $dag(w)$ admits an odd ranking. The 'if' half is easy, while the 'only if' part is more involved.

Exercise 24. Show: If $dag(w)$ admits an odd ranking, then w is rejected by $\mathcal{A}$.

Proposition 4.1.5 *If w is rejected by $\mathcal{A}$, then $dag(w)$ admits an odd ranking.*

Proof: In the sequel, given two DAGs D, D', we denote by $D' \subseteq D$ the fact that D' can be obtained from D through deletion of some nodes and their adjacent edges.

Assume that $\mathcal{A}$ rejects w. We describe an odd ranking for $dag(w)$. We say that a node $\langle q, l \rangle$ is *endangered* in a (possibly finite) DAG $D \subseteq dag(w)$ iff only finitely many nodes of D are reachable from $\langle q, l \rangle$. The node $\langle q, l \rangle$ is *safe* in D iff all the nodes in D that are reachable from $\langle q, l \rangle$ are not accepting. In particular, safe nodes are not accepting. We inductively define an infinite sequence $D_0 \supseteq D_1 \supseteq D_2 \supseteq \ldots$ of DAGs by:

- $D_0 = dag(w)$;
- $D_{2i+1} = D_{2i} \setminus \{\langle q, l \rangle \mid \langle q, l \rangle \text{ is endangered in } D_{2i}\}$;

- $D_{2i+2} = D_{2i+1} \setminus \{\langle q, l\rangle \mid \langle q, l\rangle \text{ is safe in } D_{2i+1}\}$.

Figure 4.10 shows D_0, D_1, and D_2 for $dag(aba^\omega)$. D_3 is the empty dag.

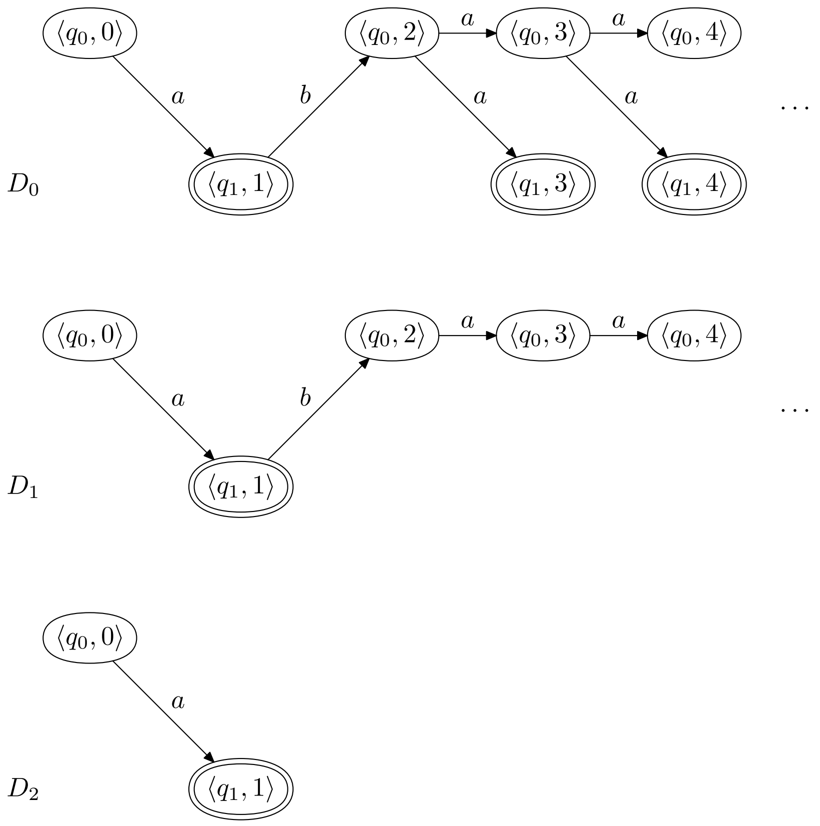


Figure 4.10: The DAGs D_0, D_1, D_2 for $dag(aba^\omega)$

Consider the function f that assigns to each node of $dag(w)$ a natural number as follows:

$$f(\langle q, l\rangle) = \begin{cases} 2i & \text{if } \langle q, l\rangle \text{ is endangered in } D_{2i} \\ 2i+1 & \text{if } \langle q, l\rangle \text{ is safe in } D_{2i+1} \end{cases}$$

We prove that f is an odd ranking. The proof is divided into three parts:

(1) f assigns all nodes a number in the range $[0 \ldots 2n]$.

(2) If $\langle q', l'\rangle$ is a child of $\langle q, l\rangle$, then $f(\langle q', l'\rangle) \leq f(\langle q, l\rangle)$.

(3) If $\langle q, l\rangle$ is an accepting node, then $f(\langle q, l\rangle)$ is even.

We leave parts (2) and (3) as exercises.

Exercise 25. Prove that f satisfies condition (2) above. Hint: prove first that for every node $\langle q, l\rangle$ of $dag(w)$, if $\langle q, l\rangle \notin D_i$, then $f(q, l) < i$.
Exercise 26. Prove that f satisfies condition (3) above. For part (1), we

show that for every $i \geq 0$ there exists a number l_i such that for all $l \geq l_i$, the DAG D_{2i} contains at most $n - i$ nodes of the form $\langle q, l\rangle$. This implies that D_{2n} is finite, and so that D_{2n+1} is empty, which in turn implies that f assigns all nodes a number in the range $[0 \ldots 2n]$.

The proof is by an induction on i. The case where $i = 0$ follows from the definition of G_0. Indeed, in $dag(w)$ all levels $l \geq 0$ have at most n nodes of the form $\langle q, l\rangle$. Assume now that the hypothesis holds for i; we prove it for $i+1$. Consider the DAG D_{2i}. We distinguish between two cases. First, if D_{2i} is finite, then D_{2i+1} is empty; D_{2i+2} is empty as well, and we are done. Otherwise, we claim that there must be some safe node in D_{2i+1}. To see this, assume, by way of contradiction, that D_{2i} is infinite and no node in D_{2i+1} is safe. Since D_{2i} is infinite, D_{2i+1} is also infinite. Moreover, since D_{2i+1} is obtained by removing all endangered nodes from D_{2i}, every node of D_{2i+1} has at least one child. Let $\langle q_0, l_0\rangle$ be an arbitrary node of D_{2i+1}. Since, by the assumption, it is not safe, there exists an accepting node $\langle q_0', l_0'\rangle$ reachable from $\langle q_0, l_0\rangle$. Let $\langle q_1, l_1\rangle$ be a child of $\langle q_0', l_0'\rangle$. By the assumption, $\langle q_1, l_1\rangle$ is also not safe. Hence, there exists an accepting node $\langle q_1', l_1'\rangle$ reachable from $\langle q_1, l_1\rangle$. Let $\langle q_2, l_2\rangle$ be a child of $\langle q_1', j_1'\rangle$. By the assumption, $\langle q_2, l_2\rangle$ is also not safe. We can thus construct an infinite sequence of nodes $\langle q_j, l_j\rangle$, $\langle q_j', l_j'\rangle$ such that for all i the node $\langle q_j', l_j'\rangle$ is accepting, reachable from $\langle q_j, l_j\rangle$, and $\langle q_{j+1}, l_{j+1}\rangle$ is a child of $\langle q_j', l_j'\rangle$. Such a sequence, however, corresponds to a path in $dag(w)$ visiting infinitely many accepting nodes, which contradicts the assumption that $\mathcal{A}$ rejects w.

So, let $\langle q, l\rangle$ be a safe node in D_{2i+1}. We claim that we can take $l_{i+1} = l$. That is, we claim that for all $j \geq l$, there are at most $n - (i+1)$ nodes of the form $\langle q, j\rangle$ in D_{2i+2}. Since $\langle q, l\rangle$ is in D_{2i+1}, it is not endangered in D_{2i}. Thus, there are infinitely many nodes in D_{2i} that are reachable from $\langle q, l\rangle$. Hence, by König's Lemma, D_{2i} contains an infinite path $\langle q, l\rangle, \langle q_1, l+1\rangle, \langle q_2, l+2\rangle, \ldots$. For all $k \geq 1$, D_{2i} contains infinitely many nodes reachable from $\langle q_k, l+k\rangle$, and thus $\langle q_k, l+k\rangle$ is not endangered in D_{2i}. Therefore, the path $\langle q, l\rangle, \langle q_1, l+1\rangle, \langle q_2, l+2\rangle, \ldots$ exists also in D_{2i+1}. Recall that $\langle q, l\rangle$ is safe. Hence, being reachable from $\langle q, l\rangle$, all the nodes $\langle q_k, l+k\rangle$ in the path are safe as well. Therefore, they are not in D_{2i+2}. It follows that for all $j \geq l$ the number of nodes of the form $\langle q, j\rangle$ in

D_{2i+2} is strictly smaller than their number in D_{2i}, which, by the induction hypothesis, is $n-i$. So there are at most $n-(i+1)$ nodes of the form $\langle q, j\rangle$ in D_{2i+2}. $\square$

The automaton $comp(\mathcal{A})$

By Exercise 24 and Proposition 4.1.5, an automaton that complements $\mathcal{A}$ can proceed on an input word w by guessing an odd ranking for the run DAG of $\mathcal{A}$ on w. We now define such an automaton $comp(\mathcal{A})$ formally. We need some preliminaries.

A *level ranking* for $\mathcal{A}$ and w is a function $g : Q \to [0..2n] \cup \{\bot\}$, such that if $g(q)$ is odd, then $q \notin \alpha$. Let $\mathcal{R}$ be the set of all level rankings. Any ranking r of $dag(w)$ can be decomposed into an infinite sequence $lr_1, lr_2, \ldots$ of level rankings by defining $lr_i(q) = r(\langle q, i\rangle)$ if $\langle q, i\rangle$ is a node of $dag(w)$, and $lr_i(q) = \bot$ otherwise. This allows us to look at a ranking as an infinite word over the alphabet of level rankings. For example, if we represent a level ranking lr of our running example by the column vector

$$\begin{bmatrix} lr(q_0) \\ lr(q_1) \end{bmatrix},$$

then the rankings of Figure 4.9 correspond to the words

$$\begin{bmatrix} 2 \\ \bot \end{bmatrix}\begin{bmatrix} \bot \\ 2 \end{bmatrix}\begin{bmatrix} 1 \\ \bot \end{bmatrix}\begin{bmatrix} 1 \\ 0 \end{bmatrix}^{\omega}$$

$$\begin{bmatrix} 1 \\ \bot \end{bmatrix}\begin{bmatrix} 1 \\ 0 \end{bmatrix}\left(\begin{bmatrix} 0 \\ \bot \end{bmatrix}\begin{bmatrix} 0 \\ 0 \end{bmatrix}\right)^{\omega}$$

For two level rankings g and g', we say that g' *covers* g if for all q and q' in Q, if $g(q) \geq 0$ and $q' \in \delta(q, \sigma)$, then $0 \leq g'(q') \leq g(q)$.

The automaton $comp(\mathcal{A}) = (\overline{\Sigma}, \overline{Q}, \overline{q}_0, \overline{\delta}, \overline{\alpha})$ is formally defined as follows:

- $\overline{\Sigma} = \Sigma$, $\overline{Q} = \mathcal{R} \times 2^Q$, $\overline{\alpha} = \mathcal{R} \times \emptyset$.
- $\overline{Q}_0 = \langle lr_0, \emptyset\rangle$, where $lr_0(q) = 2n$ if $q \in Q_0$, and $lr_0(q) = \bot$ otherwise.
- $\langle lr', P'\rangle \in \overline{\delta}(\langle lr, P\rangle, a)$ iff $\langle lr', P'\rangle$ satisfies the following three conditions:
  - lr' covers lr;
  - if $P \neq \emptyset$, then $P' = \{q' \in Q \mid q' \in \delta(q, a)$ for some $q \in P$ and $lr'(q')$ is even$\}$;

– if $P = \emptyset$, then $P' = \{q' \in Q \mid lr'(q') \text{ is even}\}$.

Thus, when $comp(\mathcal{A})$ reads the lth letter in the input, for $l \geq 1$, it guesses the level ranking for level l in the run DAG. This level ranking should cover the level ranking of level $l-1$. In addition, in the P component, $comp(\mathcal{A})$ keeps track of states whose corresponding nodes at the current level have even ranks. Paths that traverse such nodes should eventually reach a node with an odd rank. When all the paths of the DAG have visited a node with an odd rank, the set P becomes empty, and is initiated by new obligations for visits in odd ranks according to the current level ranking. The acceptance condition $\mathcal{R} \times \{\emptyset\}$ then checks that there are infinitely many levels in which all the obligations have been fulfilled.

Exercise 27. Construct the automata $comp(\mathcal{A})_1$ and $comp(\mathcal{A})_2$ for $(\{a\}, \{q\}, \{q\}, \delta, \{q\})$ and $(\{a\}, \{q\}, \{q\}, \delta, \emptyset)$, where $\delta(q, a) = \{q\}$.

The size of $comp(\mathcal{A})$

Let n be the number of states of $\mathcal{A}$. Since a level ranking is a mapping $lr\colon Q \rightarrow \{\perp\} \cup [0, 2n]$, there are at most $(2n+2)^n$ level rankings. So $comp(\mathcal{A})$ has at most $(2n+2)^n \cdot 2^n \in n^{O(n)}$ states. Since $n^{O(n)} = 2^{O(n \cdot \log n)}$, we have introduced an extra $\log n$ factor in the exponent with respect to the subset construction for automata on finite words. We show that this factor is unavoidable. We exhibit a family $\{L_n\}_{n \geq 1}$ of infinitary languages such that L_n is accepted by an automaton with $n+2$ states and any Büchi automaton accepting the complement of L_n has at least $n! \in n^{O(n)}$ states. The family is due to Michel [**?**].

Let $\Sigma_n = \{1, \ldots, n, \#\}$. We associate to a word $w \in \Sigma_n^\omega$ the following directed graph $G(w)$: the nodes of $G(w)$ are $1, \ldots, n$ and there is an edge from i to j if w contains infinitely many occurrences of the word ij. Define L_n as the language of infinite words $w \in A^\omega$ for which $G(w)$ has at least a cycle and define $comp(L)_n$ as the complement of L_n.

Exercise 28. Show that for all $n \geq 1$, L_n is recognized by a Büchi automaton with $n+2$ states.

Proposition 4.1.6 *For all $n \geq 1$, if a Büchi automaton recognizes $comp(L)_n$, then it has at least $n!$ states.*

Proof: We need some preliminaries. Given a permutation $\tau = \langle \tau(1), \ldots, \tau(n) \rangle$ of $\langle 1, \ldots, n \rangle$, we identify τ and the word $\tau(1) \ldots \tau(n)$. We make two observations:

(a) $(\tau\#)^\omega \in comp(L)_n$ for every permutation τ.
The edges of $G((\tau\#)^\omega)$ are $\langle\tau(1), \tau(a_2)\rangle$, $\langle\tau(a_2), \tau(a_3)\rangle$, …, $\langle\tau(a_{n-1}), \tau(a_n)\rangle$, and so $G((\tau\#)^\omega)$ is acyclic.

(b) If a word w contains infinitely many occurrences of two different permutations τ and τ' of $1 \ldots n$, then $w \in L_n$.
Since τ and τ' are different, there are i and j in $\{1, \ldots, n\}$ such that i precedes j in τ and j precedes i in τ'. Since w contains infinitely many occurrences of τ, $G(w)$ has a path from i to j. Since it contains infinitely many occurrences of τ', $G(w)$ has a path from j to i. So $G(w)$ contains a cycle, and so $w \in L_n$.

Now, let $\mathcal{A}$ be a Büchi automaton recognizing $comp(L)_n$, and let τ, τ' be two arbitrary permutations of $(1, \ldots, n)$. By (a), there exist runs ρ and ρ' of A accepting $(\tau\#)^\omega$ and $(\tau'\#)^\omega$, respectively. We prove that the intersection of $\inf(\rho)$ and $\inf(\rho')$ is empty. This implies that A contains at least as many final states as permutations of $(1, \ldots, n)$, which proves the Proposition.

We proceed by contradiction. Assume $q \in \inf(\rho) \cap \inf(\rho')$. We build an accepting run ρ'' by "combining" ρ and ρ' as follows:

(0) Starting from the initial state of ρ, go to q following the run ρ.

(1) Starting from q, follow ρ' until having gone through a final state, and having read at least once the word τ'; then go back to q (always following ρ').

(2) Starting from q, follow ρ until having gone through a final state, and having read at least once the word τ; then go back to q (always following ρ).

(3) Go to (1).

The word accepted by ρ'' contains infinitely many occurrences of both τ and τ'. By (b), this word belongs to L_n, contradicting the assumption that $\mathcal{A}$ recognizes $comp(L)_n$. □

4.2 Decision Procedures

The results for Büchi automata are analogous to the results in Section 3.2.

Proposition 4.2.1 [?, ?, ?] *The nonemptiness problem for Büchi automata is decidable in linear time and is NLOGSPACE-complete.*

Proof: Let $\mathcal{A} = (\Sigma, Q, Q_0, \delta, \alpha)$ be a Büchi automaton. We claim that $\mathcal{L}_\omega(\mathcal{A})$ is nonempty if and only if there are states $q_0 \in Q_0$ and $f \in \alpha$ such that the graph $D_\mathcal{A}$ contains a path leading from q_0 to f and a circuit going though f. Suppose first that $\mathcal{L}_\omega(A)$ is nonempty. Then there is an accepting run $r = q_0, q_1, \ldots$ of $\mathcal{A}$ on some input word, which corresponds to an infinite path of $D_\mathcal{A}$. Since r is accepting, some $f \in \alpha$ occurs in r infinitely often; in particular, there are i, j, where $0 < i < j$, such that $f = q_i = q_j$. Thus, $q_0, \ldots, q_i$ corresponds to a path from q_0 to f, and $q_i, \ldots, q_j$ to a circuit going through f.

Conversely, assume that $D_\mathcal{A}$ contains a path leading from q_0 to f and a circuit going though f. We can then construct an infinite path of $D_\mathcal{A}$ starting at q_0 and visiting f infinitely often. This path defines a run whose corresponding word is recogized by $\mathcal{A}$.

Thus, Büchi automata nonemptiness is reducible to graph reachability, as was the case for automata on finite words.

To check emptiness in linear time, use a depth-first-search algorithm to construct a decomposition of the graph into strongly connected components [**?**]. A is nonempty iff from a component that intersects Q_0 nontrivially it is possible to reach a nontrivial component that intersects α nontrivially. (A strongly connected component is nontrivial if it contains an edge, which means, since it is strongly connected, that it contains a cycle).

The algorithm proving membership in NLOGSPACE first guesses a state $q_0 \in Q_0$, then guesses a state q_1 that is directly connected to q_0, then guesses a state q_2 that is directly connected to q_1, etc., until it reaches a state $f \in \alpha$. At that point the algorithm remembers f and it continues to move nondeterministically from a state q to a state q' that is directly connected to q until it reaches f again. Clearly, the algorithm needs only logarithmic memory, since it needs to remember (a description of) at most three states at each step.

NLOGSPACE-hardness follows from NLOGSPACE-hardness of nonemptiness for automata on finite words. □

Proposition 4.2.2 [**?**] *The nonuniversality problem for Büchi automata is decidable in exponential time and PSPACE-complete.*

Proof: Again $\mathcal{L}_\omega(A) \neq \Sigma^\omega$ iff $\Sigma^\omega - \mathcal{L}_\omega(A) \neq \emptyset$ iff $\mathcal{L}_\omega(comp(A)) \neq \emptyset$, where $comp(A)$ is the complementary automaton of A (see Section 4.1.3). Thus, to test A for nonuniversality, it suffices to test $comp(A)$ for nonemptiness. Since $comp(A)$ is exponentially bigger than A and nonemptiness can be tested in linear time, it follows that nonuniversality can be tested in exponential time. Also, since nonemptiness can be tested in nondeterministic logarithmic space, nonuniversality can be tested in polynomial space. Again, the polynomial-space algorithm constructs *comp*(A) "on-the-fly".

PSPACE-hardness follows easily from the PSPACE-hardness of the universality problem for automata on finite words [?]. □

4.3 Other acceptance conditions

The acceptance condition of Büchi automata (some state of a given set must be visited infinitely often) can be replaced by many others, thus generating a wide variety of automata for infinite words. Fortunately, in all cases the expressive power remains the same, and so it makes sense to speak of *the* ω-regular languages. However, the fact that they accept the same class of languages does not mean that all acceptance conditions are equally good for all purposes. In this section we review some acceptance conditions and discuss some of their advantages and disadvantages.

4.3.1 Generalized Büchi automata

In a *generalized Büchi automaton*, the acceptance condition is a *collection of sets of states* $\{\alpha_1, \ldots, \alpha_m\}$. A run is accepting if it visits some state of each α_i infinitely often. Formally, a run ρ is accepting if $\inf(\rho) \cap \alpha_1 \neq \emptyset$ for each $i \in \{1, \ldots, m\}$. Ordinary Büchi automata correspond to the special case $m = 1$.

An advantage of generalized Büchi automata is a better construction for computing language intersections. Recall that given two Büchi automata $\mathcal{A}_1$ and $\mathcal{A}_2$ with n_1 and n_2 states, the Büchi automaton $\mathcal{A}$ of Proposition 4.1.2 satisfying $\mathcal{L}_\omega(\mathcal{A}) = \mathcal{L}_\omega(\mathcal{A}_1) \cap \mathcal{L}_\omega(\mathcal{A}_2)$ has $2n_1n_2$ states. A generalized Büii automaton needs only n_1n_2.

Generalized Büchi automata can be translated into ordinary Büchi automata with a linear blow-up in the number of states given by the number of α-sets.

Proposition 4.3.1 *Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be generalized Büchi automata. Then there is a generalized Büchi automaton $\mathcal{A}$ such that $\mathcal{L}_\omega(\mathcal{A}) = \mathcal{L}_\omega(\mathcal{A}_1) \cap \mathcal{L}_\omega(\mathcal{A}_2)$.*

Proof: Let $\mathcal{A}_1 = (\Sigma, Q_1, Q_1^0, \delta_1, \{\alpha_1, \ldots, \alpha_{m_1}\})$ and $\mathcal{A}_2 = (\Sigma, Q_2, Q_2^0, \delta_2, \{\beta_1, \ldots, \beta_{m_2}\})$. Let $\mathcal{A} = (\Sigma, Q, Q^0, \delta, \gamma)$, where

- $Q = Q_1 \times Q_2$,
- $Q^0 = Q_1^0$,
- $\delta(\langle s, t\rangle, a) = \delta_i(s, a) \times \delta_2(t, a)$, and
- $\gamma = \{\alpha_1 \times Q_2, \ldots, \alpha_{m_1} \times Q_2, Q_1 \times \beta_1, \ldots, Q_1 \times \beta_{m_2}\}$.

It is easy to show that $\mathcal{L}_\omega \mathcal{A} = \mathcal{L}_\omega(\mathcal{A}_1) \cap \mathcal{L}_\omega(\mathcal{A}_2)$. □

Proposition 4.3.2 *Let $\mathcal{A}$ be a generalized Büchi automaton with n states and m accepting sets. There is a Büchi automaton $\mathcal{A}'$ with $n \cdot m$ states such that $\mathcal{L}_\omega(\mathcal{A}') = \mathcal{L}_\omega(\mathcal{A})$.*

Proof: Let $\mathcal{A} = (\Sigma, Q, Q_0, \delta, \{\alpha_0, \ldots, \alpha_{m-1}\}$. Informally, $\mathcal{A}'$ consists of m copies of $\mathcal{A}$, say $\mathcal{A}_1, \ldots, \mathcal{A}_m$. $\mathcal{A}'$ stays in $\mathcal{A}_i$ until it hits an accepting state of α_i, in which case it moves to the next copy modulo m. $\mathcal{A}'$ accepts if it visits some accepting state of $\mathcal{A}_1$ infinitely often.

Formally, $\mathcal{A}' = (\Sigma, Q', Q'_0, \delta', \alpha)$ where:

- $Q' = Q \times \{0, \ldots, m-1\}$,
- $Q'_0 = Q \times \{0\}$,
- $\delta(\langle q, i\rangle, a) = \delta(q, a) \times \{j\}$, where $j = i$ if $q \notin \alpha_i$ and $j = (i+1) \bmod m$ if $q \in \alpha_i$, and
- $\alpha = \alpha_1 \times \{0\}$.

□

4.3.2 Muller automata

In a *Muller automaton*, the acceptance condition is a collection of sets of states $\{\alpha_1, \ldots, \alpha_m\}$, as it was for generalized Büchi automata. However, a run is now accepting if the set of states it visits infinitely often is equal to one of the α_i's. Formally, a run ρ is accepting if $\inf(\rho) = \alpha_i$ for some $i \in \{1, \ldots, m\}$.

Muller automata have two nice features:

(a) Any boolean combination of predicates of the form "state q is visited infinitely often" can be formulated as a Muller acceptance condition. It suffices to put in the condition all sets of states for which the predicate holds. For instance, the Büchi condition "q is visited infinitely often" corresponds to putting in the Muller condition all sets of states that contain q. Since in any accepting run of the Büchi automaton, the set of states visited infinitely often is some superset of $\{q\}$, the run is also be accepted by the Muller automaton, and vice versa.

(b) The languages accepted by nondeterministic and deterministic Muller automata coincide [**?**].

Exercise 29. Give a deterministic Muller automaton for the language $(a \vee b)^* b^\omega$.

Because of (a), Muller automata are a very general model. Both Büchi and generalized Büchi automata, as well as the Rabin and Street automata we introduce below, can be easily translated into Muller automata. The obvious disadvantage is that the translations involve an exponential blow-up in the size of the acceptance condition: a Büchi automaton with n states and acceptance condition $\{q\}$ is transformed into a Muller automaton whose acceptance condition contains $2^{(n-1)}$ sets.

On the other hand, the translation from Muller to Büchi automata involves only a quadratic blow-up in the number of states:

Proposition 4.3.3 *Let $\mathcal{A}$ be a Muller automaton having n states and m accepting states. There is a Büchi automaton $\mathcal{A}'$ with at most $2n^2m$ states such that $\mathcal{L}_\omega(\mathcal{A}') = \mathcal{L}_\omega(\mathcal{A})$.*

Proof: Let $\mathcal{A} = (\Sigma, Q, Q_0, \delta, \{\alpha_1, \ldots, \alpha_m\})$. It is easy to see that $\mathcal{L}_\omega(\mathcal{A}) = \cup_{i=1}^{k} \mathcal{L}_\omega(\mathcal{A}_i)$, where $\mathcal{A}_i = (\Sigma, Q, Q_0, \delta, \{\alpha_i\})$, and so it suffices to prove the claim for Muller conditions that consist of a single set, say $\alpha = \{q_1, \ldots, q_k\}$.

We first construct a generalized Büchi automaton $\mathcal{A}_g$, and then apply Proposition 4.3.2. The idea of the construction is to take two copies of $\mathcal{A}$, say $\mathcal{A}_1$ and $\mathcal{A}_2$. The generalized Büchi automaton $\mathcal{A}_g$ starts in $\mathcal{A}_1$ and stays there as long as it "wants". At some point it nondeterministically makes a transition into $\mathcal{A}_2$ and it stays there avoiding all the states of $Q \setminus \alpha$ and visiting all states of α infinitely often.

Formally, $\mathcal{A}_g = (\Sigma, Q_g, Q_g^0, \delta_g, \alpha_g)$, where

- $Q_g = (Q \times \{0\}) \cup \alpha$;
- $Q_g^0 = Q_0 \times \{0\}$;
- $\delta_g(\langle q, 0\rangle, a) = (\delta(q, a) \times \{0\}) \cup (\delta(q, a) \cap \alpha)$, and $\delta_g(q, a) = \delta(q, a) \cap \alpha$ for $q \in \alpha$;

- $\alpha_g = \langle \{q_1\}, \ldots \{q_k\} \rangle$.

Since $\mathcal{A}_g$ has at most $2n$ states and at most n acceptance sets, an equivalent Büchi automaton having at most $2n^2$ states can be constructed. The automaton $\mathcal{A}'$ has at most $2n^2m$ states. □

4.3.3 Rabin automata

In a *Rabin automaton*, the acceptance condition is a set of pairs $\{\langle \alpha_1, \beta_1 \rangle, \ldots, \langle \alpha_m, \beta_m \rangle\}$, where α_i, β_i are sets of states. A run is accepting if there is a pair $\langle \alpha_i, \beta_i \rangle$ such that the run visits infinitely often *some* state of α_i and *no* state of β_i. Formally, a run ρ is accepting if there is $i \in \{1, \ldots, m\}$ such that $\inf(\rho) \cap \alpha_i \neq \emptyset$ and $\inf(\rho) \cap \beta_i = \emptyset$.

Rabin automata share with Muller automata the property that the deterministic and nondeterministic versions accept exactly the same languages [**?**]. Moreover, a Büchi automaton can be transformed into a Rabin automaton without any exponential blow-up: a Büchi condition $\{q_1, \ldots, q_k\}$ corresponds to the Rabin condition $\{(\{q_1\}, \emptyset), \ldots, (\{q_n\}, \emptyset)\}$. The converse translation does not involve any blow-up either:

Proposition 4.3.4 *Let $\mathcal{A}$ be a Rabin automaton with n states and m accepting pairs. There is a Büchi automaton $\mathcal{A}'$ with at most $2nm$ states such that $\mathcal{L}_\omega(\mathcal{A}') = \mathcal{L}_\omega(\mathcal{A})$.*

Proof: Let $\mathcal{A} = (\Sigma, Q, Q_0, \delta, \{(\alpha_1, \beta_1), \ldots, (\alpha_m, \beta_m)\})$. As in the case of Muller automata, it is easy to see that $\mathcal{L}_\omega(\mathcal{A}) = \cup_{i=1}^{k} \mathcal{L}_\omega(\mathcal{A}_i)$, where $\mathcal{A}_i = (\Sigma, Q, Q_0, \delta, \{\langle \alpha_i, \beta_i \rangle\})$. Since Büchi automata are closed under union by Proposition 4.1.1, it suffices to prove the claim for Rabin conditions that consists of a single pair, say (α, β).

As for generalized Büchi automata, the idea of the construction is to take two copies of $\mathcal{A}$, say $\mathcal{A}_1$ and $\mathcal{A}_2$. The Büchi automaton $\mathcal{A}_b$ starts in $\mathcal{A}_1$ and stays there as long as it "wants". At some point it nondeterministically makes a transition into $\mathcal{A}_2$ and it stays there avoiding β and visiting α infinitely often. Formally, $\mathcal{A}_b = (\Sigma, Q_b, Q_b^0, \delta_b, \alpha)$, where

- $Q_b = (Q \times \{0\}) \cup (Q \setminus \beta)$;
- $Q_b^0 = Q_0 \times \{0\}$;
- $\delta_b(\langle q, 0 \rangle, a) = (\delta(q, a) \times \{0\}) \cup (\delta(q, a) \setminus \beta)$, and $\delta_b(q, a) = \delta(q, a) \setminus \beta$ for $q \in Q \setminus \beta$.

Since $\mathcal{A}_b$ has at most $2n$ states, $\mathcal{A}'$ has at most $2nm$ states. □

4.3.4 Street automata

In a *Street automaton*, the acceptance condition is again a set of pairs $\{\langle\alpha_1, \beta_1\rangle, \ldots, \langle\alpha_m, \beta_m\rangle\}$, where α_i, β_i are sets of states. A run is accepting if for every pair $\langle\alpha_i, \beta_i\rangle$, if the run visits some state of α_i infinitely often, then it also visits some state of β_i infinitely often. Formally, a run ρ is accepting if for every $i \in \{1, \ldots, m\}$ $\inf(\rho) \cap \alpha_i \neq \emptyset$ implies $\inf(\rho) \cap \beta_i \neq \emptyset$ (equivalently, if for every $i \in \{1, \ldots, m\}$ $\inf(\rho) \cap \alpha_i = \emptyset$ or $\inf(\rho) \cap \beta_i \neq \emptyset$).

As Rabin and Muller automata, the deterministic and nondeterministic versions of Street automata accept exactly the same languages [**?**]. As happened with Rabin automata, a Büchi acceptance condition $\{q_1, \ldots, q_k\}$ can be easily encoded into the Street acceptence condition containing only the pair $(Q, \{q_1, \ldots, q_k\})$, where Q is the set of all states of the Büchi automaton. However, unlike Büchi automata, we do not have a polynomial transformation from Street to Büchi automata.

Proposition 4.3.5 *Let $\mathcal{A}$ be a Street automaton with n states and m accepting states. There is a Büchi automaton $\mathcal{A}'$ with at most $2n^2 2^n$ states such that $\mathcal{L}_\omega(\mathcal{A}') = \mathcal{L}_\omega(\mathcal{A})$.*

Proof: Transform the Street automaton into a Muller automaton with the same number of states by putting into the Muller acceptance condition all possible sets of states of $\mathcal{A}$ satisfying the Street acceptance condition. The Muller automaton has at most 2^n acceptance sets. Apply now Proposition 4.3.3. □

The following exercise shows that no transformation with polynomial blow-up exists:

Exercise 30. [**?**]
IS THIS TRUE??
Let $AP_n = \{P_1, \ldots, P_n, Q_1, \ldots, Q_n\}$ be a set of atomic propositions and let $\Sigma = 2^{AP_n}$. Let L_n be the language in which, for each $i \in \{1, \ldots, n\}$, if $P_i \in c(j)$ for infinitely many j's then $Q_i \in c(k)$ for infinitely many k's. Show that L_n can be accepted by a Street automaton with $2n$ states and $2n$ accepting pairs, but cannot be accepted by any Büchi automaton having less than $O(2^n)$ states.

THIS IS TRUE
Let $\Sigma = \{0, 1, 2\}$, and let $x_1, x_2, \ldots$ be an infinite sequence of vectors of dimension n with components in Σ. We represent this sequence as an infinite

word $x_1x_2\ldots$. Let L_n be the language in which, for each component $i \in \{1,\ldots,n\}$, $x_j(i) = 1$ for infinitely many j's if and only if $x_k(i) = 2$ for infinitely many k's. Show that L_n can be accepted by a Street automaton with $3n$ states and $2n$ accepting pairs, but cannot be accepted by any Büchi automaton having less than 2^n states.

So Street automata can be exponentially more succint than Büchi automata.

Exercise 31. Give a quadratic algorithm to check emptiness of Street automata. Hint: use Tarjan's algorithm for determining the maximal strongly connected components of the automaton in linear time.

4.4 Applications

In Chapter 3, we reduced the problem of checking regular safety properties of finite Kripke structures to the emptiness problem for finite automata. Using the theory of this Chapter we now reduce the model-checking problem for ω-regular properties and finite Kripke structures to the emptiness problem of Büchi automata.

Recall that a Kripke structure K satisfies a property L if $\mathcal{L}_\omega(K) \subseteq L$. If K is finite, we can easily construct a Büchi automaton recognizing $\mathcal{L}_\omega(K)$: we just take the automaton $\mathcal{A}_K$ of Chapter 3, but this time with Büchi acceptance condition. Since all the states of $\mathcal{A}_K$ are accepting, all computations of K are recognized.

If L is ω-regular, then there is a Büchi automaton $\mathcal{A}$ recognizing L, and a Büchi automaton $comp(\mathcal{A})$ recognizing the complement $comp(L)$ of L. So we have

$$\begin{array}{ll} & \mathcal{L}_\omega(K) \subseteq L \\ \text{iff} & \mathcal{L}_\omega(K) \cap comp(L) = \emptyset \\ \text{iff} & \mathcal{L}_\omega(\mathcal{A}_K) \cap \mathcal{L}_\omega(comp(\mathcal{A})) = \emptyset \\ \text{iff} & \mathcal{L}_\omega(\mathcal{A}_K \times comp(\mathcal{A})) = \emptyset \end{array}$$

Notice that in the last line we are making use of the fact that all the states of $\mathcal{A}_K$ are accepting. Proposition 4.1.3 allows us to use the construction of Theorem 3.1.2, instead of the more involved of Theorem 4.1.2.

Using further results of the chapter, we obtain the following theorem:

Theorem 4.4.1 *Let K be a Kripke structure, and let L be an ω-regular property. Let $\mathcal{A}$ and $comp(\mathcal{A})$ be Büchi automata accepting L and $comp(L)$,* *respectively. The problem of checking if some computation of K violates L, i.e., if it belongs to $comp(L)$, can be solved*

a) in $O(|K| \cdot |comp(\mathcal{A})|)$ time and is NLOGSPACE-complete if the input is K and $comp(\mathcal{A})$;

b) in $|K| \cdot 2^{O(|A| \log |\mathcal{A}|)}$ time and is PSPACE-complete if the input is K and $\mathcal{A}$.

Proof: The size of the automaton $\mathcal{A}_K$ is clearly $O(|K|)$. By Theorem 3.1.2, the size of $\mathcal{A}_K \times comp(\mathcal{A})$ is $O(|K| \cdot |comp(\mathcal{A})|)$. Some computation of K belongs to $comp(L)$ if and only if the language $\mathcal{L}_\omega(\mathcal{A}_K \times comp(\mathcal{A}))$ is nonempty. Now a) follows from Proposition 4.2.1.

The $|K| \cdot 2^{O(|\mathcal{A}| \log |\mathcal{A}|)}$ upper bound in b) follows from from the construction for complementation of Büchi automata, which shows that there is a Büchi automaton $comp(\mathcal{A})$ of size $2^{O(|\mathcal{A}| \log |\mathcal{A}|}$ such that $\mathcal{L}_\omega(comp(\mathcal{A})) = \Sigma^\omega \setminus \mathcal{L}_\omega(\mathcal{A})$.

The PSPACE upper bound follows from the fact that the automaton $\mathcal{A}_K \times comp(\mathcal{A})$ can be explored 'on-the-fly' for nonemptiness. That is, we consider the same nondeterministic algorithm of Proposition 4.2.1, but the algorithm now computes a succesor of the current state of $\mathcal{A}_K \times comp(\mathcal{A})$ whenever it is needed, using the definition of $comp(\mathcal{A})$. Since the nondeterministic algorithm of requires to store at most three states of $\mathcal{A}_K \times comp(\mathcal{A})$ and a state of $comp(\mathcal{A})$ can be stored using polynomial space in $|\mathcal{A}|$, the algorithm runs in nondeterministic polynomial space. By Savitch's theorem, there is also a polynomial space deterministic algorithm.

PSPACE-hardness can be obtained by reduction from the nonuniversality problem, which is PSPACE-hard by Proposition 4.2.2. Given a Büchi automaton $\mathcal{A}$, we construct in polynomial time a Kripke structure K and a Büchi automaton $\mathcal{A}'$ such that $\mathcal{L}_\omega(\mathcal{A}) \neq \Sigma^\omega$ if and only if some computation of K belongs to $comp(L)$, i.e., if and only if $\mathcal{L}_\omega(\mathcal{A}_K) \cap \mathcal{L}_\omega(comp(\mathcal{A})) \neq \emptyset$. The trick is to choose the Kripke structure $K = \langle AP, W, R, W_0, \ell \rangle$ so that $\mathcal{L}_\omega(\mathcal{A}_K) = \Sigma^\omega$[1]. This is achieved by taking $AP = W = W_0 = \Sigma$, $R = \Sigma \times \Sigma$, and $\ell(w) = \{w\}$ for every $w \in W$. For the Büchi automaton we just choose $\mathcal{A}' = \mathcal{A}$. Then, we have

$$\begin{array}{ll} & \mathcal{L}_\omega(\mathcal{A}) = \Sigma^* \\ \text{iff} & \Sigma^* \subseteq \mathcal{L}_\omega(\mathcal{A}) \\ \text{iff} & \mathcal{L}_\omega(\mathcal{A}_K) \subseteq \mathcal{L}_\omega(\mathcal{A}') \\ \text{iff} & \mathcal{L}_\omega(\mathcal{A}_K) \cap \mathcal{L}_\omega(comp(\mathcal{A}')) = \emptyset \end{array}$$

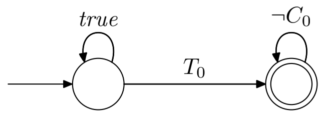
[1]In fact, since the alphabet of $\mathcal{A}_K$ is $2^{AP} = 2^\Sigma$, we identify σ and $\{\sigma\}$ for $\mathcal{L}_\omega(\mathcal{A}_K) = \Sigma^\omega$ to hold.

□

Consider now the finite waiting property for process i that we introduced in Section 2.4 as an interesting property of mutual exclusion algorithms:

$$L^i_{fw} = \{c \in (2^{AP})^\omega \mid \forall j \geq 0.(T_i \in c(j) \Rightarrow \exists k > j.C_i \in c(k))\}$$

The Büchi automaton shown at the top of Figure 4.11 recognizes $comp(L^0_{fw})$. So we can reduce the problem of checking this property to the emptiness problem of Büchi automata.



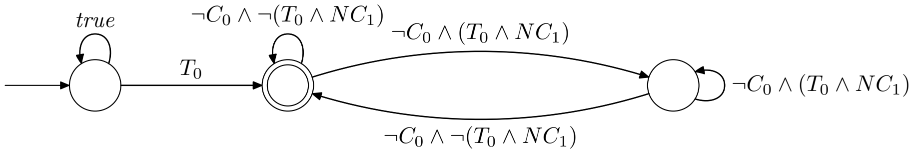


Figure 4.11: Büchi automata recognizing $comp(L^0_{fw})$ and $comp(L^0_{fw} \cup F)$

Recall that the finite waiting property does not hold for process 0 of Lamport;s algorithm, and this made us introduce the fairness assumption

$$F = \{d \cdot \{T_0, NC_1\}^\omega \mid d \in (2^{AP})^*\}\ .$$

Checking if the finite waiting property holds for process 0 under the assumption F amounts to deciding if $\mathcal{L}_\omega(K) \cap (comp(L^0_{fw} \cup F)) = \emptyset$. The Büchi automaton shown at the bottom of Figure 4.11 recognizes $comp(L^0_{fw} \cup F)$.

Exercise 32. Construct a Büchi automaton recognizing the language $comp(L^0_{fw} \cup F)$ by giving automata recognizing $comp(L^0_{fw})$ and $comp(F)$, and then applying the construction of Proposition 4.1.2. Compare it with the automaton of Figure 4.11.

Exercise 33. Construct Büchi automata for several kinds of fairness constraints. Justice, compassion, etc.