Discrete Event Systems - Notes and Exercises

Dr Michael Hoffmann
Original notes: Dr Nir Piterman

January 21, 2015

Chapter 1

Finite Automata and Regular Languages

1.1 Deterministic Automata

1.1.1 Definition of Deterministic Finite Automata

We abstract computation by constructing simple machines that recognize finite strings of symbols. A machine will read an input string and say either "yes" or "no". We start with a few definitions.

An *alphabet* Σ is a finite set of symbols (digits, charaters, letters, ...). For example $\{a, b\}$ is the two-letter alphabet consisting of the letters a and b. A *string* or a *word* over an alphabet Σ is a finite sequence of symbols from Σ. For example, ab is a word of length 2 over the alphabet $\Sigma = \{a, b\}$. The *empty string*, denoted ϵ, is a word of length 0. The length of a string (its number of letters) is denoted by $|w|$. For example $|\epsilon| = 0$, and $|abb| = 3$. Many times we use a "generic description of a word: $a_0a_1 \cdots a_{n-1}$ is a word of length n with a_0 being its first letter, a_1 being its second letter, and a_{n-1} being its n-th letter.

A finite automaton is a 5-tuple $A = (Q, \Sigma, \delta, q_0, F)$, where the components of A are:

- Q is a finite set of states, usually referred to as the *state set* of the automaton.
- Σ is the alphabet of the automaton.
- δ is the *transition function* of the automaton, it is a function $\delta : Q \times \Sigma \to Q$ associating a state $q \in Q$ and an alphabet letter $a \in \Sigma$

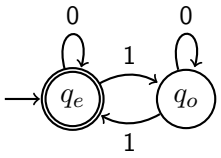


Figure 1.1: The automaton even.

with the successor of q reading a.

- $q_0 \in Q$ is an *initial state* or *start state*.
- $F \subseteq Q$ is a set of *accepting states* (sometimes *final states*).

Notice that the input itself is not embodied in the definition of the automaton.

1.1.2 The Even Automaton

We construct an automaton that accepts all the words with an even number of 1s over the alphabet $\Sigma = \{0, 1\}$. The automaton is depicted in Figure 1.1. For a word w and a letter a, we denote by $\sharp_a(w)$ the number of as appearing in w. For example, $\sharp_1(001) = 1$ and $\sharp_1(0110) = 2$.

Formally, the automaton is $A = \langle Q, \Sigma, \delta, q_e, F \rangle$, where the components of A are as follows:

- $Q = \{q_e, q_o\}$
- $\Sigma = \{0, 1\}$
- $\delta(q_e, 0) = q_e \quad \delta(q_o, 0) = q_o$
  $\delta(q_e, 1) = q_o \quad \delta(q_o, 1) = q_e$
- $F = \{q_e\}$

We show that the language of A is the set of words that have an even number of ones.

Lemma 1.1 *For every word w, the run of A on w ends in state q_e if $\sharp_1(w)$ is even and ends in state q_o if $\sharp_1(w)$ is odd.*

Proof We prove this by induction on the length of the word. Consider the empty word ϵ. Clearly, $\sharp_1(\epsilon) = 0$, which is even. The run of A on ϵ is q_e, proving the induction base.

Suppose that for every words w of length at most n the run of A on w ends in q_e if $\sharp_1(w)$ is even and ends in q_o if $\sharp_1(w)$ is odd. Consider a word wa of length $n+1$. Let $q_0, \ldots, q_n, q_{n+1}$ be the run of A on wa. However, $q_0, \ldots, q_n$ is the run of A on w. By assumption $q_n = q_e$ if and only if $\sharp_1(w)$ is even. There are four possible cases. ,

- If $\sharp_1(w)$ is even and a is 1. Then, by assumption $q_n = q_e$. By definition $\delta(q_e, 1) = q_o$, hence $q_{n+1} = q_o$. However, if $\sharp_1(w)$ is even, then $\sharp_1(w \cdot 1)$ is odd, proving the claim.

- If $\sharp_1(w)$ is even and a is 0. Then, by assumption $q_n = q_e$. By definition $\delta(q_e, 0) = q_e$, hence $q_{n+1} = q_e$. However, if $\sharp_1(w)$ is even, then $\sharp_1(w \cdot 0)$ is even, proving the claim.

- If $\sharp_1(w)$ is odd and a is 1. Then, by assumption $q_n = q_o$. By definition $\delta(q_o, 1) = q_e$, hence $q_{n+1} = q_e$. However, if $\sharp_1(w)$ is odd, then $\sharp_1(w \cdot 1)$ is even, proving the claim.

- If $\sharp_1(w)$ is odd and a is 0. Then, by assumption $q_n = q_o$. By definition $\delta(q_o, 1) = q_o$, hence $q_{n+1} = q_o$. However, if $\sharp_1(w)$ is odd, then $\sharp_1(w \cdot 0)$ is odd, proving the claim.

Corollary 1.2 $L(A) = \{w : \sharp_1(w)$ *is even*$\}$

Proof Follows immediately from q_e being the only accepting state.

1.2 Constructions for Deterministic Automata

1.2.1 Union of Deterministic Automata

Given two automaton $A_i = \langle Q_i, \Sigma, \delta_i, q_0^i, F_i \rangle$, where $i \in \{1, 2\}$, we construct the automaton $A^\cup$ such that $L(A^\cup) = L(A_1) \cup L(A_2)$. Formally, $A^\cup = \langle Q^\cup, \Sigma, \delta^\cup, q_0^\cup, F^\cup \rangle$, where the elements of $A^\cup$ are defined as follows.

- $Q^\cup = Q_1 \times Q_2$
- For every $(q_1, q_2) \in Q^\cup$ and $a \in \Sigma$ we define $\delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a))$.
- $q_0^\cup = (q_0^1, q_0^2)$
- $F^\cup = (F_1 \times Q_2) \cup (Q_1 \times F_2)$

We show that the language of $A^\cup$ is union of the languages of A_1 and A_2.

Lemma 1.3 $L(A^{\cup}) = L(A_1) \cup L(A_2)$.

Proof Consider a word $w \in L(A_1)$. Let $r_1 = q_0^1, \ldots, q_n^1$ be the run of A_1 on w and let $r_2 = q_0^2, \ldots, q_n^2$ be the run of A_2 on w. Then, r_1 is accepting and $q_n^1 \in F_1$. Consider now the run $(q_0^1, q_0^2), \ldots, (q_n^1, q_n^2)$. It is a run of $A^{\cup}$. Indeed, it starts in the initial state and for every $0 \leq j < n$ we have $(q_{j+1}^1, q_{j+1}^2) = \delta^{\cup}((q_j^1, q_j^2), a_j)$ because $\delta_i(q_j^i, a_j) = q_{j+1}^i$ for $i \in \{1, 2\}$. Furthermore, as $q_n^1 \in F_1$ it follows that $(q_n^1, q_n^2) \in F_1 \times Q_2$ implying that $(q_n^1, q_n^2) \in F^{\cup}$. Hence, w is accepted by $A^{\cup}$ and $L(A_1) \subseteq L(A^{\cup})$.

Showing that $L(A_2) \subseteq L(A^{\cup})$ is similar. Thus, $L(A_1) \cup L(A_2) \subseteq L(A^{\cup})$.

We now prove that $L(A^{\cup}) \subseteq L(A_1) \cup L(A_2)$. Consider $w \in L(A^{\cup})$. Let $(q_0^1, q_0^2), \ldots, (q_n^1, q_n^2)$ the accepting run of $A^{\cup}$ on w. Then, either q_n^1 is in F_1 or q_n^2 is in F_2. Suppose that q_n^1 is in F_1. Then, $q_0^1, \ldots, q_n^1$ is an accepting run of A_1 on w. Indeed, it starts in the initial state, for every $0 \leq j < n$ we have $q_{j+1} = \delta_1(q_j, a_j)$, and $q_n^1 \in F_1$. Thus, $w \in L(A_1)$. If $q_n^2 \in F_2$ we show that $w \in L(A_2)$ in a similar way. It follows $L(A^{\cup}) \subseteq L(A_1) \cup L(A_2)$.

1.2.2 Complementation of Deterministic Automata

Given an automaton $A = \langle Q, \Sigma, \delta, q_0, F \rangle$ we construct the automaton A^c such that $L(A^c) = \Sigma^* - L(A)$. Formally, $A^c = \langle Q, \Sigma, \delta, q_0, Q - F \rangle$.

Lemma 1.4 $L(A^c) = \Sigma^* - L(A)$

Proof Consider a word $w \in L(A)$. We show that $w \notin L(A^c)$. Let $r = q_0, \ldots, q_n$ be the accepting run of A on w. But r is also a run of A^c on w. However, in A^c the state q_n is not accepting. Thus, $w \notin L(A^c)$.

Consider a word $w \notin L(A)$. We show that $w \in L(A^c)$. Let $r = q_0, \ldots, q_n$ be the rejecting run of A on w. But r is also a run of A^c on w. However, in A^c the state q_n is accepting. Thus, $w \in L(A^c)$.

1.2.3 Intersection of Deterministic Automata

Given two automaton $A_i = \langle Q_i, \Sigma, \delta_i, q_0^i, F_i \rangle$, where $i \in \{1, 2\}$, we construct the automaton $A^{\cap}$ such that $L(A^{\cap}) = L(A_1) \cap L(A_2)$. Formally, $A^{\cap} = \langle Q^{\cap}, \Sigma, \delta^{\cap}, q_0^{\cap}, F^{\cap} \rangle$, where the elements of $A^{\cap}$ are defined as follows.

- $Q^{\cap} = Q_1 \times Q_2$
- For every $(q_1, q_2) \in Q^{\cap}$ and $a \in \Sigma$ we define $\delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a))$.
- $q_0^{\cap} = (q_0^1, q_0^2)$

- $F^{\cap} = F_1 \times F_2$

We show that the language of $A^{\cap}$ is intersection of the languages of A_1 and A_2.

Lemma 1.5 $L(A^{\cap}) = L(A_1) \cap L(A_2)$.

Proof Consider a word $w \in L(A_1) \cap L(A_2)$. Let $r_1 = q_0^1, \ldots, q_n^1$ be the run of A_1 on w and let $r_2 = q_0^2, \ldots, q_n^2$ be the run of A_2 on w. Then, r_1 is accepting and $q_n^1 \in F_1$. Similarly, r_2 is accepting and $q_n^2 \in F_2$. Consider now the run $(q_0^1, q_0^2), \ldots, (q_n^1, q_n^2)$. It is a run of $A^{\cap}$. Indeed, it starts in the initial state and for every $0 \leq j < n$ we have $(q_{j+1}^1, q_{j+1}^2) = \delta^{\cap}((q_j^1, q_j^2), a_j)$ because $\delta_i(q_j^i, a_j) = q_{j+1}^i$ for $i \in \{1, 2\}$. Furthermore, as $q_n^1 \in F_1$ and $q_n^2 \in F_2$ it follows that $(q_n^1, q_n^2) \in F_1 \times F_2$ implying that $(q_n^1, q_n^2) \in F^{\cap}$. Hence, w is accepted by $A^{\cap}$ and $L(A_1) \cap L(A_2) \subseteq L(A^{\cap})$.

We now prove that $L(A^{\cap}) \subseteq L(A_1) \cap L(A_2)$. Consider $w \in L(A^{\cap})$. Let $(q_0^1, q_0^2), \ldots, (q_n^1, q_n^2)$ the accepting run of $A^{\cap}$ on w. Then, q_n^1 is in F_1 and q_n^2 is in F_2. Then, $q_0^1, \ldots, q_n^1$ is an accepting run of A_1 on w. Indeed, it starts in the initial state, for every $0 \leq j < n$ we have $q_{j+1} = \delta_1(q_j, a_j)$, and $q_n^1 \in F_1$. Thus, $w \in L(A_1)$. Similarly, $q_0^1, \ldots, q_n^n$ is an accepting run of A_2 on w. It follows $L(A^{\cap}) \subseteq L(A_1) \cap L(A_2)$.

1.3 Constructions for Nondeterministic Automata

1.3.1 Union of Nondeterministic Automata

Given two automaton $A_i = \langle Q_i, \Sigma, \delta_i, q_0^i, F_i \rangle$, where $i \in \{1, 2\}$, we construct the automaton $A^{\cup}$ such that $L(A^{\cup}) = L(A_1) \cup L(A_2)$. Assume without loss of generality that $Q_1 \cap Q_2 = \emptyset$. Formally, $A^{\cup} = \langle Q^{\cup}, \Sigma, \delta^{\cup}, q_0^{\cup}, F^{\cup} \rangle$, where the elements of $A^{\cup}$ are defined as follows.

- $Q^{\cup} = Q_1 \cup Q_2 \cup \{q_0^{\cup}\}$
- For every $q \in Q_1$ and $a \in \Sigma$ we set $\delta^{\cup}(q, a) = \delta_1(q, a)$. For every $q \in Q_2$ and $a \in \Sigma$ we set $\delta^{\cup}(q, a) = \delta_2(q, a)$. For $q_0^{\cup}$ we set and $a \in \Sigma$ we set $\delta^{\cup}(q_0^{\cup}, a) = \delta(q_0, a) \cup \delta(q_0', a)$.
- $F^{\cup} = F_1 \cup F_2 \cup \{q_0^{\cup} \mid q_0 \in F_1 \vee q_1 \in F_2\}$

We show that the language of $A^{\cup}$ is union of the languages of A_1 and A_2.

Lemma 1.6 $L(A^{\cup}) = L(A_1) \cup L(A_2)$.

Proof First it is clear that $\epsilon \in L(A^{\cup})$ if and only if $\epsilon \in L(A_1)$ or $\epsilon \in L(A_2)$. In the rest of the proof we restrict attention to words of length at least 1.

Consider a word $w \in L(A_1)$.

Let $r = q_0, q_1, \ldots, q_n$ be an accepting run of A_1 on w. Then, $q_n \in F_1$. Consider now the run $r' = q_0^{\cup}, q_1, \ldots, q_n$ obtained from r by replacing q_0 as the first state by $q_0^{\cup}$. It is a run of $A^{\cup}$. Indeed, it starts in the initial state. The first transition is a legal transition of $q_0^{\cup}$ as from $q_1 \in \delta(q_0, a_0)$ we deduce that $q_1 \in \delta(q_0^{\cup}, a_0)$. Every other transition is a legal transition in $\delta^{\cup}$ as for every $q \in Q_1$ we set $\delta^{\cup}(q, a) = \delta_1(q, a)$. It follows that r' is an accepting run of $A^{\cup}$ on w and $w \in L(A^{\cup})$. Thus, $L(A_1) \subseteq L(A^{\cup})$.

Showing that $L(A_2) \subseteq L(A^{\cup})$ is similar. Thus, $L(A_1) \cup L(A_2) \subseteq L(A^{\cup})$.

We now prove that $L(A^{\cup}) \subseteq L(A_1) \cup L(A_2)$. Consider $w \in L(A^{\cup})$. Let $q_0^{\cup}, q_1, \ldots, q_n$ be the accepting run of $A^{\cup}$ on w. It must be the case that $q_n \in F_1$ or $q_n \in F_2$. Then the only way to get from $q_0^{\cup}$ to a state in Q_1 is by taking a transition $q_1 \in \delta(q_0^{\cup}, a_0)$ such that $q_1 \in \delta(q_0^1, a_0)$. It follows that the run $q_0^1, q_1, \ldots, q_n$ is an accepting run of A_1 on w. Thus, $w \in L(A_1)$. If $q_n \in F_2$ we show that $w \in L(A_2)$ in a similar way. It follows $L(A^{\cup}) \subseteq L(A_1) \cup L(A_2)$.

1.3.2 Concatenation of Nondeterministic Automata

Given two automaton $A_i = \langle Q_i, \Sigma, \delta_i, q_0^i, F_i \rangle$, where $i \in \{1, 2\}$, we construct the automaton A such that $L(A) = L(A_1) \cdot L(A_2)$. Assume without loss of generality that $Q_1 \cap Q_2 = \emptyset$. Formally, $A = \langle Q, \Sigma, \delta, q_0, F \rangle$, where the elements of A are defined as follows.

- $Q = Q_1 \cup Q_2$
- For every $q \in Q_1 - F_1$ and $a \in \Sigma$ we set $\delta(q, a) = \delta_1(q, a)$. For every $q \in F_1$ and $a \in \Sigma$ we set $\delta(q, a) = \delta_1(q, a) \cup \delta(q_0^2, a)$. For every $q \in Q_2$ and $a \in \Sigma$ we set $\delta(q, a) = \delta_2(q, a)$.
- $F = F_2 \cup \{q \in F_1 \mid q_0^2 \in F_2\}$

Notice that if q_0^1 is an accepting state of A_1 then its transition is changed just like all other accepting states and that if q_0^2 is accepting then all accepting states of A_1 remain accepting. We show that the language of A is concatenation of the languages of A_1 and A_2.

Lemma 1.7 $L(A) = L(A_1) \cdot L(A_2)$.

Proof Consider words $w_1 \in L(A_1)$ and $w_2 \in L(A_2)$. Let $r_1 = q_0^1, q_1^1, \ldots, q_n^1$ be an accepting run of A_1 on w_1. Let $r_2 = q_0^2, q_1^2, \ldots, q_m^2$ be an accepting run of A_2 on w_2. Then, $q_n^1 \in F_1$, and $q_m^2 \in F_2$. Consider now the run $r = q_0^1, \ldots, q_n^1, q_1^2, \ldots, q_m^2$ obtained from r by removing q_0^2 from r_2 and concatenating r_1 with the result of this removal. It is a run of A. Indeed, it starts in the initial state. All the transitions up to i_n are legal transitions as $\delta_1(q, a) \subseteq \delta(q, a)$ for all $q \in Q_1$. The transition from q_n^1 to q_1^2 is a legal transition as $q_n^1 \in F_1$, $q_1^2 \in \delta(q_0^2, a)$ and $\delta(q, a) = \delta_1(q, a) \cup \delta_2(q_0^2, a)$ for $q \in F_1$ and $a \in \Sigma$. Finally, all the transitions from $n+1$ to $n+m$ are legal transitions as $\delta(q, a) = \delta_2(q, a)$ for every $q \in Q_2$. Finally, $q_m^2 \in F_2$ showing that r is accepting. Notice that if the length of r_2 is 1 (i.e., $w_2 = \epsilon$) then q_0^2 is accepting leading to every accepting state of A_1 being accepting also in A and having the run r accepting as well. Thus, $L(A_1) \cdot L(A_2) \subseteq L(A^{\cup})$.

We now prove that $L(A) \subseteq L(A_1) \cdot L(A_2)$. Consider $w \in L(A)$. Let $r = q_0, q_1, \ldots, q_n$ be the accepting run of A on w. Now, q_0 is q_0^1 and either $q_n \in F$. If $q_n \in F_1$ then it follows that $q_0^2 \in F_2$, ϵ is accepted by A_2 and r is also an accepting run of A_1 on w. However $w \cdot \epsilon = w$ showing that indeed $w \in L(A_1) \cdot L(A_2)$. Otherwise, $q_n \in F_2$. The only possible transitions from Q_1 to Q_2 are transitions from states in F_1. There are no transitions from Q_2 to Q_1. Thus, there is some j such that $q_j \in F_1$ and $q_{j+1} \in \delta(q_0^2, a_j)$. Let $w_1 = a_0 \cdots a_{j-1}$ and let $w_2 = a_j \cdots a_{n-1}$. Clearly, $w = w_1 \cdot w_2$. Then, $q_0, \ldots, q_j$ is an accepting run of A_1 on w_1. Indeed, it starts in the initial state, for every $0 \leq k < j$ we have $q_{k+1} \in \delta_1(q_k, a_k)$, and $q_j \in F_1$. Thus, $w_{\in} L(A_1)$. Similarly, $q_0^2, q_{j+1}, \ldots, q_n$ is an accepting run of A_2 on w_2. Indeed, it starts in the initial state, we have already shown that $q_{j+1} \in \delta(q_0^2, a_j)$ and for every $0 \leq k < |w| - j$ we have $q_{k+1} \in \delta_2(q_k, a_{k-j})$, and $q_n \in F_2$.

1.3.3 Kleene Star of Nondeterministic Automata

Given an automaton $A = \langle Q, \Sigma, \delta, q_0, F \rangle$ we construct the automaton A^* such that $L(A^*) = (L(A))^*$. Formally, $A^* = \langle Q \cup \{q_0'\}, \Sigma, \delta^*, q_0', F^* \rangle$, where the elements of A^* are defined as follows.

- For every $a \in \Sigma$ we set $\delta^*(q_0', a) = \delta(q_0, a)$. For every $q \in Q - F$ and $a \in \Sigma$ we set $\delta^*(q, a) = \delta(q, a)$. For every $q \in F$ and $a \in \Sigma$ we set $\delta^*(q, a) = \delta(q, a) \cup \delta(q_0, a)$.
- $F^* = F \cup \{q_0'\}$

We show that the language of A^* is $(L(A))^*$.

Lemma 1.8 $L(A) = (L(A))^*$.

Proof Consider a word $w \in (L(A))^*$. It follows that either $w = \epsilon$ or there is a partition $w_0, \ldots, w_k$ of w such that for every $0 \leq j \leq k$ we have $w_j \in L(A)$ and $w_j \neq \epsilon$. If $w = \epsilon$, then clearly $\epsilon \in L(A^*)$ as q'_0 is accepting. In the second case, for every $0 \leq j \leq k$ there is an accepting run $r_j = q_0^j, q_1^j, \ldots, q_{n_j}^j$ of A on w_j. Then, for every $0 \leq j \leq k$ we have $q_{n_j}^j \in F$. Then, $q_1^{j+1} \in \delta(q_{n_j}^j, a_0^{j+1})$, where a_0^{j+1} is the first letter of w_{j+1}. Consider now the run

$$r = q'_0, q_1^1, \ldots q_{n_1}^1, q_1^2, \ldots, q_{n_2}^2, \ldots, q_1^k, \ldots, q_{n_k}^k$$

obtained from r by starting in q'_0 and concatenating all the runs from which the first state is removed. It is a run of A^*. Indeed, it starts in the initial state. All the transitions up to $i_{n_1}^1$ are legal transitions as $\delta(q, a) \subseteq \delta^*(q, a)$ for all $q \in Q$. The transition $q_1^2 \in \delta(q_{n_1}^1, a_{n_1})$ is a legal transition as $q_{n_1}^1 \in F$ and $q_1^2 \in \delta(q_0, a_{n_1})$. Similarly, the transition $q_1^{j+1} \in \delta(q_{n_j}^j, a_0^{j+1})$ is a legal transition as $q_{n_j}^j \in F$ and $q_1^{j+1} \in \delta(q_0, a_0^{j+1})$. Finally, $q_n^k \in F^*$ showing that $w \in L(A^*)$. Thus, $(L(A))^* \subseteq L(A^*)$.

We now prove that $L(A^*) \subseteq (L(A))^*$. Consider $w \in L(A^*)$. If $|w| = 0$, then, clearly, $w \in (L(A))^*$. Otherwise, let $q'_0, q_1, \ldots, q_n$ be the accepting run of A^* on w. Let $j_1, \ldots, j_k$ denote the locations such that the $q_{j_{m_l}} \in F$ and the transition from $q_{j_{m_l}}$ to $q_{j_{m_l}+1}$ results from the inclusion of $\delta(q_0, a_{j_{m_l}})$ in the transition of $q_{j_{m_l}}$. Let w_l denote the word consisting of the letters $j_{m_{l-1}}$ to $j_{m_l} - 1$. Thus $w = w_1 \cdot w_2 \cdots w_k$. Then, $q_0, q_{m_{l-1}+1}, \ldots, q_{j_{m_l}}$ is an accepting run of A on w_l. Indeed, it starts in the initial state, every transition is legal, reads the entire word, and ends in an accepting state. Thus, $w_l \in L(A)$. We conclude that $w \in (L(A))^*$.

1.4 Determinization

Given a nondeterministic automaton $A = \langle Q, \Sigma, \delta, q_0, F \rangle$ we construct the deterministic automaton $A^{\mathcal{P}}$ such that $L(A^{\mathcal{P}}) = L(A)$. Let $A^{\mathcal{P}} = \langle Q^{\mathcal{P}}, \Sigma, \delta^{\mathcal{P}}, q_0^{\mathcal{P}}, F^{\mathcal{P}} \rangle$, where the elements of $A^{\mathcal{P}}$ are defined as follows.

- $Q^{\mathcal{P}} = \mathcal{P}(Q)$
- For every $S \subseteq Q$ and $a \in \Sigma$ we set

$$\delta^{\mathcal{P}}(S, a) = \{s' \mid s' \in \delta(s, a) \text{ for some } s \in S\}$$

- $q_0^{\mathcal{P}} = \{q_0\}$

- $F^{\mathcal{P}} = \{S \mid S \cap F \neq \emptyset\}$

It is simple to see that $A^{\mathcal{P}}$ is indeed deterministic. In order to show that the languages of A and $A^{\mathcal{P}}$ are equivalent we prove two stronger claims.

Lemma 1.9 *For every word w and every run $q_0, \ldots, q_n$ of A on w, we have $q_n \in S_n$, where $S_0, \ldots, S_n$ is the unique run of $A^{\mathcal{P}}$ on w.*

Proof We prove the claim by induction on the length of w. Consider the empty word ϵ. Let q_0 be a run of A on ϵ. Then, by definition of $q_0^{\mathcal{P}}$ we have $q_0 \in q_0^{\mathcal{P}}$ and $q_0^{\mathcal{P}}$ is *the* run of $A^{\mathcal{P}}$ on ϵ.

Suppose that the claim holds for words of length up to m. That is, for every w of length up to m and every run $q_0, \ldots, q_n$ of A on w, we have $q_n \in S_n$, where $S_0, \ldots, S_n$ is the unique run of $A^{\mathcal{P}}$ on w.

Consider a word wa of length $m+1$. Let $q_0, \ldots, q_{m+1}$ be a run of A on wa. Consider the run $q_0, \ldots, q_m$ of A. It is simple to see that it is a run of A on w. Then, by assumption, the unique run $S_0, \ldots, S_m$ of $A^{\mathcal{P}}$ on w fulfils $q_m \in S_m$. However, as $q_{m+1} \in \delta(q_m, a)$ it follows that $q_{m+1} \in \delta^{\mathcal{P}}(S_{|w|}, a)$. Thus, *the* run of $A^{\mathcal{P}}$ on wa is $S_0, \ldots, S_m, S_{m+1}$ and $q_{m+1} \in S_{m+1}$.

Lemma 1.10 *For a word w, let $S_0, \ldots, S_m$ be the run of $A^{\mathcal{P}}$ on w. Then, for every $q \in S_m$ there is a run $q_0, \ldots, q_m$ of A on w such that $q_m = q$.*

Proof We prove the claim by induction on the length of w. Consider the empty word ϵ. The run of $A^{\mathcal{P}}$ on ϵ is $Q^{\mathcal{P}}$. By definition, q_0 is a run of A on ϵ.

Suppose that the claim holds for words of length up to m. That is, for every w of length up to m and every q such that $q \in S_m$, where $S_0, \ldots, S_m$ is the run of $A^{\mathcal{P}}$ on w, there is a run $q_0, \ldots, q_m$ of A on w such that $q_m = q$.

Consider a word wa of length $m+1$. Let $S_0, \ldots, S_m, S_{m+1}$ be the run of $A^{\mathcal{P}}$ on wa. Consider a state $q \in S_{m+1}$. By definition, $S_{m+1} = \delta^{\mathcal{P}}(S_m, a)$. Thus, there is some state $p \in S_m$ such that $q \in \delta(p, a)$.

Consider the run $S_0, \ldots, S_m$. It is the run of $A^{\mathcal{P}}$ on w. As $p \in S_m$ we know that there is a run $q_0, \ldots, q_m$ of A such that $q_m = p$. Then, $q_0, \ldots, q_m, q$ is a run of A on wa as required.

Corollary 1.11 $L(A) = L(A^{\mathcal{P}})$

Proof Consider a word $w \in L(A)$. It follows that there is a run $q_0, \ldots, q_n$ such that $q_n \in F$ on w. By Lemma 1.9 we know that $q_n \in S_n$, where

$S_0, \ldots, S_n$ is the run of $A^{\mathcal{P}}$ on w. But as $q_n \in F$ we know that $S_n \in F^{\mathcal{P}}$ and w is accepted by $A^{\mathcal{P}}$.

Consider a word $w \in L(A^{\mathcal{P}})$. It follows that $S_0, \ldots, S_n$ is accepting and there is some $q \in F$ such that $q \in S_n$. By Lemma 1.10 we know that there is a run $q_0, \ldots, q_n$ of A on w such that $q_n = q$. Thus, w is accepted by A.

1.5 Synchronous Composition

Given two automaton $A_i = \langle Q_i, \Sigma_i, \delta_i, q_0^i, F_i \rangle$, where $i \in \{1, 2\}$, we construct the automaton $A^{\|}$ (or sometimes $A_1 \| A_2$) that is the synchronous parallel composition of A_1 and A_2.

Formally, $A^{\|} = \langle Q^{\|}, \Sigma^{\|}, \delta^{\|}, q_0^{\|}, F^{\|} \rangle$, where the elements of $A^{\|}$ are defined as follows.

- $Q^{\|} = Q_1 \times Q_2$
- $\Sigma^{\|} = \Sigma_1 \cup \Sigma_2$
- For every $(q_1, q_2) \in Q^{\|}$ and $a \in \Sigma^{\|}$ we define $\delta((q_1, q_2), a)$ as follows.

$$\delta((q_1, q_2), a) = \begin{cases} \{(q_1', q_2) \mid q_1' \in \delta_1(q_1, a)\} & \text{If } a \in \Sigma_1 - \Sigma_2 \\ \{(q_1, q_2') \mid q_2' \in \delta_2(q_2, a)\} & \text{If } a \in \Sigma_2 - \Sigma_1 \\ \{(q_1', q_2') \mid q_1' \in \delta_1(q_1, a) \text{ and } q_2' \in \delta_2(q_2, a)\} & \text{If } a \in \Sigma_1 \cap \Sigma_2 \end{cases}$$

- $q_0^{\|} = (q_0^1, q_0^2)$
- $F^{\|} = F_1 \times F_2$

We show that every word in language of $A^{\|}$ is an interleaving of words in the languages of A_1 and A_2.

Lemma 1.12 *For every word $w \in (\Sigma^{\|})^*$ we have $w \in L(A^{\|})$ iff $P_{\Sigma_1}(w) \in L(A_1)$ and $P_{\Sigma_2}(w) \in L(A_2)$.*

Proof Consider a word $w \in (\Sigma^{\|})^*$. Suppose that $w_1 = P_{\Sigma_1(w)} \in L(A_1)$ and $w_2 = P_{\Sigma_2(w)} \in L(A_2)$. Let $r_1 = q_0^1, q_1^1, \ldots, q_n^1$ be the run of A_1 on w_1 and let $r_2 = q_0^2, q_1^2, \ldots, q_m^2$ be the run of A_2 on w_1. Then, r_1 is accepting and $q_n^1 \in F_1$. Similarly, r_2 is accepting and $q_m^2 \in F_2$. We construct a run of $A^{\|}$ on w as follows. The following algorithm constructs a run of $A^{\|}$ from the runs of A_1 and A_2.

```
1   p1=0;p2=0;p=0;
4   r=(q^1_{p1}, q^2_{p2})
5   while (p<|w|)
6     if (a_p ∈ Σ1 − Σ2)
7       p1=p1+1;p=p+1;
9       r=r·(q^1_{p1},q^2_{p2});
10    else if (a_p ∈ Σ2 − Σ1)
11      p2=p2+1;p=p+1;
13      r=r·(q^1_{p1},q^2_{p2});
14    else
15      p1=p1+1;p2=p2+1;p=p+1;
18      r=r·(q^1_{p1},q^2_{p2});
19  end -- while
```

It is simple to see that the algorithm constructs a valid and accepting run of $A^{\|}$ on w.

Suppose that $r = (q_0^1, q_0^2), \ldots, (q_n^1, q_n^2)$ is an accepting run of $A^{\|}$ on w. Then, $q_0^1, q_1^1, \ldots, q_n^1$ can be pruned to an accepting run of A_1 on w_1. Similarly, for w_2.

1.6 Regular Expressions

1.6.1 Translating Automata to Regular Expressions

Extra material

Given a nondeterministic automaton $A = \langle Q, \Sigma, \delta, q_0, F\rangle$ we construct a regular expression for the language of A. Let $Q = \{q_0, \ldots, q_{n-1}\}$ be the states of the automaton. We construct the regular expression for the language of A gradually. Effectively, we construct regular expressions for the words that lead from state to state in A. Thus, the regular expression $r_{i,j,k}$ defines the language of all words that take A from state q_i to state q_k using in between at most the states in $\{q_0, \ldots, q_{j-1}\}$. Let $L_{i,j,k}$ denote the language defined by $r_{i,j,k}$. Notice that the languages $L_{i,0,k}$ allow passing nowhere between q_i and q_k. The language $L_{i,j,k}$ can be also thought of as the language of the automaton $A_{i,j,k}$ where $A_{i,j,k} = \langle Q', \Sigma, \delta', \dot{q}_i, \dot{q}_k\rangle$ with the following components:

- $Q' = \{q_0, \ldots, q_{j-1}\} \cup \{\dot{q}_i, \dot{q}_k\}$.

  That is, states $\dot{q}_i$ and $\dot{q}_k$ are special copies of q_i and q_k that are distinct from them. Notice that if $i = k$ then $\dot{q}_i$ and $\dot{q}_k$ are the same state.

- For $q \in \{q_0, \ldots, q_{j-1}\}$ and letter $a \in \Sigma$ we set $\delta'(q, a) = \delta(q, a) \cap \{q_0, \ldots, q_{j-1}\} \cup \{\dot{q}_k \mid q_k \in \delta(q, a)\}$.
- For every $a \in \Sigma$ we set $\delta(\dot{q}_i, a) = \delta(q_i, a) \cap \{q_0, \ldots, q_{j-1}\} \cup \{\dot{q}_k \mid q_k \in \delta(q, a) \text{ and } i \neq k\}$.
- For every $a \in \Sigma$ we set $\delta(\dot{q}_k, a) = \emptyset$. Notice that in case that $i = k$ then $\dot{q}_i = \dot{q}_k$ cannot have self loops.

That is, apart from the initial and final state, restrict A only to use the states $\{q_0, \ldots, q_{j-1}\}$ and consider the language leading from q_i to q_k.

Formally, we define the regular expressions $r_{i,j,k}$ as follows.

- For every i and k:

  $$L_{i,0,k} = \{a \in \Sigma \mid q_k \in \delta(q_i, a)\} \cup \{\epsilon \mid i = k\}$$

  That is, $L_{i,0,k}$ includes the letters in the alphabet for which q_k is *directly* reachable from q_i. If in addition $i = k$, then $L_{i,0,k}$ includes ϵ. If $i \neq k$ and there is no direct transition from q_i to q_k, then $L_{i,0,k} = \emptyset$.

  Then, the regular expression $r_{i,0,k}$ is the disjunction of the letters in $L_{i,0,k}$ (including ϵ) or $\emptyset$ if $L_{i,0,k}$ is empty.

- Suppose that we have defined the regular expressions $r_{i,j,k}$ for all i and k and for all $j \leq l$. We now define the regular expression $r_{i,l+1,k}$, for every i and k.

  $$r_{i,l+1,k} = r_{i,l,k} \mid r_{i,l,l} \cdot r^*_{l,l,l} \cdot r_{l,l,k}$$

  It is simple to see that the language of $r_{i,l+1,k}$ is indeed $L_{i,l+1,k}$. Consider a word in $L_{i,l+1,k}$. Then, there is a run that starts in q_i and ends in q_k and passes in between at most in state q_l. If the run does not visit q_l at all, then the word labeling this run is in $L_{i,l,k}$ and in particular in $r_{i,l+1,k}$. If the run visits q_l then it can be partitioned to segments that go between the different visits to q_l. Then, the word is in $r_{i,l,l} \cdot r^*_{l,l,l} \cdot r_{l,l,k}$.

  In the other direction, for similar reasons, words in $r_{i,l+1,k}$ can be shown to label runs in A leading from q_i to q_k visiting at most q_l.

We note that the language $L_{i,n,k}$ does not restrict the states in the automaton that are allowed to use. Indeed, it allows to use *all* states $\{q_0, \ldots, q_n\}$.

Now we are ready to construct a regular expression for the language of A. Let $F = \{q_{f_1}, \ldots, q_{f_m}\}$ be the set of accepting states of A. Then, $r_A = r_{0,n,f_1} \mid r_{0,n,f_2} \mid \cdots \mid r_{0,n,f_m}$. Clearly, the language of r_A is the language of A.

Corollary 1.13 *For every automaton A there is a regular expression r such that $L(r) = L(A)$.*

1.7 Minimizing Deterministic Automata

Consider a deterministic automaton $A = \langle Q, \Sigma, \delta, q_0, F \rangle$. Denote $Q = \{q_0, \ldots, q_{n-1}\}$. For a word w and a state q, let $\delta(q, w)$ denote the unique state that is reached from q after reading the word w. Formally, $\delta(q, \epsilon) = q$ and for every $w \cdot a$ we set $\delta(q, wa) = \delta(\delta(q, w), a)$. We have shown the following algorithm for minimization of A.

```
1   for every i < j
2     if ((q_i ∈ F && q_j ∉ F) || (q_i ∉ F && q_j ∈ F))
3       different(i,j) := true
4     else
5       different(i,j) := false
6   found := true;
7   while (found);
8     found := false;
9     for (i in 0..n-1)
10      for (j in i+1..n-1)
11        if (!different(i,j))
12          foreach (σ ∈ Σ)
13            if (different(δ(q_i,σ),δ(q_j,σ)))
14              different(i,j) := true;
15              found := true;
16          end -- foreach (σ ∈ Σ)
17      end -- for (j ...)
18    end -- for (i ...)
19  end -- while (found)
```

Now, define two states q_i and q_j as equivalent if `different(i,j)` is `false`, denoted $q_i = q_j$. Clearly, $q_i = q_i$ for every i. For every state q_i let $\mathtt{rep}(i)$ denote the minimal j such that $q_i = q_j$.

Lemma 1.14 *For every two states q_i, q_j we have $q_i \neq q_j$ iff word w such that either $\delta(q_i, w) \in F$ and $\delta(q_i, w) \notin F$ or $\delta(q_i, w) \notin F$ and $\delta(q_j, w) \in F$.*

Proof We show that if $q_i \neq q_j$ such a word w exists. The proof is by induction on the number of times the algorithm passes through the while loop before marking `different(i,j)` as `true`.

If the initialization stage sets `different(`i`,`j`)` as `true`, then clearly ϵ fulfils the lemma.

Suppose that this is true for all states marked as different in the kth run through the while loop. Consider two states q_i and q_j such that `different(`i`,`j`)` is set `true` during the $k+1$th run through the loop. Then, there are states q'_i and q'_j such that `different(`i'`,`j'`)` during the $k+1$th run through the loop and such that for some letter σ we have $q_{i'} = \delta(q_i, \sigma)$ and $q_{j'} = \delta(q_j, \sigma)$. Then, by the induction assumption there is w such that (without loss of generality) $\delta(q_{i'}, w) \in F$ and $\delta(q_{j'}, w) \notin F$. Then, $\delta(q_i, \sigma \cdot w) \in F$ and $\delta(q_j, \sigma \cdot w) \notin F$.

Suppose that there is a word w such that $\delta(q_i, w) \in F$ and $\delta(q_j, w) \notin F$. Then, the unique runs reading w starting from q_i and q_j define a sequence of pairs that must be inequivalent in the algorithm. It follows that $q_i \neq q_j$.

It follows that if $q_i = q_{i'}$ then $\mathsf{r}ep(i) = \mathsf{r}ep(i')$. Otherwise, there will be a word distinguishing $q_{\mathsf{r}ep(i)}$ from $q_{\mathsf{r}ep(i')}$ and that word would be used to distinguish q_i from $q_{i'}$.

Now, we construct the minimized automaton. Let $A' = \langle Q', \Sigma, \delta', q_0, F' \rangle$, with the following components.

- $Q' = \{q_i \mid i = rep(i)\}$
- $\delta'(q_i, \sigma) = \mathsf{r}ep(\delta(q_i, \sigma))$.
- $F' = \{q_f \mid q_f \in F \text{ and } f = rep(f)\}$.

Lemma 1.15 *$q_j = \delta(q_i, \sigma)$ iff $q_{\mathsf{r}ep(j)} = \delta(q_{\mathsf{r}ep(i)}, \sigma)$.*

Proof Suppose that $q_j = \delta(q_i, \sigma)$, $q_l = \delta(q_{\mathsf{r}ep(i)}, \sigma)$, and $q_l \neq q_{\mathsf{r}ep(j)}$. Then, $q_l \neq q_j$. But this implies that $q_i \neq q_{\mathsf{r}ep(i)}$.

Corollary 1.16 *$L(A) = L(A')$*

Proof Consider a word w accepted by A. Then, there is a run $r = q_0, q_{i_1}, \ldots, q_{i_m}$ showing that w is accepted. Then, by Lemma 1.15, $r' = q_{\mathsf{r}ep(0)}, q_{\mathsf{r}ep(i_1)}, \ldots, q_{\mathsf{r}ep(i_m)}$ is the run of A' on w. If r' is not accepting then it shows for each of the states that it is not equivalent to its representative.

The other direction is similar.

1.8 The Pumping Lemma

Extra Material

Lemma 1.17 *For every regular language L there is a natural number n such that for every word $w \in L$ such that $|w| > n$ we can find $w = xyz$ such that $|y| > 0$ and $|xy| \leq n$ such that $xy^p z \in L$ for every $p \geq 0$.*

Proof By assumption L is regular. Then, there is an automaton A accepting the language L. Let n be the number of states of A. We claim that n satisfies the requirements of the Lemma.

Consider a word w such that $|w| > n$. Let $q_0, q_1, \ldots, q_m$ be an accepting run of A on w. By assumption $m > n$. By the pigeon-hole principle there must be $j < k$ such that $q_j = q_k$.

Let $x = a_0, \ldots a_j$, $y = a_{j+1}, \ldots, a_k$, and $z = a_{k+1}, \ldots, a_{m-1}$.

First, as $k \leq n$ we conclude that $|xy| \leq n$. Furthermore, as $j < k$ we conclude that $|y| > 0|$.

Finally, the run $q_0, \ldots q_j, q_{k+1}, \ldots, q_m$ is an accepting run of A on xz. Then, for every $p > 1$ the following is an accepting run of A on $xy^p z$.

$$\begin{array}{rcl} r & = & q_0, \ldots, q_j, \\ & & q_{j+1}, \ldots, q_k, \\ & & q_{j+1}, \ldots, q_k, , \\ & & \ldots \\ & & q_{j+1}, \ldots, q_k, \\ & & q_{k+1}, \ldots, q_m \end{array}$$

There is a more elaborate version of the pumping lemma that identifies more concretely *parts* of words that are long enough.

Lemma 1.18 *For every regular language L there is a natural number n such that for every word $w \in L$ and every partition $w = xyz$ such that $|y| > n$ there is a partition $y = uvt$ such that $|v| > 0$ and $|uv| \leq n$ such that $xuv^p tz \in L$ for every $p \geq 0$.*

The proof of this stronger version is a variant of the proof above.

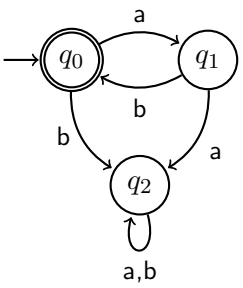
1.9 Exercises

Exercises marked with ^{s} have solutions in Section 1.10.

The topics of exercises are:

- Nondeterministic Constructions - 17-23,25,26.
- Modelling - 16,27
- Regular Expressions - 24, 31, 33, 40, 44
- Minimization - 32, 36

^{s}1. Consider the following automaton over the alphabet $\Sigma = \{a, b\}$:



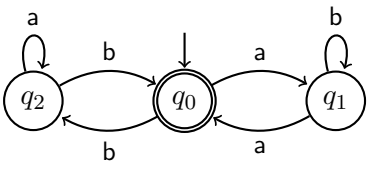
(a) For each of the following words w over $\{a, b\}$, determine the state of the automaton after processing w. Which of these words are accepted by the automaton?

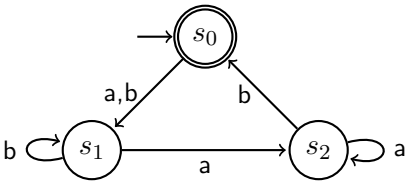
- ϵ
- $aabaa$
- $ababab$

(b) What is the language of the automaton? (For $w \in \Sigma^*$, write w^n for $\overbrace{w \cdots w}^{n \text{ times}}$.) Justify your answer briefly.

(c) Write down a formal definition of this *deterministic* automaton as a 5-tuple $A = (Q, \Sigma, \delta, q_0, F)$ specifying how each of the individual components is defined.

^{s}2. Consider the following automata over the alphabet $\Sigma = \{a, b\}$:

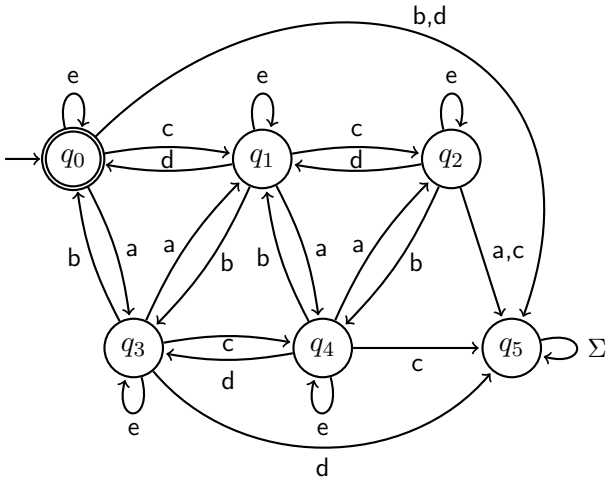




(a) For each of the following words w over $\{a, b\}$, write the run of the two automata on them and determine which are accepted.

- ϵ
- *aabaa*
- *ababab*

3. Consider the following automaton A over the alphabet $\Sigma = \{a, b, c, d, e\}$:



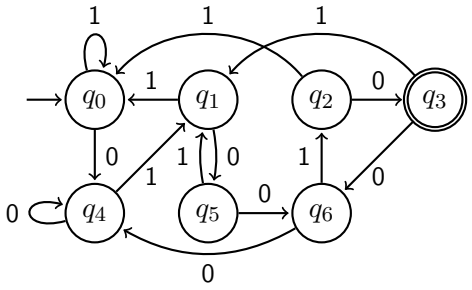
(a) Write down a formal definition of the automaton A as a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ specifying how each of the individual components is defined.

(b) Find three long words accepted by the automaton. Write down the accepting runs of all three of them.

(c) What is the language L accepted by the automaton A? Justify your answer briefly.

(d) Draw the diagram of the automaton accepting $\Sigma^* \setminus L$.

4. Consider the following automaton A over the alphabet $\Sigma = \{0, 1\}$:



(a) Write down a formal definition of the automaton A as a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ specifying how each of the individual components is defined.

(b) What is the shortest word accepted by the automaton? Find two more words accepted by the automaton and write down the accepting runs of each of the three words.

(c) What is the language L accepted by the automaton A? Justify your answer briefly.

(d) Draw the diagram of the automaton accepting $\Sigma^* \setminus L$.

5. Let $\Sigma = \{0, 1, 2\}$. Construct a deterministic automaton A such that

$$L(A) = \{s \in \Sigma^* : s \text{ does not contain 01202 as a subword}\}.$$

For example, every word shorter than five letters (such as ϵ, 210, or 2222) is in L. Also 012010 and 012012012 are in L; But 01202, 00120221, and 11012021 are not in L.

Test your automaton on four strings from Σ^* of length at least seven, two of which are in $L(A)$ and two of which are not. Explain what happens in each case.

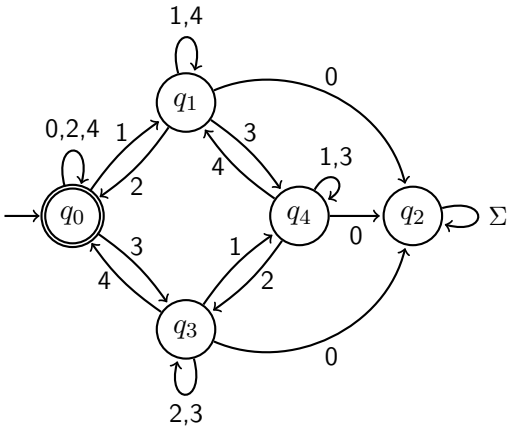
6. Let $\Sigma = \{0, 1\}$. Construct a deterministic automaton A such that

$$L(A) = \{w \in \Sigma^* : w \text{ ends with } 0010100\}.$$

Test your automaton on four strings from Σ^* of length at least 10, two of which are in $L(A)$ and two of which are not. Explain what happens in each case.

A diagram is sufficient as an answer to this question.

7. Consider the following automaton A over the alphabet $\Sigma = \{0, 1, 2, 3, 4\}$:



(a) Write down a formal definition of the automaton A as a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ specifying how each of the individual components is defined.

(b) Find three long words accepted by the automaton. Write down the accepting runs of all three of them.

(c) What is the language L accepted by the automaton A? Justify your answer briefly.

(d) Draw the diagram of the automaton accepting $\Sigma^* \setminus L$.

[s] 8. Consider the alphabet $\Sigma = \{0, 1\}$ and recall the automaton A discussed in Lecture 2 that accepts precisely those binary words (i.e. words over Σ) that contain an even number of 1's.

(a) By suitably modifying A, define a deterministic automaton B that accepts precisely those binary words that contain an even number of 0's.

(b) Use the construction of the union automaton to obtain a deterministic automaton that accepts precisely those binary words that contain either an even number of 0's or an even number of 1's.

(c) Use the construction of the intersection automaton to obtain a deterministic automaton that accepts precisely those binary words that contain both an even number of 0's and an even number of 1's.

9. Let Σ be the alphabet $\{0, 1\}$ and then let:

$$L_1 = \{w \in \Sigma^* : w = w'abc \text{ and a=c}\};$$

$$L_2 = \{w \in \Sigma^* : w = aw'b \text{ and a=b}\}.$$

That is, L_1 includes all words in which the last letter and the third letter from the end are the same. Language L_2 includes all words in which the first and the last letters are the same.

(a) Construct a deterministic automaton A_1 accepting L_1.

(b) Construct a deterministic automaton A_2 accepting L_2.

(c) Construct a deterministic automaton A_3 accepting $L_1 \cup L_2$.

(d) Construct a deterministic automaton A_4 accepting $L_1 \cap L_2$.

10. Let Σ be the alphabet $\{0, 1, 2\}$ and then let:

$$L_1 = \left\{ w \in \Sigma^* \;\middle|\; \begin{array}{l} w = 1w' \text{ and } w' \in \{0,1\}^* \text{ or} \\ w = 2w' \text{ and } w' \in \{0,2\}^* \end{array} \right\};$$

$$L_2 = \left\{ a_0 a_1 \cdots a_{n-1} \in \Sigma^* \;\middle|\; \forall i \in [0..n-1] \; \begin{array}{l} a_i = 0 \to i = 0 \bmod 3 \\ a_i = 1 \to i = 1 \bmod 3 \end{array} \right\}.$$

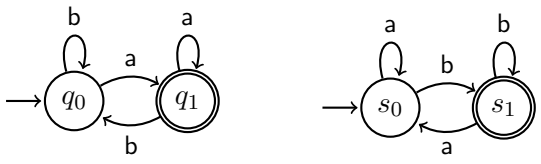
That is, a word in L_1 either starts with 1 and then includes only 0 and 1 or starts with 2 and then includes only 0 and 2. A word is in L_2 if whenever 0 appears it is in a location divisible by 3 and whenever 1 appears it is in a location whose remainder when divided by 3 is 1.

(a) Construct deterministic automata A_1 and A_2 accepting L_1 and L_2.

(b) Construct a deterministic automaton A_3 accepting $L_1 \cup L_2$.

(c) Construct a deterministic automaton A_4 accepting $L_1 \cap L_2$.

[s] 11. Consider the two automata over the alphabet $\Sigma = \{a, b\}$.



(a) Apply the product construction for the union and intersection of the two automata. What is the language of their union? What is the language of their intersection?

(b) Construct an automaton for the intersection of their complements. What is the language of this automaton?

[s] 12. Construct a deterministic automaton for the language

$$L = \{w \in \Sigma^* : |w| \geq 2,\ w \text{ begins with an } a \text{ and ends with an } a\}$$

over the alphabet $\Sigma = \{a, b\}$. Draw a diagram that represents your automaton and also write down the formal definition of your automaton in terms of a 5-tuple.

[s] 13. Construct a deterministic automaton for the language

$$L = \{w \in \Sigma^* : w \text{ ends in } 01100\}$$

over the alphabet $\Sigma = \{0, 1\}$. Draw a diagram that represents your automaton and also write down the formal definition of your automaton in terms of a 5-tuple.

14. Let $\Sigma = \{0, 1, 2\}$. Construct a deterministic automaton A such that

$$L(A) = \{w \in \Sigma^* : w \text{ ends with } 0120101\}.$$

Test your automaton on four strings from Σ^* of length at least 10, two of which are in $L(A)$ and two of which are not. Explain what happens in each case.

[s] 15. Consider the alphabet $\Sigma = \{a, b\}$. For a word w and a letter $\sigma \in \Sigma$ denote by $\#_\sigma(w)$ the number of σs appearing in w. Construct an automaton for the set of words w such that (a) $\#_a(w) = \#_b(w)$ and (b) in every prefix u of w we have $0 \leq \#_a(u) - \#_b(u) \leq 2$.

For example, every word starting with b is not in this language. As for the prefix b we have $\#_a(b) - \#_b(b) = -1$. Similarly, the word *abaabaa* and all its possible extensions is not in the language. The word *aabb* is in the language and so is *abaababb*. The word $w = aabbaba$ is not in the language as $\#_a(w) = 4$ and $\#_b(w) = 3$.

Do you think that the language $\{w \mid \#_a(w) = \#_b(w)\}$ is regular?

[s]16. Consider a database system that is used by two parties. Each party can read from the database and is also able to change its content. Events are created by each party when:

- it starts reading from the database;
- it stops reading from the database;
- it starts writing to the database;
- it stops writing to the database.

(a) What events are needed to model the above setup?

By a *transaction* we mean a finite sequence consisting of read/write requests and completion events. We call a transaction *legal* if:

- every completion event is preceded by the corresponding event that signifies the beginning of that operation;
- each party can only write to the database if no other reading or writing action is currently taking place;
- each party must terminate a read/write action before embarking on a new operation of the same type;
- all actions have been terminated.

A read operation by one party is allowed to commence whilst a read operation by the other party is being executed.

(a) Design an automaton model for the above system. The automaton should accept the language consisting of the set of all legal transactions. Note that legal transactions can be arbitrarily long. Explain which situation is reflected by each of the states in your automaton. (A diagram is quite sufficient here. There will be a "fail state"; to avoid a cluttered diagram, you need not draw the arrows leading to the fail state.)

(b) Check your automaton by choosing both a legal transaction l and an illegal transaction i each of length at least 5; verify that the automaton accepts l and rejects i.

[s]17. Let Σ be the alphabet $\{0, 1\}$ and then let:

- $L_1 = \{w \in \Sigma^* : w \text{ contains the subword } 010\}$
- $L_2 = \{s \in \Sigma^* : w \text{ contains an odd number of 0's}\}$

(a) Construct nondeterministic automata A_1 and A_2 accepting L_1 and L_2 respectively.

(b) Construct a nondeterministic automaton accepting $L_1 \cdot L_2$.

(c) Determinize the automaton for $L_1 \cdot L_2$.

18. Let Σ be the alphabet $\{0, 1, 2\}$ and then let:

$$L_1 = \{a_0 a_1 \cdots a_{n-1} \in \Sigma^* : a_{n-1} = a_{n-3}\};$$

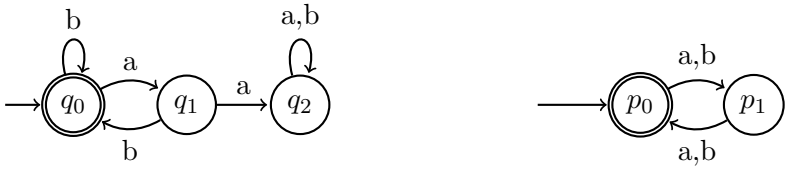
$$L_2 = \{a_0 a_1 \cdots a_{n-1} \in \Sigma^* : a_0 = a_{n-1}\}.$$

(a) Construct nondeterministic automata A_1 and A_2 accepting L_1 and L_2 respectively.

(b) Construct a nondeterministic automaton accepting $L_1 \cup L_2$.

(c) Construct a nondeterministic automaton accepting $L_1 \cdot L_2$.

[s] 19. Consider the following automata over the alphabet $\Sigma = \{a, b\}$:



(a) Which of these automata is deterministic?

(b) Use the construction described in the lectures to obtain an automaton A that accepts precisely those strings accepted by both the automata shown above.

[s] 20. Consider the alphabet $\Sigma = \{h, w, c\}$ whose elements stand for:

- h: at home;
- w: at work;
- c: at the cafe,

and consider a person that follows the following strategy:

- after leaving work in the evening, I either go to the cafe or straight home.
- sometimes, I go from work to the cafe to have lunch, but, whenever I do that, I return to work afterwards.

- I never go to the cafe from home, only from work.

(a) Construct a nondeterministic finite automaton that accepts precisely those strings of locations that adhere to the above strategy and which start and end at home. For example, *h*, *hwcwch*, *hwh* and *hwcwchwh* are valid strings whereas *hcwh* and *hwhwcwcwh* are not.

(b) Convert the resulting automaton to a deterministic automaton.

21. Let Σ be the alphabet $\{0, 1, 2\}$ and then let:

$$L_1 = \{a_0 a_1 \cdots a_{n-1} \in \Sigma^* : a_0 = a_{n-2}\};$$

$$L_2 = \{w \in \Sigma^* : w = w'012210w''\}.$$

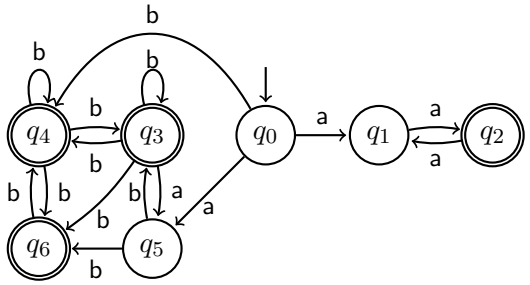
That is, L_1 is the language of words that have their first and one before last letters the same and L_2 is the language of words that contain 012210 as a subword.

(a) Construct nondeterministic automata A_1 and A_2 accepting L_1 and L_2 respectively.

(b) Construct a nondeterministic automaton accepting $L_1 \cup L_2$.

(c) Construct a nondeterministic automaton accepting $L_1 \cdot L_2$.

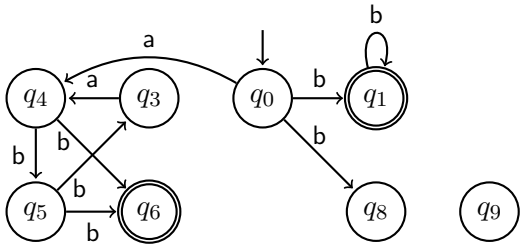
[s] 22. Consider the following nondeterministic automaton A over the alphabet $\Sigma = \{a, b\}$:



(a) Give a formal definition of the automaton as a 5-tuple $(Q, \Sigma, \delta, q_0, F)$. Make sure that you specify each component of the 5-tuple fully.

(b) Construct a deterministic automaton B that recognizes $L(A)$.

[s] 23. Consider the following nondeterministic automaton A over the alphabet $\Sigma = \{a, b\}$:



(a) Give a formal definition of the automaton as a 5-tuple $(Q, \Sigma, \delta, q_0, F)$. Make sure that you specify each component of the 5-tuple fully.

(b) Construct a deterministic automaton B that recognizes $L(A)$.

(c) Write down a regular expression r that represents the same language, i.e. write down a regular expression r such that $L(r) = L(A)$. Justify your answer.

It may be easier to consider the nondeterministic automaton.

[s] 24. Construct nondeterministic automata over the alphabet $\Sigma = \{a, b, c\}$ for the following languages.

(a) $L = \{w \mid w \text{ contains the subword } aabab\}$.

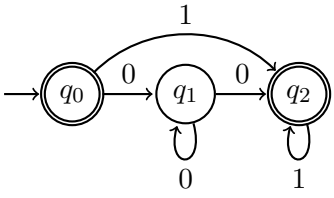
(b) $L = \{w \mid w \text{ ends with } aabab\}$.

(c) $L = \{w \mid \text{if } w \text{ contains } c \text{ then it starts with } a\}$.

(d) $L = \{w \mid b \text{ appears only directly after } a\}$.

(e) $L = \{w \mid w = uv \text{ such that } \#_a(u) \text{ is even and } \#_b(v) \text{ is even}\}$.

[s] 25. Consider the following nondeterministic automaton A over the alphabet $\Sigma = \{0, 1\}$:



(a) Give a formal definition of the automaton A as a 5-tuple $(Q, \Sigma, \delta, q_0, F)$. Make sure that you specify each component of the 5-tuple fully.

(b) Construct a deterministic finite automaton that accepts the same language as A.

(c) Describe the language accepted by A.

[s] 26. What are the languages of these two automata over the alphabet $\Sigma = \{a, b\}$.

Apply the subset construction on the two automata. Does the result support your previous answer?

[s] 27. Consider two processors P_1 and P_2 that process jobs (of the same type). The jobs arrive at a queue Q which can hold at most one job. If a job arrives and finds that the queue is full, then it is simply rejected. The rejection of jobs should not be part of the model that you are being asked to develop here.

The queue dispatches jobs to the processors P_1 and P_2 according to the following strategy:

- if processor P_1 is *not* busy, then the queue dispatches the job to processor P_1;
- if P_1 is busy and P_2 is not busy, then the job is dispatched to processor P_2;
- if P_1 and P_2 are both busy, then the queue does nothing.

(a) What events are needed to model the above system? Write down the events, together with a short explanation of what each means in the model.

(b) Write down automata that model the queue Q, the processor P_1 and the processor P_2.

(c) Identify those events that need to happen at the same time to guarantee that the combined system will have the desired behaviour. Justify your answer briefly.

(d) Change the automaton models of the processors P_1, P_2 and the queue Q such that any events which you have identified in part c) as needing to happen simultaneously actually have the same name (assuming, of course, that this is not already the case). Construct an automaton representing the combined system involving the queue and the two processors.

28. We are going to model a simple production line. The model includes two input trays for items of types A and B, and a robot arm that removes the items from the input trays. They operate as follows:

    - The input tray for items of type A has 0, 1, or "2 or more" items to be processed.
    - The input tray for items of type B has 0, or "1 or more" items to be processed.
    - Each input tray has sensors that tell it when an item is put on the tray and when an item is taken from the tray.
    - The robot arm can either take an A item or a B item from its respective tray and then passes it on for further production.
    - The robot arm can carry at most one item at a time.

    (a) Write down automata that model the two input trays and the robot arm.

    (b) Identify those events in the automata that need to happen at the same time to guarantee that the combined system will have the desired behaviour. Justify your answer briefly.

    (c) Change the automaton models of the trays and the arm such that all events that you have identified in part (b) as needing to happen simultaneously actually have the same name (assuming, of course, that this is not already the case).

    (d) Construct an automaton representing the combined system involving all the parts.

29. We are going to model a variant of the consumer producer problem. In this case, there are two producers, putting information into a buffer, and one consumer, reading information from the buffer. They operate as follows:

    - Each producer can either idle or produce.
    - After production, a producer can pass a message to the buffer and returns to the idle state.
    - The buffer has two places.
    - The consumer can either idle or consume.
    - In order to consume, the consumer must take a message from the buffer. After consuming it returns to the idle state.

- The buffer cannot deliver messages when it is empty nor receive messages when it is full.

(a) Write down automata that model the two producers, the buffer, and the consumer.

(b) Identify those events in the automata that need to happen at the same time to guarantee that the combined system will have the desired behaviour. Justify your answer briefly.

(c) Change the automaton models of the producers, the buffer, and the consumer such that all events that you have identified in part (b) as needing to happen simultaneously actually have the same name (assuming, of course, that this is not already the case).

(d) Construct an automaton representing the combined system involving all the parts.

30. We are going to model a single-lane bridge. The model includes a single-lane bridge and sensors sensing whether cars are coming from the left and the right. They operate as follows:

- Each sensor can sense whether there are 0, 1, or "2 or more" cars waiting to enter the bridge from its side.
- Each sensor can sense when a car enters the bridge (i.e., leaves the queue on its side).
- The bridge can sense when a car enters it from either side and when a car leaves it.
- At any given time there are at most two cars on the bridge travelling in the same direction.

(a) Write down automata that model the two sensors and the bridge.

(b) Identify those events in the automata that need to happen at the same time to guarantee that the combined system will have the desired behaviour. Justify your answer briefly.

(c) Change the automaton models of the sensors and the bridge such that all events that you have identified in part (b) as needing to happen simultaneously actually have the same name (assuming, of course, that this is not already the case).

(d) Construct an automaton representing the combined system involving all the parts.

(e) Does your model have an execution where cars from one side wait forever?

[s] 31. Consider the alphabet $\Sigma = \{a, b\}$ and write down regular expressions for the following languages:

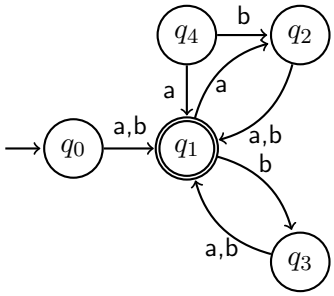
(a) $L_1 = \{s \in \Sigma^* : s \text{ has odd length}\}$.

(b) $L_2 = \{s \in \Sigma^* : s \text{ contains } aba\}$.

(c) $L_3 = \{s \in \Sigma^* : s \text{ ends in } a\}$.

(d) $L_4 = \{s \in \Sigma^* : s \text{ does not contain } aa\}$.

[s] 32. Consider the following automaton over the alphabet $\Sigma = \{a, b\}$:



(a) Use the construction presented in Lecture 9 to optimise this automaton.

(b) What is the language accepted by this automaton? (A regular expression denoting the language is sufficient here.) Justify your answer.

[s] 33. Consider the following regular expressions over the alphabet $\Sigma = \{0, 1\}$. Explain what is the language of each regular expression. Give a word in the language and a word not in the language for each.

(a) $(0^*(10^+)^*)^*$

(b) $1^*(1^*01^*01^*01^*)^*$

(c) $\Sigma^*010\Sigma^*$

(d) $(\Sigma^*0)^*$

(e) $(\Sigma 0)^*$

(f) Σ^*0

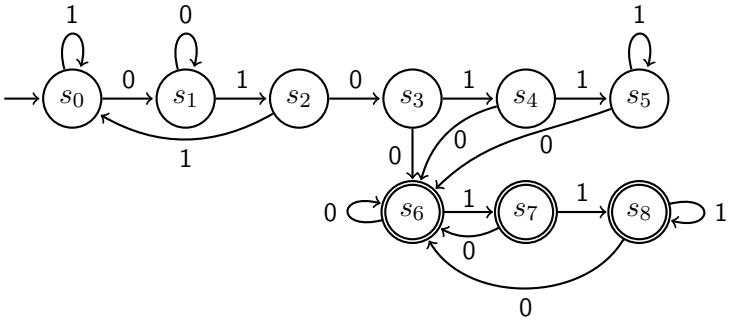
34. Write down regular expressions r that represents the languages of the automata in questions 18, 19, 21, and 22

    It may be easier to consider the nondeterministic automaton.

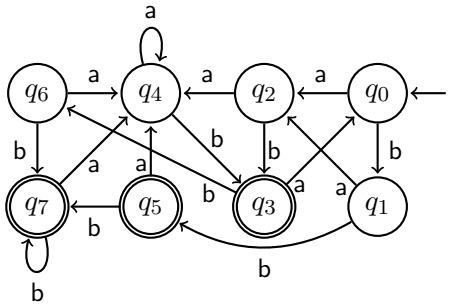
35. Consider the alphabet $\Sigma = \{a, b\}$ and write down regular expressions for the following languages:

    (a) $L_1 = \{w \in \Sigma^* : \sharp_b(w)\%3 = 0\}$.
    (b) $L_2 = \{w \in \Sigma^* : \text{the 5th letter from the end in } w \text{ is } b\}$.
    (c) $L_3 = \{w \in \Sigma^* : \text{after every } a \text{ there's a } b\}$.
    Notice that the b does not have to appear right after the a.
    (d) $L_4 = \{w \in \Sigma^* : \text{ the number of } a\text{s in } w \text{ is } 4\}$.
    (e) $L_5 = \{w \in \Sigma^* : \text{ the first letter and the last letter in } w \text{ are the same}\}$.

[s] 36. Minimize the following automaton (from Surgery 2).

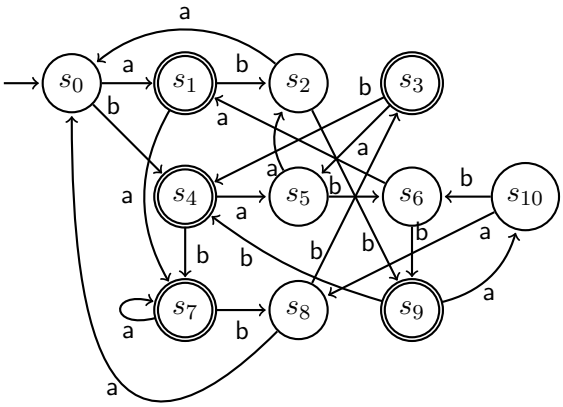


37. Minimize the following automaton over the alphabet $\Sigma = \{a, b\}$:



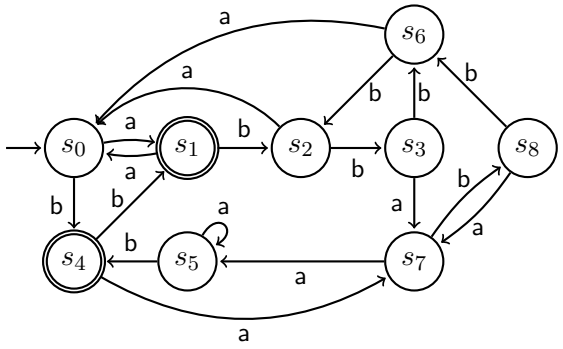
    Either show the steps in your minimization or write in the table a word that distinguishes the two states.

38. Minimize the following automaton over the alphabet $\Sigma = \{a, b\}$:



Either show the steps in your minimization or write in the table a word that distinguishes the two states.

39. Minimize the following automaton over the alphabet $\Sigma = \{a, b\}$:



Either show the steps in your minimization or write in the table a word that distinguishes the two states.

[s] 40. Construct nondeterministic automata for the following regular expressions.

(a) $((a^*b)^*(b^*a)^*)^*$

(b) $(a(a|d)^*b(a|b|d)^*c(a|b|c|d)^*)^*$

(c) $((a|c|d)^*(bb)^*)^*$

(d) $(a|c|d)^*(b(b|c|d)^*)^*$

41. Let Σ be the alphabet $\{a, b\}$ and then let:

$$L_1 = \{w \in \Sigma^* \mid \sharp_b(w) = 2\}$$

$$L_2 = \{w \in \Sigma^* \mid \sharp_b(w) = 3\}$$

That is, a word in L_1 has 2 bs and a word in L_2 has 3 bs.

(a) Construct deterministic automata A_1 and A_2 accepting L_1 and L_2.

(b) Construct a deterministic automaton A_3 accepting $L_1 \cup L_2$.

(c) Construct a nondeterministic automaton A_4 accepting $L_1 \cup L_2$.

(d) Construct a nondeterministic automaton A_5 accepting $(L(A_4))^*$.

(e) Determinize the automaton A_5.

42. Consider the alphabet $\Sigma = \{a, b\}$ and write down regular expressions for the following languages:

(a) $L_3 = \{w \in \Sigma^* : w$ starts with a and ends in $a\}$.

(b) $L_4 = \{w \in \Sigma^* : w$ starts with aa or ends in $bb\}$.

(c) $L_1 = \{w \in \Sigma^* : \sharp_a(w)$is a multiple of 3$\}$.

(d) $L_2 = \{w \in \Sigma^*$:w=uv, where $|u|$ is even and $\sharp_b(v) \neq 3\}$.

43. Consider the alphabet $\Sigma = \{a, b\}$ and write down regular expressions for the following languages:

(a) $L_1 = \{w \in \Sigma^* :$ the second letter and the one before last letter in w are the same$\}$.

(b) $L_2 = \{w \in \Sigma^* : \sharp_a(w)$ is even or w ends with $abba\}$.

(c) $L_3 = \{w \in \Sigma^* :$ the distance between every two adjacent b's in w is a multiple of 3$\}$.

(d) $L_4 = \{w \in \Sigma^* :$ the distance between every two b's in w is a multiple of 3$\}$.

(e) $L_5 = \{w \in \Sigma^*$:w=uvt, where $\sharp_b(u)$ is even, $v = aaa$, and $\sharp_b(t)$ divides by 3$\}$.

[s] 44. Translate the regular expressions $((ab)^*(ba)^*)^*$ and $((ab^*)(ba)^*)$ to nondeterministic autoamta. Determinize and minimize them.

45. We show that nondeterministic automata are exponentially more succinct than deterministic automata.

The powerset construction starts from a nondeterministic automaton with n states and, in the worst case, produces a deterministic automaton with 2^n states. This is quite bad. Unfortunately, it is unavoidable.

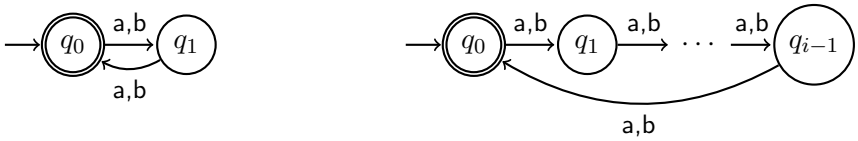
In order to show that it is impossible to find a better construction for determinization you have to prove the following theorem.

Theorem *There is a family of languages $\{L_n\}_{n\in\mathbb{N}}$ such that for every $n \in \mathbb{N}$ there is a nondeterministic automaton accepting L_n with number of states $c \cdot n$ and the minimal deterministic automaton accepting L_n has $2^{d \cdot n}$ states, for some constants c and d.*

A family of languages is a parameterized set of languages. For example, let

$$L_n = \{w \ : \ |w| \text{ is a multiple of } n\}.$$

So L_1 is Σ^*, L_2 is $(\Sigma \cdot \Sigma)^*$, and L_i is $(\Sigma^i)^*$. This is an infinite set of languages, where every language is parameterized by the index i. In this case, L_i is easy to identify with a deterministic automaton with i states. For example, if $\Sigma = \{a, b\}$ then A_2, the automaton that accepts L_2, and A_i, the automaton that accepts L_i, are:



Your task is to prove the theorem. This includes the following parts:

- Find the required family of languages.
- Find the constant c.
- Show that every deterministic automaton with less than $2^{c \cdot n}$ states must make mistakes if it accepts all the words in L_n. This is similar to the proof that $L = \{a^n b^n \mid n \in \mathbb{N}\}$ is not regular. Every automaton that has less than $2^{c \cdot n}$ states and accepts all the words in L **must** accept other words as well.

Notice, that you are free to choose the alphabet as you like but the alphabet $\{a, b\}$ should be sufficient.

46. We show that a variant of the algorithm for minimization can reduce the size of nondeterministic automata.

    Consider a nondeterministic automaton $A = (Q, \Sigma, \delta, q_0, F)$. A relation $R \subseteq Q \times Q$ is a *bisimulation relation* if for every pair $(q, p) \in R$ the following conditions hold:

    - We have $q \in F$ iff $p \in F$.
    - For every $\sigma \in \Sigma$ and $q' \in \delta(q, \sigma)$ there is $p' \in \delta(p, \sigma)$ such that $(q', p') \in R$.
    - For every $\sigma \in \Sigma$ and $p' \in \delta(p, \sigma)$ there is $q' \in \delta(q, \sigma)$ such that $(q', p') \in R$.
    - For every $q' \in \delta(q, \epsilon)$ either $(q', p) \in R$ or there is some $p' \in \delta(p, \epsilon)$ such that $(q', p') \in R$.
    - For every $p' \in \delta(p, \epsilon)$ either $(q, p') \in R$ or there is some $q' \in \delta(q, \epsilon)$ such that $(q', p') \in R$.

    We say that q is bisimilar to p if there is a bisimulation relation R such that $(q, p) \in R$.

    Your task is:

    (a) Show a variant of the minimization algorithm that computes a bisimulation relation (this will actually be the maximal bisimulation relation but we don't want to get into that).

    (b) Show that if two states are found to be not bisimilar by the algorithm then there is a word w accepted by the first and rejected by the second.

    (c) Show that the automaton resulting from unifying bisimilar states accepts the same language as the original.

47. A *universal* automaton is structurally like a nondeterministic automaton. That is $U = (Q, \Sigma, \delta, q_0, F)$, where Q is a finite set of states, Σ is a finite alphabet, $\delta : Q \times \Sigma_\epsilon \to 2^Q$ is a transition relation, q_0 is an initial state, and $F \subseteq Q$ is a set of accepting states. Runs and accepting runs for universal automata are defined just like runs of nondeterministic automata. However, the automaton accepts a word if *all* runs over this word are accepting.

    (a) Show that given a nondeterministic automaton $N = (Q, \Sigma, \delta, q_0, F)$ then the automaton $U = (Q, \Sigma, \delta, q_0, Q \setminus F)$ is its complement. That is, prove that $L(U) = \Sigma^* \setminus L(N)$.

(b) Given a universal automaton U, show how to construct an equivalent deterministic automaton D.

48. Let $N = (Q, \Sigma, \delta, q_0, F)$ be a nondeterministic automaton. That is, $\delta : Q \times \Sigma_\epsilon \to 2^Q$.

    (a) Show that ϵ-transitions are not required.

    That is, construct a nondeterministic automaton $N' = (Q', \Sigma, \delta', q_0', F')$ such that:

    - $L(N) = L(N')$.
    - Q' is equivalent to Q with possibly minor changes.
    - For every $q' \in Q'$ we have $\delta(q', \epsilon) = \emptyset$.

    The trivial solution, where N is converted to an equivalent deterministic automaton using the subset construction, will not be accepted. Clearly, this is possible but then Q' is not equivalent (or similar) to Q.

    (b) Show that ϵ-transitions are sufficient.

    That is, construct a *deterministic* automaton *with* ϵ-transitions $D' = (Q', \Sigma, \delta', q_0', F')$ such that:

    - $L(N) = L(D')$.
    - The number of states of Q' is proportional to the size of Q and Σ. That is, there are at most $p(|Q|, |\Sigma|)$ states in Q', where p is a polynomial in two variables with degree less than 2.
    - For every $q' \in Q'$ and $a \in \Sigma_\epsilon$ we have $|\delta(q', a)| \leq 1$.

    Again, the deterministic automaton obtained by the subset construction is not a good solution.

49. We consider the *until* operation on two languages. Given two languages L_1 and L_2, the language L_3 is the *until* composition of L_1 and L_2, denoted $L_3 = L_1 \,\mathrm{U}\, L_2$, if every word in L_3 can be decomposed such that it has a suffix in L_2 and every strictly longer suffix is in L_1. Formally,

$$L_3 = \left\{ w = a_0 \cdots a_{n-1} \;\middle|\; \begin{array}{l} \text{There is } j \leq n_1 \text{ such that } a_j \cdots a_{n-1} \in L_2 \text{ and} \\ \text{for every } 0 \leq k < j \text{ we have } a_k \cdots a_{n_1} \in L_1 \end{array} \right\}$$

    Given automata $A_1 = (Q, \Sigma, \delta, q_0, F)$ and $A_2 = (S, \Sigma, \rho, s_0, G)$ accepting L_1 and L_2, respectively, construct an automaton for $L_3 = L_1 \,\mathrm{U}\, L_2$.

1.10 Solutions to Selected Exercises

1. (a) The runs of the automaton on the different words are given below:
   - If $w = \epsilon$ the run is q_0. Hence ϵ is accepted;
   - If $s = aabaa$ the run is $q_0, q_1, q_2, q_2, q_2, q_2$. Hence $aabaa$ is rejected;
   - If $s = ababab$ the run is $q_0, q_1, q_0, q_1, q_0, q_1, q_0$. Hence $ababab$ is accepted.

   (b) The language L accepted by the automaton consists of all the words that consist of some number of copies of ab one after another. More formally, we have $L = \{(ab)^n : n \geq 0\}$.

   (c) We let $A = (Q, \Sigma, \delta, q_0, F)$ where:

   - $Q = \{q_0, q_1, q_2\}$;
   - $\Sigma = \{a, b\}$ (as stated);
   - $F = \{q_0\}$,

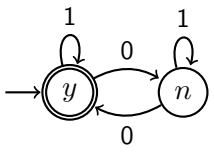
   and the state transition function δ is defined by

   $$\begin{array}{lll} \delta(q_0, a) = q_1; & \delta(q_0, b) = q_2; & \delta(q_1, a) = q_2; \\ \delta(q_1, b) = q_0; & \delta(q_2, a) = q_2; & \delta(q_2, b) = q_2. \end{array}$$

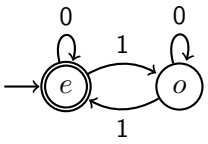
   Here q_2 is a "rejecting sink": once the automaton is in state q_2, it can never leave it again.

2. (a) The runs of the automata on the different words are given below:
   - If $w = \epsilon$ the run of the first automaton is q_0 and the run of the second automaton is s_0. Both automata accept ϵ;
   - If $s = aabaa$ the run of the first automaton is $q_0, q_2, q_2, q_0, q_2, q_2$ and the run of the second automaton is $s_0, s_1, s_2, s_0, s_1, s_2$. Both automata reject $aabaa$;
   - If $s = ababab$ the run of the first automaton is $q_0, q_2, q_0, q_2, q_0, q_2, q_0$ and the run of the second automaton is $s_0, s_1, s_1, s_2, s_0, s_1, s_1$. Hence $ababab$ is accepted by the first and rejected by the second.

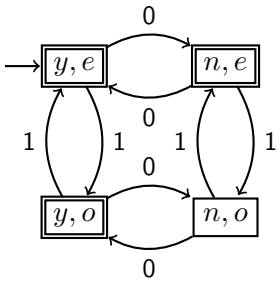
8. (a) We change the 0's to 1's and vice-versa to obtain the automaton that accepts the words with an even number of 0's:



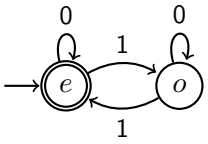
(b) The union automaton that results from running the automaton in part (a) in parallel with the automaton



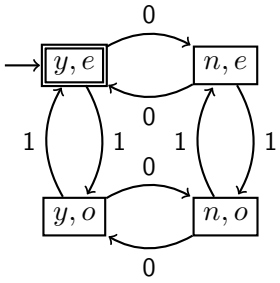
discussed in the lecture gives



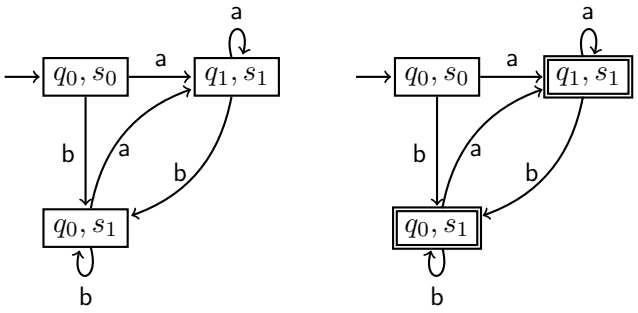
(c) The intersection automaton that results from running the automaton in part (a) in parallel with the automaton



discussed in the lecture gives

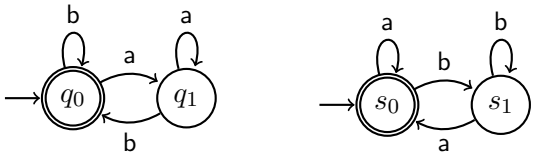


11. (a) The product construction for the intersection and union of the two automata are on the left and right, respectively, below.

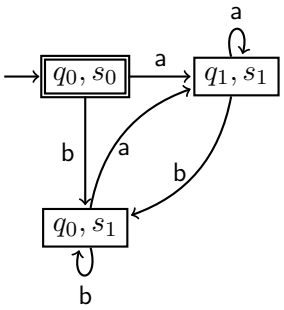


The language of their union is Σ^+ the language of their intersection is $\emptyset$.

(b) The complements of the two automata, respectively, are:



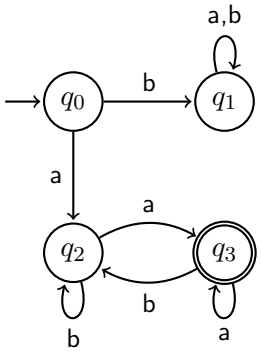
Their intersection is:



and its language is $\{\epsilon\}$.

12. We obtain the following automaton that accepts

$$L = \{w \in \Sigma^* : |w| \geq 2, w \text{ begins with an } a \text{ and ends with an } a\}.$$



Written down as a 5-tuple, we obtain $A = (Q, \Sigma, \delta, q_0, F)$ where:

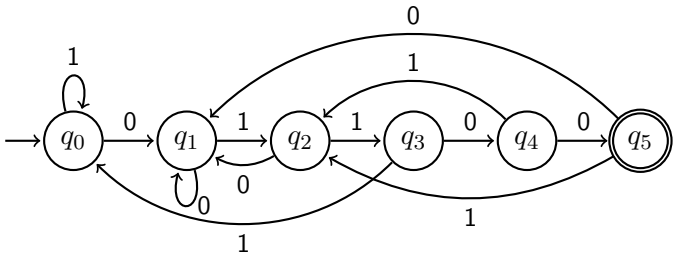
- $Q = \{q_0, q_1, q_2, q_3\}$;
- $\Sigma = \{a, b\}$ (as stated);
- $F = \{q_3\}$;

and the state transition function δ is given by the following table:

	q_0	q_1	q_2	q_3
a	q_2	q_1	q_3	q_3
b	q_1	q_1	q_2	q_2

13. We obtain the following automaton that accepts

$$L = \{w \in \Sigma^* : w \text{ ends in } 01100\}.$$



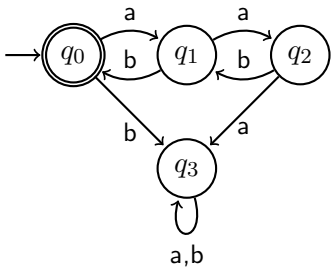
Written down as a 5-tuple, we obtain $A = (Q, \Sigma, \delta, q_0, F)$ where:

- $Q = \{q_0, q_1, q_2, q_3, q_4, q_5\}$;
- $\Sigma = \{0, 1\}$ (as stated);
- $F = \{q_5\}$;

and the state transition function δ is given by the following table:

	q_0	q_1	q_2	q_3	q_4	q_5
0	q_1	q_1	q_1	q_4	q_5	q_1
1	q_0	q_2	q_3	q_0	q_2	q_2

15. The automaton for the language $0 \leq \#_a(w) - \#_b(w) \leq 2$ is:



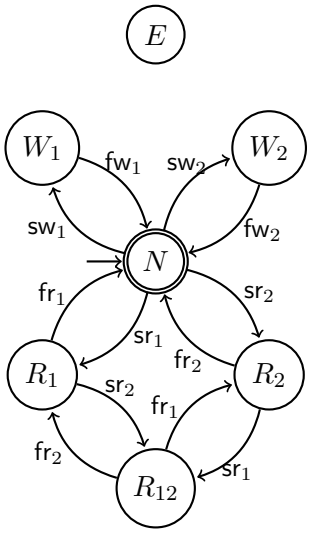
The automaton is in state q_0 when $\#_a(w) = \#_b(w)$. The automaton is in state q_1 when $\#_a(w) - \#_b(w) = 1$. The automaton is in state q_2 when $\#_a(w) - \#_b(w) = 2$. The state q_3 is a rejecting sink state reached after the difference between number of as and bs leaves the allowed range.

16. (a) We allow for one event to start reading, to start writing, to finish reading and to finish writing for each party. This leads us to the event set (or alphabet) Σ consisting of:
    - sr_1, fr_1: party 1 starts / finishes reading;
    - sr_2, fr_2: party 2 starts / finishes reading;
    - sw_1, fw_1: party 1 starts / finishes writing;
    - sw_2, fw_2: party 2 starts / finishes writing.

    We thus have $\Sigma = \{sr_1, fr_1, sw_1, fw_1, sr_2, fr_2, sw_2, fw_2\}$.

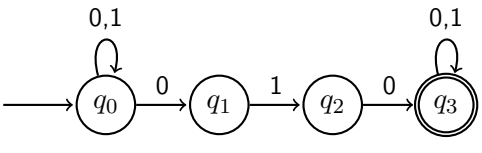
    (b) We consider an automaton that has the following states:
    - N: nobody is currently reading or writing;
    - R_1, R_2, R_{12}: party 1 / party 2 / both of parties 1 and 2 are currently reading (but no writing is taking place);
    - W_1, W_2: party 1 / party 2 is currently writing (but no reading is taking place);
    - E: an error has occurred (E is the "fail state").

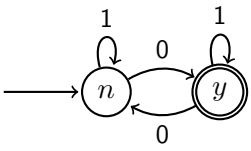
When visualising the state transition diagram of the automaton, we use the following convention for readability: if there is no outgoing edge labelled with $a \in \Sigma$ from some state q, then we implicitly add a transition from this state with label a to E. In other words: our picture just shows the legal moves; all the other moves automatically lead to the error state E. (Drawing all the transitions to E would overload the picture.)



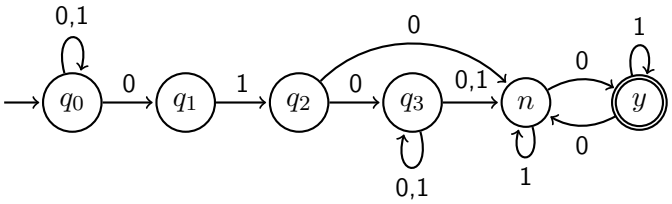
(c) We pick the legal transaction $l = sr_1, sr_2, fr_2, fr_1, sw_1, fw_1$, and running the automaton on this sequence yields the sequence $N, R_1, R_{12}, R_1, N, W_1, N$ of states ending in the accepting state N. For the illegal sequence $i = sw_1, sw_2, fw_2, fw_1, sr_1, sr_2$ we obtain the sequence N, W_1, E, E, E, E, E of states (note the implicit convention: as there is no outgoing edge from state W_1 with label sw_2, we end up in state E).

17. (a) Automata accepting L_1 and L_2 respectively are:

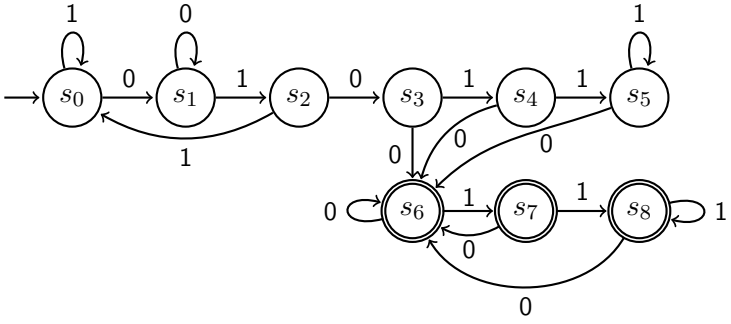




(b) An automaton accepting $L_1 \cdot L_2$ is:



(c) A deterministic automaton accepting $L_1 \cdot L_2$ is:

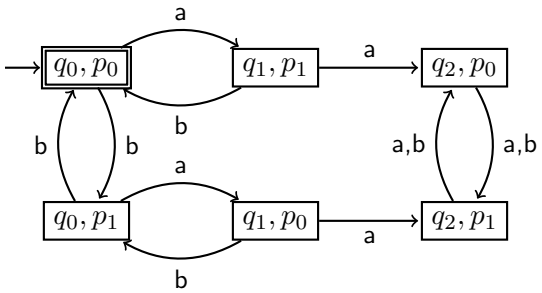


Where, the states correspond to:

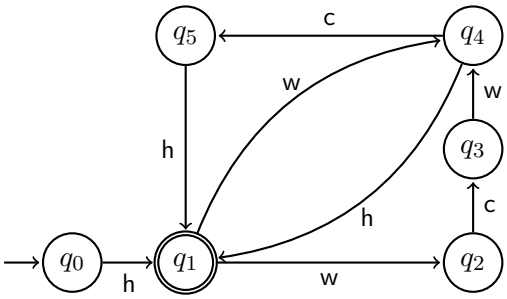
s_0	$\{0\}$	s_1	$\{0,1\}$	s_2	$\{0,2\}$
s_3	$\{0,1,3,n\}$	s_4	$\{0,2,3,n\}$	s_5	$\{0,3,n\}$
s_6	$\{0,1,3,n,y\}$	s_7	$\{0,2,3,n,y\}$	s_8	$\{0,3,n,y\}$

19. (a) Both these automata are deterministic: each state has two outgoing edges, one of which is labelled with a and the other with b.

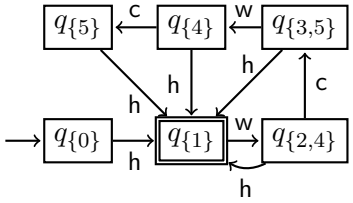
(b) We obtain the following automaton that accepts the strings which are accepted by both automata:



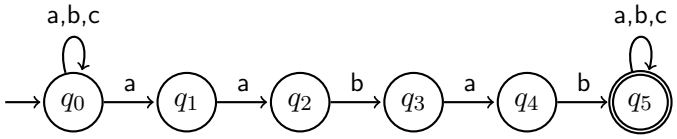
20. (a) Here is one possible model:



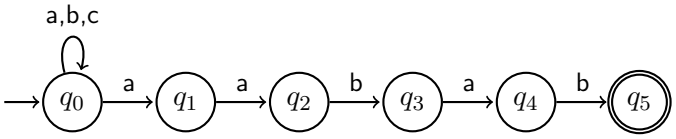
(b) Determinizing the above automaton we get the following automaton. All transitions to the rejecting sink state $q_\emptyset$ are not drawn.



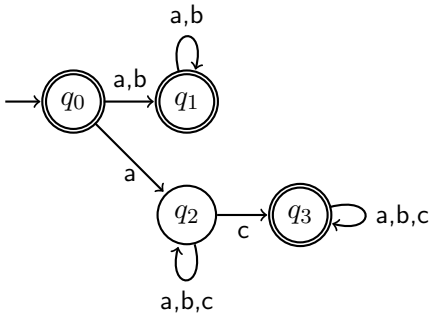
24. (a) The autoamton for $L = \{w \mid w = u \cdot aabab \cdot u'\}$ is:



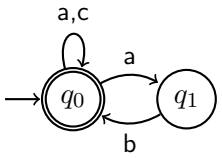
(b) The automaton for $L = \{w \mid w = u \cdot aabab\}$ is:



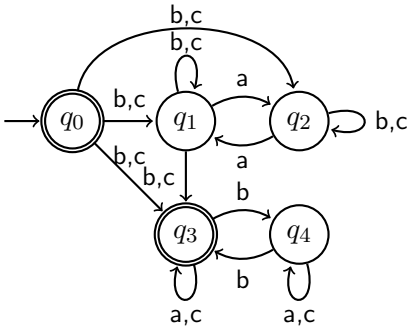
(c) The automaton for $L = \{w \mid \text{if } w \text{ contains } c \text{ then it starts with } a\}$ is:



(d) The automaton for $L = \{w \mid b \text{ appears only directly after } a\}$ is:



(e) The automaton for $L = \{w \mid w = uv \text{ such that } \#_a(u) \text{ is even and } \#_b(v) \text{ is even}\}$ is:



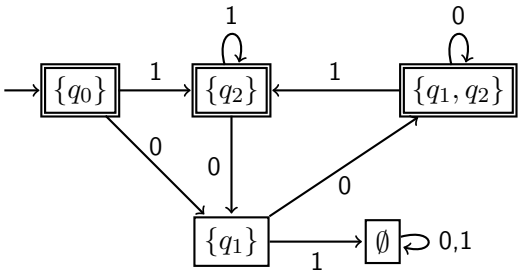
25. (a) The automaton can be formally described as the quintuple $A = (Q, \Sigma, \delta, q_0, F)$ where

- $Q = \{q_0, q_1\}$;
- $\Sigma = \{0, 1\}$;
- $\delta : Q \times \Sigma_\epsilon \to 2^Q$ is given below;
- $F = \{q_1\}$.

Here δ is defined as follows:

$$\begin{aligned}
\delta(q_0, 0) &= \{q_1\};\\
\delta(q_0, 1) &= \{q_2\};\\
\delta(q_1, 0) &= \{q_1, q_2\};\\
\delta(q_1, 1) &= \emptyset;\\
\delta(q_2, 0) &= \emptyset;\\
\delta(q_2, 1) &= \{q_2\}.
\end{aligned}$$

(b) Using the powerset construction, we obtain the following automaton:

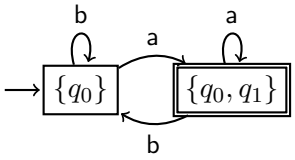


(c) The automaton A accepts all strings made of an arbitrary number of 0's followed by an arbitrary number of 1's. More formally

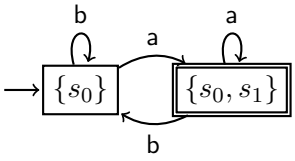
$$L(A) = \{0^k 1^\ell : k, \ell \geq 0\}.$$

26. The language of both automata is $(a|b)^*a$.

The powerset construction when applied to the first automaton produces the following deterministic automaton.



The powerset construction when applied to the second automaton produces the following deterministic automaton.

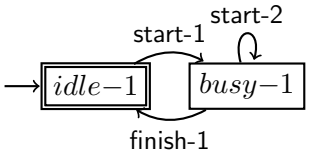


This is not surprising as both have the same language.

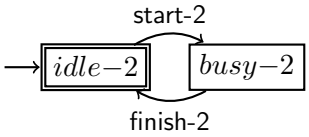
27. (a)

start-1	P_1 starts processing a job;	start-2	P_2 starts processing a job;
finish-1	P_1 finishes processing a job;	finish-2	P_2 finishes processing a job;
dispatch-1	Q sends a job to P_1;	dispatch-2	Q sends a job to P_2;
in	a job arrives in Q.		

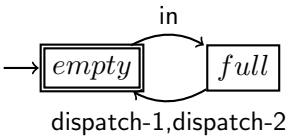
(b) *Processor 1*



Processor 2

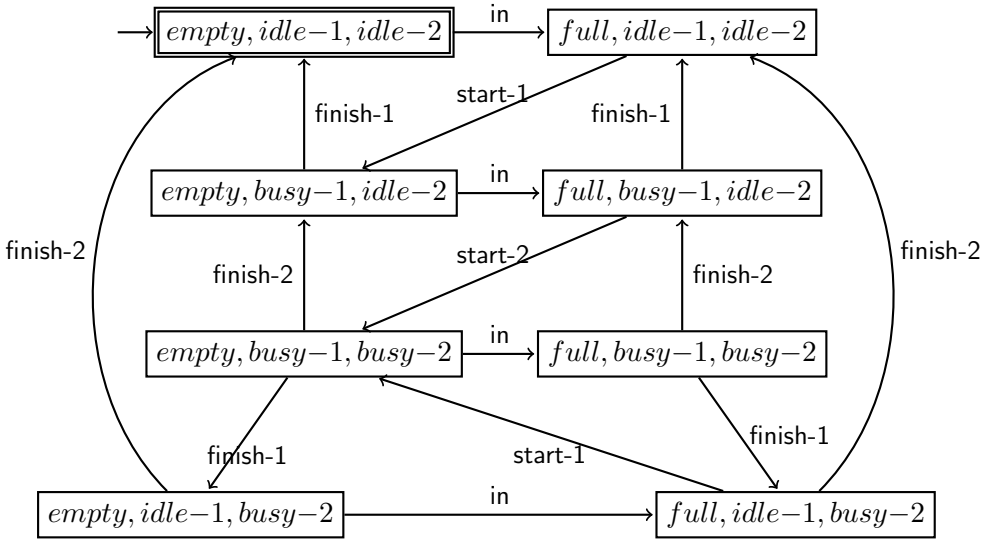


Queue



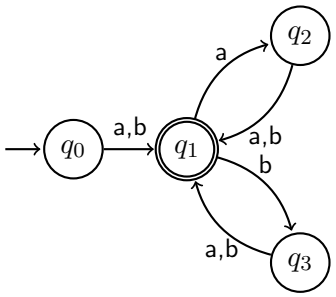
(c)
- dispatch-1 needs to happen at the same time as start-1 (when Q sends a job to P_1, P_1 starts processing it);
- dispatch-2 needs to happen at the same time as start-2 (when Q sends a job to P_2, P_2 starts processing it).
- notice that start-2 is part of the alphabet of P_1. Thus, start-2 is synchronized between all three automata.

(d) The combined system (where dispatch-1 has been identified with start-1 and dispatch-2 has been identified with start-2).



31. (a) $(aa|ab|ba|bb)^*(a|b)$;
    (b) $(a|b)^*aba(a|b)^*$;
    (c) $(a|b)^*a$;
    (d) $(b|ab)^*(a|\epsilon)$.

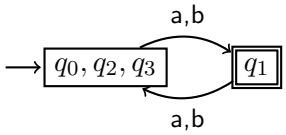
32. (a) **Step 1.** We first remove the unreachable state q_4. This leaves us with the following automaton:



Step 2. We produce a table of all pairs of states and mark those pairs (p, q) where $p \in F$ and $q \notin F$, or vice-versa.

	q_0	q_1	q_2	q_3
q_0	–	✓		
q_1		–	✓	✓
q_2			–	
q_3				–

Step 3. We consider all pairs (p, q) that are *not* marked and mark them if $p \xrightarrow{x} p'$ and $q \xrightarrow{x} q'$ for a *marked* pair (p', q') of states and an input x. This does not add any marks to the table. We conclude that all the pairs that are *not* marked are equivalent. Hence q_0 is equivalent to both q_2 and q_3; so the states q_0, q_2 and q_3 are all equivalent, and can be collapsed into a single state. This gives the following automaton:



(b) An accepted word must start with a or b (the empty word ϵ is not accepted); we are now in state q_1. We must now read an even number of characters to return to q_1; we see that the language accepted by this automaton is the set of all words of odd length.

33. (a) Every 1 is followed by a 0. The word 10 is in the language. The word 1 is not.

(b) The number of 0s in the word is divisible by 3. The word ϵ is in the language. The word 00 is not.

(c) A word in the language contains 010 as a subword. The word 010 is in the language. The word 011 is not.

(d) A word is in the language if it ends in 0 or is the empty word. The word 11 is not in the language. The word 0 is in the language.

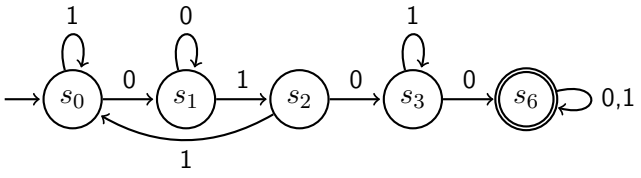
(e) A word is in the language if every odd position in the word is a 0. The word 00 is in the language, the word ϵ, is in the language, but the word 01 is not.

(f) A word is in the language if it ends in 0. The word 11 is not in the language. The words 0 and ϵ are not in the language.

36. The following table shows for each pair of inequivalent states a word that is accepted from one and not from the other.

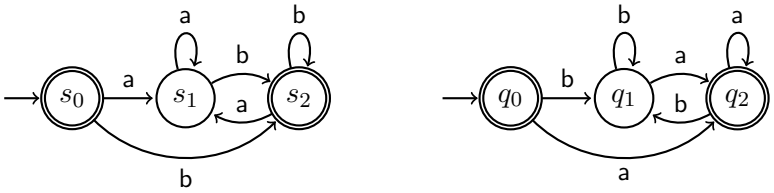
	s_0	s_1	s_2	s_3	s_4	s_5	s_6	s_7
s_1	100	–						
s_2	00	00	–					
s_3	0	0	0	–				
s_4	0	0	0		–			
s_5	0	0	0			–		
s_6	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	–	
s_7	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ		–
s_8	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ		

It follows that states s_3, s_4, and s_5 are equivalent and so are s_6, s_7, and s_8. The minimal equivalent deterministic automaton is:

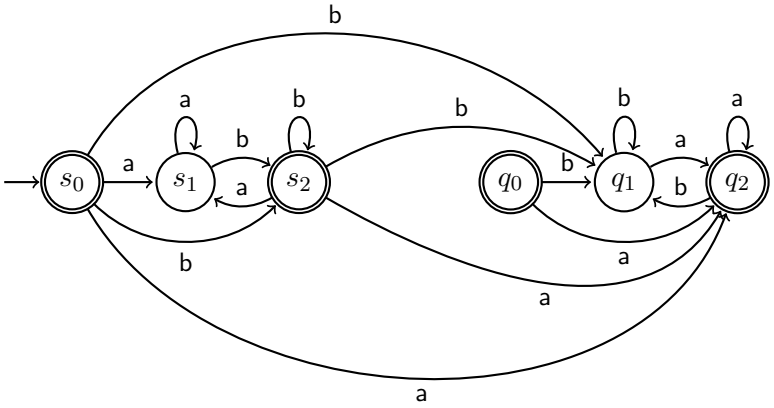


40. (a) Automata for a^*b and b^*a are:

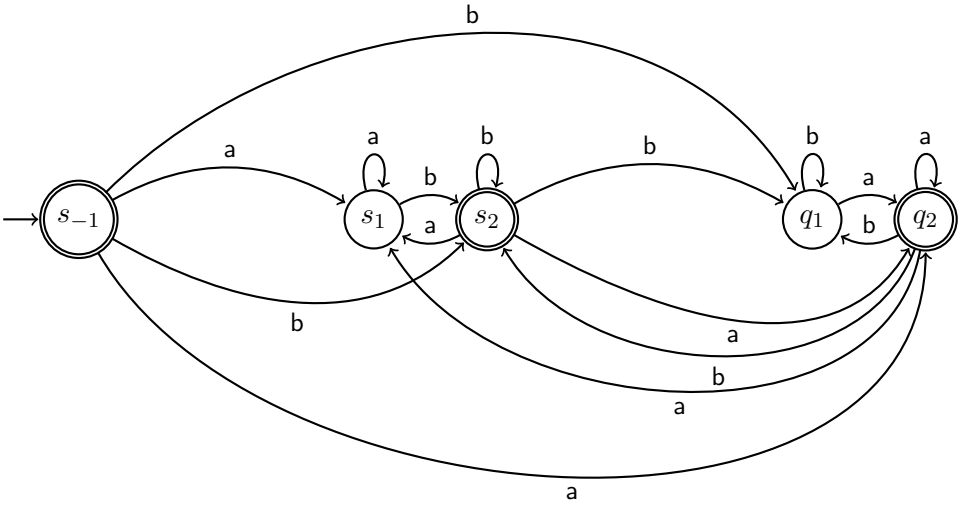
The automata for $(a^*b)^*$ and $(b^*a)^*$ are:



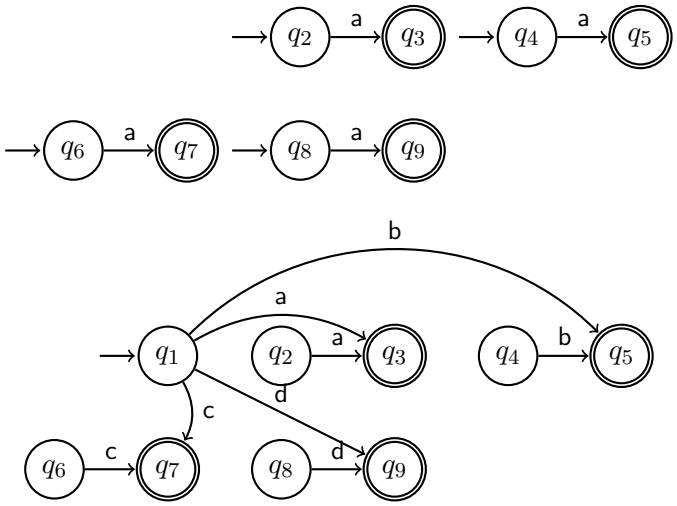
Connecting them to an automaton for $(a^*b)^*(b^*a)^*$ we get:

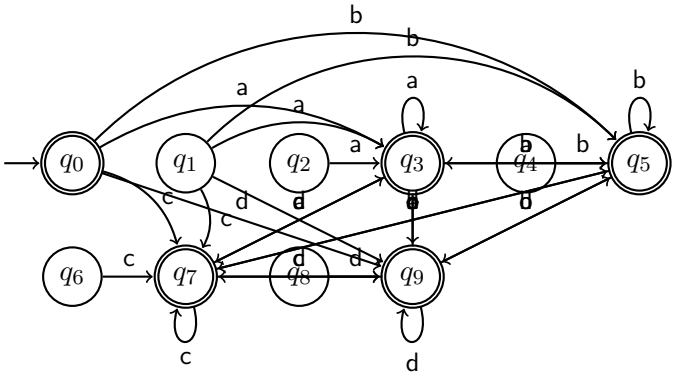


Finally, getting an automaton for $((a^*b)^*(b^*a)^*)^*$ (and removing unreachable states):

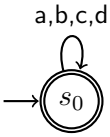


(b) The automaton for $(a|b|c|d)^*$ is obtained by the following sequence of automata and their combinations. But it gets less and less clear as the process progresses.

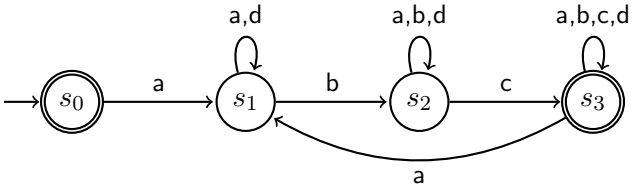




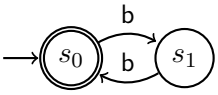
But the good news is that this smaller automaton does the same:



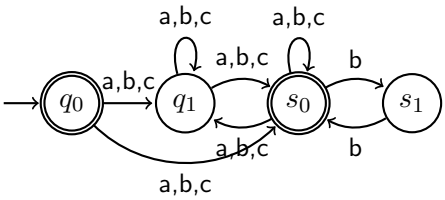
So putting all the parts together we get:



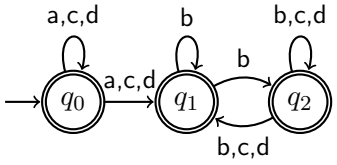
(c) The automaton for $(a|c|d)^*$ is similar to the automaton above. The automaton for $(bb)^*$ is:



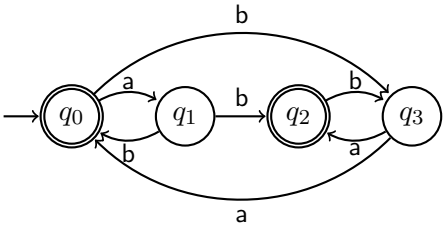
Putting the two together we get:



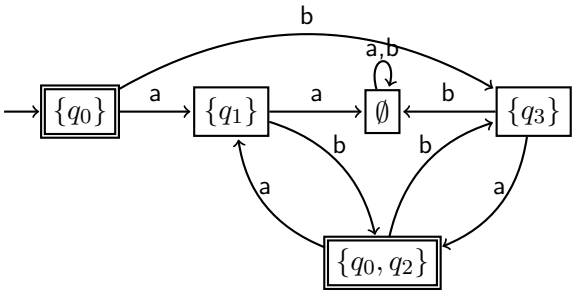
(d) This is quite similar to the automata consturcted earlier. So the final automaton is:



44. (a) The automaton for $((ab)^*(ba)^*)^*$ is:



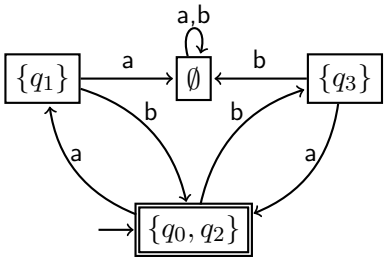
It's deterministic equivalent is:



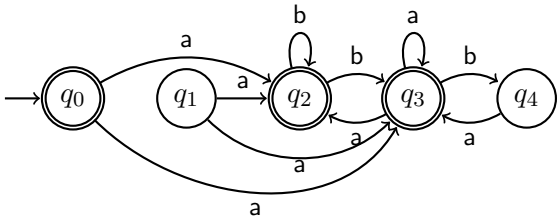
In order to minimize it consider the following table:

	0	1	0, 2
3	ϵ	b	ϵ
0, 2		ϵ	$-$
1	ϵ	$-$	$-$

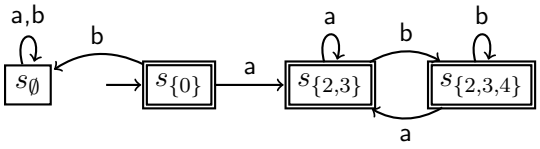
Leading to the automaton obtained from unifying $\{1\}$, $\{3\}$, $\{0, 2\}$ and $\emptyset$:



(b) The automaton for $((ab^*)(ba)^*)^*$ is



It's deterministic equivalent is:



In order to minimize it consider the following table:

	$s_{\{0\}}$	$s_{\{2,3\}}$	$s_\emptyset$	
$s_{\{2,3,4\}}$	b		ϵ	
$s_\emptyset$	ϵ	ϵ	$-$	
$s_{\{2,3\}}$	b	$-$	$-$	

Leading to the automaton obtained from unifying $s_{\{2,3\}}$ and $s_{\{2,3,4\}}$: