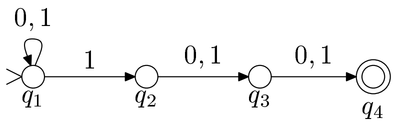
T-79.148 **Autumn 2003**
Introduction to Theoretical Computer Science
Tutorial 4
Solutions for demonstration exercises

4. **Problem**: Construct a nondeterministic finite automaton that tests whether in a given binary input sequence the third-to-last bit is a 1. Make the automaton deterministic using the subset construction.

   Answer:: The language $L = \{w \in \{0,1\}^* \mid \text{the third-to-last-bit of } w \text{ is } 1\ \}$ is recognized by the following nondeterministic automatomaton $M = (Q, \Sigma, \delta, q_1, F)$, where

$$\begin{aligned} Q &= \{q_1, q_2, q_3, q_4\} \\ \Sigma &= \{a, b\} \\ F &= \{q_4\}, \end{aligned}$$

   and the transition function δ is defined as follows:

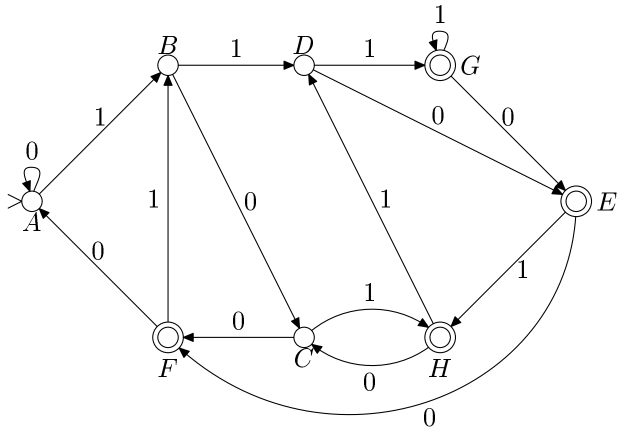


   We now construct a deterministic automaton M' that recognizes the same language as M. We take as the states of M' all subsets of Q ($Q' = \mathcal{P}(Q)$). These sets of states encode all possible computations of M. For example, when M has read the input 010, it may be either in state q_1 or q_3. So, after M' has read the same input, it has to be in the state $\{q_1, q_3\}$.

   We construct the transtition function δ':

q	0	1	new name
$\{q_1\}$	$\{q_1\}$	$\{q_1, q_2\}$	A
$\{q_1, q_2\}$	$\{q_1, q_3\}$	$\{q_1, q_2, q_3\}$	B
$\{q_1, q_3\}$	$\{q_1, q_4\}$	$\{q_1, q_2, q_4\}$	C
$\{q_1, q_2, q_3\}$	$\{q_1, q_3, q_4\}$	$\{q_1, q_2, q_3, q_4\}$	D
$\{q_1, q_3, q_4\}$	$\{q_1, q_4\}$	$\{q_1, q_2, q_4\}$	$E\times$
$\{q_1, q_4\}$	$\{q_1\}$	$\{q_1, q_2\}$	$F\times$
$\{q_1, q_2, q_3, q_4\}$	$\{q_1, q_3, q_4\}$	$\{q_1, q_2, q_3, q_4\}$	$G\times$
$\{q_1, q_2, q_4\}$	$\{q_1, q_3\}$	$\{q_1, q_2, q_3\}$	$H\times$

   All states that contain a final state of M are final states of M'. In the above table these are marked by $\times$.



5. **Problem**: Show that if languages A and B over the alphabet $\Sigma = \{a, b\}$ are recognized by some finite automata, then so are the languages $\bar{A} = \Sigma^* - A$, $A \cup B$, and $A \cap B$.

   Solution: Let $A, B \subseteq \Sigma^*$ be languages that can be recognized by finite automata. We now show that $\bar{A}$, $A \cup B$, and $A \cap B$ can also be recognized by finite automata by showing how such automata can be constructed.

$\bar{A}$: Let $M_A = (Q, \Sigma, \delta, q_0, F)$ be a deterministic automaton[1] that recognizes A ($L(M_A) = A$). We define an automaton $M_{\bar{A}}$ as follows:

$$M_{\bar{A}} = (Q, \Sigma, \delta, q_0, Q - F) \ .$$

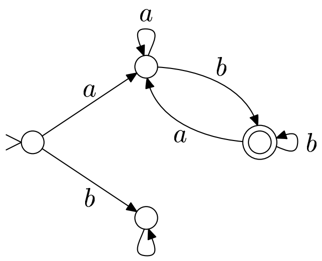
Automaton $M_{\bar{A}}$ works otherwise just as M_A, but its accecpting states are exchanged with its rejecting states. Thus, $M_{\bar{A}}$ accepts precisely those strings that M_A rejects, and rejects those that M_A accepts, so $L(M_{\bar{A}}) = \bar{A}$.

For example, consider an automaton that recognizes the language:

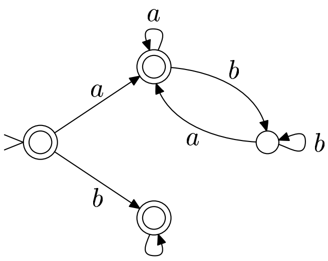
$$A = \{w \in \Sigma^* \mid w \text{ is of the form } axb, \text{ where } x \in \Sigma^*\} \ .$$

All strings that start with an a and end with a b are in L. The following two automata recognize languages A and $\bar{A}$:

M_A:



$M_{\bar{A}}$:



Note that this construction works only if M_A is deterministic. Try to find a counter example for the nondeterministic case.

$A \cup B$: Let $M_A = (Q_A, \Sigma, \delta_A, s_A, F_A)$ and $M_B = (Q_B, \Sigma, \delta_B, s_B, F_B)$ be finite automata that recognize languages A and B. We suppose that the state sets are distinct, that is, $Q_A \cap Q_B = \emptyset$. This is not a serious limitation since the states of one of the automata can be renamed if necessary.

We construct a nondeterministic finite automaton $M_{A \cup B}$ as follows:

[1] M_A necessarily exists since any nondeterministic finite automaton can be transformed into an equivalent deterministic one.

$$M_{A\cup B} = (Q, \Sigma, \delta, s, F) \ ,$$

where

$$\begin{aligned} Q &= Q_A \cup Q_B \cup \{s\} \\ F &= F_A \cup F_B \\ \delta &= \delta_A \cup \delta_B \cup \{(s, \varepsilon, s_a), (s, \varepsilon, s_b)\} \ . \end{aligned}$$

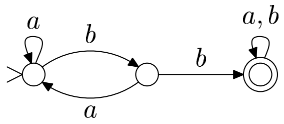
We construct $M_{A\cap B}$ by combining the automata M_A and M_B. The state s is a new initial state and from there is a nondeterministic ε-transition to initial states of M_A and M_B.

If string $x \in A$, $M_{A\cup B}$ accepts it by first making a nondeterministic transition to s_A, and then doing the same sequence of transitions that M_A would have done. Similarily, if $x \in B$, the first transition is to s_B.

For example, consider the automaton M_A that was presented above and a new automaton M_B that recognizes the language:

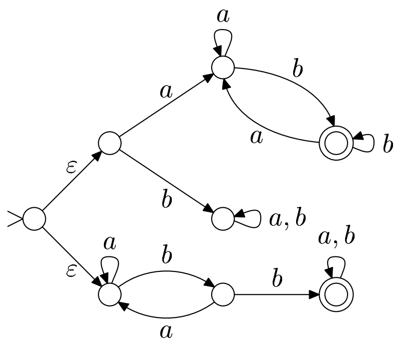
$$B = \{w \in \Sigma^* \mid w \text{ has a substring } bb\} \ .$$

M_B:



The language $A \cup B$ can be recognized by the following automaton:

$M_{A\cup B}$:



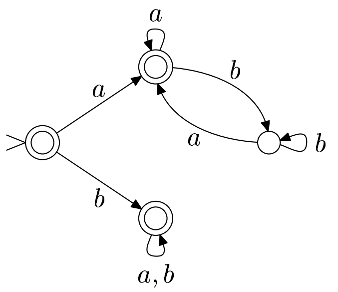
Often also a new final state f and transitions $\{(f', \varepsilon, f) \mid f' \in F_A \cup F_B\}$ are added to $M_{A\cup B}$. In this case $F = \{f\}$.

$A \cap B$: This claim is a corollary of the two previous constructions, since

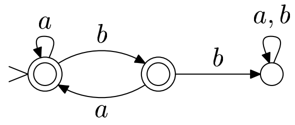
$$A \cap B = \overline{\overline{A} \cup \overline{B}} \ .$$

Let us examine again the above automata M_A and M_B construct $M_{A\cap B}$ using the above DeMorgan rule.

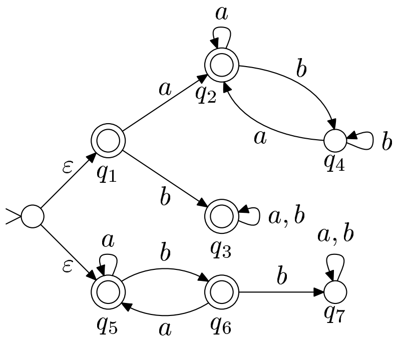
$M_{\bar{A}}$:



$M_{\bar{B}}$:

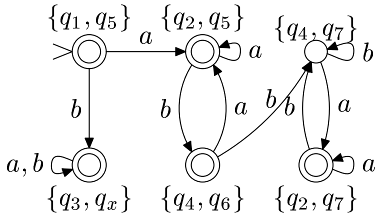


$M_{\bar{A}\cup\bar{B}}$:



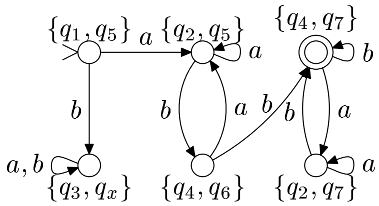
Before $M_{\bar{A}\cup\bar{B}}$ can be complemented, it has to be determinised (the following automaton is minimal, details of minimization are left in appendix):

$M'_{\bar{A}\cup\bar{B}}$:



We get the desired automaton by exchanging the accepting and rejecting states:

$M_{A\cap B}$:



We could also define the intersection of two automata directly, using a method that is analogous to the solution for the next exercise.

6. **Problem**: *(*Application)

Many methods for analyzing data transfer protocols construct the state space of the system, which can be examined to find problems, e.g., deadlocks. One way of constructing the state space of the system is to model each participant of the protocol with a finite automaton and join these two into one big state machine.

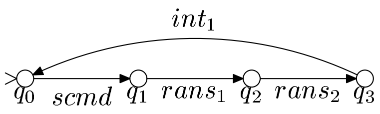
Let $M_1 = (K_1, \Sigma_1, \Delta_1, s_1, \emptyset)$ and $M_2 = (K_2, \Sigma_2, \Delta_2, s_2, \emptyset)$ be nondeterministic automata. The joint state machine $M = (K, \Sigma, \Delta, s, \emptyset)$ is constructed in the following way:

– $K = K_1 \times K_2$

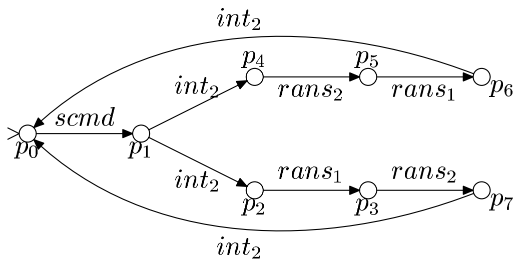
- $\Sigma = \Sigma_1 \cup \Sigma_2$
- $s = (s_1, s_2)$
- The transition $(p_1, p_2) \xrightarrow{a} (q_1, q_2)$ is in the relation Δ if any of the following conditions hold:
  1. $a \in \Sigma_1 \cap \Sigma_2$, $(p_1, a, q_1) \in \Delta_1$ and $(p_2, a, q_2) \in \Delta_2$.
  2. $a \in \Sigma_1$, $a \notin \Sigma_2$, $(p_1, a, q_1) \in \Delta_1$ and $p_2 = q_2$.
  3. $a \notin \Sigma_1$, $a \in \Sigma_2$, $(p_2, a, q_2) \in \Delta_2$ and $p_1 = q_1$.

  Let M_1 and M_2 be as below. Construct the joint state machine M and show that the system has a deadlock (i.e. from all states there is at least one transition)

M_1:



M_2:



Solution: We use as the joint state set K the cartesian product of the state sets K_1 and K_2. For example, if $K_1 = \{q_1, q_2, q_3\}$ ja $K_2 = \{p_1, p_2\}$, then the joint states are:

$$K = \{(q_1, p_1), (q_2, p_1), (q_3, p_1), (q_1, p_2), (q_2, p_2), (q_3, p_2)\} \ .$$

The idea is that the joint state holds the states of both individual automata.

The transitions of the individual automata are divided conceptually into two classes: internal and external. A transition $q \xrightarrow{a} p$ is internal if a does not occur in the alphabet of the other automaton. In this problem the only internal transition of M_1 is $q_3 \xrightarrow{int_1} q_0$. The automaton M_2 has four internal transitions that are all taken with input int_2. If a input symbol of a transition occurs in both alphabets, the transition is external.

The transition function of the joint automaton is defined such that the automata synchronize on the external transitions; an automaton can always take internal transitions, but external transitions can be made only if both automata read the same symbol at the same time.

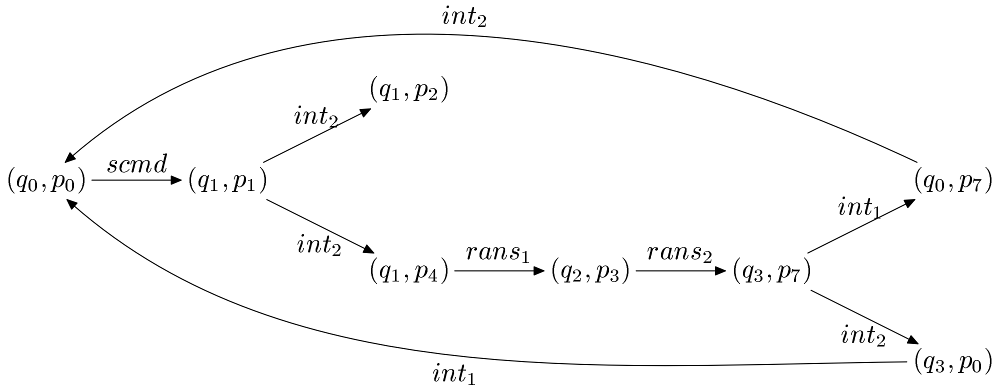
In this case the state sets of the automata are:

$$K_1 = \{q_0, q_1, q_2, q_3\}$$
$$K_2 = \{p_0, p_1, p_2, p_3, p_4, p_5, p_6, p_7\} \ ,$$

so the joint automaton has $4 \times 8 = 32$ states. However, most of these cannot be reached so they may be left out, and in the end the state space is:

$$K = \{(q_0, p_0), (q_1, p_1), (q_1, p_2), (q_1, p_4), (q_2, p_3), (q_3, p_7), (q_0, p_7), (q_3, p_0)\}$$

The transition function is defined as follows:



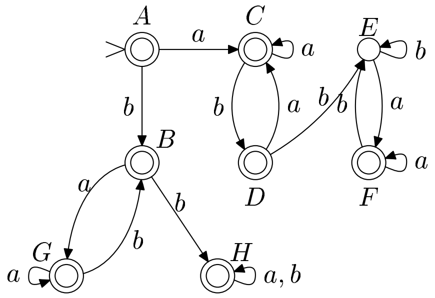
As we see, there is no transitions from the state (q_1, p_2) and we can reach it starting from the initial state using the following computation:

$$(q_0, p_0) \overset{scmd}{\to} (q_1, p_1) \overset{int_2}{\to} (q_1, p_2) \ .$$

Thus, there is a reachable deadlock in the system.

Appendix: minimizing an automaton

Using the determinising algorithm we can transform the automaton $M_{\bar{A} \cup \bar{A}}$ (of exercise 5) into the following form:



Now we want to find the minimal deterministic automaton that recognizes the same language. One algorithm is to define an equivalence relation $\overset{0}{\equiv}$ on the set of states and refine it step-by-step until we reach the desired relation $\equiv$.

In the first phase of the algorithm we remove all unreachable states. Since in this case all states are reachable, nothing has to be done.

Next, we construct the first equivalence partition such that all accepting states are in one class and all rejecting states in another:

0-equivalence:

Class	State	a	b
I	A	C (I)	B (I)
	B	G (I)	H (I)
	C	C (I)	D (I)
	D	C (I)	E (II)
	F	F (I)	E (II)
	G	G (I)	B (I)
	H	H (I)	H (I)
II	E	F (I)	E (II)

We see from the table that from the class I states D and F the b-transition leads to a state in class II, while for all other class I states the same transition leads to a class I state. So we separate the two distinct states into their own class:

1-equivalence:

Class	State	a	b
I	A	C (I)	B (I)
	B	G (I)	H (I)
	C	C (I)	D (III)
	G	G (I)	B (I)
	H	H (I)	H (I)
II	E	F (III)	E (II)
III	D	C (I)	E (II)
	F	F (III)	E (II)

This time states C and F do not fit in their classes and they have to be separated. This procedure is continued until all classes are consistent:

2-equivalence:

Class	State	a	b
I	A	C (IV)	B (I)
	B	G (I)	H (I)
	G	G (I)	B (I)
	H	H (I)	H (I)
II	E	F (V)	E (II)
III	D	C (IV)	E (II)
IV	C	C (IV)	D (III)
V	F	F (V)	E (II)

3-equivalence:

Class	State	a	b
I	A	C (IV)	B (VI)
II	E	F (V)	E (II)
III	D	C (IV)	E (II)
IV	C	C (IV)	D (III)
V	F	F (V)	E (II)
VI	B	G (VI)	H (VI)
	G	G (VI)	B (VI)
	H	H (VI)	H (VI)

All classes are now consistent so we can construct an automaton whose states are the equivalence classes. The minimized automaton is shown as a state diagram in the solution for exercise 5.

As a term, k-equivalence means that all states in a equivalence class treat all inputs that are at most k symbols long in the same way; either they all accept the input or they all reject it.