T-79.148 **Kevät 2002**
Tietojenkäsittelyteorian perusteet
Harjoitus 5
Answers for demonstration exercises

4. **Problem**: Simplify the following regular expressions (i.e., design simpler expressions describing the same languages):

   1. $(\emptyset^* \cup a)(a^*)^*(b \cup a)b^*$
   2. $(a \cup b)^* \cup \emptyset \cup (a \cup b)b^*a^*$
   3. $a(b^* \cup a^*)(a^*b^*)^*$

   Solution:

   (a)

   $$\begin{aligned}(\emptyset^* \cup a)(a^*)^*(b \cup a)b^* =&(\varepsilon \cup a)(a^*)^*(b \cup a)b^* \\ =&(\varepsilon \cup a)a^*(b \cup a)b^* \\ =&a^*(b \cup a)b^*\end{aligned}$$

   (b)

   $$(a \cup b)^* \cup \emptyset \cup (a \cup b)b^*a^* = (a \cup b)^*$$

   The result can be immediately seen from the fact that the first subexpression of the union already generates all words in Σ^*.

   (c)

   $$a(b^* \cup a^*)(a^*b^*)^* = a(a \cup b)^*$$

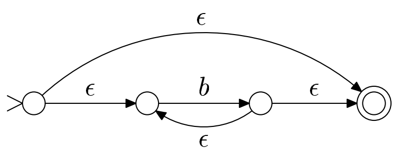
   Here we note that the subexpression $R_2 = (a^*b^*)^*$ generates all strings that can be generated by $R_1 = (b^* \cup a^*)$, so R_1 may be removed.

5. **Problem**: Determine whether the regular expressions $r_1 = b^*a(a^*b^*)^*$ and $r_2 = (a \cup b)^*a(a \cup b)^*$ describe the same language, by constructing the (minimal) finite state machines corresponding to them.

   Solution: For each regular language there exists a unique[1] minimal deterministic finite automaton. Thus, we can check the equivalence of two regular expressions by generating their corresponding minimal automata, and then checking whether they are identical.

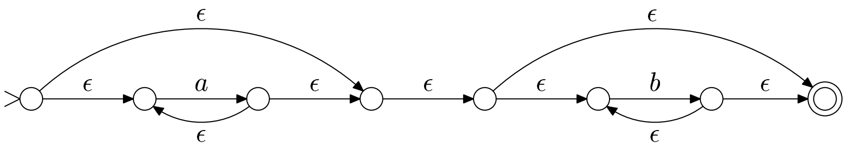
   We first construct the automaton corresponding to the regular expression r_1:

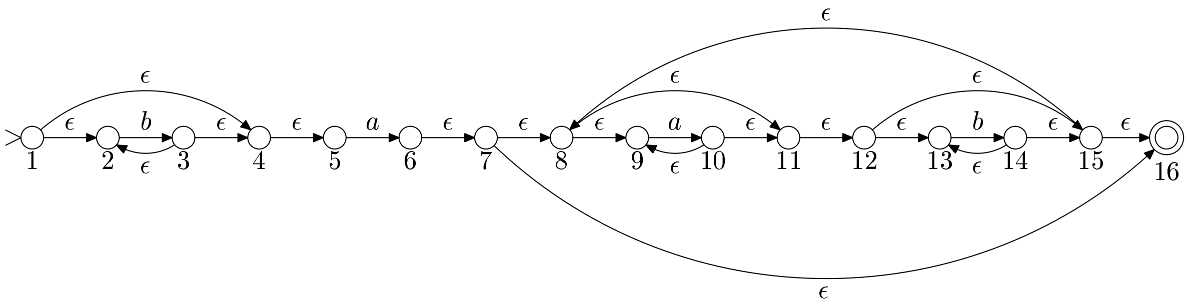
   b^*:



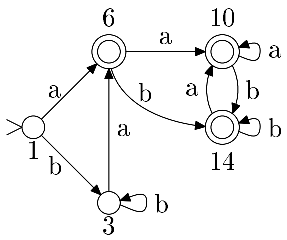
[1] Up to the naming of the states.

a^*b^*:

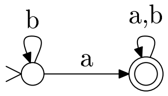




Next, we remove the ε-transitions from the automaton:

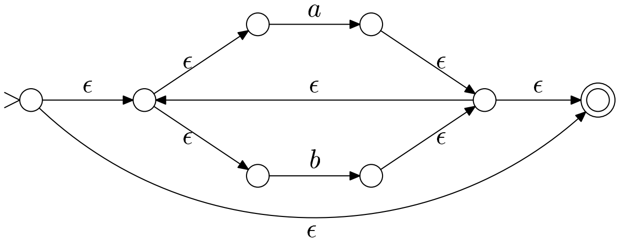


Now we notice that the automaton is already deterministic, so we may directly skip to the minimization phase. The minimization algorithm identify states 1 and 2 together, as well as states 6, 01, and 14. The resulting minimal automaton M_{r_1} is thus:

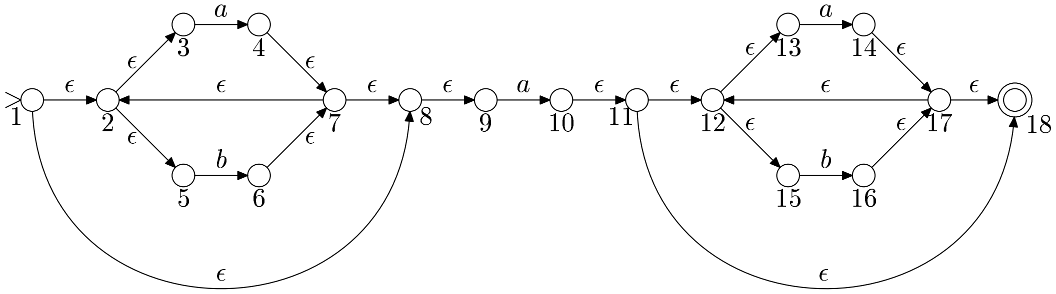


Next we do the same construction for r_2:

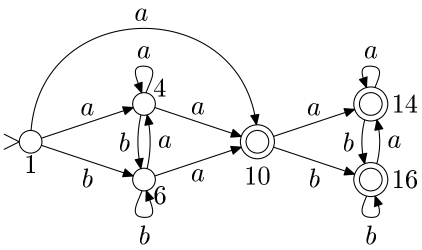
$(a \cup b)^*$



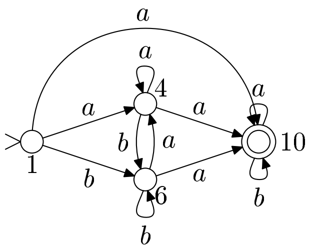
$(a \cup b)^* a (a \cup b)^*$



Remove ε-transitions:

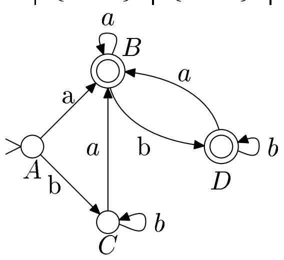


We note that this automaton may be simplified by combining all accepting states into one state:

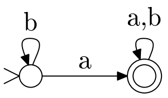


Next, we determinize this automaton:

Det. state	Nondet. states	a	b	
A	$\{1\}$	$\{4, 10\}$	$\{6\}$	
B	$\{4, 10\}$	$\{4, 10\}$	$\{6, 10\}$	$\times$
C	$\{6\}$	$\{4, 10\}$	$\{6\}$	
D	$\{6, 10\}$	$\{4, 10\}$	$\{6, 10\}$	$\times$



When we minimize this automaton, the result is:



Because both r_1 and r_2 lead to the same minimal automaton, the languages $L(r_1)$ and $L(r_2)$ are equivalent.

6. **Problem**: Prove that if L is a regular language, then so is $L' = \{xy \mid x \in L, y \notin L\}$.

   Solution: The easiest way to prove that a language is regular is to use the closure properties of regular languages; the class of regular languages is closed under the union, concatenation, Kleene star, complementation, and the intersection.

   In the language we are given a regular language L and we define a new language by it:

$$L' = \{xy \mid x \in L \text{ ja } y \notin L\}$$

   The language L' is the concatenation of languages L and its complement $\overline{L}$ ($y \notin L \Rightarrow y \in \overline{L}$). Since regular languages are closed under complementation and concatenation, L' is regular.

   We may also construct a finite-state automaton that decides the language L'. Since L is regular, there is some deterministic automaton M that decides it. We now construct an automaton $\overline{M}$ that is otherwise similar to M except that all accepting states are made rejecting and vice verse. Now $\overline{M}$ accepts the complement of L. We combine these two machines into a new non-deterministic automaton M' by adding a non-deterministic e-transition from all accepting states of M to the initial state of $\overline{M}$. Now M' decides L' so L' is regular.