Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich
Networked Systems Group (NSG)

HS 2024

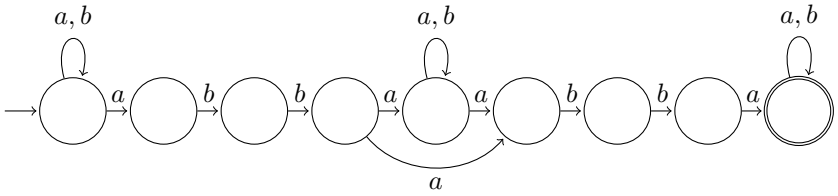
Prof. L. Vanbever / R. Schmid
based on Prof. R. Wattenhofer's material

Discrete Event Systems

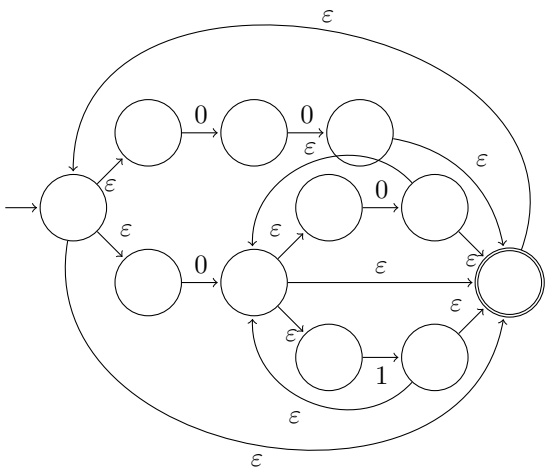
Solution to Exercise Sheet 2

1 Nondeterministic Finite Automata

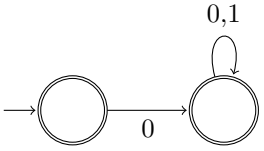
a) The following automaton also accepts strings containing $a\,b\,b\,a\,b\,b\,a$ as a substring.



b) The following automaton is obtained using the transformations presented in the lecture.

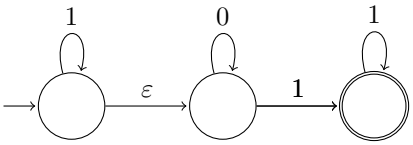


This automaton, however, is not minimal. If you take a closer look at the regular expression $(00 \cup (0(0 \cup 1)^*))^*$, you will see that it can be simplified to $(0(0 \cup 1)^*)^*$ and this in turn to $\varepsilon \cup 0(0 \cup 1)^*$. A minimal automaton for this regular expression is given by



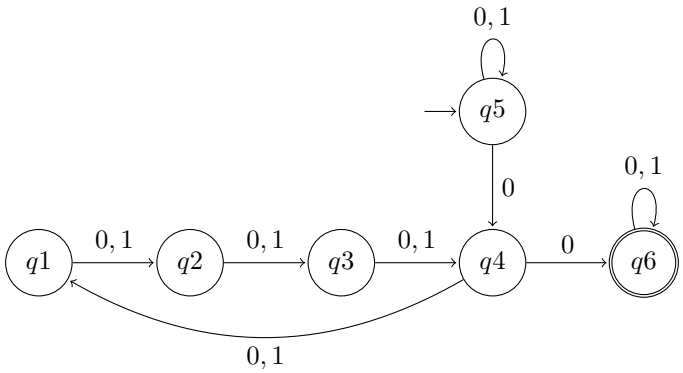
Note: The absence of a transition with label 1 in the left state means that if a 1 is read, the automaton goes into a non-accepting state and stays there for the rest of the execution. NFAs may be under-determined in this notion, but FAs must provide a transition for every symbol of the alphabet and potentially a crash state to "capture" non-accepting executions.

c) Three states are enough if we can use NFAs. With FAs, we needed at least four states.



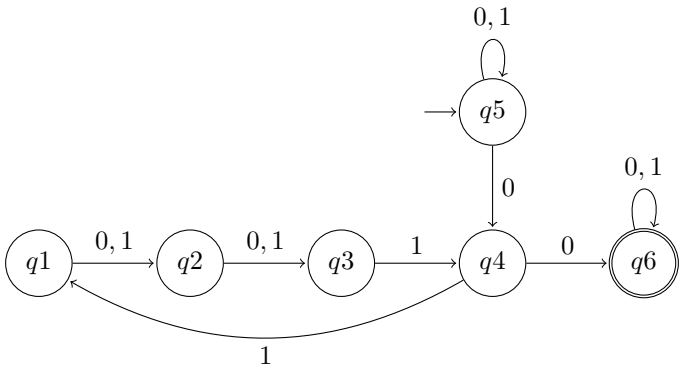
d) A deterministic machine for which every state is an accepting state, accepts *every* string of the corresponding alphabet. However, this does not hold for a nondeterministic automaton, namely if it is under-determined.

2 Exam question [2018]



Don't overthink things!

One may be tempted to optimise the NFA further, i.e. remove any information that is not necessary. However, it usually requires more work to convince ourselves that it is indeed a correct solution. For instance, here we could be tempted to modify two transitions ($q4 \to q1$ and $q3 \to q4$) from the solution above to make the automaton "slightly more deterministic":



However, it is now harder to convince yourself that this still recognises the language $L = \{w|0\{0,1\}^{4i}0 \subseteq w, i \geq 0\}$. One way to see this would be to prove the following statement:

$$\exists k > 0 : (01\{0,1\}^{4k-2}10 \cup 00) \subseteq w \implies w \in L$$

Proof. (By contraposition) Assume $w \notin L$. This means that $w \in \{w|0\{0,1\}^{4i}0 \subseteq w, i \geq 0\}^C$. Now if we make a case distinction on i and see what this means for w:

$i = 0$**:** $w \in \{w|0\{0,1\}^{4i}0 \subseteq w, i = 0\}^C$

$\Leftrightarrow w \in \{w|00 \subseteq w\}^C \Rightarrow 00 \nsubseteq w$

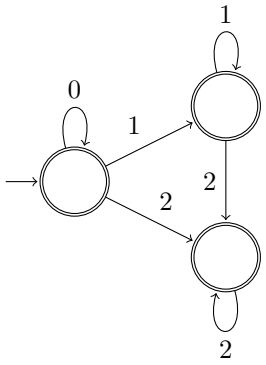
$i > 0$**:** $w \in \{w|0\{0,1\}^{4i}0 \subseteq w, i > 0\}^C$

$\Leftrightarrow w \in \{w|(00\{0,1\}^{4i-2}00 \cup 01\{0,1\}^{4i-2}00 \cup 00\{0,1\}^{4i-2}10 \cup 01\{0,1\}^{4i-2}10) \subseteq w, i > 0\}^C \Rightarrow (00 \cup 01\{0,1\}^{4i-2}10) \nsubseteq w$

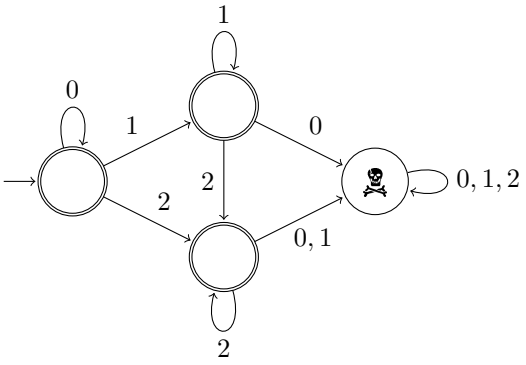
Both cases imply that $\nexists i > 0 : (01\{0,1\}^{4i-2}10 \cup 00) \subseteq w$. By contraposition this proves our initial statement. □

3 De-Randomization

a) The automaton accepts strings adhering to the regular expression $0^*1^*2^*$. Without ε-transitions we get the following automaton.

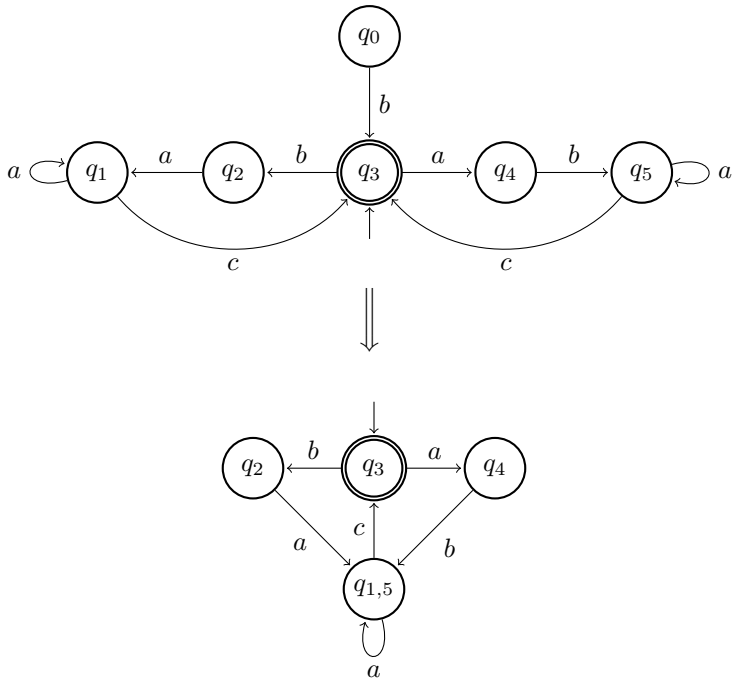


b) The deterministic automaton can be found by applying the power set construction presented in the lecture followed by the state minimization algorithm. However, it is obvious that the automaton shown below does the job.



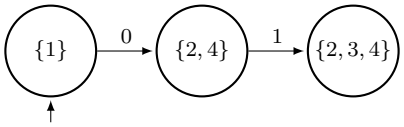
4 States Minimization

State q_0 can be omitted as it is not reachable. Moreover, states q_1 and q_5 can be merged, as there is no input sequence which will show a difference between these two states. A regular expression of the language is $((ba \cup ab)a^*c)^*$.

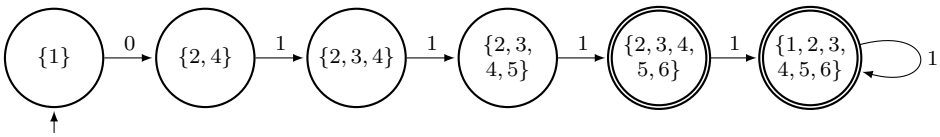


5 Derandomizing a large NFA [Exam HS14]

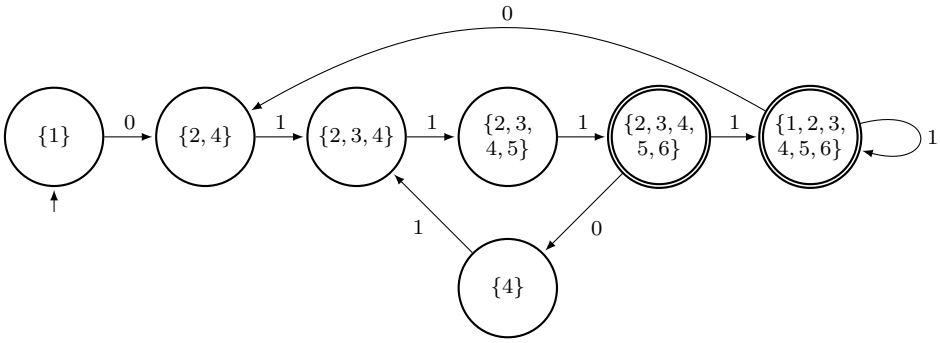
We could use the systematic transformation scheme presented in the lecture. Considering the large number of states, however, this will easily lead to an explosion of states in the derandomized automaton. Hence, we build the deterministic finite automaton in a step-wise manner, only creating those states that are actually required (i.e. reachable by any sequence from the starting state): Initially, the automaton requires a 0. Subsequently, only a 1 is accepted. Including the various transitions, this 1 can lead to three different states, namely states 2, 3, and 4.



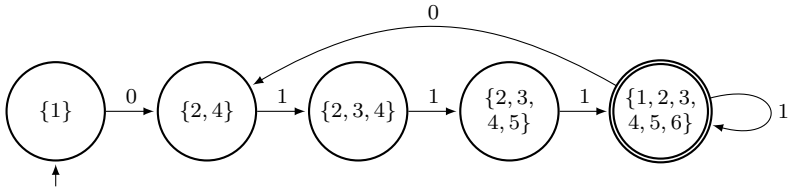
In any of the states 2, 3, and 4, only a 1 is accepted. Assume that the automaton is currently in state 2, this 1 can lead to states $\{2,3,4\}$ when including all ε-transitions. When in state 3, the 1 leads to states $\{2,3,4,5\}$ and finally, when being in state 4, the reachable states given a 1 are $\{2,3,4\}$. Hence, a 1 leads from state $\{2,3,4\}$ to state $\{2,3,4,5\}$. Repeating the same process for state $\{2,3,4,5\}$, we can see that, again, only a 1 is accepted, which leads to state $\{2,3,4,5,6\}$. Because the state 6 in the original NFA was an accepting state, $\{2,3,4,5,6\}$ is also accepting in the DFA. From state $\{2,3,4,5,6\}$, an additional 1 will lead to another accepting state $\{1,2,3,4,5,6\}$. And from this state, any subsequent 1 returns to state $\{1,2,3,4,5,6\}$ as well.



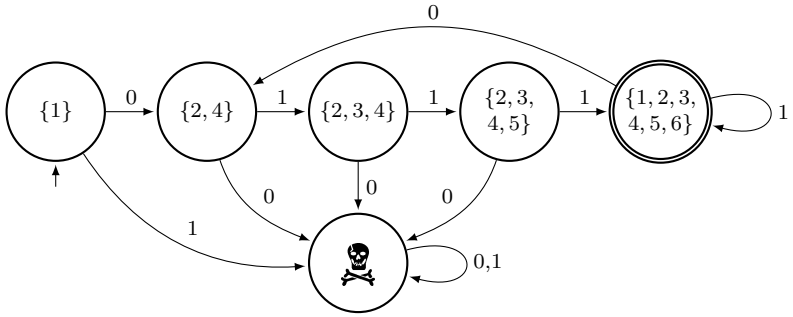
What happens if a 0 occurs in the input? This is feasible only when the deterministic state includes either state 1 or state 6. In state $\{2,3,4,5,6\}$, a 0 necessarily leads to state $\{4\}$, whereas in state $\{1,2,3,4,5,6\}$ a 0 leads to state $\{2,4\}$. In both of these states, the only acceptable input symbol is a 1 and leads to the state $\{2,3,4\}$. Hence, the deterministic finite automaton looks like this:



Observe that the states $\{4\}$ are equivalent $\{2, 4\}$ and can thus be merged. Subsequently, the states $\{2, 3, 4, 5, 6\}$, $\{1, 2, 3, 4, 5, 6\}$ can also be merged and the automaton can be reduced as follows:



This is not a DFA yet, because the crash state is still missing. The final deterministic automaton looks like this:



6 "Regular" Operations in UNIX

In UNIX, the special symbol "$" stands for the end of a line. We have:

```
egrep 'passwor(d|t)(a|e|i|o|u|A|E|I|O|U)*$' <file>
```