INF210: Modelling of Computing

Oblig 2

Åsmund Aqissiaq Arild Kløvstad

November 4, 2020

main.pdf 5.7

[coming soon]

Textbook 3.3

Exercise 3) and 4) use the procedure described by the text book in section 3.3 to mark pairs of states that cannot be collapsed, and then collapsing the remaining pairs.

3) with one step labelled "a" from s4 to s2

Step one marks $\{s_0, s_2\}, \{s_0, s_4\}, \{s_1, s_2\}, \{s_1, s_4\}, \{s_2, s_3\}, \{s_3, s_4\}$.
Step two marks $\{s_0, s_1\}$ and $\{s_0, s_3\}$. The final two pairs are not marked in step three and we collapse $\{s_1, s_3\}$ and $\{s_2, s_4\}$ resulting in the minimal automata in figure 1.

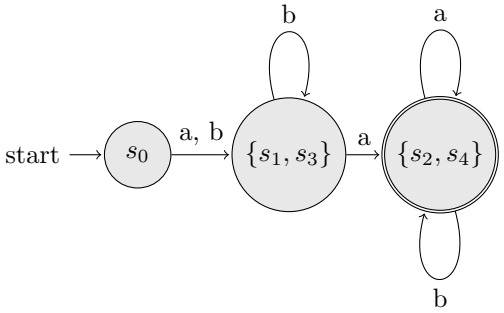

Figure 1: Minimal automata with "a"-step from s_4 to s_2

3) as given

Same as above, but this time step two also marks $\{s_2, s_4\}$ and only $\{s_1, s_3\}$ is collapsed resulting in figure 2

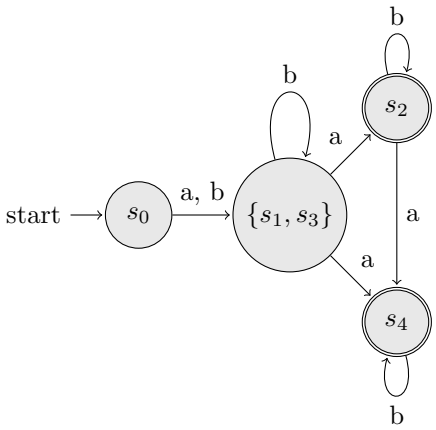

Figure 2: Minimal automata as given

4)

Step one marks $\{s_0, s_1\}, \{s_0, s_2\}, \{s_1, s_3\}, \{s_2, s_3\}$.
Step two marks $\{s_1, s_3\}$ and $\{s_0, s_3\}$ is collapsed.

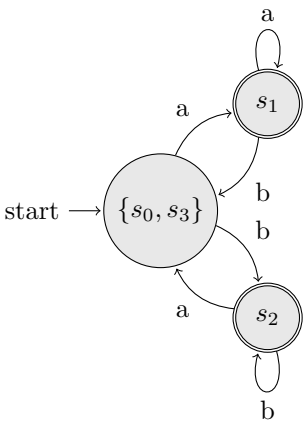

Figure 3: Minimal automata 4)

6) Find minimal automaton for $a(b \mid c)^*bb^*$

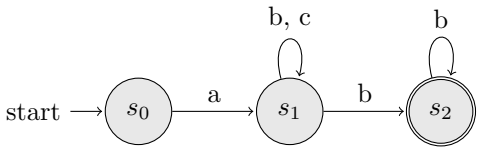

Figure 4: Minimal automaton for $a(b \mid c)^*bb^*$

Textbook 3.4

6) Determine whether the language $\{\mathbf{ww} \mid \mathbf{w} \in \Sigma^*, |\Sigma| = 2\}$ is regular.

Assume regular. Then by pumping lemma there exists an $n > 0$ such that for any word $\mathbf{xyz}$ in the language of length n or greater, $\mathbf{xy}^k\mathbf{z}$ is also in the language for all k.
We consider $\mathbf{w} \in \Sigma^*$ such that $|\mathbf{w}| = n$ and say $\mathbf{w} = \mathbf{uv}$ with distinct $\mathbf{u}, \mathbf{v}$ without loss of generality. Then $\mathbf{ww}$ is large enough for the pumping lemma and in the language by construction. Now let $\mathbf{ww} = \mathbf{xyz}$ in order to apply the pumping lemma. There are two ways this could split $\mathbf{ww}$: either $\mathbf{x} = \mathbf{w} = \mathbf{yz}$ or $\mathbf{y} = \mathbf{vu}$, splitting $\mathbf{w}$.
In the first case $\mathbf{y}^k\mathbf{z} \neq \mathbf{w}$ for $k \neq 1$ and the pumping lemma does not hold. (More precisely they are not *necessarily* equal for an arbitrary $\mathbf{w}$).
In the second case $\mathbf{xy}^k\mathbf{z} = \mathbf{uvu}^k\mathbf{v}$. For $k = 0$ we have $\mathbf{uv}$, which is certainly not in the language (assuming distinct $\mathbf{u}, \mathbf{v}$) and so the pumping lemma does not hold.
We conclude that the language is not regular.

7) Determine whether the language $\{a^{2n} \mid n \geq 1\}$ is regular.

We note that the language is described by the regular expression $(aa)^+$ and conclude it is regular.

10) Determine whether the language $\{\mathbf{ww}^R \mid \mathbf{w} \in \{a, b\}^*, |\mathbf{w}| \leq 3\}$ is regular.

The language is finite, and therefore regular. It is described by the (exhaustive) regular expression $aaaaaa \mid bbbbbb \mid abaaba \mid baaaab \mid aabbaa \mid aaaa \mid bbbb \mid abba \mid baab \mid aa \mid bb$.

Textbook 3.5

1) Prove there is an algorithm to decide if a regular language $L = \Sigma^*$

We construct a 3-step algorithm based on the observation that $\Sigma^* \setminus \Sigma^* = \emptyset$

1. Construct a finite automaton M_L accepting L
2. Swap final and non-final states of M_L to obtain an automaton for the complement of $L = \Sigma^* \setminus L$.
3. For each final state in $M_{\Sigma^* \setminus L}$, check if reachable from initial state. If yes, then $L \neq \Sigma^*$. If no final states are reachable then $L = \Sigma^*$.

6) Prove there is an algorithm for determining if there is a word in a regular language that begins with a given letter.

Assume we have a finite automaton M_L accepting L. Then for each state q_i reachable from q_0 by the given letter and each final state f_i: if f_i is reachable from q_i, return "yes". Then if we exhaust the search, return "no".

7) Prove there is an algorithm to determine if a regular language contains a word of even length.

We construct a 2-step algorithm based on the observation that any even number is a multiple of 2, and the assumption that we can find a path between two nodes in a graph. We also assume a finite automaton M_L accepting L:

1. construct a graph G such that $V(G) = Q$ and $E(G) = \{(q_i, q_j) \mid \text{there is a path of length 2 from} q_i \text{to} q_j \text{in} M_L\}$
2. for each final state f_i, look for a path from q_0 to f_i. If such a path exists return "yes", otherwise return "no".

Textbook 3.6

18) Construct a pushdown automaton accepting $L = \{\mathbf{w}c\mathbf{w}^r \mid \mathbf{w} \in \{a, b\}^*\}$.

Use PDA model with empty stack-acceptance and stack alphabet $\{\alpha, \beta\}$.

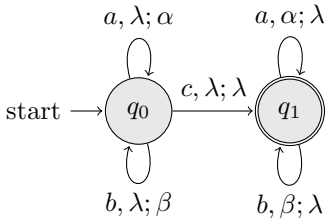

Figure 5: pushdown automaton accepting $L = \{\mathbf{w}c\mathbf{w}^r \mid \mathbf{w} \in \{a, b\}^*\}$

union, concat and kleene star go here

Textbook 4.1

14) Construct a grammar for the language $(ab)^* \mid (ac)^*$

$$\begin{array}{lcl} S & \rightarrow & abA \mid acB \\ A & \rightarrow & abA \mid \lambda \\ B & \rightarrow & acB \mid \lambda \end{array}$$

21) Construct a grammar for the language $(a|b)^*(aa|bb)(a|b)^*$

$$\begin{array}{lcl} S & \rightarrow & aA \mid bA \mid B \\ A & \rightarrow & aA \mid bA \mid B \\ B & \rightarrow & aaC \mid bbC \\ C & \rightarrow & aC \mid bC \mid \lambda \end{array}$$

24) Find an automaton which accepts the language generated by $S \rightarrow aB$, $A \rightarrow aB$, $B \rightarrow bA \mid b$

We construct an automaton with states $N \cup \{f\}$ and initial state S following the procedure:

- $A \rightarrow aB$ becomes the rule (A, a, B)
- $A \rightarrow a$ becomes the rule (A, a, f)
- $A \rightarrow \lambda$ makes A final

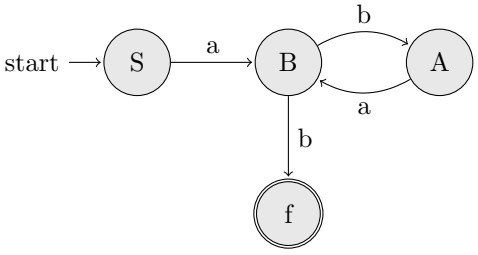

Figure 6: The finite automaton for **24)**

36) Construct a grammar that generates the language accepted by a given finite automaton.

We apply the procedure from the previous question in reverse.

- the rule (Q, a, Q') becomes $Q \to aQ'$
- if $Q' \in F$, add $Q \to a$ to our grammar rules
- if $Q \in F$, add $Q \to \lambda$ to our rules

Also note that the state S_4 is a dead end, and will not contribute to the accepted language. The resulting grammar has $N = \{S_0, S_1, S_2, S_3\}$, $\Sigma = \{a, b\}$ and production rules:

$S_0 \to aS_1 \mid bS_2$

$S_1 \to bS_1 \mid aS_3 \mid a$

$S_2 \to aS_1 \mid bS_3 \mid b$

$S_3 \to \lambda$