ALBERT-LUDWIGS-
UNIVERSITÄT FREIBURG

Prof. Dr. Andreas Podelski
Matthias Heizmann
Christian Schilling

May 20th, 2014

1. Presence Exercise Sheet for the Lecture

Computer Science Theory

WITH PROPOSALS FOR SOLUTIONS

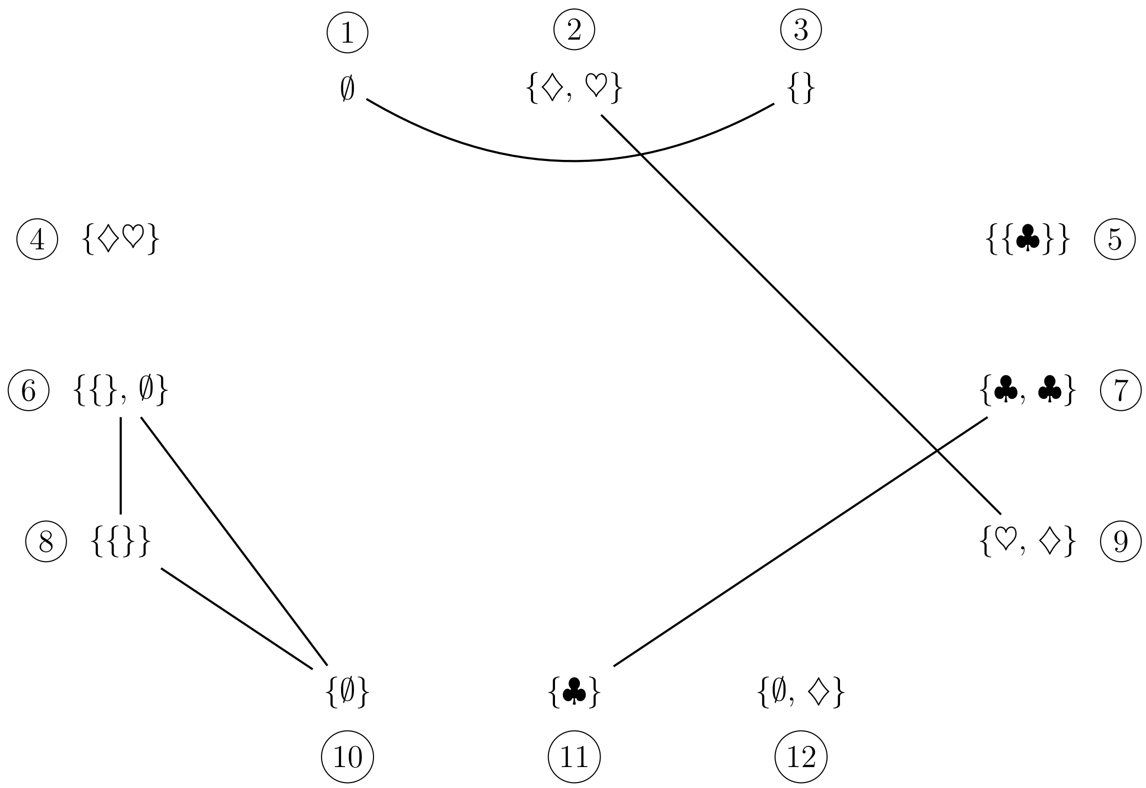
Exercise 1: Sets

(a) Two sets are equal if and only if <u>they contain the same elements</u>.

Write down all elements (without the duplicates) for the twelve sets below.

①: (none)	②: $\diamondsuit, \heartsuit$	③: (none)
④: $\diamondsuit\heartsuit$	⑤: $\{\clubsuit\}$	⑥: $\{\}$
⑦: $\clubsuit$	⑧: $\{\}$	⑨: $\diamondsuit, \heartsuit$
⑩: $\{\}$	⑪: $\clubsuit$	⑫: $\{\}, \diamondsuit$

Draw lines between those sets which are equal.



(b) Apply the following set operations and give the number (i), yes/no (ii–iii), and the resulting sets (iv–vi).

(i) $|S|$ for finite set S is defined as <u>the number of elements in S</u>.

$$|\{\}| = \underline{0}$$
$$|\{\heartsuit, \clubsuit\}| = \underline{2}$$
$$|\{\{\}, \{\diamondsuit\}\}| = \underline{2}$$
$$|\{\{\heartsuit, \clubsuit\}\}| = \underline{1}$$

(ii) $e \in S$ if and only if <u>e is contained in S</u>.

$\{\} \in \{\}$	no
$\{\} \in \{\heartsuit, \{\}\}$	yes
$\diamondsuit \in \{\heartsuit, \{\diamondsuit\}\}$	no
$\{\heartsuit\} \notin \{\heartsuit, \{\diamondsuit\}\}$	yes

(iii) $S_1 \subseteq S_2$ if and only if <u>every element in S_1 is contained in S_2</u>.

$\{\} \subseteq \{\}$	yes
$\{\} \subseteq \{\{\diamondsuit\}\}$	yes
$\{\diamondsuit, \clubsuit\} \subseteq \{\diamondsuit, \heartsuit, \{\clubsuit\}\}$	no
$\{\diamondsuit, \clubsuit\} \subseteq \{\clubsuit, \heartsuit, \diamondsuit\}$	yes

(iv) $S_1 \cup S_2$ is the set which <u>contains all elements in S_1 or in S_2</u>.

$$\{\} \cup \{\} = \underline{\{\}}$$
$$\{\} \cup \{\heartsuit\} = \underline{\{\heartsuit\}}$$
$$\{\diamondsuit, \heartsuit\} \cup \{\heartsuit, \clubsuit\} = \underline{\{\diamondsuit, \heartsuit, \clubsuit\}}$$

(v) $S_1 \cap S_2$ is the set which <u>contains all elements both in S_1 and in S_2</u>.

$$\{\} \cap \{\} = \underline{\{\}}$$
$$\{\} \cap \{\heartsuit\} = \underline{\{\}}$$
$$\{\diamondsuit, \heartsuit\} \cap \{\heartsuit, \clubsuit\} = \underline{\{\heartsuit\}}$$

(vi) $S_1 \setminus S_2$ is the set which <u>contains all elements in S_1 but not in S_2</u>.

$$\{\clubsuit\} \setminus \{\} = \underline{\{\clubsuit\}}$$
$$\{\} \setminus \{\heartsuit\} = \underline{\{\}}$$
$$\{\diamondsuit, \heartsuit\} \setminus \{\heartsuit, \clubsuit\} = \underline{\{\diamondsuit\}}$$

Exercise 2: Natural numbers as a language
Consider the alphabet $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$.

Give a definition for a language L over Σ containing exactly all natural numbers ($\mathbb{N}$) without leading zeros. This means we do not want to have 1 and 001 (but only 1).

Hint: You can define the language directly or you can apply set operations.
Ask yourself: are we interested in how many leading zeros a word has?
Do not forget 0 (zero).

.... Sketch of solution ....

$$
\begin{aligned}
&L = ((\Sigma^* \setminus \{w \mid w \in \Sigma^* \text{ and } w = 0v \text{ for some } v \in \Sigma^*\}) \setminus \{\varepsilon\}) \cup \{0\} \\
\text{or } &L = ((\Sigma^* \setminus \{w \mid w \in \Sigma^* \text{ and } \exists v \in \Sigma^* : w = 0v\}) \setminus \{\varepsilon\}) \cup \{0\} \\
\text{or } &L = \left(\Sigma^+ \setminus \{w \mid w \in \Sigma^* \text{ and } \exists v \in \Sigma^* : w = 0v\}\right) \cup \{0\} \\
\text{or } &L = \Sigma^+ \setminus \{w \mid w \in \Sigma^* \text{ and } \exists v \in \Sigma^+ : w = 0v\} \\
\text{or } &L = \{w \mid w \in \Sigma^+ \text{ and } \exists v \in \Sigma^+ : w \neq 0v\} \\
\text{or } &L = \{w \in \Sigma^+ \mid \exists v \in \Sigma^+ : w \neq 0v\}
\end{aligned}
$$

Exercise 3: Deterministic finite automata
Consider the following picture:

We start at ① and want to get to ③. We can move from ① to ②, from ② to both ① and ③, and from ③ to ②.

(a) Your task is to represent the language of all valid moves from ① to ③ as a DFA.

  (i) Let $\Sigma = \{r\}$. For each r we go one step to the right.

  (ii) Let $\Sigma = \{\ell\}$. For each ℓ we go one step to the left.

  (iii) Let $\Sigma = \{r, \ell\}$.

(b) How many words are accepted in each case?

(c) How can we modify the automaton from (iii) if

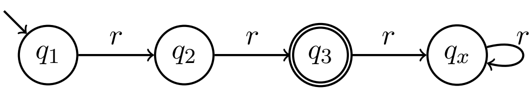
  (i) we start at ②?

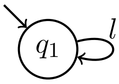
  (ii) we want to get to ② instead?

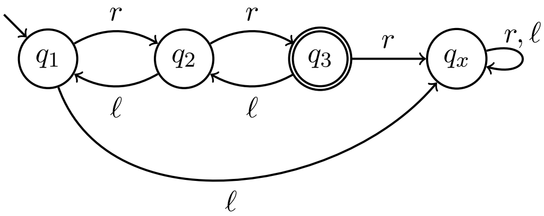
  (iii) we want to get to ② or ③?

.... Sketch of solution ....

(a) (i)



(ii) 

(iii) 

(b) (i) 1; (ii) 0; (iii) infinitely many

(c) (i) Make q_2 the initial state.

(ii) Make only q_2 final.

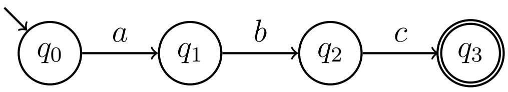
(iii) Make both q_2 and q_3 final.

Exercise sheet 3

Exercise 1: Reverse Operator

Consider $\Sigma = \{a, b, c\}$.

(a) What is the language $L(\mathcal{A})$ and its reverse language $L(\mathcal{A})^R$ for the NFA $\mathcal{A}$ below?



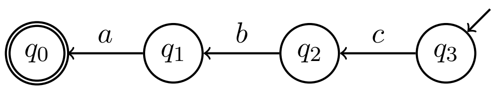
$L(\mathcal{A}) = \underline{\{abc\}}$ $\qquad$ $L(\mathcal{A})^R = \underline{\{cba\}}$

Construct an NFA that recognizes the reverse language $L(\mathcal{A})^R$.

(b) What is the problem with the construction if we have more than one final state?

.... Sketch of solution ....

(a) idea: swap initial and final state, turn around the transitions



(b) We get more than one initial state. This can be easily solved with an ε-NFA.

Exercise 2: Regular Expressions
Construct regular expressions for the following languages over the alphabet $\Sigma = \{a, b\}$.

(a) $L_1 = \{a, b, ab\}$

(b) $L_2 = \Sigma^*$

(c) $L_3 = \Sigma^+$

(d) $L_4 = \{w \in \Sigma^* \mid w \text{ starts with } a\}$

.... Sketch of solution ....

(a) $a + b + ab$

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

(c) $(a + b) \cdot (a + b)^* = (a + b)^* \cdot (a + b)$

(d) $a \cdot (a + b)^*$

Exercise 3: Pumping Lemma
The proof always works as follows:

(a) Assume the language L is regular. Then the pumping lemma must hold.

(b) Assume some $n \in \mathbb{N}$ from the pumping lemma. You must *not* make any assumptions on n.

(c) Smartly choose a word $z \in L$ (usually depending on n) with $|z| \geq n$.

(d) Assume some decomposition $z = uvw$ (with the rules given in the pumping lemma).

(e) Smartly choose some $i \in \mathbb{N}$ such that $uv^iw \notin L$ (often $i = 0$ or $i = 2$ suffices).

With this "algorithm" in mind, read the example provided in the lecture script on page 27.

In general, you may need to make a case distinction in step (d). Then for each case you have to find some i in the next step. But in this exercise it is not necessary.