

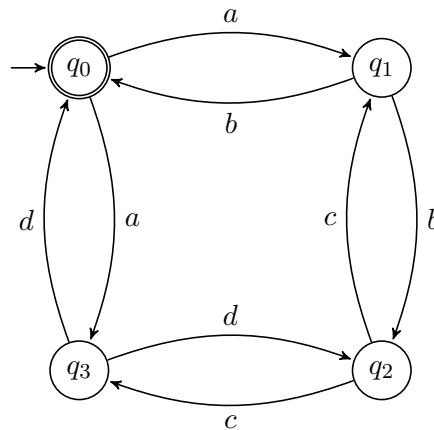
**Exercise Sheet 1**

Jun.-Prof. Roland Meyer, Reiner Hüchting

Due: Wed, Nov 5

**Exercise 1.1 NFA  $\Rightarrow$  REG**

Use equations and Arden's Lemma to find a regular expression for the following NFA:



**Exercise 1.2 Arden's Lemma**

Consider the following extension: If  $U, V \subseteq \Sigma^*$  and  $\varepsilon \in U$  then all solutions  $L \subseteq \Sigma^*$  of the equation  $L = UL \cup V$  are precisely the elements of  $\mathcal{L} = \{U^*V' \mid V \subseteq V' \subseteq \Sigma^*\}$ .

Prove the extension by solving (a) and (b) below:

- (a) Show that if  $L$  is a solution of  $L = UL \cup V$  then  $L \in \mathcal{L}$ .
- (b) Show that every  $L \in \mathcal{L}$  satisfies  $L = UL \cup V$ .

**Exercise 1.3 Language Shuffle as Intersection**

- (a) Consider NFAs  $A_1$  and  $A_2$  over alphabet  $\Sigma$ . Construct an NFA  $A_1 \cap A_2$  over  $\Sigma$  that satisfies  $L(A_1 \cap A_2) = L(A_1) \cap L(A_2)$ .
- (b) Consider NFAs  $A_1$  and  $A_2$  over disjoint alphabets  $\Sigma_1 \cap \Sigma_2 = \emptyset$ . Give NFAs  $A'_1$  and  $A'_2$  over  $\Sigma_1 \cup \Sigma_2$  so that  $L(A'_1) \cap L(A'_2) = L(A_1) \sqcup L(A_2)$ .

For (b), we define the *shuffle operation* on languages  $L_1 \subseteq \Sigma_1^*$  and  $L_2 \subseteq \Sigma_2^*$  by

$$L_1 \sqcup L_2 := \bigcup_{u \in L_1, v \in L_2} \{u_1v_1 \dots u_nv_n \mid u = u_1 \dots u_n, u_i \in \Sigma_1^*, v = v_1 \dots v_n, v_i \in \Sigma_2^*\}.$$

Note that  $L_1 \sqcup L_2$  is over alphabet  $\Sigma_1 \cup \Sigma_2$ . For example,  $\{a.b\} \sqcup \{x\} = \{a.b.x, a.x.b, x.a.b\}$ .

### Exercise 1.4 (Parallel) Programs as Automata

*Boolean Programs* are programs where all variables take Boolean values from  $\mathbb{B} := \{0, 1\}$ . For simplicity, we consider programs that only have reads  $r(x, v)$  and writes  $w(x, v)$ . Here,  $x$  is taken from a finite set of variables  $\text{Var}$  and  $v \in \mathbb{B}$ . Reads are meant to be blocking, so  $r(y, 0)$  cannot be executed if  $y$  is 1. Initially, all variables are set to 0. The following is an example program:

```
P1
10: w(x, 1) goto 11
11: r(y, 0) goto 12
11: r(y, 1) goto 11
12:                //execution stops here
```

A finite run of program  $P$  is a sequence of read and write statements, i.e., a word over the alphabet  $\Sigma := \{r, w\} \times \text{Var} \times \mathbb{B}$ . We denote the set of all finite runs of  $P$  by  $L_P$ .

- (a) Construct an NFA  $A_{P_1}$  such that  $L(A_{P_1}) = L_{P_1}$ .
- (b) Parallel programs  $P_1 \parallel P_2$  execute multiple programs over the same variables  $\text{Var}$ . Assume program  $P_1$  above is executed in parallel with

```
P2
1a: w(y, 1) goto 1b
1b: r(x, 0) goto 1c
1b: r(x, 1) goto 1b
1c:                //execution stops here
```

Change your construction into an NFA  $A_{P_1 \parallel P_2}$  with  $L(A_{P_1 \parallel P_2}) = L_{P_1 \parallel P_2}$ .

What do you observe about  $l2$  in  $P_1$  and  $1c$  in  $P_2$ . If you are interested, search the web for *Dekker's protocol*.

- (c) **This exercise is optional.** We can also directly understand the programs  $P_1$  and  $P_2$  as finite automata  $C_{P_1}$  and  $C_{P_2}$  that only reflect the control flow. This means the labels of  $P_1$  are the states of  $C_{P_1}$  and the instructions are the transitions. In this setting, we additionally have to encode the behaviour of reads from and writes to variables. To this end, construct an automaton  $A_{\text{Var}}$  so that

$$(L(C_{P_1}) \sqcup L(C_{P_2})) \cap L(A_{\text{Var}}) = L_{P_1 \parallel P_2}.$$

Can you draw a connection between reachability and language emptiness?