CS-320

Computer Language Processing

Exercise Session 1

October 2, 2017

Overview

We will recap and do exercises on the following topics:

1. Regular languages,
2. Finite state machines,
3. how to *determinize* them, and
4. how to *minimize* them.

Regular languages

Alphabet Σ is a set of symbols $\{a, b, c, \dots\}$.

A word w is a sequence of symbols $s_i \in \Sigma$.

We denote the empty word by ϵ.

A language L is a set of words.

Operations on regular languages

We define several operations on regular languages:

- Concatenation $L_1 \cdot L_2$,
- Union $L_1 \cup L_2$, and
- Kleene closure L^*.

Other operations such as $\cdot^+$, $\cdot?$ can be expressed using the above.

Finite-state automata

A deterministic finite-state automaton (DFA) is defined by a quintuple $\langle \Sigma, Q, s_0, \delta, F \rangle$ where

- Σ is a (finite) set of symbols called the alphabet,
- Q is the finite set of states,
- $s_0 \in Q$ is the initial state,
- $\delta : (Q \times \Sigma) \rightarrow Q$ is called the transition function, and
- $F \subseteq Q$ is the set of accepting states.

For nondeterministic finite-state automatons (NFAs) δ is not necessarily a function, i.e., in general we only have $\delta \subseteq Q \times \Sigma \times Q$.

A simple regular language

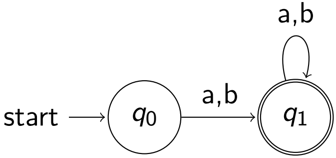
Exercise 1

- ▷ Find a finite-state automaton that accepts the language given by $(a \mid b)^+$.

A simple regular language

Exercise 1

▷ Find a finite-state automaton that accepts the language given by $(a \mid b)^{+}$.



Even binary numbers

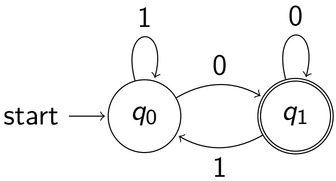
Exercise 2

- ▷ Find a finite-state automaton that accepts the even binary numbers (e.g., 0, 10, 100, 110, . . .).

Even binary numbers

Exercise 2

▷ Find a finite-state automaton that accepts the even binary numbers (e.g., 0, 10, 100, 110, . . .).



Binary numbers divisible by three

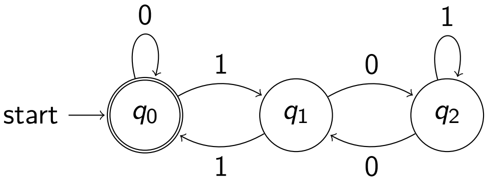
Exercise 3

- Find a finite-state automaton that accepts all binary numbers divisible by three.

Binary numbers divisible by three

Exercise 3

▷ Find a finite-state automaton that accepts all binary numbers divisible by three.



All but one

Exercise 4

▷ Find a regular expression that describes the language of all words over alphabet $\{a, b, c\}$ which contain at most two of the three symbols (e.g., a, $acac$, $ccccbbbbbb$, $\ldots$).

All but one
Exercise 4

▷ Find a regular expression that describes the language of all words over alphabet $\{a, b, c\}$ which contain at most two of the three symbols (e.g., a, $acac$, $ccccbbbbbb$, ...).

$$(a \mid b)^* \mid (a \mid c)^* \mid (b \mid c)^*$$

All but one

Exercise 4

▷ Find a regular expression that describes the language of all words over alphabet $\{a, b, c\}$ which contain at most two of the three symbols (e.g., a, $acac$, $ccccbbbbbb$, $\ldots$).

$(a \mid b)^* \mid (a \mid c)^* \mid (b \mid c)^*$

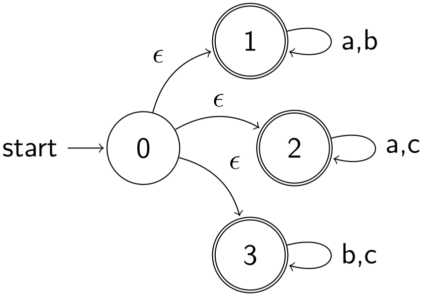
▷ Find an NFA which accepts the language.

All but one: NFA

Exercise 4

$$(a \mid b)^* \mid (a \mid c)^* \mid (b \mid c)^*$$

$$\Updownarrow$$

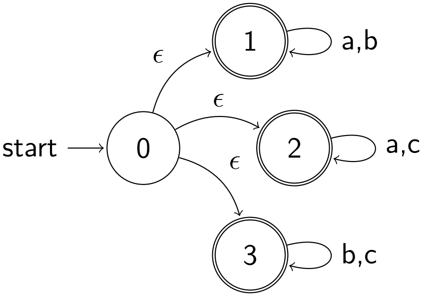


All but one: NFA

Exercise 4

$$(a \mid b)^* \mid (a \mid c)^* \mid (b \mid c)^*$$

$$\Updownarrow$$



▷ What does an equivalent DFA look like?

Recap: Determinization

For each NFA $\langle \Sigma, Q, q_0, \delta, F \rangle$ there is an equivalent DFA $\langle \Sigma, 2^Q, q_0', \delta', F' \rangle$ with

$$\begin{aligned} q_0' &= E(q_0), \\ \delta'(q', a) &= \bigcup_{\exists q_1 \in q'} E(\delta(q_1, a)), \text{ and} \\ F' &= \{q' \mid q' \in 2^Q \wedge q' \cap F \neq \emptyset\}. \end{aligned}$$

Recap: Determinization

For each NFA $\langle \Sigma, Q, q_0, \delta, F \rangle$ there is
an equivalent DFA $\langle \Sigma, 2^Q, q_0', \delta', F' \rangle$ with

$$
\begin{aligned}
q_0' &= E(q_0), \\
\delta'(q', a) &= \bigcup_{\exists q_1 \in q'} E(\delta(q_1, a)), \text{ and} \\
F' &= \{ q' \mid q' \in 2^Q \wedge q' \cap F \neq \emptyset \}.
\end{aligned}
$$

Note that for undefined transitions on symbol a in state q we get

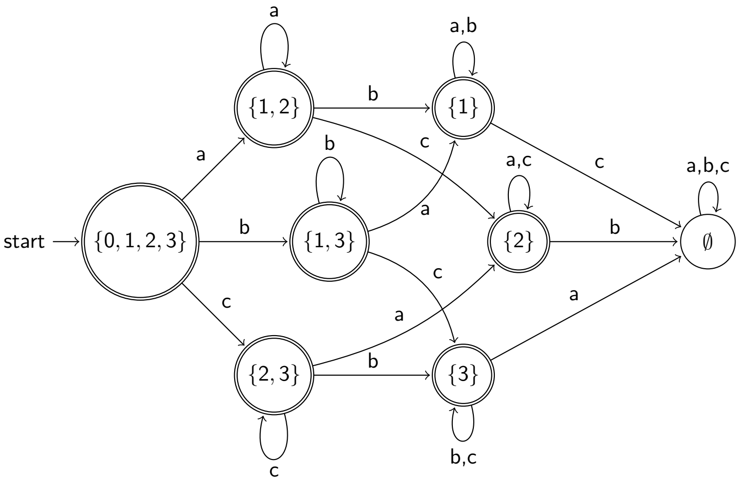
$$\delta'(\{q\}, a) = \emptyset,$$

and similarly for the trap state $\emptyset$ we get

$$\delta'(\emptyset, a) = \emptyset.$$

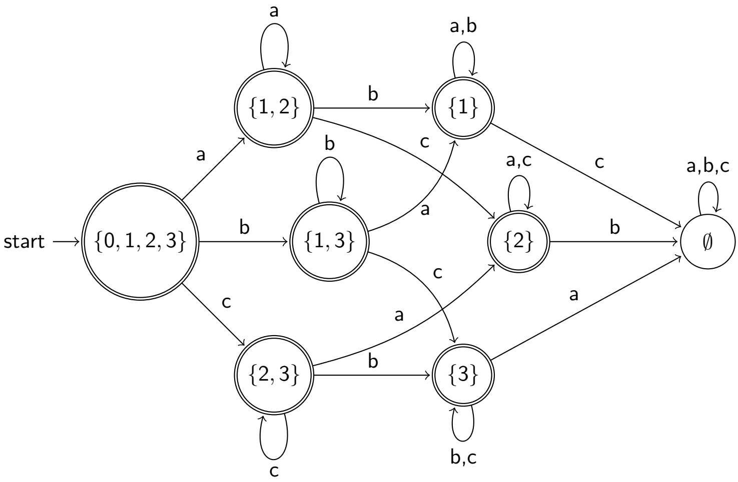
All but one: DFA

Exercise 4



All but one: DFA

Exercise 4



▷ What is the significance of the intermediate states?

Recap: Minimization

We can *minimize* DFAs by collapsing *equivalent* states.

We will consider two states s_1 and s_2 equivalent, if they are indistinguishable wrt. acceptance.

That is, s_1 is equivalent to s_2, if, for any word w, following the automaton's transitions from state s_1, respectively s_2, we end up in two accepting or two rejecting states.

Recap: Minimization

In the lecture we already touched upon one algorithm for minimizing DFAs:

We use a table to gradually mark all non-equivalent pairs of states.

1. *Initialize* the table by marking all pairs of states where one is accepting and the other is not.
2. For every symbol a and for every pair of states s_1 and s_2, mark the pair, if $\delta(s_1, a)$ is not equivalent to $\delta(s_2, a)$.
3. *Repeat* the second step until no more additional non-equivalent pairs are found.

Recap: Minimization

In the lecture we already touched upon one algorithm for minimizing DFAs:

We use a table to gradually mark all non-equivalent pairs of states.

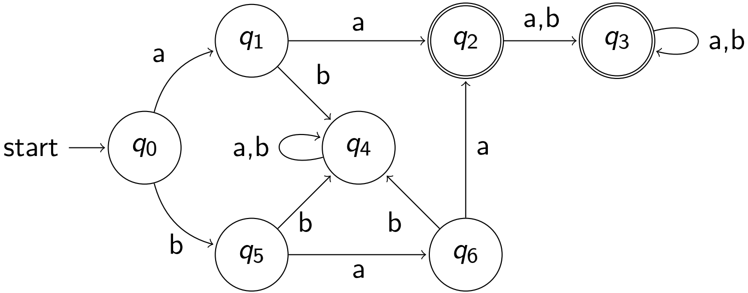
1. *Initialize* the table by marking all pairs of states where one is accepting and the other is not.
2. For every symbol a and for every pair of states s_1 and s_2, mark the pair, if $\delta(s_1, a)$ is not equivalent to $\delta(s_2, a)$.
3. *Repeat* the second step until no more additional non-equivalent pairs are found.

▷ How do you extract the minimal DFA from this table?
▷ Is the resulting DFA unique?

Minimization

Exercise 5

▷ Minimize the following DFA.



Minimization

Exercise 5

Minimized: