Introduction to Language Theory and Compilation: Exercises

Session 1: Regular languages

About these sessions

- Assistants: Jean-Sébastien
  E-mail: `Jean-Sebastien.Lerat@ulb.ac.be`
  Office: P.NO.2.O8.212
- Resources on *Université Virtuelle*:
  `webctapp.ulb.ac.bel`
- Exercises originally by Gilles Geeraerts, Sébastien Collette, Anthony Piron, Joël Goossens, etc.
- Slides based on former slides by Gilles Geeraerts, translated by Markus Lindström

Languages and operations

Let Σ[1] be a (finite) alphabet. A *language* is a set of *words* defined on a given alphabet. Let L, L_1 and L_2 be languages, we can then define some operations:

Definition (Union)

$L_1 \cup L_2 = \{w \mid w \in L_1 \text{ or } w \in L_2\}$

Definition (Concatenation)

$L_1 \cdot L_2 = \{w_1 w_2 \mid w_1 \in L_1 \text{ and } w_2 \in L_2\}$

Definition (Kleene closure)

$L^* = \{\varepsilon\} \cup \{w \mid w \in L\} \cup \{w_1 w_2 \mid w_1, w_2 \in L\} \cup \cdots$

ULB

[1] Sigma

Regular languages

Regular languages are defined inductively:

Definition (Regular language)

- $\emptyset$ is a regular language
- $\{\varepsilon\}$ is a regular language
- For all $a \in \Sigma$, $\{a\}$ is a regular language

If L, L_1, L_2 are regular languages, then:

- $L_1 \cup L_2$ is a regular language
- $L_1 \cdot L_2$ is a regular language
- L^* is a regular language

Finite automata (FA)

$M = \langle Q, \Sigma, \delta, q_0, F \rangle$ where:

- Q is a *finite* set of *states*
- Σ is the *input alphabet*
- δ is the *transition function*
- $q_0 \in Q$ is the *start state*
- $F \subseteq Q$ is the set of *accepting states*

M is a *deterministic* finite automaton (DFA) if the transition function $\delta : Q \times \Sigma \to Q$ is total. In other words, on each input, there is *one and one only* state to which the automaton can transition from its current state.

ULB

Exercise 1

Consider the alphabet $\Sigma = \{0, 1\}$. Using the inductive definition of regular languages, prove that the following languages are regular:

1. The set of words made of an arbitrary number of ones, followed by 01, followed by an arbitrary number of zeroes.
2. The set of odd binary numbers.

Exercise 1.1 – solution

- $1 \in \Sigma$ and $0 \in \Sigma$, thus $\{1\}$ and $\{0\}$ are both regular languages.
- The Kleene closure of a regular language is also a regular language, thus $\{1\}^*$ and $\{0\}^*$ are regular languages.
- The concatenation of regular languages is a regular language, thus $\{1\}^* \cdot \{0\} \cdot \{1\} \cdot \{0\}^*$ is a regular language.

Exercise 1.2 – solution

- An odd binary number always ends with a 1.
- $\{1\}$ et $\{0\}$ are regular languages.
- $\{1\} \cup \{0\}$ is regular.
- $(\{1\} \cup \{0\})^*$ is regular.
- $(\{1\} \cup \{0\})^* \cdot \{1\}$ is regular.

Exercise 2

1. Prove that any finite language is regular.
2. Is the language $L = \{0^n 1^n \mid n \in \mathbb{N}\}$ regular? Explain.

Exercise 2.1 – solution

- Let $L = \{w_1, w_2, \ldots, w_n\}$ be a finite language.
- As each word $w_i \in L$ is a finite concatenation of characters in Σ, $\{w_i\}$ is a regular language as well for all $1 \leqslant i \leqslant n$.
- Thus, $\{w_1\} \cup \{w_2\} \cup \cdots \cup \{w_n\} = L$ is regular.

Exercise 2.2 – solution

- 'fraid not! Let's prove it by contradiction.
- Let's assume L is regular.
- Thus, there exists a finite automaton $M = \langle Q, \Sigma, \delta, q_0, F \rangle$ that accepts L.
- Since Q is finite, there exists a word $w \in L$ such that the automaton is forced to pass twice through a given state q.

ULB

Exercise 2.2 – solution (ctd.)

- Let's consider the word $w = 0^{2|Q|}1^{2|Q|}$.
- There exists a path from q_0 to q labeled by 0^{k_1}, followed by a loop from q to q labeled by 0^{k_2}, followed by a path from q to $q' \in F$ labeled by $0^{k_3}1^{2|Q|}$, such that $k_1 + k_2 + k_3 = 2|Q|$.
- If that is the case, then the automaton can also accept the word $0^{k_1}0^{k_3}1^{2|Q|} \notin L$.
- Contradiction! The automaton M cannot exist, which in turn implies that L cannot be regular.

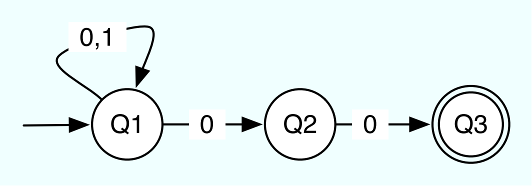
Exercise 3

For each of the following languages (defined on the alphabet $\Sigma = \{0, 1\}$), design a nondeterministic finite automaton (NFA) that accepts it.

1. The set of strings ending with 00.
2. The set of strings whose 10^{th} symbol, counted from the end of the string, is a 1.
3. The set of strings where each pair of zeroes is followed by a pair of ones.
4. The set of strings not containing 101.
5. The set of binary numbers divisible by 4.

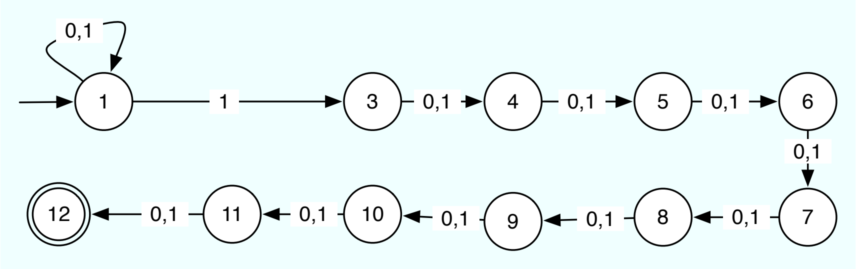
Exercise 3.1 – solution

The set of strings ending with 00.



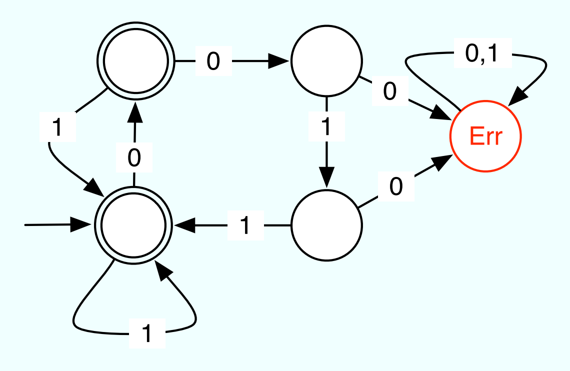
Exercise 3.2 – solution

The set of strings whose 10^{th} symbol, counted from the end of the string, is a 1.



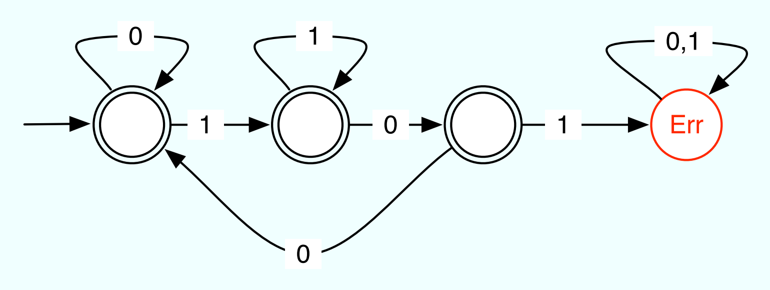
Exercise 3.3 – solution

The set of strings where each pair of zeroes is followed by a pair of ones.



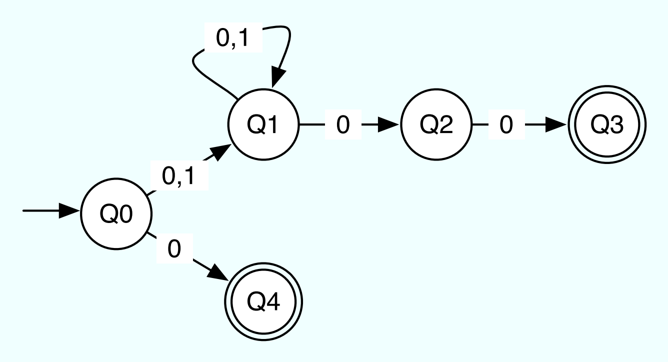
Exercise 3.4 – solution

The set of strings not containing 101.



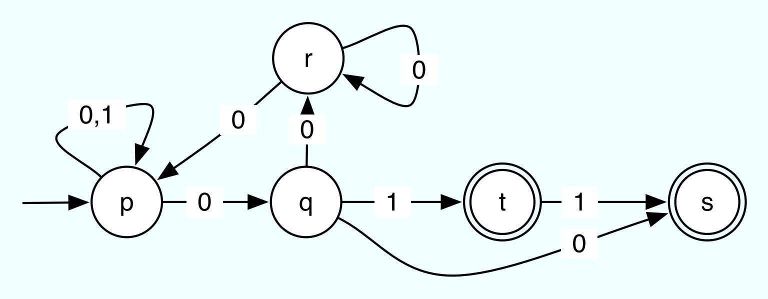
Exercise 3.5 – solution

The set of binary numbers divisible by 4.



Exercise 4.1

Transform the following NFA into a DFA:



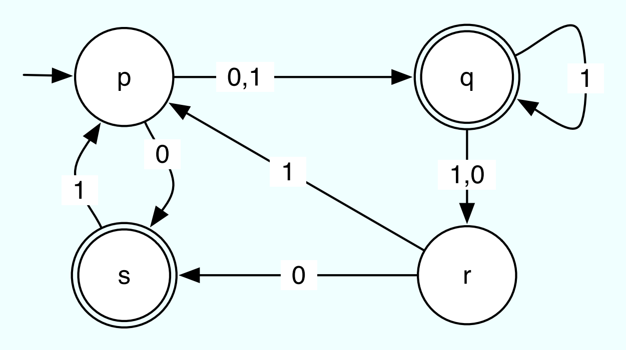
Exercise 4.1 – solution

δ	1	0
$\{p\}$	$\{p\}$	$\{p, q\}$
$\{p, q\}$	$\{p, t\}$	$\{p, q, r, s\}$
$\{p, t\}$	$\{p, s\}$	$\{p, q\}$
$\{p, q, r, s\}$	$\{p, t\}$	$\{p, q, r, s\}$
$\{p, s\}$	$\{p\}$	$\{p, q\}$

Start state: $\{p\}$.
Accepting states: $\{p, t\}$, $\{p, s\}$ and $\{p, q, r, s\}$.

Exercise 4.2

Transform the following NFA into a DFA:



Exercise 4.2 – solution

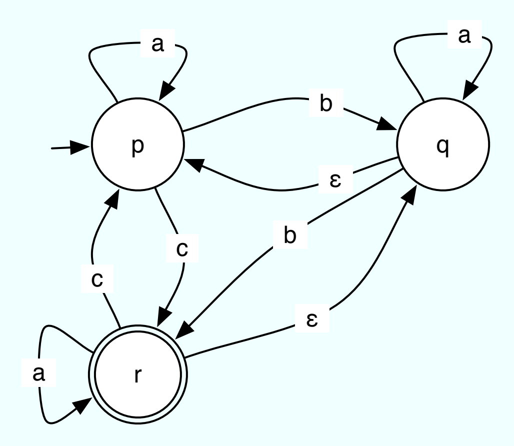
δ	1	0
$\{p\}$	$\{q\}$	$\{q, s\}$
$\{q\}$	$\{q, r\}$	$\{r\}$
$\{r\}$	$\{p\}$	$\{s\}$
$\{s\}$	$\{p\}$	$\emptyset$
$\{q, r\}$	$\{p, q, r\}$	$\{r, s\}$

δ	1	0
$\{q, s\}$	$\{p, q, r\}$	$\{r\}$
$\{r, s\}$	$\{p\}$	$\{s\}$
$\{p, q, r\}$	$\{p, q, r\}$	$\{q, r, s\}$
$\{q, r, s\}$	$\{p, q, r\}$	$\{r, s\}$
$\emptyset$	$\emptyset$	$\emptyset$

Start state: $\{p\}$.
Accepting states: $\{q\}$, $\{s\}$, $\{q, s\}$, $\{r, s\}$, $\{p, q, r\}$ et $\{q, r, s\}$.

Exercise 4.3

Transform the following ε-NFA into a DFA:



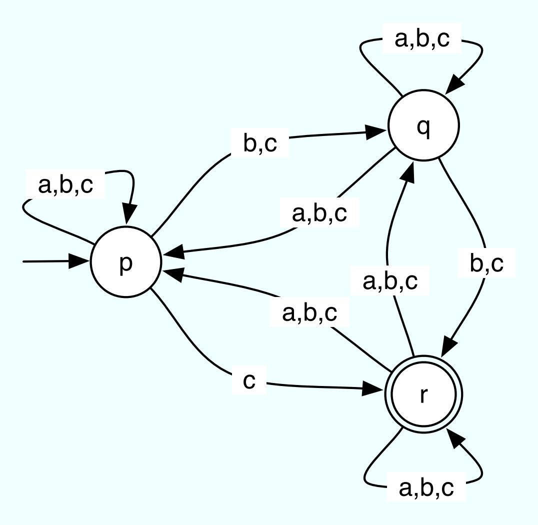
Exercise 4.3 – solution

Where can we get from each state, on each input?

	a	b	c
p	$\{p\}$	$\{p, q\}$	$\{p, q, r\}$
q	$\{p, q\}$	$\{p, q, r\}$	$\{p, q, r\}$
r	$\{p, q, r\}$	$\{p, q, r\}$	$\{p, q, r\}$

This allows us to design an NFA without ε-transitions.

Exercise 4.3 – solution (ctd.)



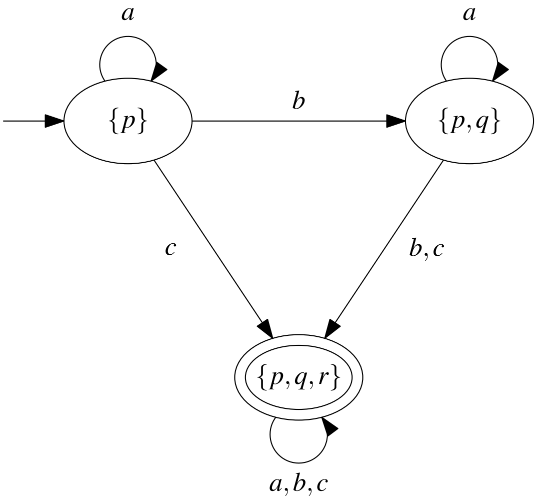
Exercise 4.3 – solution (ctd.)

Obtained deterministic transition function:

δ	a	b	c
$\{p\}$	$\{p\}$	$\{p, q\}$	$\{p, q, r\}$
$\{p, q\}$	$\{p, q\}$	$\{p, q, r\}$	$\{p, q, r\}$
$\{p, q, r\}$	$\{p, q, r\}$	$\{p, q, r\}$	$\{p, q, r\}$

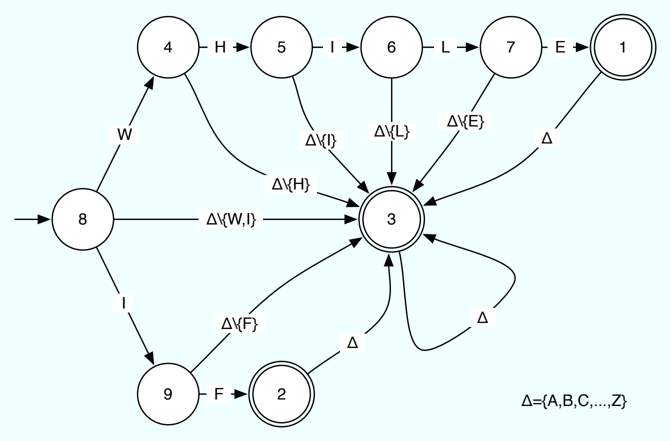
Start state: $\{p\}$.
Accepting state: $\{p, q, r\}$.

Exercise 4.3 – solution (ctd.)



Exercise 5

Write a C function that implements the following automaton and returns the accepting state number.



Exercise 5 – solution

```
char buffer; // Initialized with first input character

char next_char() {
   return buffer;
}

void read_next() {
   buffer = getchar();
}

bool is_alpha(char c) {
   return (c >= 'A' && c <= 'Z');
}
```

Exercise 5 – solution (ctd.)

```
int automaton() {
  int state = 8;
  char c;
  while (true) {
    switch state {
      case 8:
        if (next_char() == 'W') state = 4;
        else if (next_char() == 'I') state = 9;
        else if (is_alpha(c)) state = 3;
        else state = 8;
        read_next();
        break;
      case 4:
        ...
    }
  }
}
```

Exercise 5 – solution (ctd.)

```
case 2:
  if (is_alpha(c))
     state = 3;
  else
     return 2; // Do not read next character!
  read_next();
  break;
```