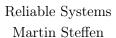
## UNIVERSITETET I OSLO Institutt for Informatikk





## INF 5110: Compiler construction

Spring 2024 Handout 2 22. 01. 2024

## Handout 2: Scanning etc.

Issued: 22. 01. 2024

The handout collects definitions in connection with scanning resp. the underlying principles and definitions. They are shown in the slides, as well, but collected in this handout for easier reference.

**Definition 1 (Alpabet**  $\Sigma$ ) An alphabet is a (finite) set of elements called "letters" or "symbols" or "characters".

**Definition 2 (Words and languages)** Given an alphabet  $\Sigma$ , a word over  $\Sigma$  is a finite sequence of letters from  $\Sigma$ . A language over alphabet  $\Sigma$  is a set of finite words over  $\Sigma$ .

**Definition 3 (Regular expressions)** A regular expression is one of the following

- 1. a basic regular expression of the form **a** (with  $a \in \Sigma$ ), or  $\epsilon$ , or  $\emptyset$
- 2. an expression of the form  $r \mid s$ , where r and s are regular expressions.
- 3. an expression of the form rs, where r and s are regular expressions.
- 4. an expression of the form  $r^*$ , where r is a regular expression.

**Definition 4 (Regular expression)** Given an alphabet  $\Sigma$ . The meaning of a regexp r (written  $\mathcal{L}(r)$ ) over  $\Sigma$  is given by equation (1).

$$\mathcal{L}(\emptyset) = \{\}$$
 empty language (1)  

$$\mathcal{L}(\epsilon) = \{\epsilon\}$$
 empty word  

$$\mathcal{L}(a) = \{a\}$$
 single "letter" from  $\Sigma$   

$$\mathcal{L}(rs) = \{w_1w_2 \mid w_1 \in \mathcal{L}(r), w_2 \in \mathcal{L}(s)\}$$
 concatenation  

$$\mathcal{L}(r \mid s) = \mathcal{L}(r) \cup \mathcal{L}(s)$$
 alternative  

$$\mathcal{L}(r^*) = \mathcal{L}(r)^*$$
 iteration

**Definition 5 (FSA)** A finite-state automaton (FSA), where  $\mathcal{A}$  over an alphabet  $\Sigma$  is a tuple  $(\Sigma, Q, I, F, \delta)$ 

- Q: finite set of states
- $I \subseteq Q$ ,  $F \subseteq Q$ : initial and final states.
- $\delta \subseteq Q \times \Sigma \times Q$ : transition relation.

Handout 2 22. 01. 2024

**Definition 6 (DFA)** A deterministic, finite automaton  $\mathcal{A}$  (DFA for short) over an alphabet  $\Sigma$  is a tuple  $(\Sigma, Q, I, F, \delta)$ 

- Q: finite set of states
- $I = \{i\} \subseteq Q, F \subseteq Q$ : initial and final states.
- $\delta: Q \times \Sigma \to Q$  transition function.

Definition 7 (Accepted words and language of an automaton) A word  $c_1c_2...c_n$  with  $c_i \in \Sigma$  is accepted by automaton A over  $\Sigma$ , if there exists states  $q_0, q_2, ..., q_n$  from Q such that

$$q_0 \xrightarrow{c_1} q_1 \xrightarrow{c_2} q_2 \xrightarrow{c_3} \dots q_{n-1} \xrightarrow{c_n} q_n$$

and where  $q_0 \in I$  and  $q_n \in F$ . The language of an FSA  $\mathcal{A}$ , written  $\mathcal{L}(\mathcal{A})$ , is the set of all words that  $\mathcal{A}$  accepts.

**Definition 8 (NFA (with**  $\epsilon$  **transitions))** A non-deterministic finite-state automaton (NFA for short)  $\mathcal{A}$  over an alphabet  $\Sigma$  is a tuple  $(\Sigma, Q, I, F, \delta)$ , where

- Q: finite set of states
- $I \subseteq Q$ ,  $F \subseteq Q$ : initial and final states.
- $\delta: Q \times \Sigma \to 2^Q$  transition function

In case, one uses the alphabet  $\Sigma + \{\epsilon\}$ , one speaks about an NFA with  $\epsilon$ -transitions.

**Definition 9 (Acceptance with**  $\epsilon$ **-transitions)** A word w over alphabet  $\Sigma$  is accepted by an NFA with  $\epsilon$ -transitions, if there exists a word w' which is accepted by the NFA with alphabet  $\Sigma + \{\epsilon\}$  according to Definition 7 and where w is w' with all occurrences of  $\epsilon$  removed.

**Definition 10** ( $\epsilon$ -closure, a-successors) Given a state q, the  $\epsilon$ -closure of q, written  $close_{\epsilon}(q)$ , is the set of states reachable via zero, one, or more  $\epsilon$ -transitions. We write  $q_a$  for the set of states, reachable from q with one a-transition. Both definitions are used analogously for sets of states.