COMP2012/G52LAC Languages and Computation Lecture 5

Regular Expressions

Henrik Nilsson

University of Nottingham

Recap: DFAs and NFAs (1)

We have so far encountered two ways of describing formal languages:

- Deterministic Finite Automata (DFA)

$$(Q, \Sigma, \delta, q_0, F)$$

- Non-deterministic Finite Automata (NFA)

$$(Q, \Sigma, \delta, S, F)$$

Recap: DFAs and NFAs (2)

Key difference: the type of the transition function:

- ***DFA:*** $\delta \in Q \times \Sigma \to Q$
- ***NFA:*** $\delta \in Q \times \Sigma \to \mathcal{P}(Q)$

Recap: DFAs and NFAs (2)

Key difference: the type of the transition function:

- ***DFA:*** $\delta \in Q \times \Sigma \to Q$
- ***NFA:*** $\delta \in Q \times \Sigma \to \mathcal{P}(Q)$

Language of an automaton: the set of all words it accepts.

Recap: DFAs and NFAs (2)

Key difference: the type of the transition function:

- ***DFA:*** $\delta \in Q \times \Sigma \to Q$
- ***NFA:*** $\delta \in Q \times \Sigma \to \mathcal{P}(Q)$

Language of an automaton: the set of all words it accepts.

As DFAs and NFAs are ***interconvertible***, these two kinds of automata defines the same ***class*** of languages.

Regular Expressions

- Automata describe languages in a somewhat indirect way: not always obvious what the defined language is.

Regular Expressions

- Automata describe languages in a somewhat indirect way: not always obvious what the defined language is.
- ***Regular Expressions*** offer a different, more direct way to describe languages.

Regular Expressions

- Automata describe languages in a somewhat indirect way: not always obvious what the defined language is.
- ***Regular Expressions*** offer a different, more direct way to describe languages.
- We will see (later) that the class of languages that can be described by regular expressions again is the same as those describable by DFAs and NFAs.

Regular Expressions

- Automata describe languages in a somewhat indirect way: not always obvious what the defined language is.
- ***Regular Expressions*** offer a different, more direct way to describe languages.
- We will see (later) that the class of languages that can be described by regular expressions again is the same as those describable by DFAs and NFAs.
- This class is called the ***regular*** languages. Hence the name regular expressions.

Syntax of Regular Expressions

1. $\emptyset$ is an RE
2. ϵ is an RE
3. For all $x \in \Sigma$, $\mathbf{x}$ is an RE
   (Handwriting convention: $\underline{x}$ is an RE)
4. If E and F are REs, so is $E + F$
5. If E and F are REs, so is EF
6. If E is an REs, so is E^*
7. If E is an REs, so is (E)

These are ***all*** regular expressions.

Conventions

- The $*$-operator has higher precedence than $+$ and sequencing.

Conventions

- The $*$-operator has higher precedence than $+$ and sequencing.
  E.g.

$$\begin{aligned} \mathbf{ab}^* &= \mathbf{a}(\mathbf{b}^*) \\ \mathbf{a}+\mathbf{b}^* &= \mathbf{a}+(\mathbf{b}^*) \end{aligned}$$

Conventions

- The $*$-operator has higher precedence than $+$ and sequencing.
  E.g.

$$\begin{aligned} \mathbf{ab}^* &= \mathbf{a}(\mathbf{b}^*) \\ \mathbf{a}+\mathbf{b}^* &= \mathbf{a}+(\mathbf{b}^*) \end{aligned}$$

- Sequencing has higher precedence than $+$.

Conventions

- The $*$-operator has higher precedence than $+$ and sequencing.
  E.g.

$$\begin{aligned} \mathbf{ab}^* &= \mathbf{a}(\mathbf{b}^*) \\ \mathbf{a} + \mathbf{b}^* &= \mathbf{a} + (\mathbf{b}^*) \end{aligned}$$

- Sequencing has higher precedence than $+$.
  E.g.

$$\mathbf{ab} + \mathbf{cd} = (\mathbf{ab}) + (\mathbf{cd})$$

Semantics of Regular Expressions

1. $L(\emptyset) = \emptyset$
2. $L(\epsilon) = \{\epsilon\}$
3. For all $x \in \Sigma$, $L(\mathbf{x}) = \{x\}$
4. $L(E + F) = L(E) \cup L(F)$
5. $L(EF) = L(E)L(F)$
6. $L(E^*) = L(E)^*$
7. $L(\ (E)\ ) = L(E)$