CS 241 – Week 4 Tutorial

Regular Languages Part 1

Spring 2018

1 Regular Languages Review

- An *alphabet* (denoted Σ) is a finite set of symbols, such as:
  - $\{a, b, c\}$
  - $\{b\}$
  - $\{to, be, or, not, \_\}$
  - {0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F}
- A *word* or *string* (over an alphabet Σ) is finite sequence of symbols from Σ, such as:
  - *bac*, *aba*, *c* given that $\Sigma = \{a, b, c\}$
  - ε,*b*, *bb*, *bbb* given that $\Sigma = \{b\}$
  - *to_be_or_not_to_be* (one word formed from the alphabet)
    $\Sigma = \{to, be, or, not, \_\}$
  - $DEADBEEF$, $FACE$ given that Σ = {0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F}
- The *empty word* (word with no letters) is usually denoted by ϵ or ε.
- A *language* is a set of words.
- A *regular* language R is a sets of words where either:
  - $R = \emptyset$: the empty language.
  - $R = \{w\}$: the language containing a single word.
  - $R = R_1 \cup R_2$: the union of two regular languages.
  - $R = R_1 \cdot R_2$: the concatenation of two regular languages.
  - $R = L^* = \bigcup_{i=0}^{\infty} L^i$ where L is a regular language, $L^0 = \{\varepsilon\}$ and for $i > 0, L^i = L \cdot L^{i-1}$.

1.1 Exercises

Build the following languages using combinations of finite languages with regular operations (set notation):

1. The language of binary strings whose second letter is a '0' and whose 5th is a '1'.
2. The language of binary strings that contain the substring "110101".

2 Deterministic Finite Automata (DFAs)

A Deterministic Finite Automaton (DFA) is a 5-tuple $(\Sigma, Q, q_0, \mathcal{A}, \delta)$ where:

- Q is a finite set of states.
- Σ is the input alphabet.
- $q_0 \in Q$ is a the starting state.
- $\mathcal{A} \subseteq Q$ (sometimes also denoted by $\mathcal{F}$) is the set of accepting states.
- $\delta : Q \times \Sigma \to Q$ is the transition function.

In the definition, $\delta(q, \sigma)$ is defined for every $q \in Q$ and $\sigma \in \Sigma$. However, when drawing the DFA we often assume that the transition from one state goes to an implicit error state if it was not shown in the diagram. An error state is any state that can never reach an accepting state by consuming any inputs, and is not itself accepting.

2.1 Exercises

Draw DFAs for the following languages:

1. The language of strings over $\Sigma = \{a, b, c\}$ that contain exactly one a and an even number of c's (no restriction on number of b's).
2. The language of strings over $\Sigma = \{a, b\}$ that contain an even number of a's and an odd number of b's.
3. Modify your previous solution by including c in the alphabet and allowing words to have any number of c's.
4. The language of strings over $\Sigma = \{0, 1\}$ that end in 1011.
5. Modify your previous solution by allowing 1011 to appear anywhere in the string.
6. The language of strings over $\Sigma = \{0, 1, 2, 3\}$ which are integers whose digits sum to 3. Leading zeros are permitted.
7. The language of strings over $\Sigma = \{a, b, c\}$ that end in cab and contain an even number of a's (no restriction on the number of b's or c's).

3 Non-deterministic Finite Automata (NFAs)

A Nondeterministic Finite Automaton (NFA) is a 5-tuple $(Q, \Sigma, q_0, \mathcal{A}, \delta)$ where:

- Q is a finite set of states.
- Σ is the input alphabet.
- $q_0 \in Q$ is a the starting state.
- $\mathcal{A} \subseteq Q$ (sometimes also denoted by $\mathcal{F}$) is the set of accepting states.
- $\delta : Q \times \Sigma \to \mathbb{P}(Q)$ is the transition function.

The key difference between NFAs and DFAs is the transition function: for DFAs it outputs a state, whereas for NFAs it outputs a set of states. This means that an NFA can be in multiple states at once.

3.1 Exercises

1. Show that every DFA is also an NFA.
2. Draw NFAs for the following languages:
   (a) The language of strings over $\Sigma = \{0, 1\}$ ending in 1011.
   (b) The language of strings over $\Sigma = \{0, 1\}$ ending in either 01 or 10.
   (c) The language of strings over $\Sigma = \{0, 1\}$ beginning with 1000.