CS 241 – Week 6 Tutorial

Regular Language Part 2: DFAs, NFAs and ϵ-NFAs

Spring 2017

1 Deterministic Finite Automata (DFAs)

A Deterministic Finite Automaton (DFA) is a 5-tuple $(\Sigma, Q, q_0, \mathcal{A}, \delta)$ where:
Σ – the input alphabet
Q – a finite set of states
$q_0 \in Q$ – the starting state in the set of states
$\mathcal{A} \subseteq Q$ – a set of accepting states
$\delta : Q \times \Sigma \to Q$ – the transition function

In the definition, $\delta(q, \sigma)$ exists for every $q \in Q$ and $\sigma \in \Sigma$. However, when drawing the DFA diagram, we often assume that the transition from one state goes to 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 error state itself cannot be an accepting state.

1.1 DFA Problems

Draw DFA diagrams 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. How would the solution change if we added c to the alphabet and words could have any number of c's?
3. The language of strings over $\Sigma = \{0, 1\}$ that end in 1011. How would the solution change if 1011 could appear anywhere in the string?
4. The language of strings over $\Sigma = \{0, 1, 2, 3\}$ which are integers whose digit sum is 3. Leading zeros are permitted.
5. 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).

2 Non-deterministic Finite Automata (NFAs)

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

Σ – the input alphabet
Q – a finite set of states
$q_0 \in Q$ – the starting state in the set of states
$\mathcal{A} \subseteq Q$ – a set of accepting states
$\delta : Q \times \Sigma \to \mathcal{P}(Q)$ – the transition function

The key difference between NFAs ands DFA is the transition function. In DFAs, $\delta(q, \sigma)$ maps to a single state for any choice of $q \in Q$ and $\sigma \in \Sigma$. For NFAs, $\delta(q, \sigma)$ maps to a subset of states, which is equivalent to $\delta(q, \sigma)$ mapping to an element of the power set $\mathcal{P}(Q)$. We can show that DFAs are also NFAs, but not vice-versa. If so, how?

2.1 ϵ-NFA

An ϵ-NFA allows the use of epsilon transitions. ϵ-transitions allow transitions from one state to another without consuming any input. This is useful when we connect multiple NFA together.

The definition of an ϵ-NFA is very similar to the definition of an NFA. The only difference is that for the transition function δ, we include ϵ as part of the definition of δ, that is $\delta : Q \times (\Sigma \cup \{\epsilon\}) \to \mathcal{P}(Q)$.

2.2 NFA and ϵ-NFA Problems

1. Draw an NFA for the following languages:
   (a) Let A be the language of strings over $\Sigma = \{0, 1\}$ ending with "0001".
   (b) Let B be the language of strings over $\Sigma = \{0, 1\}$ ending with either "01" or "10".
   (c) Let C be the language of strings over $\Sigma = \{0, 1\}$ starting with ''1000". What did you notice about the NFA you've created?
2. Give an ϵ-NFA for the language $C(A \cup B)$, where A, B and C are the languages from the previous question.
3. Convert the NFA of language B to a DFA using subset construction, where B is the language from previous question.