MSc in Computer Science
MSc in Mathematics and the Foundations of Computer Science

Michaelmas Term 2017

FOUNDATIONS OF CS

Exercise class 2 (non-regular languages; CFGs)

1. Consider the following two variants of DFAs

   - A "reverse DFA" (RDFA), which is given exactly as a DFA, but executes starting at the *end* of a string.
   - A "backwards-and-forwards DFA" (BAFDA), which is like a DFA but simultaneously reads from the beginning of the string and the end of a string.

     A BAFDA is of the form $(Q, \Sigma, \delta, (q_r, q_l), F)$, where

     – Q is a finite set of states
     – Σ is the input alphabet
     – $q_r \in Q$ is the initial state of the rightward-moving head and $q_l \in Q$ is the initial state of the leftward-moving head
     – δ is a transition function that takes a pair of states from Q and a pair of letters in Σ and returns a new pair of states from Q.
     – A set of accepting pairs of states $F = (f_1, g_1), \dots (f_n, g_n)$

     A BAFDA computes on string ω using two heads L (which moves to the left) and R (which will move to the right); during computation, both of these heads are on an element of the string and each of them have their own control state. Initially head R begins on the first (i.e. leftmost) element of ω in state q_r, while L begins on the last element of ω in state q_l; at any point the machine simultaneously moves head R one space to the right and head L one space to the left according to the transition function δ. That is, if at some point head R is sitting on an element of the input with symbol a in state q, and head L is sitting on an input element with symbol b in state q', then we find (r, r') such that $\delta((q, q'), (a, b)) = (r, r')$ and move R to the right and into state r, while moving L to the left and into state r'.

     The computation terminates when R is on the beginning of the string and L is on the end of the string, and it accepts if the state for R paired with the state for L is in F.

   Is every language accepted by a reverse DFA regular? Prove or disprove your answer.

   Is every language accepted by a BAFDA regular? Prove or disprove your answer.

2. This is Sipser problem 1.49b (page 90).

   Let $C = \{1^k y | y \in \{0, 1\}^*$ and y contains at most k occurrences of 1 for $k \geq 1\}$. Show that C is not recognizable by a DFA.

   Hint: try using the pumping lemma.

3. (A hopefully straightforward exercise on CFGs construction:)

   (a) Write down a CFG that defines the set of all words over the alphabet $\{a, b\}$ where the number of a's is equal to the number of b's. Explain how your grammar achieves this.

(b) Write down a Chomsky normal form CFG that defines palindromes over the alphabet $\{a, b, c\}$.

4. This problem concerns another way to prove that a language is *not* regular.

   Let L be a language, and consider the following relation $\equiv_L$ on strings:

   $s_1 \equiv_L s_2$ if and only if for every string w, $s_1 w$ is in L iff $s_2 w$ is in L.

   It is easy to show that this is an equivalence relation. Informally, two strings are equivalent if they have the same "impact" on membership in the language L. Let $I(L)$ be the number of equivalence classes of $\equiv_L$ – i.e. the maximal number of inequivalent elements.

   - Suppose L is recognized by a DFA A, and suppose that two strings w_1 and w_2 reach the same state when used as input to A (this state need not be an accepting state).
     - Prove that $w_1 \equiv_L w_2$.
     - Explain why this shows that if L is a language with $I(L)$ infinite, L cannot be regular.
   - Suppose L is a language and $I(L)$ is finite. Construct a DFA recognizing L that has exactly $I(L)$ states (hint: make each equivalence class into a state).
   - Consider the language $L = \{www : w \in \{a, b\}^*\}$. Show that L is not regular by giving infinitely many pairwise inequivalent elements.