CS172 Computability & Complexity, Spring 2009

Homework 2

Out: 5 Feb, 2009
Due: 12 Feb., 2009

Note: Not all questions are of equal difficulty. Questions marked with an asterisk (*) are to be handed in. The others are for practice and will not be graded. Put your solutions to the (*) problems in the homework box on Soda level 2 by 4pm on the due date. Take time to write clear *and* concise *answers; confused and long-winded solutions will be penalized. You are allowed to work in small groups to discuss the homework and gain an understanding of what's involved, but your submitted solutions must be completely your own work. Depending on grading resources, we might only grade a random subset of the required problems and simply check off the rest; so you are advised to attempt all questions.*

1. (*) An *all-paths* NFA is defined in the same way as a standard NFA, except the definition of acceptance is changed so that the machine accepts a string w if and only if *all* possible computations on input w lead to accepting states. (In other words, if *any* computation on w dies out or ends in a rejecting state, the machine rejects; else it accepts.) Show that the class of languages accepted by an all-paths NFA is exactly the class of regular languages. [HINT: One direction is obvious. For the other, think about the construction we used to convert an NFA into a DFA.]

2. Show that regular languages are closed under *reverse*. The reverse of language L is $L^R = \{\underline{w}^R : \underline{w} \in L\}$, where $(w_1 \cdots w_n)^R = w_n w_{n-1} \cdots w_1$.

   Show they are also closed under intersection; i.e., that $L_1 \cap L_2 = \{\underline{w} : \underline{w} \in L_1 \text{and } \underline{w} \in L_2\}$ is regular if L_1 and L_2 are.

3. (*) Show that regular languages are closed under *shuffle*: for languages L_1 and L_2 over Σ, their shuffle is the language $= \{\underline{w} : \underline{w} = a_1 b_1 \cdots a_k b_k, a_1 \cdots a_k \in L_1, b_1 \cdots b_k \in L_2$ and each $a_i, b_i \in \Sigma^*\}$.

4. Construct regular expressions that denote each of the following languages:

   (a) The set of all words over the English alphabet that have "kk" as a substring. [NOTE: Use the symbol Σ as shorthand for the regular expression $\mathbf{a} \cup \mathbf{b} \cup \ldots \cup \mathbf{z}$, denoting the English alphabet.]

   (b) (*) The set of all words over the English alphabet that begin with a vowel (i.e., with 'a', 'e', 'i', 'o' or 'u') and end in 'ing'.

   (c) The set of all words over the English alphabet that have at least two k's.

   (d) (*) The set of all 0-1 strings in which the number of 1's is divisible by three.

   (e) The set of all words over the English alphabet that have an even number of vowels and three consonants.

   (f) (*) The set of all 0-1 strings such that in any prefix, there is at most one more 1 than 0's and at most one more 0 than 1's. (x is a prefix of $\underline{w} \in L$ if $\underline{w} = xy$ and $|y| \geq 1$.

   (g) The set of all words over the English alphabet that have a vowel in every odd position.

5. (*) Let R, S be arbitrary regular expressions. Which of the following statements are true? If the statement is true, give a proof; if it is false, give a counterexample.

   (a) $(R \cup S)^* \equiv R^* \cup S^*$; (b) $(R^*)^* \equiv R^*$; (c) if $R^* \equiv S^*$ then $R \equiv S$.

   NOTE: The symbol '$\equiv$' denotes *equivalence* of regular expressions: $R \equiv S$ means that $L(R) = L(S)$ (R and S denote the same language).

6. In the text, the construction of an NFA for the regular expression R^* from an NFA for R involves adding a new start state q_0' with an ϵ-transition to the old start state q_0 (and also ϵ-transitions from all accepting states to q_0). Since the new NFA must accept the empty string, we make q_0' an accepting state. Suppose we didn't add q_0', but instead just made the old start state q_0 accepting (and added the other ϵ-transitions into q_0 as before). Give a small example which shows that this simpler construction does not always work correctly.