CSCI565 - Compiler Design

Spring 2016

Due Date: Feb. 05, 2016 at 1:00 PM in Class

Problem 1 [30 points]: Regular Expressions and Finite Automata

Develop a regular expression (RE) that detects the longest string over the alphabet {a-z} with the following properties:

1. The string begins with an ‘a’ character and ends with a ‘z’ character;
2. After the first ‘a’, the string can include an alternation of two subsequences, namely a sequence that begins with a ‘b’ character followed by one or more ‘s’ characters ending with an ‘e’ character and a subsequence that begins with a ‘c’ character but does not have any ‘s’ character until a terminating ‘t’ character.
3. If at any point in the subsequence there is an ‘x’ character, that specific sequence is considered terminated, i.e., the ‘x’ acts as the ‘e’ character in the first type of sequence and as the ‘t’ character in the second type of sequence.
4. The two subsequences cannot be nested but can be repeated in any alternating order.

As an example the string “absssecaaabbaftbsezw” is to be accepted as well as the string “abssxcefztz”

Questions:

a) [05 points] Develop a regular expression that captures the structure of the acceptable strings described above. Use short-hands to capture subsets of characters in the alphabet so that your description and the corresponding FA are kept short.

b) [10 points] Using the regular expression defined in section a) devise the corresponding Non-Deterministic Finite Automaton (NFA) using the Thompson construction described in class.

c) [10 points] Convert the NFA in section b) to a DFA using the subset construction. Show the mapping between the states in the NFA and the resulting DFA.

d) [05 points] Minimize the DFA derived in section c) (or show it is already minimal) using the iterative refinement algorithm described in class.
Problem 2 [20 points]: Translation from DFA to Regular Expressions

Given the DFA below over the alphabet {0,1} determine the following:

a. [15 points] Use the dynamic-programming Kleene algorithm to derive the regular expression that denotes the language accepted by it. Make sure your labelling of the DFA states is correct and that the DFA is completely specified, i.e., each state has transitions on all the alphabet characters.

b. [05 points] Describe succintly what are the words accepted by this DFA?

Problem 3 [50 points]: Predictive Top-Down Parsing

Consider the CFG grammar \( G = (N={S, A, B}, T={a, b}, P, S) \) where the set of productions \( P \) is given below:

\[
S \rightarrow A a A b \mid B b B a \\
A \rightarrow a \mid \varepsilon \\
B \rightarrow b \mid \varepsilon
\]

Questions:

a) [05 points] Can this grammar be used as presented for parsing using a predictive (backtrack-free) algorithm? Why or why not?

b) [10 points] Devise an alternative (but equivalent) grammar for the same language that has the LL(1) property.

c) [15 points] Compute the FIRST and FOLLOW sets for each production’s RHS and the non-terminal symbol respectively. Use these to show that the grammar has in fact the LL(1) property.

d) [10 points] Derive the LL(1) parsing table as described in the lectures and show that in fact the grammar is parseable using the LL(1) parsing algorithm.

e) [10 points] Show the sequence of parsing steps and the corresponding parse tree for this algorithm and the two inputs \( w_1 = “aab” \) and \( w_2 = “ba” \).