COMS W4115: Programming Languages and Translators

Homework Assignment 2

Due Sunday, November 14 at 11:59 PM EST via Courseworks

Place your answers in the boxes below and upload your solutions as a PDF file on Courseworks.
For example, you may load this PDF file into Inkscape (https://inkscape.org), edit, save each as PDF, and then combine them.

Name: Uni:

1. Write a regular expression for floating constants according to the following (after K&R):

   > A floating constant consists of an integer part, a decimal point, a fraction part, an `e` or an `E`, and an optionally negative integer exponent. The integer and fraction parts both consist of a sequence of digits. Either the integer part, or the fraction part (not both) may be missing; either the decimal point or the `e`/`E` and the exponent (not both) may be missing.

   You may use extended regular expression notation such as `[0-9]` and `+`, but explain what you mean by each. You may also use something like ocamllex's *let* notation to define shorthands.

   Hint: make sure your regular expression accepts floating constants such as `1. 0.5E-15 .3e3 .2 1e5 3.5E-4` but not integer constants such as `42`

2. Draw a DFA for a scanner that recognizes and distinguishes the following set of keywords. Draw accepting states with double lines and label them with the name of the keyword they accept. Follow the definition of a DFA given in class.

   `fun if elseif else inc`

3. Construct nondeterministic finite automata for the following regular expressions using Thompson's algorithm (Algorithm 3.23, p. 159, also shown in class), then use the subset construction algorithm to construct DFAs for them using Algorithm 3.20 (p. 153, also shown in class).

   Number the NFA states; use the numbers to label DFA states while performing subset construction, e.g., like Figure 3.35 (p. 155).

   a) $(ba(a|\varepsilon))^*$

Draw the NFA here

Draw the subset construction table here

Draw the DFA here

   b) $((a^*|b^*)bb)^*$

Draw the NFA here

Draw the subset construction table here

Draw the DFA here

4. Using this grammar, whose three terminals are *a*, *b*, and *c*,

$E \to b\ F\ c$
$E \to a$
$F \to E\ b\ F$
$F \to E$

(a) Construct a rightmost derivation for *bbabaccc*. Underline the handle of each right-sentential form.

(b) Show the steps of a shift-reduce (bottom-up) parser corresponding to this rightmost derivation.

(c) Show the concrete parse tree that would be constructed during this shift-reduce parse.

5. Draw the LR(0) automaton for the following ambiguous grammar. **if**, **else**, and **null** are terminals; the third rule indicates *T* may be the empty string. Indicate the state in which the shift/reduce conflict appears. Check your work by running ‘‘ocamlyacc -v’’ on the grammar below and looking through the ‘‘.output’’ file. Show part of your .output file below that confirms your answer.

$S' \to S$
$S \to$ **if** $S\ T$
$S \to$ **null**
$T \to$
$T \to$ **else** S