COMS W4115
Programming Languages and Translators
Homework Assignment 2

Prof. Stephen A. Edwards
Columbia University

Due June 27th, 2011
at 11:59 PM your time

CVN students: FAX the solutions to CVN.
Do this assignment alone. You may consult the instructor
or a TA, but not other students.

1. Scanners

(a) Using Ocamllex-like syntax, write a scanner for C’s
floating point numbers, as defined by Ritchie.
A floating constant consists of an integer
part, a decimal point, a fraction part, an
e, and an optionally signed 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 and the exponent (not both) may
be missing.

Hint: make sure your scanner accepts constants
such as 1.0 0.5e-15 3e+3 2 1e5 but not inte-
ger constants such as 42

(b) Draw a DFA for a scanner that recognizes and dis-
tinguishes the following set of keywords. Draw ac-
cepting states with double lines and label them with
the name of the keyword they accept. Follow the
definition of a DFA given in class.
abort abs accept access else elsif for
subtype type

2. Construct nondeterministic finite automata for the fol-
lowing regular expressions using Algorithm 3.23 (p. 159,
shown in class), then use the subset construction algo-
rithm to construct DFAs for them using Algorithm 3.20
(p. 153, also shown in class).

(a) \((a | ab)^*\)
(b) \((a (c | b))^*\)
(c) \(a (a | b)^* b\)

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

3. Using the grammar

\[
S \rightarrow (L) \mid a \\
L \rightarrow L, S \mid S
\]

(a) Construct a rightmost derivation for \((a, (a, a))\) and
show the handle of each right-sentential form.
(b) Show the steps of a shift-reduce (bottom-up) parser
considering to this rightmost derivation.
(c) Show the concrete parse tree that would be con-
structed during this shift-reduce parse.

4. Build 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.

\[
S' \rightarrow S \\
S \rightarrow if S T \\
S \rightarrow null \\
T \rightarrow \\
T \rightarrow else S
\]

Check your work by running "ocamlyacc -v" on the gram-
mar below and looking through the ".output" file.

%token IF ELSE NULL
%start s
%type <int>s
%

s : IF s t { 0 } \\
| NULL { 0 }

t : /* empty */ { 0 } \\
| ELSE s { 0 }