CPSC 510: Fall Final Midway Exam

Robin Cockett

December 14, 1998

This exam is worth 17.5% of the course and 35% of the examination credit. This is a take home exam which is due on the first class in 1999. You must complete all except for the last question of the exam BY HAND (that is in your own hand writing).

1. (15 points) For each of these grammars you are expected to calculate the vital statistics (that is nullable, endable, first, and follow properties) and to give the item automata.

   (a) (5 points) Show that the following grammar is not LR(1):

   ```
   A -> aAa | \epsilon
   ```

   Is this grammar ambiguous? Explain your answer.

   (b) (5 points) Show that the following grammar is LR(1) but not LALR(1):

   ```
   S -> a A d | b B d | a B e | b A e
   A -> c
   B -> c
   ```

   (c) (5 points) Show that the following grammar is LL(1) but not SLR(1):

   ```
   S -> A a A b | B b B a
   A -> \epsilon
   B -> \epsilon
   ```

2. (10 points) Consider the following attribute grammar:

```
based_num -> num basechar      { based_num.val = num.val
                                 num.base = base_char.base }
base_char -> o                 { base_char.base = 8 }
base_char -> d                 { base_char.base = 10 }
num_1 -> num_2 digit           { num_1.val =
                                   if digit.avl = error
                                       or num_2.val = error
                                    then error
                                   else num_2val * num_1.bas
                                         + digit.val;
                                 num_2.base = num_1.base;
                                 digit.base = num.base}
digit -> 0                     { digit.val = 0}
digit -> 1                     { digit.val = 1}
digit -> 2                     { digit.val = 2}
digit -> 3                     { digit.val = 3}
digit -> 4                     { digit.val = 4}
digit -> 5                     { digit.val = 5}
digit -> 6                     { digit.val = 6}
digit -> 7                     { digit.val = 7}
digit -> 8                     { digit.val =
                                   if digit.base = 8
                                    then error
                                   else 8}
digit -> 9                     { digit.val =
                                   if digit.base = 8
                                    then error
                                   else 9}
```

(a) Draw the plumbing diagrams for this attribute system,

(b) Show that this system is strongly non-circular. (Hint: you must provide the dependencies at the types and show that the equations given together with the dependencies expressed as functions on the "input types" give functions with dependencies which lie within the specified dependencies at the "output types.")

(c) Generate mutually recursive pseudo-code to calculate the value of the based integer.

3. (10 points) This question need not be answered by hand (i.e. you permitted to answer this question with the aid of a computer):

   Propose an extension to the language m+ which includes

   (a) boolean variables, arrays of the two basic types (booleans and integers), and pointers for these types.

   (b) where functions can have parameters which are pointers and can return pointers.

   You are expected to:

   (a) Propose a grammar for the extended language (the proposed language must be a superset of the existing language) and discuss its semantics. Pay careful attention to the following issues:

      i. The grammar should be implementable in YACC.

      ii. Can the size of a local array be determined by a variable/argument. If so what are the implementation issues?

      iii. Can a function return a pointer to a local variable? If so what are the implementation issues?

      Please note you are not expected to solve these implementation issues in detail (hint: some of these issues are discussed in the book!)

   (b) Explain the additional instructions you would expect add to the existing stack machine in order to implement this language (hint: you will certainly need new STORE and LOAD modes to accommodate offsets into arrays and access via pointers).