CSE 322
Intro to Formal Models in CS
Homework #7
Due: Friday, 3 Dec 10
20 Nov 10

W. L. Ruzzo

As usual three turn-in bundles: Problem(s) 1–2 in one, problem(s) 3–4 *and 7* in another and problem(s) 5–6 in the third. Text problems from Sipser, *US* 2^{nd} *edition*; see online scanned versions if needed.

1. 2.27a. **Extra Credit:** 2.27b. (Don't change the language, just the grammar.) Include a convincing informal argument that your grammar is unambiguous.

2. 2.10. The "or" is an "inclusive or"; i.e. both clauses may be true. Do it directly, i.e., do *not* follow either of the CFG to PDA constructions.

3. Consider the following CFG:

$$\begin{aligned} E &\rightarrow E + T \mid T \\ T &\rightarrow T * F \mid F \\ F &\rightarrow (E) \mid a \end{aligned}$$

   (a) Draw a parse tree for the string `a + a * a + a`.

   (b) Number the internal nodes (not leaves) in your tree in *postorder*, i.e., recursively explore the left subtree of the root, then the right subtree, then assign the next unused number to the root. Numbers used in the left subtree will be sequential starting at 1; those in the right subtree will be sequential, starting with one more than the largest number in the left subtree. Place these numbers to the *left* of their corresponding nodes in your tree diagram. (Although some nodes have 3 children, the middle one is always a leaf in this grammar, so "left"/"right" should be clear.)

   (c) Which grammar rule was used to produce the children of node number 7 in your tree?

   (d) Give a rightmost derivation of the same string. Number the steps in the derivation in reverse order, as on page 12 of the PDA slides.

   (e) Which grammar rule was used in the seventh rewrite step in your derivation?

   (f) Are (c) & (e) a coincidence? Briefly explain.

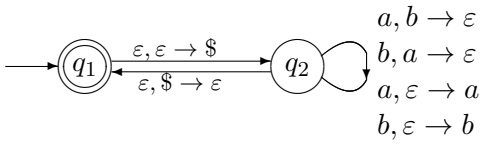
   (g) To evaluate an arithmetic expression, it is natural to use a postorder traversal: evaluate a node's children in left-to-right order, then evaluate the node (performing addition or multiplication as appropriate, based on the terminal labeling the middle child; "evaluating" a node with only one child simply means copying the child's value). Assuming the four `a`'s in the example above are placeholders for the values 1, 2, 3 and 4 in L-R order, label the nodes of the parse tree with the corresponding values. Place these labels to the *right* of each node. Does this evaluation correspond to the usual precedence/associativity conventions for the expression `1+2*3+4`?

(h) Draw the PDA for the shift-reduce parser built from this grammar, and give the steps in the PDA's computation leading to acceptance of input `a+a*a+a`. Number the reduce steps. Which rule is used for the seventh reduce transition? (Again, refer to slide 12 for a detailed example of what you are being asked to do and how to draw it.)

(i) **Extra credit:** Add exponentiation, denoted by `^`, to the grammar, so that it has higher precedence than `+`, `*`, and is *right*-associative, meaning that `7^5^3^2` is equivalent to `7^(5^(3^2))`. Informally (but convincingly) explain how your grammar reflects the desired precedence and associativity of each operator (while remaining unambiguous).

4. **Extra Credit:** Repeat steps a-f and h above, but give a *pre*order numbering, a *left*most derivation numbered in forward order, build and simulate the *top-down* PDA, as described on slides 8–10 (and Lemma 2.21 in the text), and comment on node 6, derivation step 6, and the 6th of the "production" steps by the PDA (i.e., corresponding to the numbered steps in the "Deriv" column on slide 10.)

5. Consider the following PDA M:



(a) Show that M is nondeterministic by giving a short input on which it has two computations. Show them.

(b) Describe in English the language L recognized by M. I want a *nonprocedural* description; don't say "do this then if that do something else..." (Hint: it has a very simple description.)

(c) Prove informally that M recognizes this language. Don't forget to argue that it rejects all strings *not* in L, on *all* (it's nondeterministic) possible computations. You don't need a detailed induction proof or the like, but do give a thorough and convincing argument.

(d) Show that M is "ambiguous" by giving a short input on which it has two accepting computations. Show them.

6. Apply the construction given in the proof of lemma 2.27 (PDA slides 23,24) to build a grammar for the language accepted by the PDA in problem 5. For this grammar, give a derivation tree for the string `abba`.

7. [Extra Credit:] Text 2.31.