CSE 322
Intro to Formal Models in CS
Homework #7
Due: Friday, 26 Feb 10
19 Feb 10

W. L. Ruzzo

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

1. 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 in your tree in *preorder*, i.e., assign number 1 to the root, then recursively number the internal nodes in the left subtree starting from the next available number (2), then do the same to the right subtree (if any), starting with the next number not already used. Place these numbers to the *left* of their corresponding nodes in your tree diagram. (Note that for this problem you should not number the leaves. 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 6 in your tree?

   (d) Give a leftmost derivation of the same string.

   (e) Which grammar rule was used in the sixth 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 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) **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).

2. 2.16. Just do star; I started concatenation in lecture Friday; will finish it Monday. Union is similar.

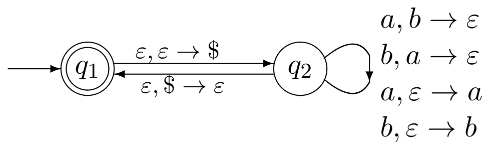
3. 2.27a. **Extra Credit:** 2.27b. Include a convincing informal argument that your grammar is unambiguous.

4. Give a formal description of a PDA recognizing the language defined by the following CFG:

$$\begin{array}{rcl} E & \to & E + T \mid T \\ T & \to & T * F \mid F \\ F & \to & (E) \mid a \end{array}$$

   You do not need to follow either of the CFG-to-PDA constructions, nor give a formal correctness proof, but do give a convincing informal argument/explanation for why/how it works.

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.