University of Central Florida

School of Computer Science

COT4210 Fall 2004

Prof. Rene Peralta

T3 Solutions (by TA Robert Lee)

1. Let L be the set of well-formed parenthetical expressions. Examples of strings in L are $(())()$ and $((()()))$.

   (a) (5 pts) Prove that L is not a regular language.

   Solution

   We use the Pumping Lemma for Regular Languages to prove that L is not a regular language. Proof by contradiction. Assume L is regular. The Pumping Lemma gives p, the pumping length, for L. Choose the string $s = (^p)^p$. Note that $s \in L$ because the left and right parentheses are balanced, and $|s| = 2p \geq p$. Choose the partition $s = xyz$, where $x = \epsilon$, $y = (^p$, and $z =)^p$, and note that $|y| = p \geq p$. For all divisions $y = uvw$, we have $v = (^m$, where $0 < m \leq p$. Choose $i = 2$: the pumped string is $xuv^2wz = (^{p+m})^p$, and is not in L because the left and right parentheses are not balanced. This is a contradiction. Therefore L is not a regular language.

   (b) (10 pts) Give a context-free grammar for L.

   Solution

   $S \rightarrow (S) \mid SS \mid \epsilon$ generates L. The first rule increases the parenthesis nesting level; the second rule generates sets of parentheses at the same level of nesting; the third rule is the termination rule. (This grammar assumes that $\epsilon \in L$.)

(c) (10 pts) Give a PDA that accepts L.

Solution

One way to solve this problem is to convert our CFG from part (a) to a PDA, following the procedure given in the proof of Lemma 2.13 on page 107 of the textbook. There are three states in our PDA: q_{start}, q_{loop}, and q_{accept}. The transition from q_{start} to q_{loop} is given by the rule $\epsilon, \epsilon \rightarrow S\$$; the $\$$ marks the bottom of the stack. The transition from q_{loop} to q_{accept} is given by the rule $\epsilon, \$ \rightarrow \epsilon$; this pops off the bottom-of-stack marker at the end of the derivation. The transitions from q_{loop} to itself consist of two types. There is one transition for each rule in the CFG; our CFG has three rules, and the corresponding transitions are $\epsilon, S \rightarrow (S)$; $\epsilon, S \rightarrow SS$; and $\epsilon, S \rightarrow \epsilon$ (note that these transitions use the shorthand notation for pushing an entire string onto the stack). Also, there is one transition for each terminal in the CFG; our CFG has two terminals, with the corresponding transitions $(, (\rightarrow \epsilon$ and $),) \rightarrow \epsilon$. This completes the conversion from our CFG that generates L to a PDA that recognizes L.

2. (25 pts) Consider the machine M in figure 1. Give the sequence of configurations that M enters when the input is 000. Be careful!

Solution

(See page 132 in the textbook for another example of this procedure.)

$$D000$$
$$\sqcup A00$$
$$\sqcup xB0$$
$$\sqcup Cx0$$
$$\sqcup xC0$$
$$\sqcup xxB\sqcup$$
$$\sqcup xQ_{accept}x$$

3. (25 pts) Recall the definitions

   $E_{DFA} =$ the set of strings representing DFAs that accept no strings

   and

   $EQ_{DFA} =$ the set of strings representing pairs of equivalent DFAs.

   Give a proof that EQ_{DFA} is a decidable language.

   Solution

   To prove that EQ_{DFA} is decidable, we must build a TM to decide EQ_{DFA}. Another way of writing the definition of EQ_{DFA} is

$$EQ_{DFA} = \{\langle A, B\rangle \mid A \text{ and } B \text{ are DFA's and } L(A) = L(B).\}$$

   The hint given in this problem is to think about E_{DFA}; we know that this language is decidable (see Theorem 4.4 on page 154 of the textbook). If we can convert our problem into E_{DFA}, we're done. Recall (from COT3100) that

$$\begin{aligned} L(A) = L(B) &\iff [L(A) \subseteq L(B)] \wedge [L(B) \subseteq L(A)] \\ &\iff [L(A) - L(B) = \emptyset] \wedge [L(B) - L(A) = \emptyset] \\ &\iff [L(A) \cap \overline{L(B)} = \emptyset] \wedge [L(B) \cap \overline{L(A)} = \emptyset] \\ &\iff [L(A) \cap \overline{L(B)}] \cup [L(B) \cap \overline{L(A)}] = \emptyset \\ &\iff L(A) \oplus L(B) = \emptyset. \end{aligned}$$

   Here the notation $\oplus$ represents the symmetric difference of two sets. Thus the problem of determining whether $L(A) = L(B)$ is equivalent to the problem of determining whether $L(A) \oplus L(B) = \emptyset$. Hence we build our TM as follows: given the DFA specifications $\langle A\rangle$ and $\langle B\rangle$, construct a DFA $M_{A\oplus B}$ to recognize the regular language $L(A) \oplus L(B)$. We know that $L(A) \oplus L(B) = [L(A) \cap \overline{L(B)}] \cup [L(B) \cap \overline{L(A)}]$ is regular because $L(A)$ and $L(B)$ are regular, and the regular languages are closed under union, intersection, and complementation. The procedures for constructing the union, intersection, and complementation of DFA's can be automatically performed by a TM.

   Now pass the DFA specification $\langle M_{A\oplus B}\rangle$ to a sub-TM (like a subprocedure or a function call) that decides the language E_{DFA}; by Theorem 4.4, we know this sub-TM exists. The sub-TM will accept $\langle M_{A\oplus B}\rangle$ if $L(M_{A\oplus B}) = \emptyset$; if that is the case, then $L(A) = L(B)$, so our TM must accept $\langle A, B\rangle$. The sub-TM will reject $\langle M_{A\oplus B}\rangle$ if $L(M_{A\oplus B}) \neq \emptyset$; if that is the case, then $L(A) \neq L(B)$, so our TM must reject $\langle A, B\rangle$. Our TM will halt in all cases and correctly determine whether $L(A) = L(B)$. This completes the proof that EQ_{DFA} is decidable.