CSE 2001 Second test – solutions
Summer 2007
July 18, 2007
Instructor: S. Datta

NOTE: The solutions given below skip many details for the sake of saving time. You are expected to supply these details in an exam.

1. (12 points) (a) (6 points) Suppose C is a CFG and R is a regular language. Prove that the language $C \cap R$ is context-free.

   (b) (6 points) Consider the language $A = \{w | w \in \{a, b, c\}^*, w$ contains an equal number of a's, b's and c's$\}$. Use part(a) to show that A is not a CFL.

   Solution: Solved in the text – question 2.18 pg 130

2. (12 points) Prove that the CFG with rules $S \to 0S1S|1S0S|\epsilon$ generates the language $L = \{x \in \{0,1\}^* | n_0(x) = n_1(x)\}$ where $n_0(x), n_1(x)$ are the number of 0,1 (respectively) in x.
   Hint: you must prove that all strings produced by the grammar have the aforementioned property and any string with that property is produced by the grammar.

   Solution: This is done by proving either direction.

   First, let us prove that all strings produced by this grammar as equal number of ones and zeroes. This can be done using (strong) induction on the length of the string. The base case is length 2. The grammar produces only 01 and 10, both of which have the property. Suppose the inductive hypothesis holds for all strings of length smaller than n. Then any string of length n is produced using the rules $S \to 0S1S|1S0S$. Each rule adds one 0 and one 1. The variables S produce strings with equal numbers of ones and zeroes. Therefore the hypothesis holds for length n strings. (Note that n can only be even).

   Next, we prove that any string with an equal number of 1's and 0's are produced by this grammar. This can be done using induction as well. The base case is similar to before. Suppose that the hypothesis holds for strings with length less than n. Given such a string $w_1 \dots w_{2j}$ that has $w_1 = 1$, we assign to each character w_i in the string a number – viz., the value $n_1(x) - n_0(x)$ where $x = w_1 \dots w_{i-1}$. Then w_1 gets a label 0. Let w_k be the first character after w_1 to get a label 0. Then $w_2 \dots w_{k-1}$ has an equal number of zeroes and ones, and has length less than n. Hence it can be derived from S. Similarly $w_{k+1} \dots w_{2j}$ must have an equal number of ones and zeroes and can be derived from S. Therefore the whole string can be produced using the rule $S \to 1S0S$.

   A similar argument holds for strings starting with 0.

3. (12 points) (a) (8 points) A two-stack PDA is like a one-stack PDA but has an extra stack to work with. Describe a two-stack PDA that accepts the language $\{a^n b^n c^n | n > 0\}$. Next generalize your design to handle the language $\{a_1^n a_2^n \dots a_k^n | n > 0\}$, where k is a constant.

   Solution: We start pushing the a's in stack 1. When a b is received we move to a new state, push b into stack 2 and pop stack 1. As long as we get b's we keep doing this, moving to a reject state if stack 1 is empty. Then when we get a c we pop stack 2 for each c read. We accept if and only if the stack empties when the last c is read. If we read other characters we reject.

   Note: we always push in a \$ first to make checking for an empty stack easy.

   For generalizing, we CANNOT do the above for the first three and the next three and so on. That would imply it recognizes the language $\{a_1^n a_2^n a_3^n a_4^m a_5^m a_6^m \dots\}$. Instead we use stack 1 and stack 2 alternately. So first we push a_1's into stack 1, a_2's into stack 2, a_3's into stack 1, a_4's into stack 2 and so on.

   (b) (4 points) Describe a PDA that recognizes the language $\{a^n b^{n+m} a^m | n, m \geq 0\}$.

   Solution: Initially the PDA reads a's and pushes then into the stack. When it gets a b it moves to a new state in which it reads b's and pops the stack for each a read. When the stack is empty it moves

to a new state and pushes each b read into the stack. Then when it gets a a it moves to a new stack in which it pops a b for each a read. It accepts if and only if the stack is empties when the last a is read.

4. (12 points) (a) (7 points) Consider the language $A = \{0^n 10^{2n} 10^{3n} | n > 0\}$ over the alphabet $\{0, 1\}$. Use the Pumping Lemma to show that A is not context-free.

   Solution: Assume p to be the pumping length, and choose $s = 0^p 10^{2p} 10^{3p}$. The Pumping Lemma proof is straightforward.

   (b) (5 points) Describe a Turing machine that takes an integer encoded in unary (i.e. n is input as a string 1^n), and computes $f(x) = x^2$.

   Solution: Very briefly, copy x into tape 2. Mark off each 1 on tape 1 and copy tape 2 (x) at the end of tape 3. Thus at the end (when all 1's in tape 1 are marked) the third tape has x^2.

5. (12 points) (a) (6 points) Describe a Turing Machine that takes as input a string of 0's and 1's, interprets it as the binary representation of a nonnegative integer, and leaves the output the unary representation of that integer (i.e. a string of that many 1's).

   Solution: We leave the input on tape 1, use tape 2 to maintain the value 2^i for each i, and use tapes 3 and 4 as scratch and output tapes respectively. First we move the head of tape 1 to the rightmost position (the least significant bit). Then on tape 2 we put a single one. Next we repeat the following steps until the head on tape one falls off the left side.

   1. If the character under the head in tape 1 is a 1 append tape 2 to tape 4.
   2. Using tape 3, double the number of ones in tape 2. This can be done by first copying tape 2 to 3 and then appending tape 2 with the contents of tape 3.

   The output tape contains the unary representation of the number originally given in binary.

   (b) (6 points) Describe a Turing Machine that accepts strings in the language $A = \{www | w \in \{0, 1\}^*\}$ and halts in the reject state for any string not in the language A.

   Solution: First, compute the length of w. This can be done by writing on tape 2 a single 1 for every three characters in tape 1 (the input). This also checks that the input length is a multiple of 3. Next, using tape 2 the TM writes down the three thirds on tapes 3 through 5. Finally it checks that the contents of tapes 3 through 5 are equal by sweeping across these tapes from left to right.