

COMP 481: Automata, Formal Languages, and Computability  
 Spring 2008  
 Solutions to Homework Assignment #8

1. (a) Solving this problem depends on the fact (in the notes I handed out in class) that  $L(G)$  is infinite if and only if  $G$  generates at least one string  $w$ , where  $b^n < |w| \leq b^{n+1} + b^n$ , where  $n$  is the number of non-terminals in  $G$  and  $b$  is the maximum number of symbols in the right-hand side of a production rule in  $G$ . So, first check all strings of length less than or equal to  $b^n$ . If  $G$  generates at least three of these strings, then print YES. Otherwise, check if  $G$  generates at least one string  $w$ , where  $b^n < |w| \leq b^{n+1} + b^n$ . If this is the case, print YES (since  $L(G)$  is infinite in this case). Otherwise, print NO.
  - (b) Let  $M$  be a DFA that accepts all even-length strings. Using the construction shown in the textbook, convert  $G$  into a PDA  $M'$ , build a PDA  $M''$  for  $L(M) \cap L(M')$ , and test for emptiness of  $L(M'')$ . If  $L(M'') = \emptyset$ , then print NO; otherwise, print YES.
  - (c) print YES. (every regular grammar is also a CFG!)
2. The following is an equivalent LL(1) grammar.

$$\begin{aligned} S &\rightarrow AT \\ T &\rightarrow B\$|c\$ \\ A &\rightarrow aX \\ B &\rightarrow bY \\ X &\rightarrow aX|\varepsilon \\ Y &\rightarrow bY|\varepsilon \end{aligned}$$

You can prove that it is LL(1) using the conditions stated at the top of Page 338 in your textbook.

3. (a) An equivalent CFG in CNF is:

$$\begin{aligned} E &\rightarrow EX|TY|LZ|id \\ T &\rightarrow TY|LZ|id \\ F &\rightarrow LZ|id \\ X &\rightarrow PT \\ Y &\rightarrow MF \\ Z &\rightarrow ER \\ P &\rightarrow + \\ M &\rightarrow * \\ L &\rightarrow ( \\ R &\rightarrow ) \end{aligned}$$

- (b) The table that a CYK parser builds is shown in Table 1. Given that the length of the string is  $n = 5$ , and we have that the start variable,  $E$ , is in entry  $[5, 1]$  of the table, we conclude that  $w \in L$ .

Table 1: The table that a CYK parser builds for string  $id + id * id$  using the CFG in 3a.

Row 5	E				
Row 4		X			
Row 3	E		E,T		
Row 2		X		Y	
Row 1	E,T,F	P	E,T,F	M	E,T,F
<b>Input String:</b>	<b>id</b>	<b>+</b>	<b>id</b>	<b>*</b>	<b>id</b>

4. (a) A 2-tape TM  $M$  that decides the language scans the input from left to right to check if it's of the form  $a^*b^*$ . If not, it rejects. Then, it copies the  $b$ 's to the second tape, and resets the reading heads on both tapes to the leftmost cells. Then, it moves both reading heads right as long as it's seeing an ' $a$ ' on the first tape, and a ' $b$ ' on the second tape. Once it sees a blank on the first and an ' $a$ ' on the second, it halts and accepts. If it sees two blanks on both tapes, or a blank on the second and an ' $a$ ' on the first, it halts and rejects.
- (b) A 3-tape TM  $M$  that decides the language scans the input from left to right and checks if it's of the form  $(a \cup b)^*$ . If not, it rejects. Then, it computes the length of the string, copies its second third to the second tape, and its third part to the third tape (if the string's length is not divisible by 3,  $M$  halts and rejects). Then,  $M$  moves all three reading heads to the leftmost cells, and move them right in parallel, while checking that all three heads read the same symbol at all times. If the three reading heads see blanks,  $M$  halts and accepts. Otherwise (not the same symbol on all three tapes), it halts and rejects.
- (c) A 4-tape TM  $M$  that decides the language computes the numbers of  $a$ 's,  $b$ 's, and  $c$ 's and stores them on tapes 2, 3, and 4, respectively. Then, it compares all three numbers (when they are represented in unary, for example). If they are equal, it halts and accepts; otherwise, it rejects. While counting, if  $M$  encounters any symbol not in the alphabet, it halts and rejects.