Middle East Technical University Department of Computer Engineering

CENG 280

Formal Languages and Abstract Machines

Spring 2022-2023

Homework 4

Name Surname:Doruk Berke Yurtsizoglu
Student ID: 2522225

Answer for Q1

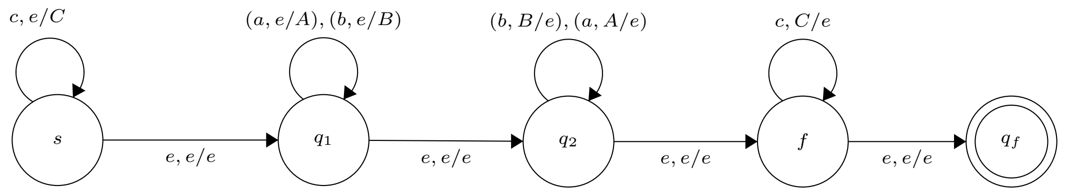
1. Context-free grammar that generates our language is:

$$S \rightarrow AX$$
$$A \rightarrow 0A0|1T1|\#R$$
$$X \rightarrow XX|0|1|\epsilon$$

The PDA of this grammar:
K = (q_0,q_1,q_2,q_f)
$\sum$ = (0,1)
γ = (0,1,S,A,X)
F = (f)

I was not able to construct the rest of the PDA. Sorry about that :((((

2. K = (s,q_1,q_2,f,q_f)
$\sum$ = (a,b,c)
γ = (A,B,C)
F = (f)
Δ:
1. ((s,c,e), (s,C))
2. ((s,e,e), (q_1,e)
3. ((q_1,a,e), (q_1,A))
4. ((q_1,b,e), (q_1,B))
5. ((q_1,e,e), (q_2,e))
6. ((q_2,a,A), (q_2,e))
7. ((q_2,b,B), (q_2,e))
8. ((q_2,e,e), (f,e))
9. ((f,c,C), (f,e))
10. ((f,e,e), (q_f,e)



Answer for Q2

Counter Example: $(w \in (0,1)^* |$length of w is odd$)$

G = (V, $\sum$, R,S), V = $(S) \cup \sum$, $\sum = (0,1)$

$$S \rightarrow 0S0|0S1|1S0|1S1|0|1$$

If we add the rule $S \rightarrow SS$ to the cfg that we defined, it will look like we get the kleene star of the language. However, kleene star of an expression includes the empty string which is not the case for our cfg since it doesn't have empty string in its rule set. That means we can't generate L^* of the language we defined even if we add the rule $S \rightarrow SS$.

Answer for Q3

1. Just L_1.

For L_1:(can be recognized as S-CFL)

$\rightarrow$ Push all a's to the stack
$\rightarrow$ When it is time to read b's, start to pop a's from the stack.
$\rightarrow$ When the process is finished; if the stack is empty
$\rightarrow$ Then the number of a's is equal to number of b's.

For L_2:(can't be recognized as S-CFL)

We can think of an algorithm which is similar to the language above, however it will not work because in the language L_2, the arrangement of a and b in the strings can be different. So we can't come up with an algorithm to recognize this language as a S-CFL.

For L_3:(can't be recognized as S-CFL)

$\rightarrow$ Divide the strings to two parts.
$\rightarrow$ $a^n b^n, b^m c^m$.
$\rightarrow$ In order to check if the condition is satisfied, we need to have two symbols for stack symbols
$\rightarrow$ Because we have to parts to compare.

2. L = $(a^n b^m | n > m)$
G = (V,$\sum$,R,S)
V = (S, A, X) $\sum$ $\sum$ = (a,b)

$$S \to Aa|XS|SXA$$
$$A \to Aa|\epsilon$$
$$X \to RR|aRb|bRa|\epsilon$$

The process of PDA will be:

$\to$ Push all as' to the stack
$\to$ When it is time to read bs', start to pop as' from the stack.
$\to$ When the reading of bs' is finished;
$\to$ If there are still elements in the stack
$\to$ Then the number of as' is greater than number of bs'.

3. One-Counter Automaton: A non-deterministic finite automaton with an addtional memory cell.

4. The purpose of using an additional memory is to hold a non-negative integer. We are using this memory to copy the behavior of stack structure. Incrementing the integer is equal to adding something to the stack, decrementing the integer is equal to removing something fromthe stack, and checking if the integer is zero is equal to checking if the stack is empty.

5. No, it is not closed under complementation. The reason for this is, the automons are non-deterministic. Let's look at the language L_1 from the part 1 of this question. The complement of L_1 is the string where the number of a is not equl to number of b. However the complement of L_1 still accepts the empty string which shouldn't be in the set of strings it accepts. So, we can say that, the class of S-CFLs is not closed under complementation.