Middle East Technical University | Department of Computer Engineering

CENG 280

Formal Languages and Abstract Machines

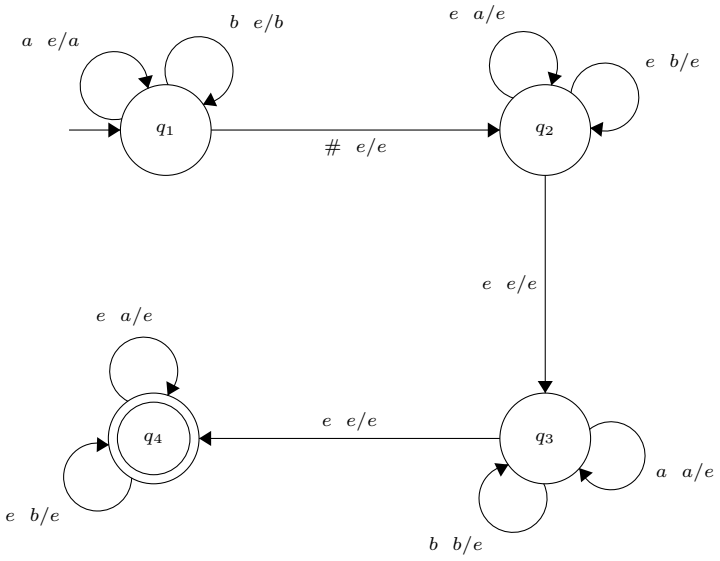
Spring 2022-2023

Homework 4

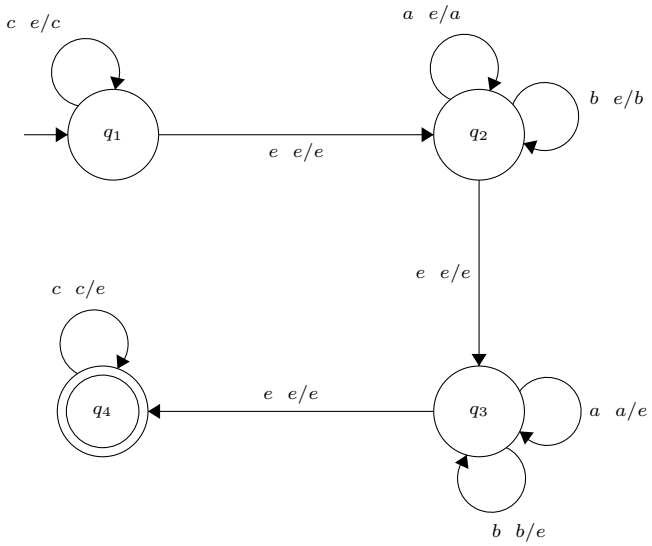
Name Surname: Yusuf Sami Lök
Student ID: 2521748

Answer for Q1

1. $M = (K, \Sigma, \Gamma, \Delta, s, F)$
K=$\{q_1, q_2, q_3, q_4\}$
$\Sigma = \{a, b, \#\}$
$\Gamma = \{a, b\}$
$F = \{q_4\}$
$s = q_1$ $\Delta = \{$
$\{(q_1, a, e), (q_1, a)\}$
$\{(q_1, b, e), (q_1, b)\}$
$\{(q_1, \#, e), (q_2, e)\}$
$\{(q_2, e, a), (q_2, e)\}$
$\{(q_2, e, b), (q_2, e)\}$
$\{(q_2, e, e), (q_3, e)\}$
$\{(q_3, a, a), (q_3, e)\}$
$\{(q_3, b, b), (q_3, e)\}$
$\{(q_3, e, e), (q_4, e)\}$
$\{(q_4, e, a), (q_4, e)\}$
$\{(q_4, e, b), (q_4, e)\}\}$



2. $M = (K, \Sigma, \Gamma, \Delta, s, F)$
K=$\{q_1, q_2, q_3, q_4\}$
$\Sigma = \{a, b, c\}$
$\Gamma = \{a, b, c\}$
$F = \{q_4\}$
$s = q_1$ $\Delta = \{$
$\{(q_1, c, e), (q_1, c)\}$
$\{(q_1, e, e), (q_2, e)\}$
$\{(q_2, a, e), (q_2, a)\}$
$\{(q_2, b, e), (q_2, b)\}$
$\{(q_2, e, e), (q_3, e)\}$
$\{(q_2, b, e), (q_2, b)\}$
$\{(q_2, e, e), (q_3, e)\}$
$\{(q_3, a, a), (q_3, e)\}$
$\{(q_3, b, b), (q_3, e)\}$
$\{(q_3, e, e), (q_4, e)\}$
$\{(q_4, c, c), (q_4, e)\}\}$



Answer for Q2

Let $L = \{a^x b^x | x \geq 1\}$.
The Context-free language belongs to the language (L defined above) defined below: $G = (V, \Sigma, R, S)$
$V = \{S, C, a, b\}$
$\Sigma = \{a, b\}$
$R = \{S \rightarrow aCb,$
$C \rightarrow aCb|e\}$
L* allows us to generate the empty string, but we cannot generate the empty string with adding this rule. Therefore, this rule does not generate empty string, even though we can generate the empty string with L*. That is why this rule does not generate L*.

Answer for Q3

1.
L_1 is S-CFL because we can describe this example with a stack item 's' by pushing an 's' onto the stack every time we see an 'a', and when we see a 'b', we transition to another state and pop an 's' item from the stack. Therefore, if the number of a's and b's are equal, the stack will be empty.

If we can not use bottom marker. L_2 is not S-CFL. We can handle the case where the number of 'a's and 'b's are equal, but we cannot handle the case where they are not equal. This is because we need to push one of the symbols and pop the other when we encounter them. However, this method does not work in all cases. For example, if we push 'a' and pop 'b', we would not be able to accept the string 'bba' with this method because we try to pop when the stack is empty. Similarly, this method also does not work when the positions of 'a's and 'b's are swapped. Therefore, we cannot describe this language with a single symbol.
If we use bottom marker, L_2 is S-CFL.

If we can use bottom marker, L_3 is S-CFL. When an 'a' is encountered, we push 'b' onto the stack, and when a 'b' is encountered, we pop a 'b' from the stack. When the number of 'b's equals the number of 'a's seen, which means the stack is empty, we can start adding 'b's to the stack because we can determine whether the stack is empty by using the bottom marker. Finally, every time we see a 'c', we pop a 'b' from the stack. If the stack is empty at the end, we accept; otherwise, we reject. If we can not use bottom marker, L_3 is not a S-CFL because we can not know whether the stack empty or not.

2. $L = \{a^m b^n c^{m+n} | m, n \in N\}$
Let M be the Context-free grammar that generates L.
$M = (K, \Sigma, \Gamma, \Delta, s, F)$
$K = \{q_1, q_2, q_3\}$
$\Sigma = \{a, b, c\}$
$\Gamma = \{a\}$
$F = \{q_3\}$
$s = q_1$
$\Delta = \{$
$\{(q_1, a, e), (q_1, a)\}$
$\{(q_1, e, e), (q_2, e)\}$
$\{(q_2, b, e), (q_2, a)\}$
$\{(q_2, e, e), (q_2, e)\}$
$\{(q_3, c, a), (q_3, e)\}\}$

3.The minimal memory element that must be added to finite automata so as to increase their computational power to recognize S-CFLs is counter.

4. The memory element that must be added to finite automata to recognize S-CFLs is a single counter variable. This counter is used to keep track of the number of times a specific symbol (e.g. "a") appears in the input string, and it is incremented each time the symbol is read by the automaton. The counter is decremented each time a different symbol (e.g. "b") is read. The automaton accepts the input string if the counter is zero at the end of the input.

Finite automata must be extended with a counter variable because a single stack symbol is not powerful enough to recognize S-CFLs. The counter provides a simple form of memory that allows the automaton to keep track of the number of occurrences of a specific symbol.

The operational semantics of the machine type we designed are as follows: The automaton starts with the counter set to 0. As it reads the input string, it increments the counter each time it encounters a specific symbol and decrements it each time it encounters a different symbol. If the counter is zero at the end of the input, the automaton accepts the input string. Otherwise, it rejects the input.

5. The class of S-CFLs is not closed under complementation. We can use the example of $a^n b^n (L_1)$ from the first part. We know the L_1 is an S-CFL from the first part. Now we need to check if its complement is also an S-CFL. Its complement is a language that contains a and b but is not in the form of $a^n b^n$. Can we define this language with an S-CFL? It is almost impossible to recognize a S-CFL that accepts everything except $a^n b^n$ with a single stack symbol because we would need to accept many different cases (similar to the L_2 example from the first part) and we cannot do that with a single symbol. Since the complement of L_1 is not an S-CFL, we can say that S-CFL is not closed under complementation.