Middle East Technical University Department of Computer Engineering

CENG 280

Formal Languages and Abstract Machines

Spring 2022-2023

Homework 3

Name Surname:Doruk Berke Yurtsizoglu
Student ID: 2522225

Answer for Q1

a. At first, we will create two sections for our states. First one will consists of non-final states, and the other one will only include the final states of our automata.
– $G_1 = (q_0, q_1, q_3, q_4)$
– $G_2 = (q_2, q_5)$

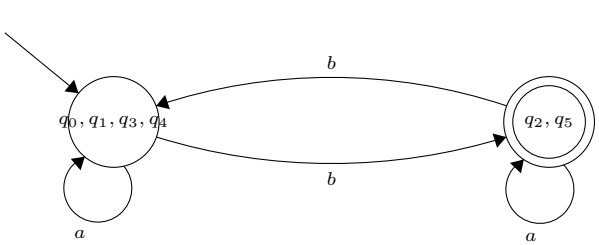
Now, we will iterate within these groups and inspect their transitions to other states. The rule when iterating is, if you find a different type of transitions for a state, create a new group for that state and so on. We will stop if we find the same groups from the previous iteration.

State	a	b
q_0	q_1	q_2
q_1	q_1	q_2
q_2	q_2	q_3
q_3	q_4	q_5
q_4	q_0	q_2
q_5	q_2	q_3

Let's interpret the relation table:
– For q_0, it goes to a non-final state when reading 'a', and to a final state when reading 'b'.
– For q_1, it goes to a non-final state when reading 'a', and to a final state when reading 'b'.
– For q_2, it goes to a final state when reading 'a', and to a non-final state when reading 'b'.
– For q_3, it goes to a non-final state when reading 'a', and to a final state when reading 'b'.
– For q_4, it goes to a non-final state when reading 'a', and to a final state when reading 'b'.
– For q_5, it goes to a final state when reading 'a', and to a non-final state when reading 'b'.

If we classify our states according to the informations above, we will get:
– $C_1 = (q_0, q_1, q_3, q_4)$
– $C_2 = (q_2, q_5)$

We can observe that the groups G_1-C_1 and G_2-C_2 are identical. This means, we need to stop iterating. We have reach the minimized version of the automata which looks like:



b. There are two equivalence classes for this automata:
1-) Strings that have even number of b's
2-) Strings that have odd number of b's

Let's define these equivalence classes with regular expressions:
(e) = $(a^* \cup a^*ba^*ba^*(ba^*ba^*)^*) = Lb$
(b) = $(a^*ba^*(ba^*ba^*)^*) = L$
Note: L = Strings that have odd number of b's.

c. .
Let string1 $(s_1) = a^n b^m c^k$
Let string2 $(s_2) = a^x b^y c^z$ where $n \neq x, m \neq y, k \neq z$.
Let string3 $(s_3) = d^u$ where u satisfies $m + n = k + 2u$.

If we concatenate s_1 and s_3 ($s_1 s_3$) we get strings in the format of $a^n b^m c^k d^u$ which are in our language L'.
However if we do the same thing with s_2 and s_3 ($s_2 s_3$), the new string can't be in L' since $n \neq x, m \neq y, k \neq z$.

Therefore, we can conclude that $\approx_L$ has infinetly many equivalence classes and hence by MyHill-Nerode Theorem L' is not regular.

Answer for Q2

1. G = (V, $\sum$, R, S) where
$\sum$ = (a,b), V = (S, A, X)$\cup \sum$

R =
S $\rightarrow$ Ab | XS | SXA
A $\rightarrow$ Ab | e
X $\rightarrow$ e | XX | aXb | bXa

A variable is equal to b^* which makes the strings of this grammar unbalanced.
X variable irepresents the balanced parts of the strings.
S determines where the unbalanced part is. It can be at the beginning, in the middle or at the end.

2. G = (V, $\sum$, R, S) where
$\sum$ = (0,1,2), V = (S, A, X)$\cup \sum$

R =
S $\rightarrow$ AX
A $\rightarrow$ 0A1 | e

X → 1X2 | e

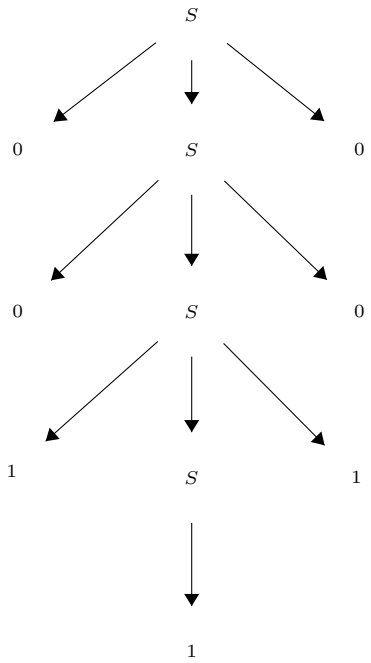
Separate the desired string ($0^i1^j2^k$) into two parts:
– 0^i1^i
– 1^k2^k
This way, we will have i + k amount of 1's in our strings.

A variable is equal to 0^i1^i.
X variable is equal to 1^k2^k.
S variable concatenates A and X.

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

R =
S → 0 | 1 | 0S0 | 0S1 | 1S0 | 1S1

Base cases are just 0 or 1. We will construct other strings from these base cases.
Strings can start with 0 and end with 1, or start with 0 and end with 0, or start with 1 and end with 1, or start with 1 and end with 0.



Answer for Q3

1. Strings that start and end with either 1 or 0.

2. Strings that contain at least two 1's.