CS - 275

Automata Theory and Formal Languages

Coursework 2 - James Smith 863266

1)

a) $L = \{ a^{2n}ba^{3n} \mid n \geq 0 \}$

aba

aabaaa

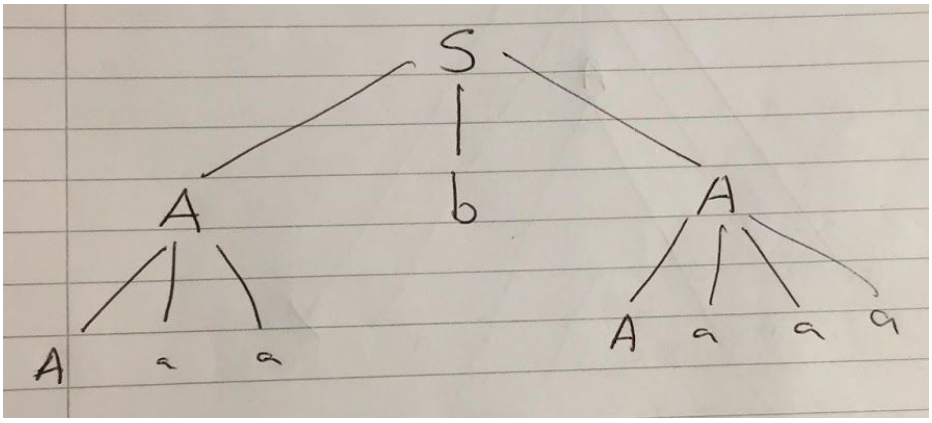
aaaabaaaaaa

S -> AbA

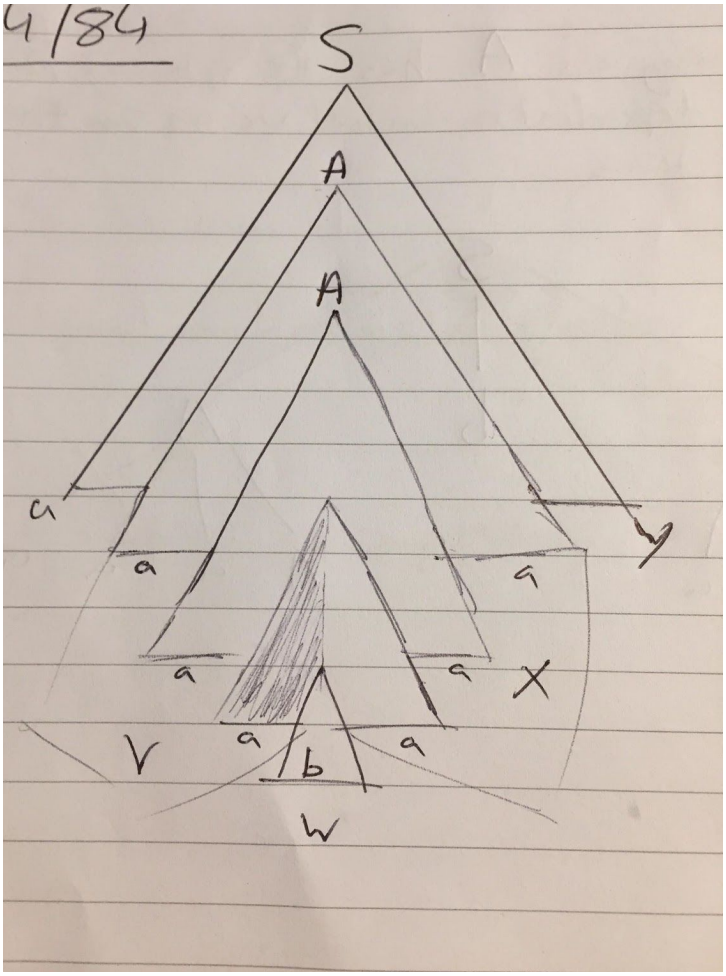
A -> Aaa, aaaA

A -> aa, aaa

Seeing as A has the option of being either aa or aaa dependant on the context we can see this language is context free




b) Pumping Lemma for regular languages states that if a string derived by an FSA is sufficiently big, we pass through at least one state twice.
If we take L as our context free language, then there exists a constant K s.t. for all strings z of L s.t. $|z| \geq k$ there exist u, v, w, x, y s.t.
$Z = uvwxy$, $|v\,w\,x| \leq k$, $|vx| \geq 1$, $\forall i \geq 0.u\ v^i\ w\ x^i\ y \in L$.
If we derive a tree from this language we are given this.




Which shows the values of V and X have a different quantity of subderivations. (2 for V and 3 for X) therefor the language is not regular.

2) f: N^2 -> N, f( x, y ) := 3( y - x )

First we need to compute Y - X = Z
Then Z * 3 = Output

For Y- X = Z (where X, Y & Z denote registers)
And A dot minus B := max{0 , a-b}
Z := Y - X

```
Z := Y
aux := X
While aux != 0 do{
aux := aux - 1
X := x - 1 }
```

For *3 while X != 0
X = N Y = 0
X = X - 1
Y = Y + 3

Combine these to form 3( y - x )

```
R0 = X,  R1 = Y,  R2 = Z, R3 = 0

0: if R0 = 0 goto 5
1:   R0 := R0 - 1
2:   R1 := R1 - 1
3: if R3 = 0 goto 0
4: if R1 = 0 goto 10
5:   R1 := R1 - 1
6:   R2 := R2 +1
7:   R2 := R2 +1
8:   R2 := R2+1
9: if R3 = 0 goto 5
```

3) To derive the function from the code:

```
0: if R0 = 0 goto 5
1:   R0 := R0 - 1
2:   R1 := R1 + 1
3:   R1 := R1 + 1
4: if R2 = 0 goto 0
5: if R1 = 0 goto 10
6:   R1 := R1 - 1
7:   R0 := R0 + 1
8:   R0 := R0 + 1
9: if R3 = 0 goto 5
```

We first record the results given when we use a set of natural inputs

For R0 = 1, R1 = 2, R2 = 0

| Instructions | R0 | R1 | R2 |
|---|---|---|---|
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 2 | 0 | 2 | 0 |
| 3 | 0 | 3 | 0 |
| 4 | 0 | 3 | 0 |
| 5 | 0 | 3 | 0 |
| 0 | 0 | 3 | 0 |
| 5 | 0 | 3 | 0 |
| 6 | 0 | 2 | 0 |
| 7 | 1 | 2 | 0 |
| 8 | 2 | 2 | 0 |
| 9 | 2 | 2 | 0 |
| 5 | 2 | 2 | 0 |
| 6 | 2 | 1 | 0 |
| 7 | 3 | 1 | 0 |
| 8 | 4 | 1 | 0 |
| 9 | 4 | 1 | 0 |
| 5 | 4 | 1 | 0 |
| 6 | 4 | 0 | 0 |
| 7 | 5 | 0 | 0 |
| 8 | 6 | 0 | 0 |
| 9 | 6 | 0 | 0 |
| 5 | 6 | 0 | 0 |

This gives us the output of `( 1, 1) = 6`

Following this through with other values we get;

```
( 1, 2) = 8
( 1, 3) = 10
( 2, 1) = 10
( 2, 2) = 12
```

The most useful output being;

```
( 0, 2) = 4
( 2, 0) = 8
```

Solving these as a graph and using the X and Y intercepts we find the original equation to be

```
f:N^2 -> N, f(x, y):+ (4X + 2Y)
```

4) i)

A Turing machine is a hypothetical machine thought up by Alan Turing used to simulate the outputs of certain algorithms. It consists of an infinitely long tape which acts like memory inside a computer. The tape is split up into boxes containing either 1, 0 or " " (blank). This is a typical three symbol Turing machine.
The machine moves along the tape reading one symbol at a time. It can edit this symbol by writing an new one or erasing it entirely. The machine can also move itself left or right to edit a neighbouring square.

4)ii)

A Turing machine consists of the 5 components $\Sigma$, S, I, [ ], $S_0$. These 5 symbols make up the language that defines all the functions possible in the machine. The machine follows the instructions on the form

s ----a/b,D-----> t
Where the symbols a, b and D translate too; if the head reads a, change it to b, D is the direction the head moves after the state has been changed.

5) i)

The head reads the first symbols, as they are all either 1 or 0 it moves to the right, once it reaches an empty tile it changes it to a 1 and moves left to state q1, moving on from state q1 it writes 1 if it is not empty. In state q2 if it reads either a 1 or 0 it changes it to a 0. Once it reaches a space it moves right to the final finishing state.

ii)