Indian Institute of Technology Kanpur
CS345: Theory of Computation, 17–18

Student Name: Gurpreet Singh
Roll Number: 150259
Date: October 24, 2017

ASSIGNMENT 1

Question 1

Part 1

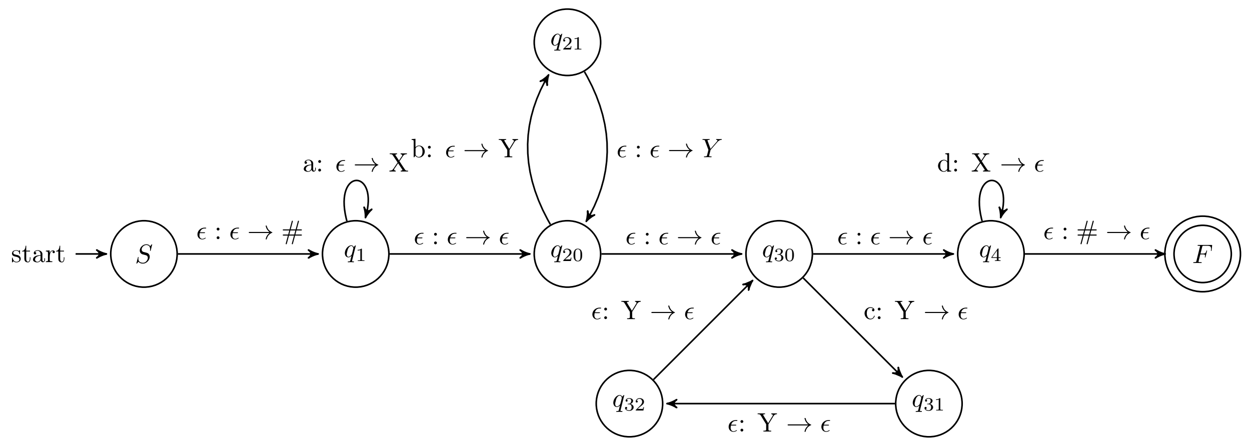
start
S
$\epsilon : \epsilon \to \#$
q_1
a: $\epsilon \to$ X
$\epsilon : \epsilon \to \epsilon$
b: $\epsilon \to$ Y
q_{21}
$\epsilon : \epsilon \to Y$
q_{20}
$\epsilon : \epsilon \to \epsilon$
q_{30}
$\epsilon : \epsilon \to \epsilon$
q_4
d: X $\to \epsilon$
$\epsilon : \# \to \epsilon$
F
ϵ: Y $\to \epsilon$
c: Y $\to \epsilon$
q_{32}
q_{31}
ϵ: Y $\to \epsilon$

Figure 1: PDA for L_1

Part 2

Reference: Stack Overflow (Referred to the CFG only — proved myself)

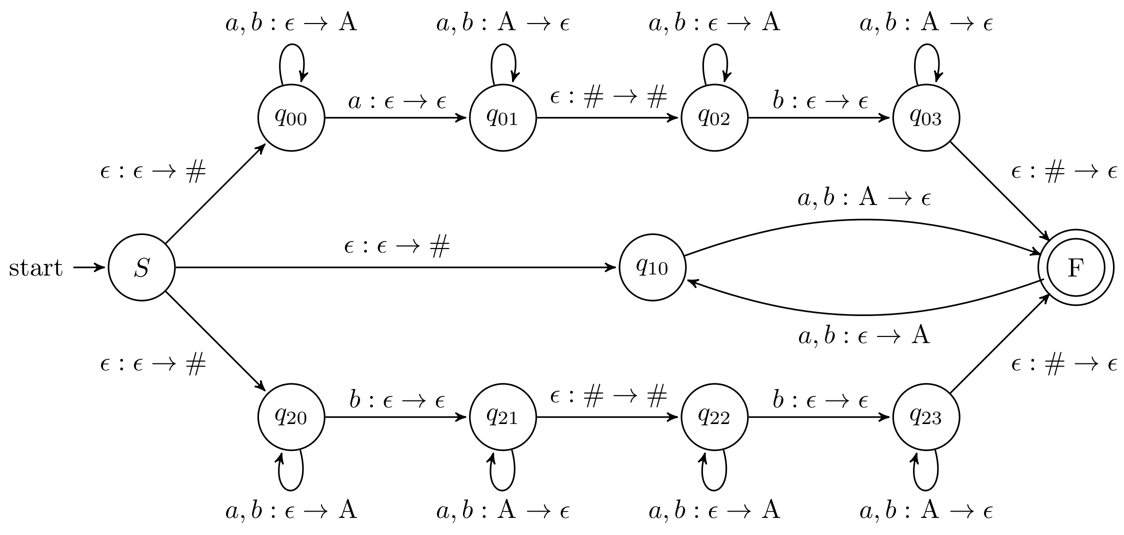
$a, b : \epsilon \to$ A
$a, b :$ A $\to \epsilon$
$a, b : \epsilon \to$ A
$a, b :$ A $\to \epsilon$
q_{00}
$a : \epsilon \to \epsilon$
q_{01}
$\epsilon : \# \to \#$
q_{02}
$b : \epsilon \to \epsilon$
q_{03}
$\epsilon : \epsilon \to \#$
$\epsilon : \# \to \epsilon$
$a, b :$ A $\to \epsilon$
start
S
$\epsilon : \epsilon \to \#$
q_{10}
F
$a, b : \epsilon \to$ A
$\epsilon : \epsilon \to \#$
$\epsilon : \# \to \epsilon$
q_{20}
$b : \epsilon \to \epsilon$
q_{21}
$\epsilon : \# \to \#$
q_{22}
$b : \epsilon \to \epsilon$
q_{23}
$a, b : \epsilon \to$ A
$a, b :$ A $\to \epsilon$
$a, b : \epsilon \to$ A
$a, b :$ A $\to \epsilon$

Figure 2: PDA for L_1

Question 2

$$\begin{aligned} L &= \{a^i b^j c^k d^l \mid i = k \text{ and } j = 2l\} \\ M &= \{a^i b^j c^k d^l \mid i = k \text{ or } j = 2l\} \end{aligned}$$

Part 1

M is context free wheras L is not context free.

Part 2

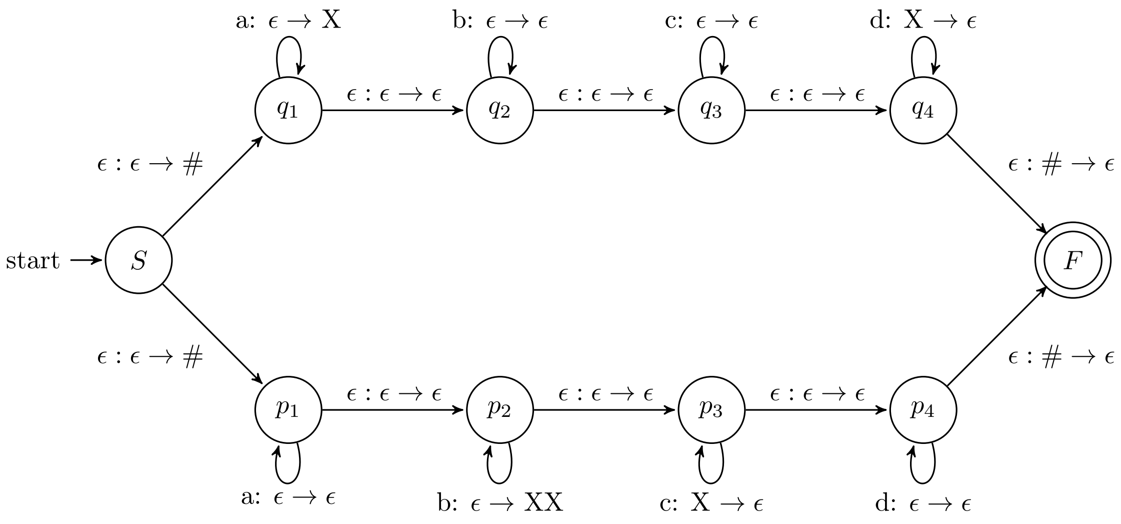

Figure 3: PDA for M

Part 3

We need to prove that L is not context free. We prove this using the contrapositive statement of the pumping lemma.

Consider that L is context free, and hence it must satisfy pumping lemma for CFLs. Given p, we choose a string $w \in L$ such that $w = a^p b^{2p} c^p d^p$.

Consider all partions of the string w as $uvxyz$, such that $|vxy| \leq p$ and $|vy| > 0$. The string vy, thus, can be represented as $a^{t_a} b^{t_b} c^{t_c} d^{t_d}$. For all such partitions, we need to find an $i \geq 0$ such that $uv^i xy^i z \notin L(L)$. Since $|vy| \leq p$, only the following cases are possible

Case 1 $t_a + t_b \geq 0$ *and* $t_c = 0 = t_d$

> For this case, if we choose $i = 0$, and if $t_a > 0$, then the constraint $i = k$ will not hold (**Note:** This i is different and is only to represent the constraint), or if $t_b > 0$, the constraint $j = 2l$ will no longer hold.
>
> Hence, for this case, we can say that $uxz \notin L(L)$.

Case 2 $t_b + t_c \geq 0$ *and* $t_a = 0 = t_d$

For this case, if we choose $i = 0$, and if $t_b > 0$, then the constraint $j = 2l$ will not hold. In the same way, if $t_c > 0$, then the constraint $i = k$ will not hold.

Hence, for this case too, we can say that $uxz \notin L(L)$.

CASE 3 *$t_c + t_d \geq 0$ and $t_a = 0 = t_b$*

For this case, if we choose $i = 0$, and if $t_c > 0$, then the constraint $i = k$ will break, or if $t_d > 0$, then the constrain $j = 2l$ will no longer hold.

Hence, for this case as well, we can say that $uxz \notin L(L)$.

Therefore, for any partition following the above constraints, choosing $i = 0$, we can find a string that is not in the language defined by L. Hence, using pumping lemma, we can say that L is not context free.

Question 3

In order to prove the decidability of a language, we need to construct a halting turing machine, M, for the language *i.e.* M either *accepts* or *rejects.*

Part 1

$$L_1 = \left\{ \langle M \rangle \;\middle|\; M \text{ is a DFA which does not accept any string that contains 101 as a substring} \right\}$$

We construct a turing machine M_1 which accepts the language L_1.

Description of the Turing Machine M_1

Input: A DFA D

1. List all the reachable states from the start state in the DFA D in the tape
2. Starting from all these states (reachable states), simulate the DFA on the string '101'
3. Starting from all the corresponding states (after simulation) one by one, check if any of the accepted states is reachable. If an accepted state is reachable for any state, *reject*
4. If no accepted state is reachable from these states, *accept*

Since all steps are feasible in closed time, we say that the turing machine M_1 is a halting turing machine.

CLAIM *The machine M_1 accepts the language L_1*
PROOF

Firstly we prove that if a DFA D accepts a string $w = w_1 101 w_2$, then the machine M_1 rejects this DFA.

Let $p \in Q(D)$ such that $p = \delta(s, w_1)$ *i.e.* p is the state reached after simulating the string w on the DFA D. Since we are reaching p after simulating a string, clearly, p is reachable from the start state, and hence will be included in the tape after the first step.

As in the second step, we replace this state by another state q such that $q = \delta(p, 101)$. Now since D accepts the string w, then the state $\delta(q, w_2)$ must be an accept state. Therefore, we can reach an accept state from the state q. As per the 3rd step, we reject the DFA D.

Following from the above case, if the DFA D does not accept any string $w = w_1 101 w_2$, then it is impossible to find a path from q which ends at an accepted state. Hence, the turing machine will not accept in this case. Also, since the number of paths are finite, the machine will also halt.

> Hence, we can say that the turing machine M_1 accepts the language L_1.

Since M_1 both accepts L_1 and is halting, we can say that L_1 is finite.

Part 2

$$L_2 = \{\langle R, S\rangle \mid \text{R, S are regular expressions and } L(R) \subseteq L(S)\}$$

For this question we use the following concepts

- Every regular language has a corresponding DFA
- If L_1 and L_2 are regular languages, then we can construct a DFA in closed time for the language $L_1 \cup L_2$ (Discussed in class)
- If $L_1 \subseteq L_2$, then $L_1 \cup L_2 = L_2$

We construct a turing machine M_2 which accepts the language L_2.

Description of the Turing Machine M_2

Input: Two regular expressions R, S

1. Construct a DFA D_R for the regualar expression R and a DFA D_S for the regular expression S
2. Using the DFAs D_R and D_S, construct a DFA D for the language $L(R) \cup L(S)$
3. Using the EQ_{DFA} algorithm discussed in class, *accept* if $EQ_{DFA}(D, D_S)$ accepts, *reject* otherwise

From the construction of the turing machine M, we can say that it is halting as all the steps are halting. Also, we are accepting if and only if $D = D_S$ *i.e.* iff $R \subseteq S$. Therefore the turing machine M_2 is a halting turing machine which accepts the language L. Thus, we can say that L_2 is decidable.

Question 4

$$L = \{\langle G\rangle \mid G \text{ is a CFG over } \{0,1\}^* \text{ and } 1^* \subseteq L(G)\}$$

This is similar to question 3b. We again use the following concepts for this question

- For every CFG, we have a corresponding PDA
- Intersection is closed between CFLs and Regular Languages *i.e.* $L(G) \cap L(D)$ is a CFL (where G is a CFG and D is a DFA)
- For a regular language, we can create a DPDA
- If $1^* \subseteq G$, then the DPDA for G will accept all strings 1^*, and hence every state obtained after simulating 1^a for some $a \geq 0$ should be an accept state

We can now construct a turing machine M which accepts the language L

Description of the Turing Machine M

Input: A CFG G

1. Check if G is a CFG, if not, *reject*
2. Construct a PDA P for G and a DFA D for 1^*

3. Using P and D, construct a PDA for 1^*
4. Convert P to a DPDAİf it is not possible, *reject*
5. Set $s = s_0$ (the start state in P)
6. do while (s hasn't been visited)

```
if ($s$ is not an accept state)
    *reject* $G$
else
    $s = \delta(s, 1)$
    VISITED $(s)$ = TRUE
```

7. If not rejected yet, *accept* G

From the construction of the turing machine M, we can say that it is halting as all the steps are halting.

If the DPDA cannot be constructed, we can safely assume that the language formed after intersection is not a regular language, and hence it cannot be 1^*. Therefore we simply reject.

Also, we are accepting if and only if every state reached on simulating $1*$ on the DPDA is an accept state. If this condition is not true, then we can find a string 1^a such that 1^a is not accepted, and hence G shall not be accepted. Also, since P is a finite DPDA, if we keep on simulating the current state on the string 1, we will find a cycle. After this cycle, it is redundant to check for the above condition. Hence, we are done.

Therefore we are accepting iff $1^* \subseteq L(G)$. Therefore the turing machine M is a halting turing machine which accepts the language L. Therefore, this suggests that L is decidable.