Theory of Computing

CSE – 203

Pumping Lemma for CFG, DFA minimization, Turing Machine

Pumping Lemma

If **L** is a context-free language, there is a pumping length **p** such that any string **$w \in L$** of length **$\geq p$** can be written as **$w = uvxyz$**, where **$vy \neq \varepsilon$**, **$|vxy| \leq p$**, and for all **$i \geq 0$, $uv^ixy^iz \in L$**.

Applications of Pumping Lemma

Pumping lemma is used to check whether a grammar is context free or not.

Problem

Find out whether the language **$L = \{x^ny^nz^n \mid n \geq 1\}$** is context free or not.

Solution

Let **L** is context free. Then, **L** must satisfy pumping lemma.

At first, choose a number **n** of the pumping lemma. Then, take z as $0^n1^n2^n$.

Break **z** into **uvwxy,** where

$|vwx| \leq n$ and $vx \neq \varepsilon$.

Hence **vwx** cannot involve both 0s and 2s, since the last 0 and the first 2 are at least (n+1) positions apart. There are two cases −

Case 1 − **vwx** has no 2s. Then **vx** has only 0s and 1s. Then **uwy**, which would have to be in **L**, has **n** 2s, but fewer than **n** 0s or 1s.

Case 2 − **vwx** has no 0s.

Here contradiction occurs.

Hence, **L** is not a context-free language.

Problem 1. Show that the following language on $\Sigma = \{a, b, c\}$ is not context-free.

$L = \{a^nb^jc^k : k = jn\}$.

Solution

Assume that $L = \{a^n b^j c^k : k = jn\}$ is a context free language.

Let $w = a^m b^m c^{m^2}$, $w \in L$

By the Pumping Lemma w can be decomposed as w = uvxyz with $|vxz| \leq m$

and $|vy| \geq 1$ such that $uv^i xy^i z \in L, i \geq 0$

case 1

$$\underbrace{aaa\ldots a}_{uvxy}\underbrace{ab\cdots bc\cdots c}_{z}$$

If i=0, $uv^0 xy^0 z = a^{m-|vy|} b^m c^{m^2} \notin L$

case 2

$$\underbrace{aaa\ldots a}_{uv}\underbrace{ab}_{x}\underbrace{\ldots}_{y}\underbrace{bc\ldots}_{z}$$

If i=0, $uv^0 xy^0 z = a^{m-|v|} b^{m-|y|} c^{m^2} \notin L$

case 3

$$\underbrace{aaa\ldots ab}_{u}\underbrace{b\ldots b}_{vxy}\underbrace{bc\ldots c}_{z}$$

If i=0, $uv^0 xy^0 z = a^m b^{m-|vy|} c^{m^2} \notin L$

case 4

$$\underbrace{aaa\ldots ab}_{u}\underbrace{b\ldots b}_{v}\underbrace{bc}_{x}\underbrace{c\ldots c}_{yz}$$

If i=0, $uv^0xy^0z = a^m b^{m-|v|} c^{m^2-|y|} \notin L$

case 5

$$\underbrace{aaa\ldots ab\ldots bc}_{u}\underbrace{c\ldots c}_{vxyz}$$

If i=0, $uv^0xy^0z = a^m b^m c^{m^2-|vy|} \notin L$

case 6 v or y containing ab or bc

If I > 0, uv^ixy^iz would be $a\ldots ab\ldots ba\ldots ab\ldots b\ldots c\ldots c$

Or $a\ldots ab\ldots bc\ldots cb\ldots c$

Contrary to the assumption.

Language is not context free.

DFA Minimization using Equivalence Theorem

If X and Y are two states in a DFA, we can combine these two states into {X, Y} if they are not distinguishable. Two states are distinguishable, if there is at least one string S, such that one of δ (X, S) and δ (Y, S) is accepting and another is not accepting. Hence, a DFA is minimal if and only if all the states are distinguishable.

Algorithm

Step 1 − All the states **Q** are divided in two partitions − **final states** and **non-final states** and are denoted by $\mathbf{P_0}$. All the states in a partition are 0th equivalent. Take a counter **k** and initialize it with 0.

Step 2 − Increment k by 1. For each partition in P_k, divide the states in P_k into two partitions if they are k-distinguishable. Two states within this partition X and Y are k-distinguishable if there is an input **S** such that **δ(X, S)** and **δ(Y, S)** are (k-1)-distinguishable.

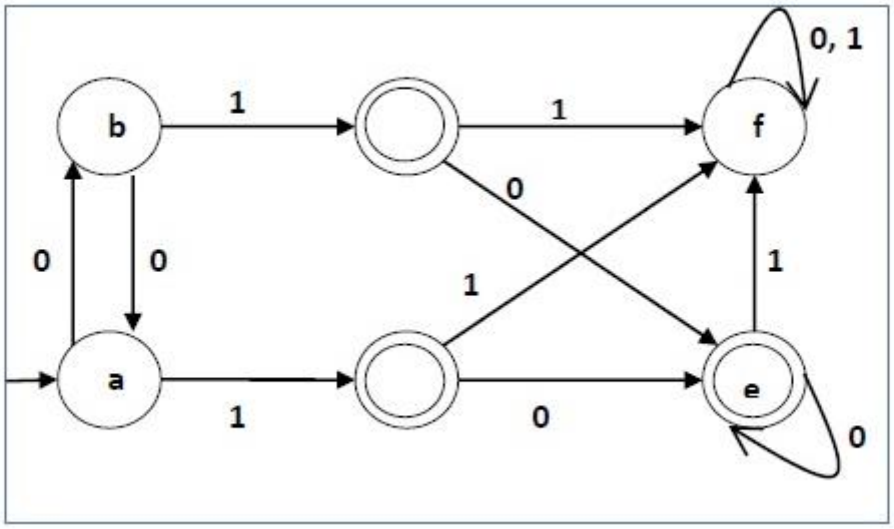
Step 3 − If $P_k \neq P_{k-1}$, repeat Step 2, otherwise go to Step 4.

Step 4 − Combine kth equivalent sets and make them the new states of the reduced DFA.

Example

Let us consider the following DFA –

q	δ(q,0)	δ(q,1)
a	b	c
b	a	d
c	e	f
d	e	f
e	e	f
f	f	f



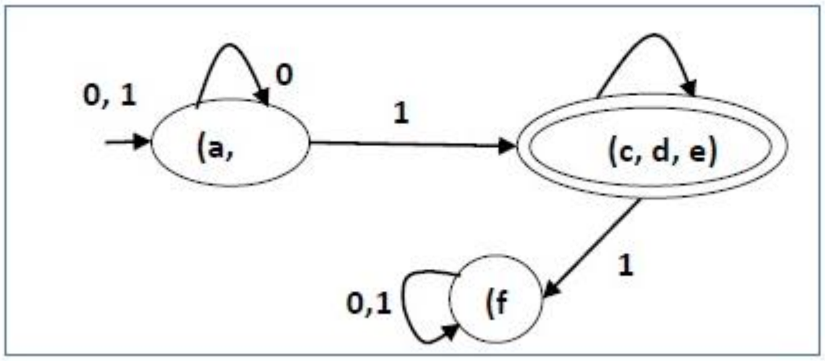
Let us apply the above algorithm to the above DFA –

- P_0 = {(c,d,e), (a,b,f)}
- P_1 = {(c,d,e), (a,b),(f)}
- P_2 = {(c,d,e), (a,b),(f)}

Hence, $P_1 = P_2$.

There are three states in the reduced DFA. The reduced DFA is as follows –

Q	δ(q,0)	δ(q,1)
(a, b)	(a, b)	(c,d,e)
(c,d,e)	(c,d,e)	(f)
(f)	(f)	(f)



Turing Machine

A Turing Machine is an accepting device which accepts the languages (recursively enumerable set) generated by type 0 grammars. It was invented in 1936 by Alan Turing.

A Turing Machine (TM) is a mathematical model which consists of an infinite length tape divided into cells on which input is given. It consists of a head which reads the input tape. A state register stores the state of the Turing machine. After reading an input symbol, it is replaced with another symbol, its internal state is changed, and it moves from one cell to the right or left. If the TM reaches the final state, the input string is accepted, otherwise rejected.

A TM can be formally described as a 7-tuple (Q, X, ∑, δ, q_0, B, F) where −

- **Q** is a finite set of states
- **X** is the tape alphabet
- **∑** is the input alphabet
- **δ** is a transition function; δ : Q × X → Q × X × {Left_shift, Right_shift}.
- **q_0** is the initial state
- **B** is the blank symbol
- **F** is the set of final states

Comparison with the previous automaton

The following table shows a comparison of how a Turing machine differs from Finite Automaton and Pushdown Automaton.

Machine	Stack Data Structure	Deterministic?
Finite Automaton	N.A	Yes
Pushdown Automaton	Last In First Out(LIFO)	No
Turing Machine	Infinite tape	Yes

Example of Turing machine
Turing machine M = (Q, X, ∑, δ, q_0, B, F) with

- Q = {q_0, q_1, q_2, q_f}
- X = {a, b}
- ∑ = {1}
- q_0 = {q_0}
- B = blank symbol
- F = {q_f }

δ is given by −

Tape alphabet symbol	Present State ‘q_0’	Present State ‘q_1’	Present State ‘q_2’
a	1Rq_1	1Lq_0	1Lq_f
b	1Lq_2	1Rq_1	1Rq_f

Here the transition 1Rq_1 implies that the write symbol is 1, the tape moves right, and the next state is q_1. Similarly, the transition 1Lq_2 implies that the write symbol is 1, the tape moves left, and the next state is q_2.

Time and Space Complexity of a Turing Machine

For a Turing machine, the time complexity refers to the measure of the number of times the tape moves when the machine is initialized for some input symbols and the space complexity is the number of cells of the tape written.

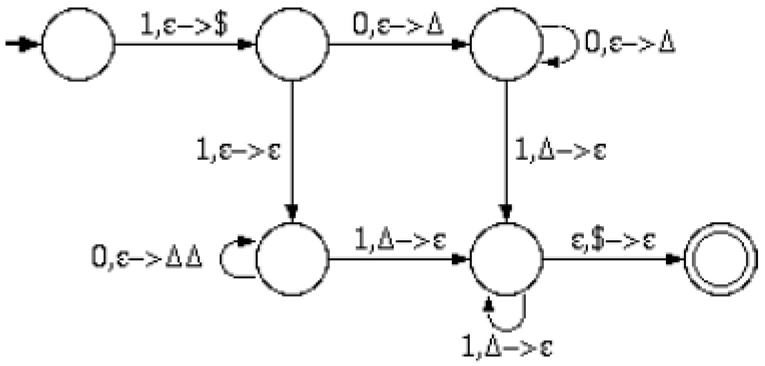
Time complexity all reasonable functions − **T(n) = O(n log n)**

TM's space complexity − **S(n) = O(n)**

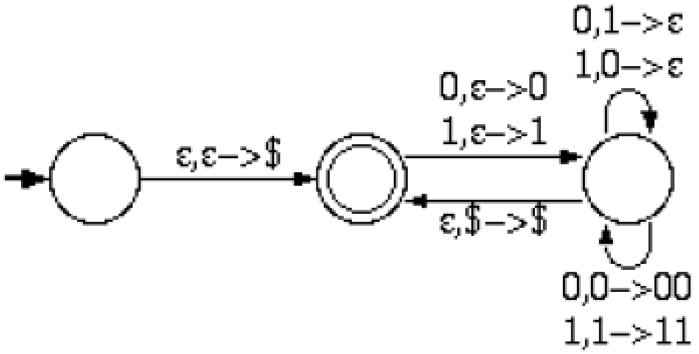
Solution to Previous Homework

Construct deterministic pushdown automata to accept the following languages.

a. $\{10^n1^n \mid n>0\} \cup \{110^n1^{2n} \mid n>0\}$



b. Binary strings that contain an equal number of 1s and 0s



Homework

1. Show that the following language on Σ = {a, b} is not context-free using pumping lemma $L = \{ ww^Ra^{|w|} : w \in \{a, b\}^*\}$

2. Construct a Turing machines that accepts the following languages on the alphabet {a, b}.
   a. $L = \{01^*0\}$
   b. $L = \{a^n b^n c^n \mid n \geq 1 \}$