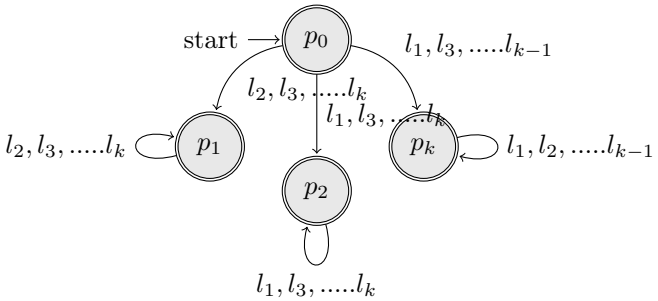
COL-352 ASSIGNMENT

DEEPANSHU YADAV(2019CS10343)
SOURAV(2019CS10404) HARIKESH(2019CS10355)

January 2022

1 Problem

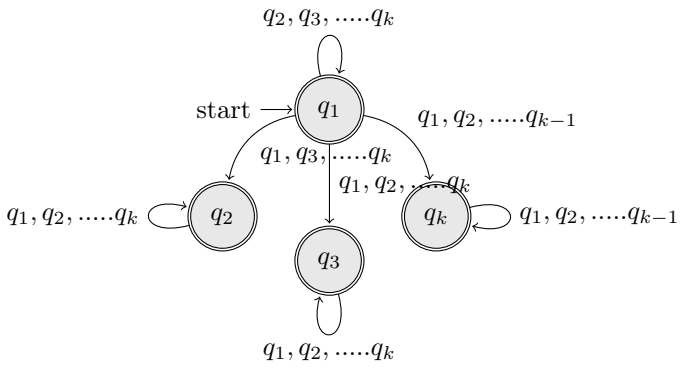


This NFA will only accept the strings that don't have all the character from Γ.
p_1 accepts all strings which don't have l_1 in it.
Similarly $\forall i$ such that $i \in$ {1,2,3,....,k}
p_i accepts all strings which don't have l_i in it.
Hence the NFA can be written as

$$\boxed{N : (Q, \Gamma, \delta, q_0, F)} \tag{1}$$

$$\delta(p_i, x) = \begin{cases} Q/(p_0 \cup p_j) & i = 0 \ \& \ x = l_j \\ p_i & i \neq 0 \ \& \ x \neq l_i \\ \phi & otherwise \end{cases}$$

Hence we get an NFA with k+1 states.



2 Problem

To Prove: All NFAs recognizes the class of regular languages.
(i) claim: All regular languages are accepted by some ALL-NFA.
Proof: Let L be a regular language hence there must be a DFA D which accepts L. i.e.

$$\boxed{L(D) = L(\text{By definition of regular languages})} \tag{2}$$

As every DFA is a All-NFA ($\because$ there can be only a single final state possible after reading input x for a DFA).
Hence, L is accepted by an all NFA D for all regular Languages L. Hence, our claim is true.
(ii) claim: Every ALL-NFA accepts some regular language.
Proof: Let,

$$\boxed{M = (Q, \Sigma, \delta, q_0, F)} \tag{3}$$

be an ALL-NFA.
We can convert M to a DFA

$$\boxed{N = (Q', \Sigma, \delta', q_0', F')} \tag{4}$$

such that

$$Q' = \text{Power Set of } Q$$

$$q_0' = \{q_0\}$$

$$\delta'(S, x) = \bigcup_{p \in S} \delta(p, x)$$

$$F' = \text{Set of all subsets of F}$$

$$F' = \{S \subseteq Q : S \subseteq F\}$$

Now as N is DFA hence it must accept some regular language.
Hence, for every All-NFA, we can find equivalent DFA which accepts regular language.
Hence, every ALL-NFA accepts some regular language.
As our **claim(i)** and **claim(ii)** are **true**, hence All-NFAs recognise the set of regular languages.

3 Problem

Suppose L is a regular language on Σ .
To Prove: Repeat(L) is also a regular language.
Proof: Let,

$$\boxed{D = (Q, \Sigma, \delta, q_0, F)} \tag{5}$$

be the DFA which accepts L.
To prove repeat(L) is a regular language, there must exist a DFA which accepts repeat(L).
We can find an NFA which accepts repeat(L) just by changing the transition function of D.

$$\boxed{N = (Q', \Sigma, \delta', q_0', F')} \tag{6}$$

will be the NFA which accepts repeat(L).
where,

$$Q' = Q \cup Q_1$$

where

$$Q_1 = \{q_{ix} | q_i \in Q' \quad \& x \quad \in \Sigma$$

$$q_0' = q_0$$

$$F' = F$$

If $q_i \in Q$ & $x \in \Sigma$

$$\delta'(q_i, x) = q_{ix}$$

If $q_{ix} \in Q_1$ & $x \in \Sigma$

$$\delta'(q_{ix}, y) = \begin{cases} \delta(q_i, x) & x = y \\ \phi & otherwise \end{cases}$$

Hence the NFA has transition to next state only if the current alphabet appears two times(for even number of times) otherwise it goes to a state from Q,and then there is no further transition possible hence can't repeat final state. Hence this kind of strings won't be accepted.
So, this NFA would accept only repeat(L).
we can find equivalent DFA which accepts repeat(L) from this NFA.
Hence repeat(L) is regular.
Hence regular languages are closed under repeat operation.

4 Problem

Given : DFA's

$$\boxed{D_1 = (Q, \Sigma, \delta, q_0, F)} \qquad \boxed{D_2 = (Q', \Sigma', \delta', q_0', F')} \tag{7}$$

Algorithm Intuition : To check whether these DFA's recognise same language or first of all we start by checking that the alphabet of these two DFA's have to be same otherwise they may have different words/strings(of different characters). Transition function and states(initial and final) may or may not be same. If everything is same then they are equal automata.
Our approach is that we will start from places where these two DFA's may have distinguishable states i.e basically one among the two states is accepting so this is basically first level of difference by the fact that ϵ is accepted by one and rejected by other in that pair of states. Let's say that pair is (p,q) where only one among these is accepting. Now we will move backwards and see all pair of unmarked points i.e indistinguishable points and for each input test it whether they point to distinguishable points in table or not. If yes then this pair is also distinguishable. This continuous till the distinguishable points are not increased. After the completion of this algorithm we check if the start states of two DFA's are distinguishable then these two DFA's are not equivalent otherwise they are equivalent .i.e recognise the same language.
Algorithm to tell if the DFA's D1 (Q, Σ, δ, q_0, F) and D2 (Q', Σ', δ', q_0', F') recognise the same language or not.

```
Def func (DFA D1, DFA d2):
    d1 ← states(D1) # number of states in d1
     d2 ← states(D2) # number of states in d2
     If (Σ ≠ Σ′) # both dfa's have different alphabets they are different dfa's
         Print ('They don't recognise same language')
     End if
     Table = Boolean Table|d1||d2| # a Boolean table of size (d1)(d2)
     Count ← 0 # stores number of distinguishable states
     For i←1 to d1: # iterate through the matix
         For j←1 to d2:
             If ((state i accepting or state j accepting)): # one of these is accepting
                  Table[i][j]←true # table is distinguishable on empty string
                 Count←count +1
             End if
             If (state i accepting and state j accepting) # both are accepting states
                 Table[i][j] ← false # not distinguishable
                 Count ← count -1
             End if
         End for
     End for
     While True: # iterate through a while loop
         Prev_count←count # make prev count = count
         For i←1 to d1:
             For j←1 to d2:
                 If table[i][j] = false: # all non-distinguishable states
                 For each symbol ∈ Σ: # for every sym in alphabet
                     If (δ(i, symbol)&δ′(j, symbol) = true):
                         Table[i][j] = true # distinguished
                         Count ← count +1 # count increased
                         Break for # distinguished atleast once
                     End if
                  End for # in this above loop we just check for each non distinguished state whether the next states are distinguishable by any symbol.
             End if
             End for
         End for
         If count = prev_count : # if no pair of state is updated break loop
             Break while
         End while
     If table[start state D1] [start state D2] = true: # if start state pair of both DFA's are distinguisged
         Print ('They don't recognise same language')
     End if
     Else:
         Print ('Both DFA's represent same language')
     End else # If start states are distinguished then we find a string accepted by one of them and rejected by other and hence we can say that they are different otherwise vice versa.
End func
```

5 Problem

Let, A be a regular language.
Let,

$$\boxed{D = (Q, \Sigma, \delta, q_0, F)} \tag{8}$$

be a DFA accepting the regular language A.
where,

$$\boxed{\delta : Q * \Sigma \to Q} \tag{9}$$

Let A^R be the reverse of language A.
To Prove: A^R is regular language.
Proof: Let, $w = w_1 w_2 .....w_n$, $w \in$ A be a string accepted by DFA D and $P_0, P_1, ..., P_n$ be the states such that

$$P_i \in Q$$

$$P_0 = q_0$$

$$\delta(P_i, w_{i+1}) = P_{i+1}$$

where i=0.....n

$$P_n \in F$$

We can create another NFA N which accepts w^R just by taking inverse of δ as the transition function and taking initial state of D as final state of N and final state of D as having ϵ transition from initial state q_{00} of N. i.e

$$\boxed{N = (Q', \Sigma, \delta', q_0', F')} \tag{10}$$

where,

$$Q' = Q \cup q_{00}$$

$$q_0' = q_{00}$$

$$\delta'(q_{00}, \epsilon) = F$$

$$\delta'(q_{00}, x) = \Phi \quad \forall x \in \Sigma$$

$$\delta'(P, x) = \{r | \delta(r, x) = P\} \quad \forall P \in Q, x \in \Sigma$$

$$F' = q_{00}$$

We can see that this NFA accepts $w^R = w_n w_{n-1} .....w_1$ as we would get the states just reverse as on running it on D.
Hence A^R is accepted by some NFA we can get equivalent DFA for that NFA
Hence A^R is accepted by some DFA.
Hence, A^R is a regular language.
Hence, regular languages are closed under reverse.

6 Problem

To Prove: The class of regular languages is closed under homomorphism where we have our string homomorphism function

$$f : \Sigma^* \rightarrow \Gamma^*$$

which is defined as below:

$$f(x) = \Sigma \quad when \quad x \in \Sigma$$

and

$$f(x.y) = f(x).f(y) \quad \forall x, y \in \Sigma^*$$

where Σ and Γ be two finite alphabets.
Proof: To prove that the class of regular languages is closed under homomorphism we have to basically prove that if any language L $\subseteq$ of Σ^* is a regular language then

$$f(L) = \{f(x) \in \Sigma^* \quad where \quad x \in L\}$$

is also a regular language.
We will start by representing the regular language in terms of regular expressions. Let's assume that R is the regex accepting L. Then we may define our regex R_f inductively as

$$R_h = \phi \quad when \quad R = \phi$$

$$R_f = \Sigma \quad when \quad R = \Sigma$$

$$R_f = f(a) \quad when \quad R = a$$

$$R_f = A_f + B_f \quad when \quad R = A + B$$

$$R_f = A_f.B_f \quad when \quad R = A.B$$

$$R_f = (A_f)^* \quad if \quad R = F^*$$

Claim: The regular expression defined above accepts the language $f(L)$.
Proof: To prove that the regular expression defined above accepts the language $f(L)$ we have to show that $L(R_f) = f(L(R))$. We will be proving this basically by induction on the number of operations in regular expression R.
Base case: When R = Σ or R = ϕ, from our regular expression R_f we know that R_f = R in both these cases. Hence $L(R_f)$ = L(R) and from the very similar logic of R_f = R in these cases our right hand side reduces to L(R). Hence for this case $L(R_f) = f(L(R))$.
For the case of R = a, we have L(R) = {a} and our $f(L(R)) = \{f(a)\}$ which in turn is equal to $L(f(a)) = L(R_f)$. Hence, the claim hold true for this case also.
Induction step: We will try the induction step on $\cup$ i.e. + operation first and for that we are having

$$R_f = A_f + B_f \quad for \quad R = A + B$$

Now, our RHS is

$$f(L(R)) = f(L(A) + L(B))$$

$$\Rightarrow f(L(R)) = f(L(A)) \cup f(L(B))$$

Now, taking the LHS we will see that

$$L(R_f) = L(f(A + B))$$

$$\Rightarrow L(R_f) = L(f(A) + f(B))$$

$$\Rightarrow L(R_f) = L(f(A)) + L(f(B))$$

$$\Rightarrow L(R_f) = f(L(A)) \cup f(L(B)) \quad \text{(from I.H)}$$

We will again try the induction step for concatenation case and for that we are having

$$R_f = A_f.B_f \quad for \quad R = A.B$$

Now lets take the RHS i.e.

$$f(L(R)) = f(L(A).L(B))$$

$$\Rightarrow f(L(R)) = f(L(A)).f(L(B))$$

Now, taking the LHS we will see that

$$L(R_f) = L(f(A.B))$$

$$\Rightarrow L(R_f) = L(f(A).f(B))$$

$$\Rightarrow L(R_f) = L(f(A)).L(f(B))$$

$$\Rightarrow L(R_f) = f(L(A)).f(L(B)) \quad \text{(from I.H)}$$

And finally, we will be doing this on the case of * operator and for that that we are having

$$R_f = (A_f)^* for R = A^*$$

Now let's take the RHS i.e.

$$f(L(R)) = f(L(A^*))$$

$$\Rightarrow f(L(R)) = f(L(A))$$

Now, taking the LHS we will see that

$$L(R_f) = L(f(A))$$

$$\Rightarrow L(R_f) = L(f(A))$$

$$\Rightarrow L(R_f) = f(L(A)) \quad \text{(from I.H)}$$

Hence, from inductive hypothesis all these pair of expressions are equal and we can say that the regular expression defined by us i.e. R_f accepts $f(L)$.
Hence, we can conclude that $f(L)$ is regular language.
Now for the next part of the problem we have to informally describe the process of getting an NFA for $f(L)$ by the DFA of L. for that Let's assume that we are having a DFA D of L. And Let

$$\boxed{D = (Q, \Sigma, \delta, q_0, F)} \tag{11}$$

and it accepts L. We can form an NFA N for accepting $f(L)$ which will be let's say

$$\boxed{N = (Q', \Gamma, \delta', q_0', F')} \tag{12}$$

where,

$$Q' = Q$$
$$q_0' = q$$
$$F' = F$$

And our transition function $\delta'(s, f(a)) = \delta(s, a)$ we can say that N will accept $f(L)$ because F' = F and our DFA D was given to be accepting L.