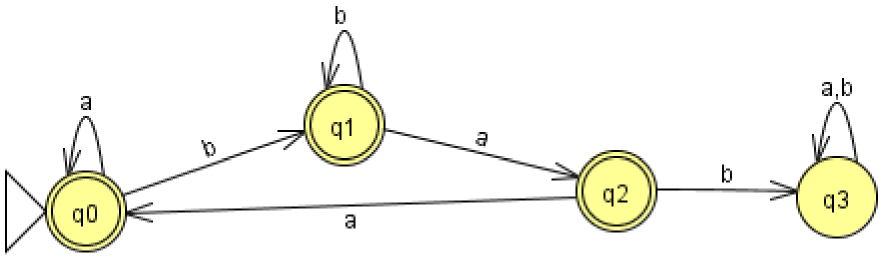
USN | | | | | | | | | |

Internal Assessment Test 2 – Dec 2021

| Sub: | Automata Theory and Computability | | | | | Sub Code: | 18CS54 | Branch: | CSE |
|---|---|---|---|---|---|---|---|---|---|
| Date: | 20/12/21 | Duration: | 90 mins | Max Marks: | 50 | Sem/Sec: | 5 A,B,C | | OBE |

Answer any FIVE FULL Questions — MARKS | CO | RBT

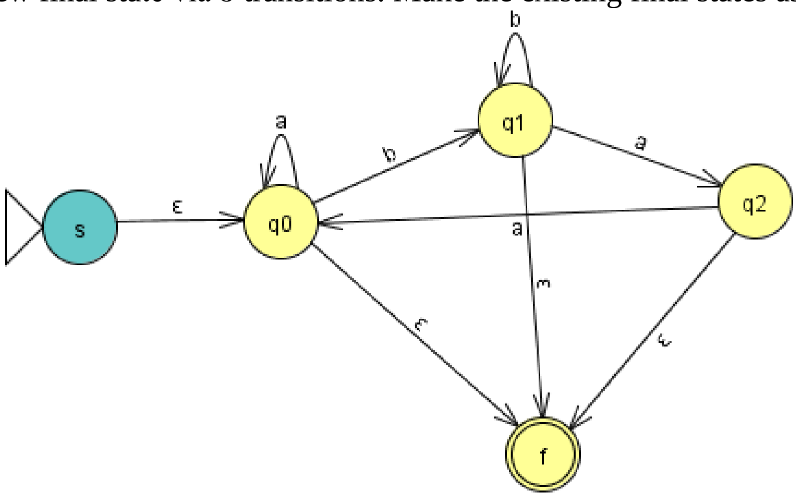
1 (a) Convert the following DFSM into a regular expression. Show steps. [6] CO2 L3




Remove q3 because it is a dead state

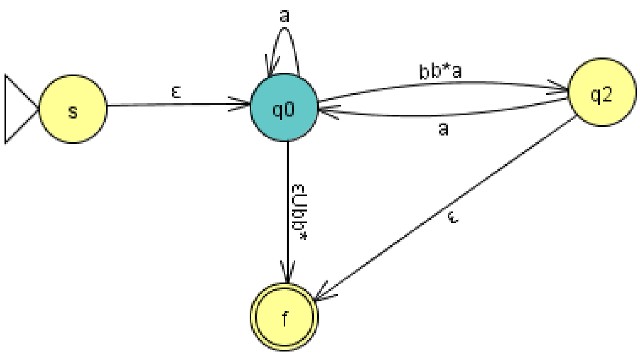
q0 is the start state and it has an incoming transition so create a new start state. Connect new start state to existing start state via ε – transition

There is more than one final state. So, create a new final state. Connect q0, q1 and q2 to new final state via ε-transitions. Make the existing final states as non-accepting.




Rip q1

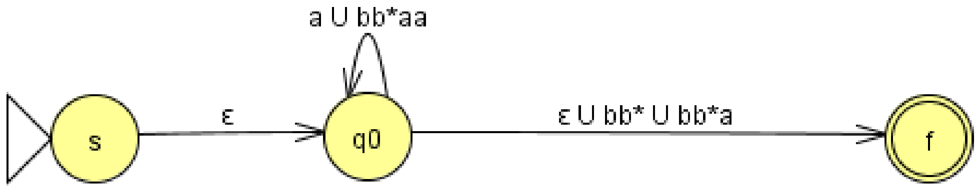
There is a path from q0-q1-p2, q0-a1-f. We will remove q1 and using the formula R(p,q) = R(p, rip) R(rip, rip)* R(rip, q), rewrite RE from q0 to q2 and q0 to f.



Rip q2

There is a path from q0 to f via q2. We rewrite that path as bb*a. Already there is a path from q0 to f. Hence bb*a will be taking a union with the existing RE.

There is also a path from q0 via q2 back to q0

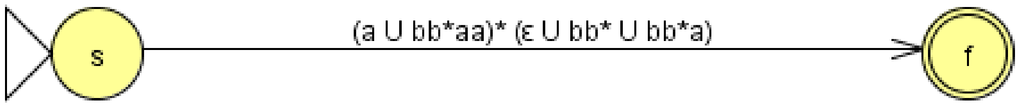



Rip q0

$R(s,f) = R(s, q0)\ R(q0,q0)^*\ R(q0,f)$

$= \varepsilon\ (a \cup bb^*aa)^*\ (\varepsilon \cup bb^* \cup bb^*a)$

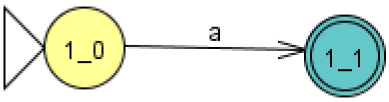
$= (a \cup bb^*aa)^*\ (\varepsilon \cup bb^* \cup bb^*a)$



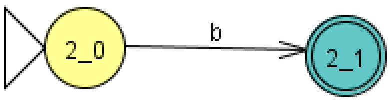

(b) Construct an NDFSM using Kleene's theorem for (a (b ∪ ε) b)*. Show steps. [4] CO2 L3

Construct FSM for primitive types

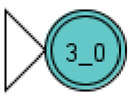
Let M1 accept L= {a}



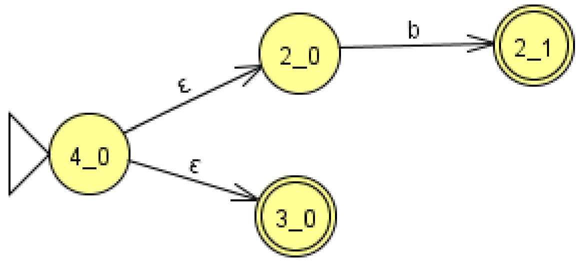

Let M2 accept L = {b}



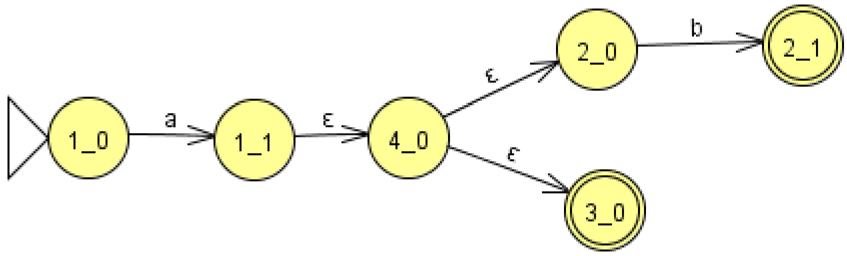

Let M3 accept L = {ε}



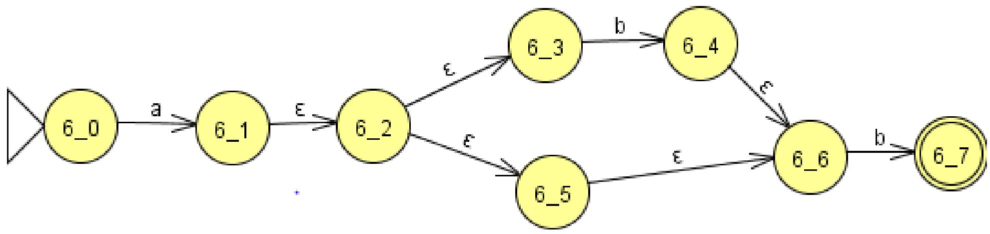

Construct M4 for (b ∪ ε). A new start state is created and connecting to existing start states to machines M2 and M3 that accept b and ε respectively. The accepting states of M2 and M3 are the accepting states of M4.



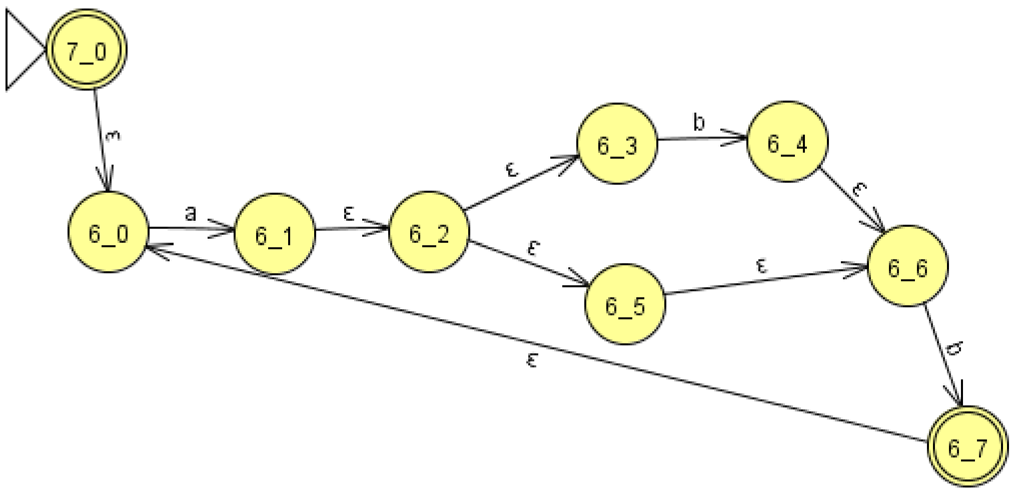

Construct M5 for a (b ∪ ε). To concatenate FSM M1 to M4, connect each final state of M1 to M4 via ε-transitions. Set start state of M5 to start state of M1. Set accepting state of M5 to accepting states of M4. Make the accepting states of M1 to non-accepting in M5.




Construct M6 for a (b ∪ ε)b following same steps as above for concatenation. Since state numbers are getting repeated, we will rename all states to 6_0, 6_1, ….




Construct M7 to accept Kleene closure of RE accepted by M6 (a (b ∪ ε)b)*. Create new start state and make it accepting. Connect to existing start sate of M6 via ε-transition. Connect each accepting state of M6 to start state of M6 via ε-transition.




| | | | | |
|---|---|---|---|---|
| 2 (a) | State and prove pumping lemma for regular languages. Prove that $a^nb^{2n}$: $n \geq 1$ is not | [05] | CO3 | L2 |

regular.
Solution:
For the language to be proved that it is regular, for any string of form w = xyz, 3 conditions must hold.

- $|xy| \leq k$, i.e. k-1 characters can be read without revisiting any states, but kth character must take DFSM M to a state it has visited before.
- $y \neq \varepsilon$ : Since M is deterministic, no transitions on ε
- $\forall q \geq 0\ (xy^qz \in L)$ : y can be pumped (q = 0 or q>1). The resulting string should be in L.

For a string $a^kb^{2k}$, |w| = 3k, So |w| is > k.
Since |xy| should be less than or equal to k, let $|xy| = a^k$
Since $y \neq \varepsilon$, let us assign x= ε and $y = a^k$
Now, let us pump y, 2 times.
We get $a^{k+2}b^{2k}$. Number of b's is no longer twice the number of a's, hence we prove that the language is not regular.
Let's take k=1

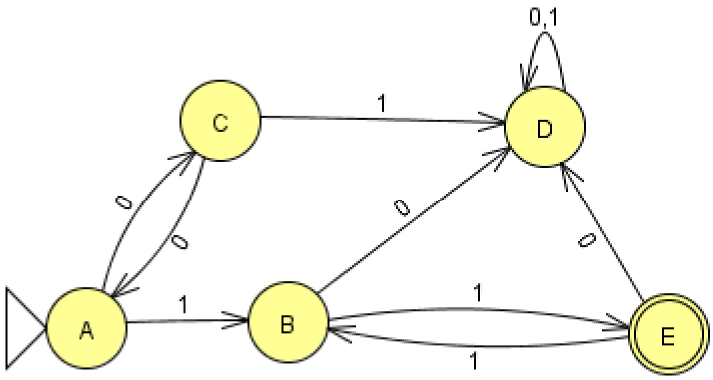
| | | |
|---|---|---|
| **ε** | **a** | **bb** |
| x | y | z |

After pumping y 2 times

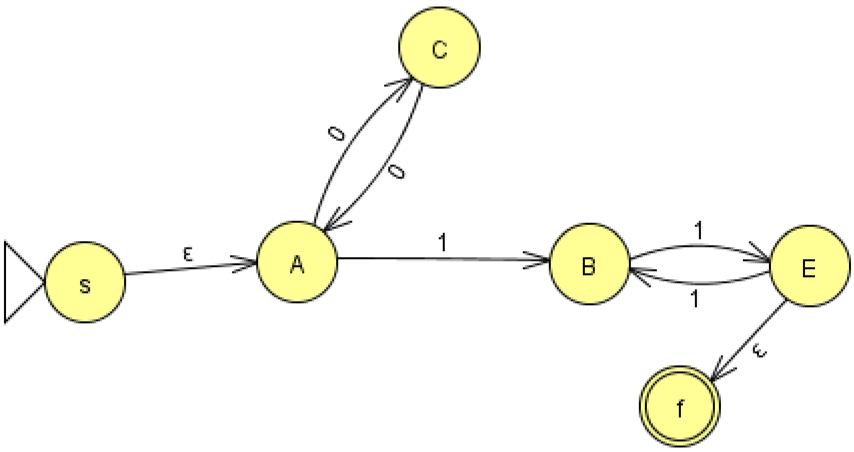
| | | |
|---|---|---|
| **ε** | **aa** | **bb** |
| x | y | z |

Since the string aabb $\notin L(a^nb^{2n})$

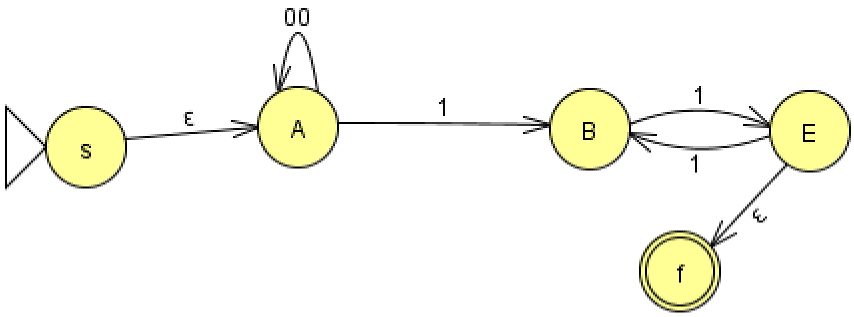
(b) Convert the following FSM into a regular expression. Show steps. [05] CO2 L3



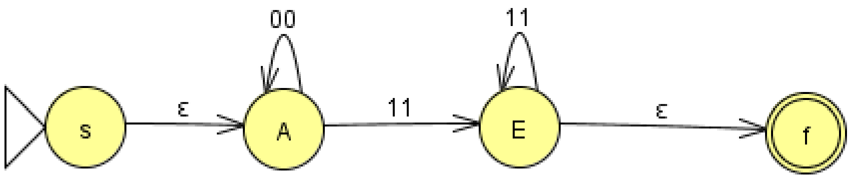

Ans: In the above DFSM, D is a dead state and can be removed. The starting state A has an incoming transition, so we will create a new start state and connect the new start to existing start state via ε-transition.
There is only one final state but it has an outgoing transition. So, we create a new accepting state and connect the existing final state, E, to the new accepting state via ε-transition. Make E as non-accepting.



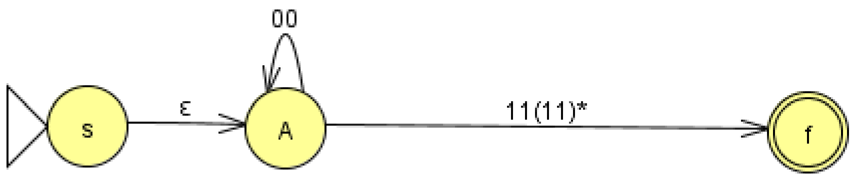

Rip C.
We rip C by replacing the transition from A to A via C.



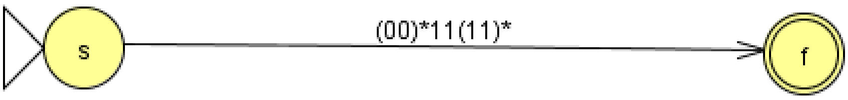

Rip B
There is a transition from A to E via B. There is also a transition from E to E via B.
We replace both.




Rip E
We use the formula to get regular expression from A to f
R(A,f) = R(A,E) R(E,E)*R(E,f) =11 (11)*ε = 11(11)*




Rip A
R(s,f) = R(s,A)R(A,A)*R(A,f) = ε.(00)* 11(11)* = .(00)* 11(11)*




3 (a) Define CFG. Write CFG for the following languages. Show derivation for given strings. [8] CO3 L3

(i) All strings over {a,b} that are even or odd palindromes., w = ababa, w=baab

(ii) $L = \{a^{2n}b^n : n \geq 0\}$, w = aaaabb

(iii) $L = \{w \in \{a,b\}^* : \#_a(w) = \#_b(w)\}$, w = abab

Context Free Language (CFG) is a quadruple (V, Σ, R, S) where

- V is the rule alphabet which contains non-terminals (symbols that are used in the grammar, but do not appear in the strings of the language and terminals.
- Σ (the set of terminals) is a subset of V.
- R(the set of rules) is a finite subset of $(V - \Sigma) \times V^*$
- S(the start symbol) can be any element of V- Σ

Each rule must have a single non-terminal in the L.H.S. It must have a R.H.S.

i) Even or odd palindromes

$S \rightarrow aSa \mid bSb \mid a \mid b \mid \varepsilon$

$G = (\{S,a,b\}, \{a,b\}, \{S \rightarrow aSa, S \rightarrow bSb, S \rightarrow a, S \rightarrow b, S \rightarrow \varepsilon\}, S)$

$S \rightarrow a$ and $S \rightarrow b$ allows for odd length palindromes and $S \rightarrow \varepsilon$ allows for even length palindromes.

**Derivation for w = ababa**

$S \Rightarrow aSa \Rightarrow a\mathbf{bSb}a \Rightarrow ab\mathbf{a}ba$

**Derivation for w = baab**

$S \Rightarrow bSb \Rightarrow b\mathbf{aSa}b \Rightarrow ba\boldsymbol{\varepsilon}ab \Rightarrow baab$

ii) $L = \{a^{2n}b^n : n \geq 0\}$, w = aaaabb

$L = \{\varepsilon, aab, aaaabb, \ldots\}$

$S \rightarrow aaSb \mid \varepsilon$

$G = (\{S,a,b\},\{a,b\},\{S \rightarrow aaSb, S \rightarrow \varepsilon\}, S)$

For every b, there are twice the number of a's and they have to be generated in tandem.

**Derivation for w = aaaabb**

$S \Rightarrow aaSb \Rightarrow aa\mathbf{aaSb}b \Rightarrow aaaa\boldsymbol{\varepsilon}bb \Rightarrow aaaabb$

iii) $L = \{w \in \{a,b\}^* : \#_a(w) = \#_b(w)\}$, w = abab

There can be ab, ba, i.e for every a there is a b, but the order does not matter. We can have ε also.

$S \rightarrow aSb \mid bSa \mid \varepsilon$

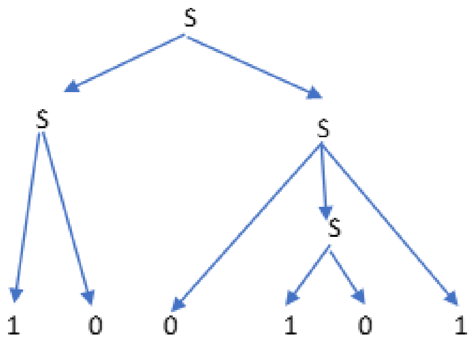
$G = (\{S,a,b\}, \{a,b\}, \{S \rightarrow aSb, S \rightarrow bSa, S \rightarrow \varepsilon\}, S)$

**Derivation for w = abab**

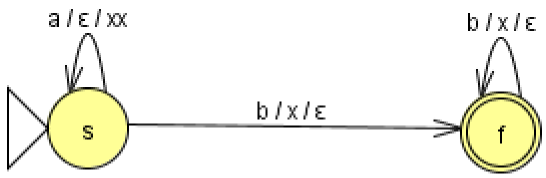
$S \Rightarrow aSb \Rightarrow a\mathbf{bSa}b \Rightarrow ab\boldsymbol{\varepsilon}ab \Rightarrow abab$

(b) For the given grammar, draw parse tree for w = 100101 [02] CO3 L2

$S \rightarrow 0S1 \mid 10 \mid SS$




4 (a) Construct PDA for L = $\{a^nb^{2n}, n>=1\}$. Write computation/instantaneous description(ID) for w = aabbbb [07] CO3 L3




In this PDA, for every a, there are two b's , so, for every a we read, we push 2 X's. Upon seeing a b, we change state to enforce all a's should precede b's. For every X we see, we pop a **b**.

M = (k, Σ, Γ, Δ, s, A)

K = {s,f}, Σ={a,b} Γ = {X}

Δ = { ((s,a,ε),(s,xx)), ((s,b,x),(f,ε)), ((f,b,x),(f,ε)) }

s=s, A = {f}

w = aabbbbb

(s,aabbbb,ε) |— (s, abbbb, XX) |— (s, bbbb, XXXX)

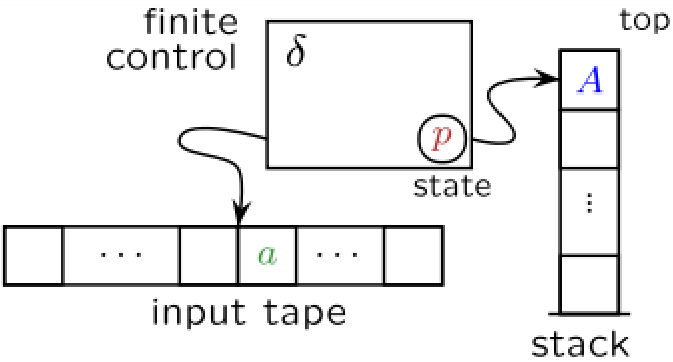
|— (f, bbb, XXX) |— (f, bb, XX) |— (f, b, X) |— (f, ε, ε)

Since (s,aabbbb,ε) |—* (f, ε, ε) , all input symbols are consumed and stack is empty and state is accepting state, i.e., f∈A, the string w = aabbbb is accepted by the PDA, M.

(b) Define a PDA. Explain the working principle of PDA with a neat diagram. [03] CO3 L2

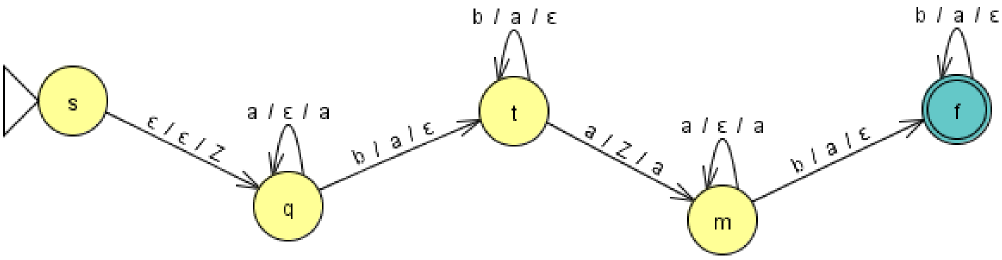
A pushdown automata M is a sixtuple (k, Σ, Γ, Δ, s, A)

- K is the finite set of states
- Σ is the input alphabet
- Γ is the stack alphabet
- s ∈ k is the start state
- A⊆k is the set of final states.
- Δ is the transition relation. It is a finite subset of
  (k x (Σ ∪ { ε }) x Γ* ) x (k x Γ*) For a state k on an input and string of symbols to pop from top of state, it goes to state k with a string of symbols to push on top of stack.




PDA reads input from input tape. There are a finite number of states and it makes use of a stack.

5 Define and construct PDA for L= $\{a^n b^n a^m b^m: n,m \geq 1\}$. Write computation/instantaneous description for w = abaabb [10] CO3 L3




In this PDA, strings accepted are {abab, aabbab, aaabbbaabb, …}
Strings that are not accepted are {ε, ab, aaabbabb, …}

Since n and m may be different values, we introduce a marker initially to keep track of when $a^n b^n$ has been read. We would reach the marker if the first part of the string is of the form $a^n b^n$.
Once we encounter the first a in $a^m$, we pop the marker and replace it with a.
All subsequent a's are pushed. Upon seeing b's, a's are popped.
If the string is $a^m b^m$, stack will be empty in state f.

M = (k, Σ, Γ, Δ, s, A)
k = { s, q, t, m, f }
Σ = {a,b}
Γ = {Z,a}
s=s
A = {f}

**<u>Computation for w = abaabb</u>**

(s,abaabb,ε) |— (q, abaabb, Z) |— (q, baabb, aZ), |— (t, aabb, Z) |— (m, abb, a) |— (m, bb, aa) |— (f, b, a) |— (f, ε, ε)
Since (s,abaabbb,ε) |—* (f, ε, ε) , all input symbols are consumed and stack is empty and state is accepting state, i.e., f∈A, the string w = abaabb is accepted by the PDA, M.

6 State the closure properties of regular languages. If L and M are regular languages, prove that L∩M and L–M, are also regular. [10] CO2 L3

Closure Properties of regular languages

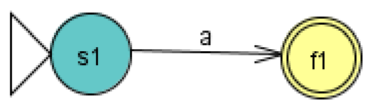
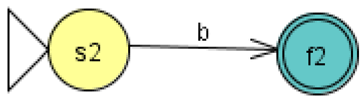
The regular languages are closed under Union, Concatenation and Kleene Star.
The regular languages are closed under complement, intersection, difference, reversal and substitution.

In order to prove that $L \cap M$ is regular, we need to prove that it is closed under complement and union.

$L \cap M = \neg \neg ( L \cap M) = \neg (\neg L \cup \neg L)$

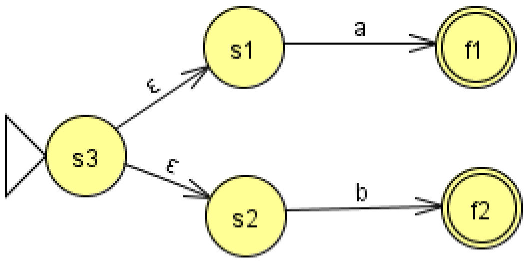
**<u>Proof that regular languages are closed under union</u>**

The proof for regular languages are closed under union is by construction. Let's take a regular expression $a \cup b$. We first construct FSM $M_1$ and $M_2$ to accept the primitives a and b.

| | |
|---|---|
|   $M_1$ that accepts RE |   $M_2$ that accepts RE b |

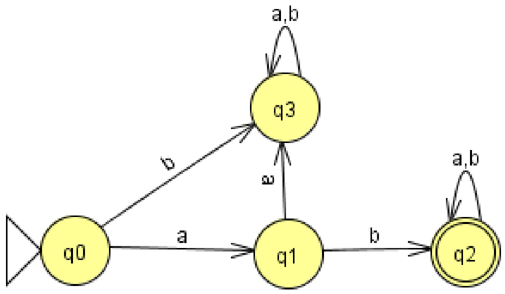
According to Kleene's theorem, for union of two languages that are regular and it's DFSM's, $M_1 = (k_1, \Sigma, \delta_1, s_1, A_1)$ and $M_2 = (k_2, \Sigma, \delta_2, s_2, A_2)$, we construct a new FSM, $M_3 = (k_3, \Sigma, \delta_3, s_3, A_3)$ such that $L(M_3) = L(M_1) \cup L(M_2)$. We rename states of $M_1$ and $M_2$ such that $k_1 \cap k_2 = \Phi$.
Create a new start state $s_3$ and connect the start states of $M_1$ and $M_2$ via $\varepsilon$-transitions. So $M_3 = (\{s_3\} \cup k_1 \cup k_2, \Sigma, \delta_3, s_3. A_1 \cup A_2)$ where $\delta_3 = \delta_1 \cup \delta_2 \cup \{(s_3, \varepsilon), s_1) \cup \{(s_3, \varepsilon), s_2)$.

So for $L(M_3) = a \cup b$, we get the machine where $M_3$ = ({s3, s1, s2, f1,f2}, {a,b}, {((s3,ε),s1), ((s3,ε),s2), ((s1,a),f1), ((s2,b),f2)}, s3, {f1,f2}.

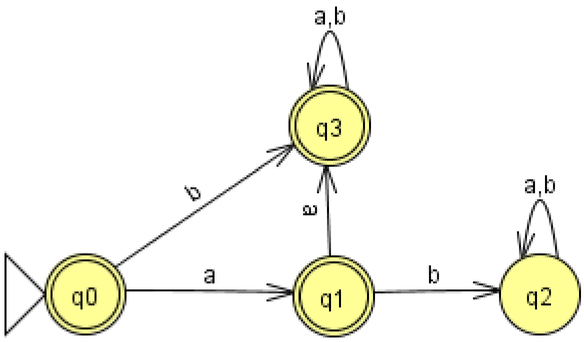



**<u>Proof that Regular Languages are closed under complement</u>**

If L is a regular language, there exists a DFSM $M_1 = (k, \Sigma, \delta, s, A)$ that accepts L.
The complement of L, $\neg L$ will be accepted by $M_2 = (k, \Sigma, \delta, s, k\text{-}A)$.
Any NDFSM has to be converted to an equivalent DFSM, then the accepting states have to be swapped with the non-accepting states.
For example, consider that language L that accepts strings that begin with 'ab' over the alphabet, $\Sigma=\{a,b\}$. The following DFSM, M = ({q0, q1, q2, q3}, {a,b}, {((q0,a),q1), ((q0,b),q3), ((q1,a),q3), ((q1,b),q2), ((q2,a),q2), ((q2,b),q2), ((q3,a),q3), ((q3,b),q3) }, q0, {q2}




The following DFSM accepts the complement of L, ¬L(M)




Hence, we proved the regular languages are closed under complement. The Accepting states are k-A.
The following DFSM, not_M = ({q0, q1, q2, q3}, {a,b}, {((q0,a),q1), ((q0,b),q3), ((q1,a),q3), ((q1,b),q2), ((q2,a),q2), ((q2,b),q2), ((q3,a),q3), ((q3,b),q3) }, q0, {q0,q1,q3}

Since regular languages are closed under union and complement and L ∩ M = ¬ ¬ ( L ∩ M) = ¬ (¬ L ∪ ¬ L), regular languages are closed under intersection.

**<u>Proof that Regular Languages are closed under Difference</u>**

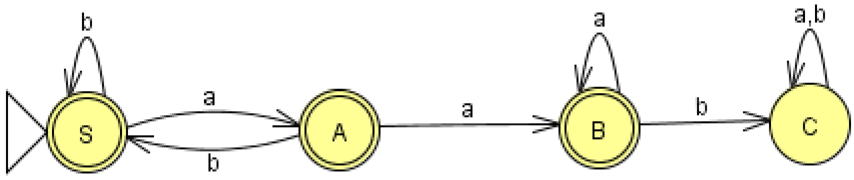
Using set theory, we can write,
L – M = L ∩ ¬M
Since we have proved that regular languages are closed under complement and intersection, it is thus proved that regular languages are also closed under difference.

| | | | | |
|---|---|---|---|---|
| 7(a) | Define regular grammar. Write regular grammar for the following languages.<br>(i) Strings of a's and b's not containing **aab** as a substring.<br>(ii) Strings of 0's and 1's ending with 1101. | [7] | CO3 | L3 |

A regular grammar is defined as G = (V, Σ, R, S)
V: The rule alphabet which contains non-terminals (symbols that are used in the grammar but do not appear in strings in the language) and terminals(symbols that can appear in strings generated by G).
Σ : (the set of terminals) is the subset of V.
R : (the set of rules) is a finite set of rules of the form X→Y
S : the start symbol which is a non-terminal.
In regular grammar there are 2 rules:
1. L.H.S. is a non-terminal
2. R.H.S. is either a ε or a single terminal or a single terminal followed by a single

non-terminal.

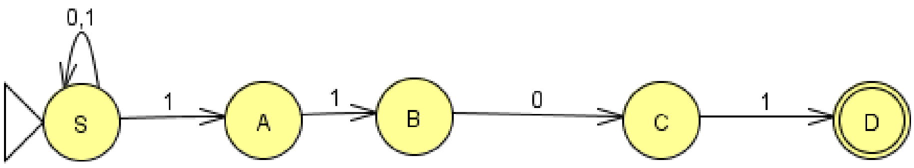
(i) Let’s draw an FSM for strings not containing aab as a substring. The dead state C need not be considered when writing grammar.




S→ ε | bS | aA
A→ aB | bS | ε
B → aB | ε
G = ({S, A, B, C, a, b}, {a, b}, { S→ ε | bS | aA, A→ aB | bS | ε, B → aB | ε }, S)
Consider production for w = bba which is a string accepted by the grammar G as it does not contain substring aab.

S ⟹ bS ⟹bbS ⟹bbaA ⟹bbaε ⟹bba

(ii) Let’s draw an FSM for string ending in 1101

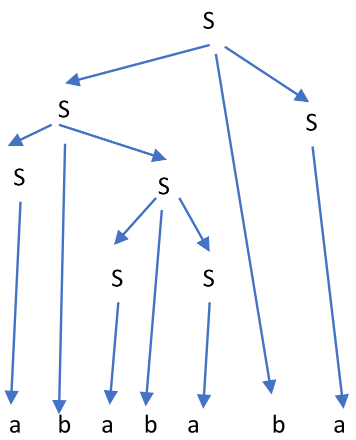


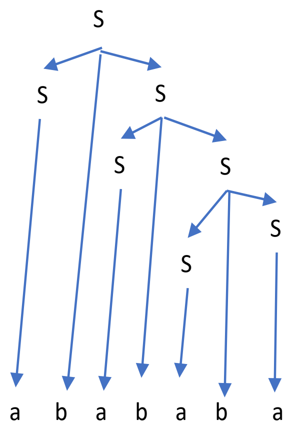

S → 0S | 1S | 1A
A → 1B
B → 0C
C → 1D
D → ε

G= ({S, A, B, C, D}, {0,1}, { S → 0S | 1S | 1A, A → 1B, B → 0C, C → 1D
D → ε
})

(b) What is ambiguous grammar? Prove that the following grammar is ambiguous for the string “**abababa**” , **S→SbS | a** [3] CO3 L3

A grammar, G is ambiguous iff there is at least one string in L(G) for which G produces more than one parse tree.




Since two different parse trees are possible for the grammar G for string w=abababa, the grammar is proved to be ambiguous.

**CO PO Mapping**

| Course Outcomes | | Modules covered | PO1 | PO2 | PO3 | PO4 | PO5 | PO6 | PO7 | PO8 | PO9 | PO10 | PO11 | PO12 | PSO1 | PSO2 | PSO3 | PSO4 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CO1 | Acquire fundamental understanding of the core concepts in automata theory and Theory of Computation | **1,2,3,4, 5** | 2 | 3 | - | - | - | 2 | - | - | - | - | - | - | - | 3 | | 3 |
| CO2 | Learn how to translate between different models of Computation | **1,2** | 2 | 3 | 2 | 2 | 2 | 2 | - | - | - | - | - | - | - | 3 | 3 | 3 |

| | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | (e.g., Deterministic and Non-deterministic and Software models). | | | | | | | | | | | | | | | | | |
| CO3 | Design Grammars and Automata (recognizers) for different language classes and become knowledgeable about restricted models of Computation (Regular, Context Free) and their relative powers. | **2,3** | 2 | 3 | 2 | 2 | 2 | 2 | - | - | - | - | - | - | 2 | - | 3 | - |
| CO4 | Develop skills in formal reasoning and reduction of a problem to a formal model, with an emphasis on semantic precision and conciseness. | **3,4** | 2 | 3 | 2 | 2 | - | 2 | - | - | - | - | - | - | 2 | 2 | 3 | 3 |
| CO5 | Classify a problem with respect to different models of Computation | **5** | 2 | 3 | 2 | 2 | - | 3 | - | - | - | - | - | - | 3 | 3 | 3 | 3 |

| COGNITIVE LEVEL | REVISED BLOOMS TAXONOMY KEYWORDS |
|---|---|
| L1 | List, define, tell, describe, identify, show, label, collect, examine, tabulate, quote, name, who, when, where, etc. |
| L2 | summarize, describe, interpret, contrast, predict, associate, distinguish, estimate, differentiate, discuss, extend |
| L3 | Apply, demonstrate, calculate, complete, illustrate, show, solve, examine, modify, relate, change, classify, experiment, discover. |
| L4 | Analyze, separate, order, explain, connect, classify, arrange, divide, compare, select, explain, infer. |
| L5 | Assess, decide, rank, grade, test, measure, recommend, convince, select, judge, explain, discriminate, support, conclude, compare, summarize. |

| PROGRAM OUTCOMES (PO), PROGRAM SPECIFIC OUTCOMES (PSO) | | | | CORRELATION LEVELS | |
|---|---|---|---|---|---|
| PO1 | Engineering knowledge | PO7 | Environment and sustainability | 0 | No Correlation |
| PO2 | Problem analysis | PO8 | Ethics | 1 | Slight/Low |
| PO3 | Design/development of solutions | PO9 | Individual and team work | 2 | Moderate/ Medium |
| PO4 | Conduct investigations of complex problems | PO10 | Communication | 3 | Substantial/ High |
| PO5 | Modern tool usage | PO11 | Project management and finance | | |
| PO6 | The Engineer and society | PO12 | Life-long learning | | |
| PSO1 | Develop applications using different stacks of web and programming technologies | | | | |
| PSO2 | Design and develop secure, parallel, distributed, networked, and digital systems | | | | |
| PSO3 | Apply software engineering methods to design, develop, test and manage software systems. | | | | |
| PSO4 | Develop intelligent applications for business and industry | | | | |