Parul® University

Theory of Computation

Prof. R Shivamallikarjun, Assistant Professor
Information Technology

Parul® University

DIGITAL LEARNING CONTENT

CHAPTER-2

Regular Languages and Finite Automata automater

Regular expressions and languages

“ Regular Expressions are used to denote regular languages. ”

- An expression is regular if:
- ϕ is a regular expression for regular language ϕ.
- ε is a regular expression for regular language {ε}.
- If a and b are regular expression, a + b is also a regular expression with language {a, b}.

Parul® University

DIGITAL LEARNING CONTENT

Regular expressions and languages

“ A language is regular if it can be expressed in terms of regular expression. ”

	REGULAR EXPRESSION	REGGULAR LANGUAGES
set of vowels	(a ∪ e ∪ i ∪ o ∪ u)	{a, e, i, o, u}
a followed by 0 or more b	(a, b^*)	{a, ab, abb, abbb, abbbb,….}
any no. of vowels followed by any no. of consonants	$v^*.c^*$ (where v – vowels and c – consonants)	{ ε , a ,aou, aiou, b, abcd…..} where ε represent empty string (in case 0 vowels and o consonants)

Regular expressions and languages

Closure Properties of Regular Languages

Union: L1 = $\{a^n \mid n \geq 0\}$ and L2 = $\{b^n \mid n \geq 0\}$
L3 = L1 $\cup$ L2 = $\{a^n \cup b^n \mid n \geq 0\}$ is also regular.

Intersection: L1= $\{a^m b^n \mid n \geq 0$ and $m \geq 0\}$ and L2= $\{a^m b^n \cup b^n a^m \mid$ $n \geq 0$ and $m \geq 0\}$
L3 = L1 $\cap$ L2 = $\{a^m b^n \mid n \geq 0$ and $m \geq 0\}$ is also regular.

Concatenation:L1 = $\{a^n \mid n \geq 0\}$ and L2 = $\{b^n \mid n \geq 0\}$
L3 = L1.L2 = $\{a^m . b^n \mid m \geq 0$ and $n \geq 0\}$ is also regular.

Regular expressions and languages

Kleene Closure:$L1 = (a \cup b)$
$L1^* = (a \cup b)^*$

Complement: $L(G) = \{a^n \mid n > 3\}$
$L'(G) = \{a^n \mid n <= 3\}$

Parul® University

DIGITAL LEARNING CONTENT

Deterministic finite automata (DFA) and

FA is characterized into two types:

- **Deterministic Finite Automata (DFA)**
- **Nondeterministic Finite Automata (NFA)**

Deterministic finite automata (DFA) and

Deterministic Finite Automata (DFA)

DFA consists of 5 tuples {Q, ∑, q, F, δ}.

- Q: set of all states.
- ∑: set of input symbols. (Symbols which machine takes as input)
- q: Initial state. (Starting state of a machine)
- F: set of final state.
- δ: Transition Function, defined as δ : Q X ∑ --> Q.

Deterministic finite automata (DFA) and

Nondeterministic Finite Automata (NFA)

NFA is similar to DFA except following additional features:
1. Null (or ε) move is allowed.
2. Ability to transmit to any number of states for a particular input.

NFA has a different transition function, rest is same as DFA.
δ: Transition Function
δ: $Q \times (\sum \cup \epsilon) \rightarrow 2^Q$.

Parul® University

DIGITAL LEARNING CONTENT

Deterministic finite automata (DFA) and

Nondeterministic Finite Automata (NFA)

For example, below is a NFA for problem

Parul® University

DIGITAL LEARNING CONTENT

Deterministic finite automata (DFA) and

Converting a DFA to its regular expression are-

1. **Arden’s Method**
2. **State Elimination Method**

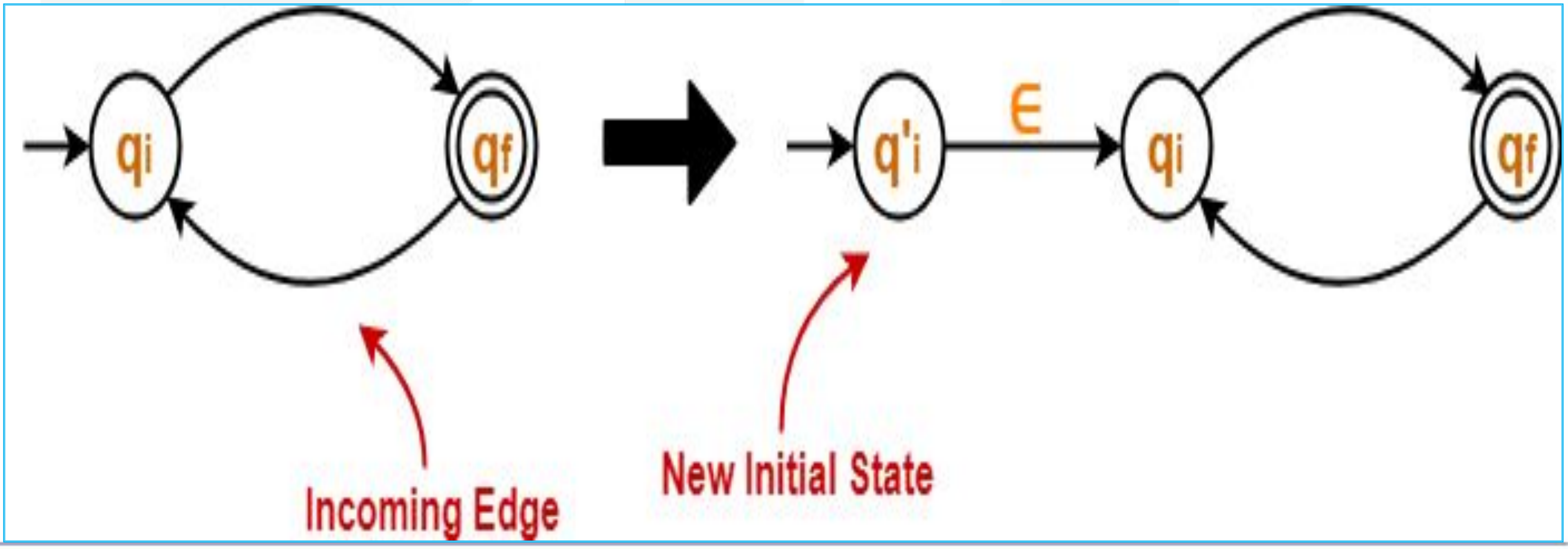
Parul® University

DIGITAL LEARNING CONTENT

Deterministic finite automata (DFA) and

State Elimination Method

1.If there exists any incoming edge to the initial state, then create a new initial state having no incoming edge to it.



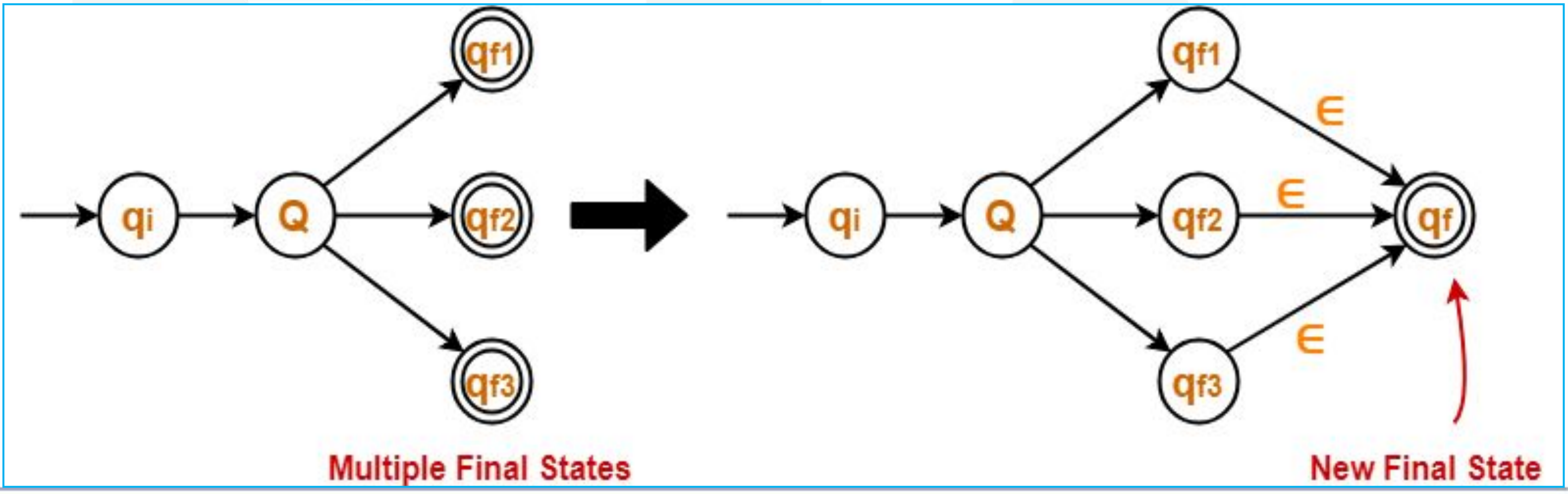
Parul® University

DIGITAL LEARNING CONTENT

Deterministic finite automata (DFA) and

State Elimination Method

2. If there exists multiple final states in the DFA, then convert all the final states into non-final states and create a new single final state.

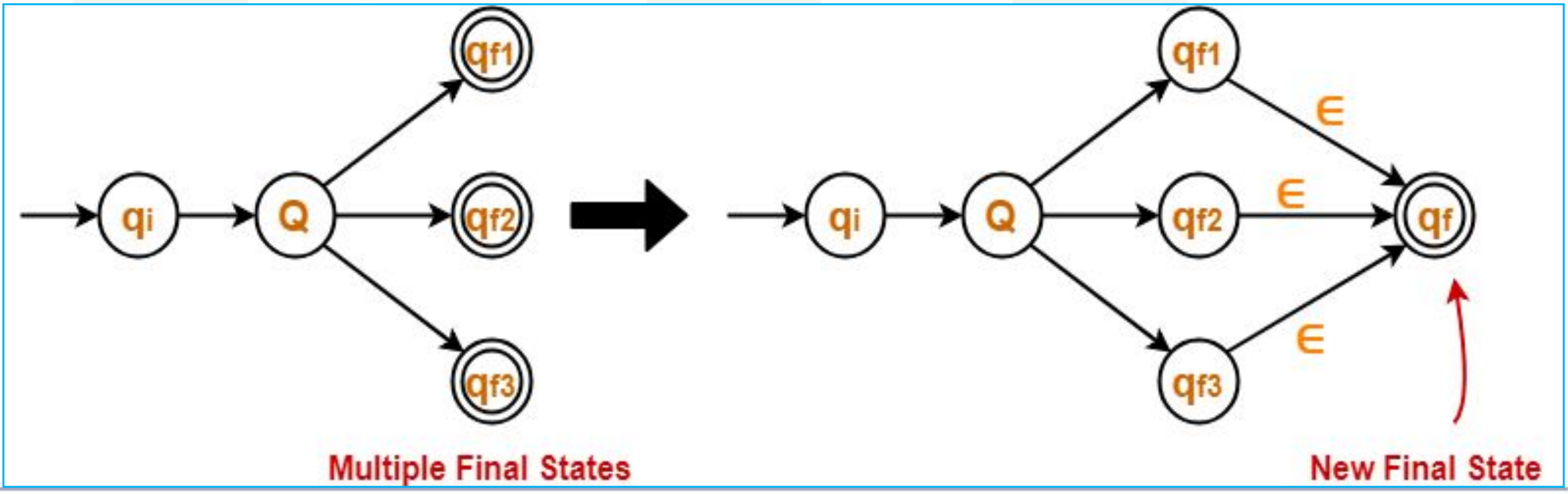


Parul® University
DIGITAL LEARNING CONTENT

Deterministic finite automata (DFA) and

State Elimination Method

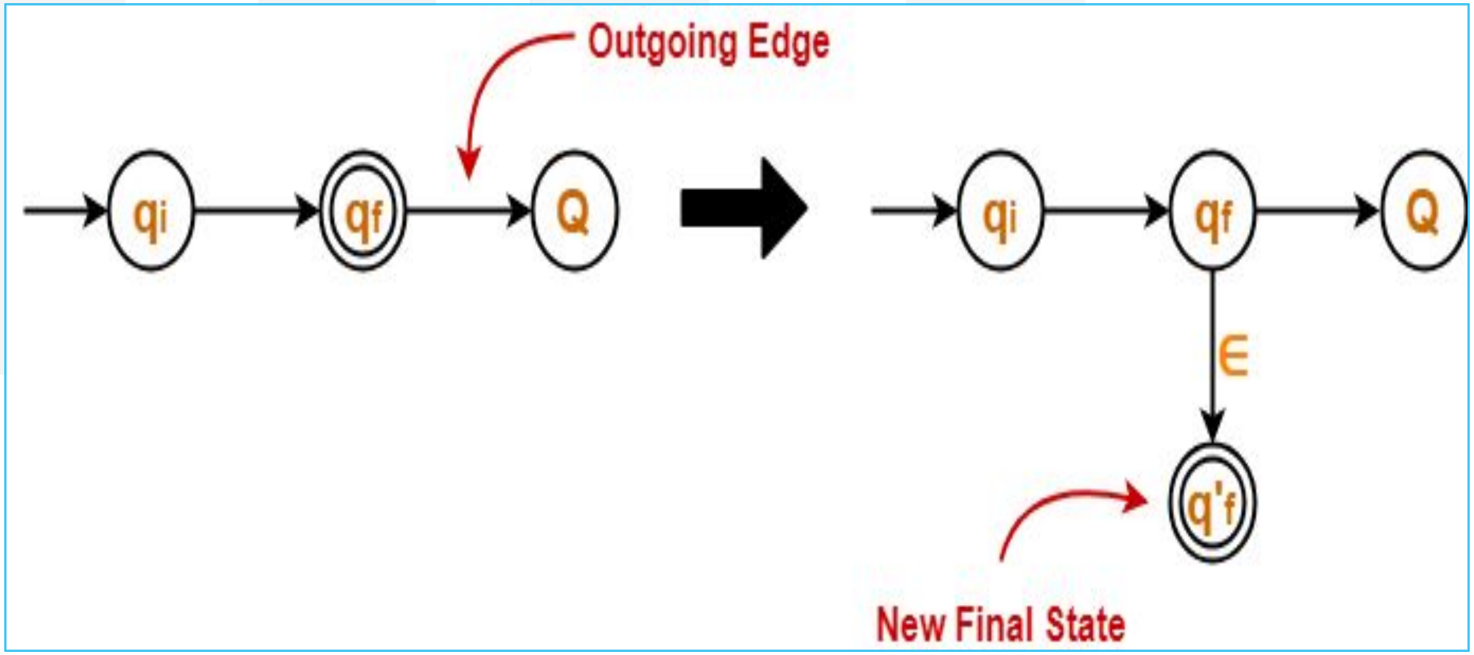
2. If there exists multiple final states in the DFA, then convert all the final states into non-final states and create a new single final state.



Deterministic finite automata (DFA) and

State Elimination Method

3. If there exists any outgoing edge from the final state, then create a new final state having no outgoing edge from it.



Deterministic finite automata (DFA) and

State Elimination Method

4. Eliminate all the intermediate states one by one.
5. These states may be eliminated in any order

The state elimination method can be applied to any finite automata. (NFA, ∈-NFA, DFA etc.)

Parul® University

DIGITAL LEARNING CONTENT

Deterministic finite automata (DFA) and

Arden's Theorem-

It states that-

Let P and Q be two regular expressions over ∑.

If P does not contain a null string ∈, then-

R = Q + RP has a unique solution i.e. R = QP*

Conditions-

To use Arden's Theorem, following conditions must be satisfied-

- The transition diagram must not have any ∈ transitions.
- There must be only a single initial state.

NFA and equivalence with DFA

Steps followed to convert a given NFA to a DFA-

Step-1:

• Let Q’ be a new set of states of the DFA. Q’ is null in the starting.

• Let T’ be a new transition table of the DFA.

Step-2:

• Add start state of the NFA to Q’.

• Add transitions of the start state to the transition table T’.

• If start state makes transition to multiple states for some input alphabet, then treat those multiple states as a single state in the DFA.

Parul® University

DIGITAL LEARNING CONTENT

NFA and equivalence with DFA

Steps followed to convert a given NFA to a DFA-

Step-3:

If any new state is present in the transition table T’,

- Add the new state in Q’.
- Add transitions of that state in the transition table T’.

Step-4:

Keep repeating Step-3 until no new state is present in the transition table T’.

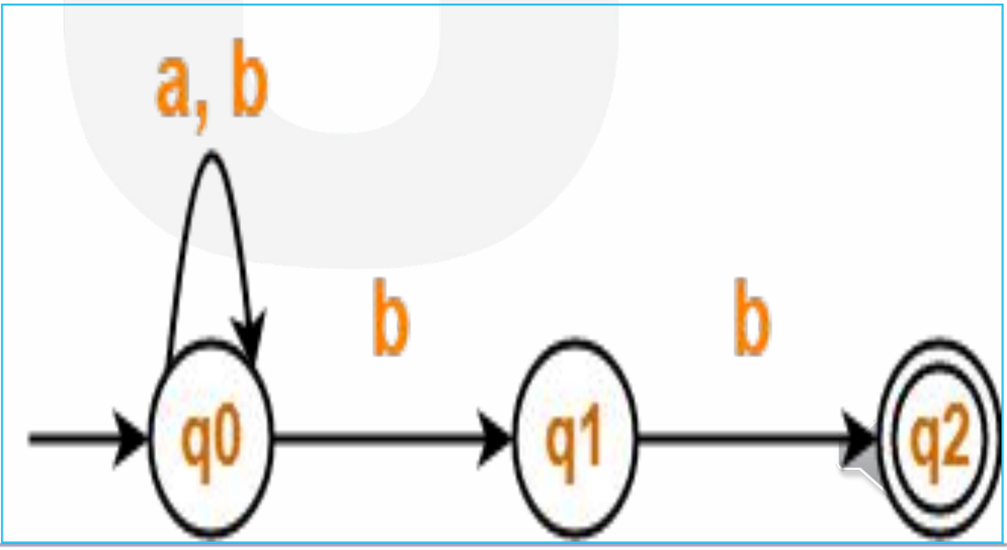
Finally, the transition table T’ so obtained is the complete transition table of the required DFA.

Parul® University

DIGITAL LEARNING CONTENT

NFA and equivalence with DFA

Convert the following Non-Deterministic Finite Automata (NFA) to Deterministic Finite Automata (DFA)-



NFA and equivalence with DFA

Transition table for the given Non-Deterministic Finite Automata (NFA) is-

State / Alphabet	A	b
→q0	q0	q0, q1
q1	–	*q2
*q2	–	–

NFA and equivalence with DFA

Step-1:

•Let Q’ be a new set of states of the Deterministic Finite Automata (DFA).
•Let T’ be a new transition table of the DFA.

NFA and equivalence with DFA

Step-2:

- Add transitions of start state q0 to the transition table T’.

State / Alphabet	A	b
→q0	q0	{q0, q1}

Parul® University

DIGITAL LEARNING CONTENT

NFA and equivalence with DFA

Step-3:

•New state present in state Q’ is {q0, q1}.
•Add transitions for set of states {q0, q1} to the transition table T’.

State / Alphabet	A	b
→q0	q0	{q0, q1}
{q0, q1}	q0	{q0, q1, q2}

NFA and equivalence with DFA

Step-4:

•New state present in state Q’ is {q0, q1, q2}.
•Add transitions for set of states {q0, q1, q2} to the transition table T’.

State / Alphabet	A	b
→q0	q0	{q0, q1}
{q0, q1}	q0	{q0, q1, q2}
{q0, q1, q2}	q0	{q0, q1, q2}

Parul® University

DIGITAL LEARNING CONTENT

NFA and equivalence with DFA

Step-5:

- Since no new states are left to be added in the transition table T’, so we stop.
- States containing q2 as its component are treated as final states of the DFA.
- Transition table for Deterministic Finite Automata (DFA) is-

State / Alphabet	a	b
→q0	q0	{q0, q1}
{q0, q1}	q0	*{q0, q1, q2}
*{q0, q1, q2}	q0	*{q0, q1, q2}

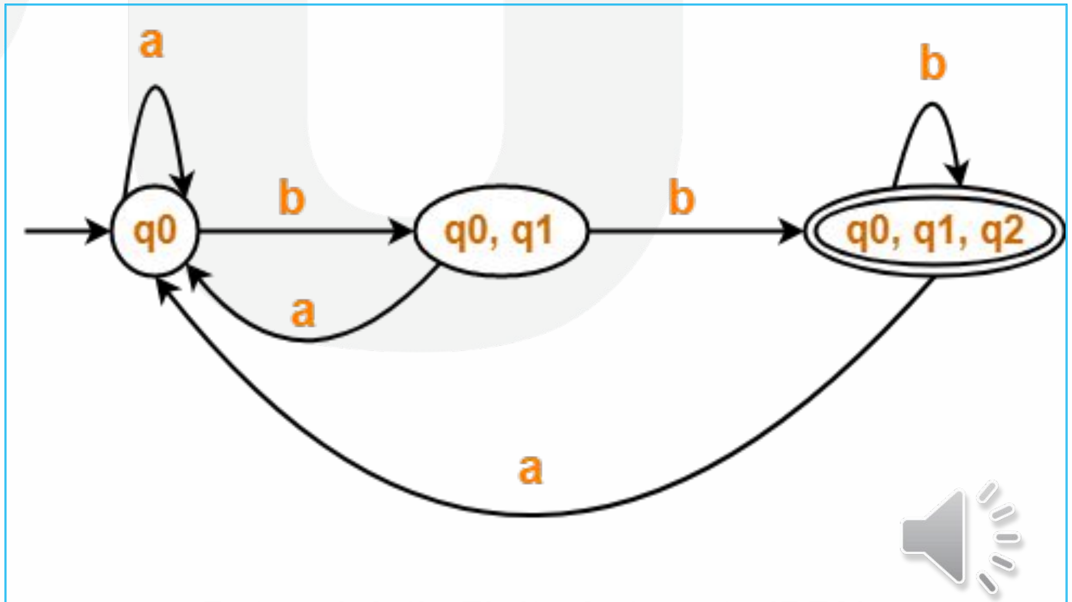
Parul® University

DIGITAL LEARNING CONTENT

NFA and equivalence with DFA

Step-5:

- Deterministic Finite Automata (DFA) is-

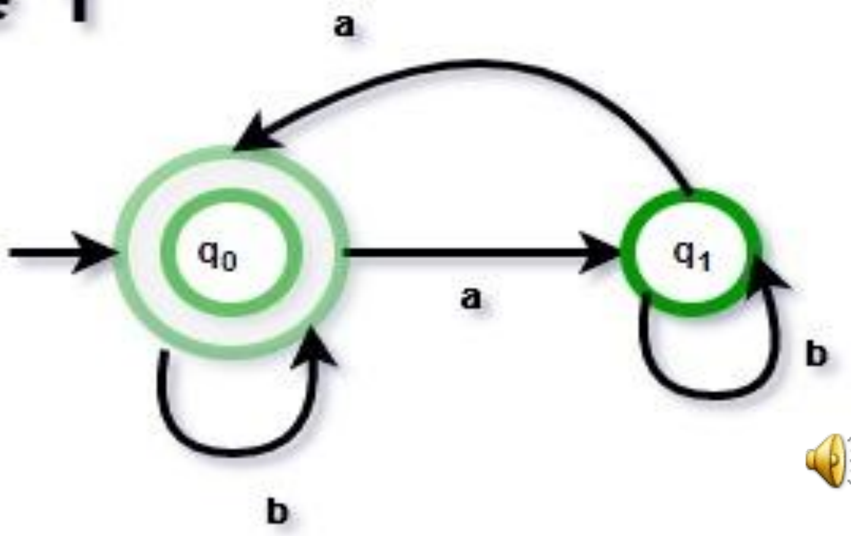


Deterministic Finite Automata (DFA)

Regular grammars and equivalence with finite automata

Even number of a's : The regular expression for even number of a's is (b|ab*ab*)*. We can construct finite automata as shown

Figure 1



Parul® University

DIGITAL LEARNING CONTENT

Properties of regular languages

1. Kleen Closure:
RS is a regular expression whose language is L, M. R* is a regular expression whose language is L*.
2. Positive closure:
RS is a regular expression whose language is L, M. R+ is a regular expression whose language is L+ .
3. Complement:
The complement of a language L (with respect to an alphabet E such that E* contains L) is E*–L. Since E* is surely regular, the complement of a regular language is always regular.

Properties of regular languagese

4. Reverse Operator:
Given language L, LR is the set of strings whose reversal is in L.
Example: L = {0, 01, 100};
LR ={0, 10, 001}.

5. Complement:
The complement of a language L (with respect to an alphabet E such E* that contains L) is E* –L. Since E* is surely regular, the complement of a regular language is always regular.

6. Union:
Let L and M be the languages of regular expressions R and S, respectively. Then R+S is a regular expression whose language is(L U M).

Properties of regular languages

7. Intersection:
Let L and M be the languages of regular expressions R and S, respectively then it a regular expression whose language is L intersection M.
8. Set Difference operator:
If L and M are regular languages, then so is L – M = strings in L but not M.
9. Homomorphism:
A homomorphism on an alphabet is a function that gives a string for each symbol in that alphabet.
10. Inverse Homomorphism :
Let h be a homomorphism and L a language whose alphabet is the output language of h. h-1(L) = {w | h(w) is in L}.

Pumping lemma for regular languages

“The tool used for proving that a language is not regular is the Pumping Lemma.”

Pumping lemma for regular languages

The Pumping Lemma
Let L be a regular language. Then there exists a constant n (which depends on L) such that for every w ∈ L, where $|w| \geq n$, there exists strings x, y, and z such that

1. $w=xyz$,

2. $|xy| \leq n$,

3. $|y| \geq 1$, and

4. For all $k \geq 0$, $xy^kz \in L$.

References

1. https://people.cs.clemson.edu/~goddard/texts/theoryOfComputation/1.pdf
2. https://en.wikipedia.org/wiki/Chomsky_hierarchy

DIGITAL LEARNING CONTENT

Parul® University

www.paruluniversity.ac.i