language is not R.L.

* $L = ww \mid w \in (a,b)^*$

given String:- abaaba

x y z

if $i=4$; $xy^iz \Rightarrow xy^4z \Rightarrow abbbbaaba$; not satisfied

$|y|>0 \Rightarrow 1>0 \Rightarrow$ satisfied

$|xy| \leq n \Rightarrow 5 \leq n \Rightarrow$ satisfied

1st condition is not satisfied so not a regular lang.

Closure properties of Regular sets :-

1. **Union**
   Union of 2 regular language is regular
   if L and M are regular languages, then so is $L \cup M$

2. **Intersection**
   Intersection of 2 regular languages are regular.
   if L and M are regular languages, then so is $L \cap M$.

3. **complement** of a language is regular.
   if L is a regular language over alphabet Σ, then $\Sigma^* - L$ is also a regular language.

4. **Difference** of two regular languages is regular.
   if L and M are regular languages, then so is $L - M$.

5. **Reversal** of a regular language is regular.
   Reversing a string means writing the string backwards. for Eg., reverse of the string abcde is the string edcba.
   The reversal of a language L is the language consisting of reversal of all its strings reversed.
   for Eg., if $L = \{001, 110\}$, then $L(R) = \{100, 011\}$.

6. **closure** of a regular language is regular.
   if L is a regular language, then so is L^*.

7. **Concatenation** of regular languages is regular
   if L and M are regular languages, then so is LM.

8. **Homomorphism** of a regular language is regular
   A homomorphism is the substitution of strings for symbols.
   for Eg:- let the function h be difined by $h(0) = a$ & $h(1) = b$. Then h applied to 0011 is simply aabb
   if h is a homomorphism on alphabet Σ and if w

is a string of symbols abcd...z, then
$h(w) = h(a)h(b)h(c)h(d)\ldots h(z)$
The mathematical definition for homomorphism is
$h: \Sigma^* \to T^*$ such that $\forall x, y \in \Sigma^*$ and $h(x), h(y) \in T^*$.

A homomorphism can also be applied to a language by applying it to each string in the language. Let L be a language over alphabet Σ, and h be a homomorphism on Σ then,
$h(L) = \{h(w) \mid w \text{ is in } L\}$
Then theorem can be stated as follows. 'if L is a regular language over alphabet Σ, and if h is a homomorphism on Σ, then h(L) is also regular'.

9. Inverse homomorphism of two regular languages is regular.
Suppose h be a homomorphism from some alphabet Σ to strings in another alphabet T and suppose L be a language over T, then h inverse of L. $h^{-1}(L)$, is the set of strings w in Σ^* such that h(w) is in L.
The theorem states that: 'if h is a homomorphism from alphabet Σ to alphabet T, and L is a regular language on T, then $h^{-1}(L)$ is also a regular language.

Eg:-
$h(0) = a$
$h(1) = b$
$h(2) = ab$
$h(L) = \{abab\}$
$h^{-1}(L) = \{0101, 22, 012, 201\}$
$h(h^{-1}(L)) = \{abab, abab, abab, abab\}$
$= \{abab\}$

Applications of Regular Expression

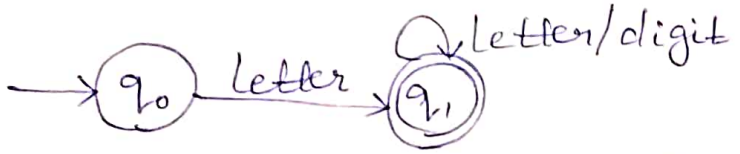
* Lexical Analysis

The compiler has a lexical analysis phase which forms the most important initial processing where the source program is scanned for recognizing the tokens. The tokens are defined by a regular expression for each pattern

Eg:- An identifier is a token which has a pattern: 'an alphabet followed by any no. of alphanumeric characters'. The corresponding regular expression is

$$[A-Za-z][A-Z|a-z|0-9]^*.$$

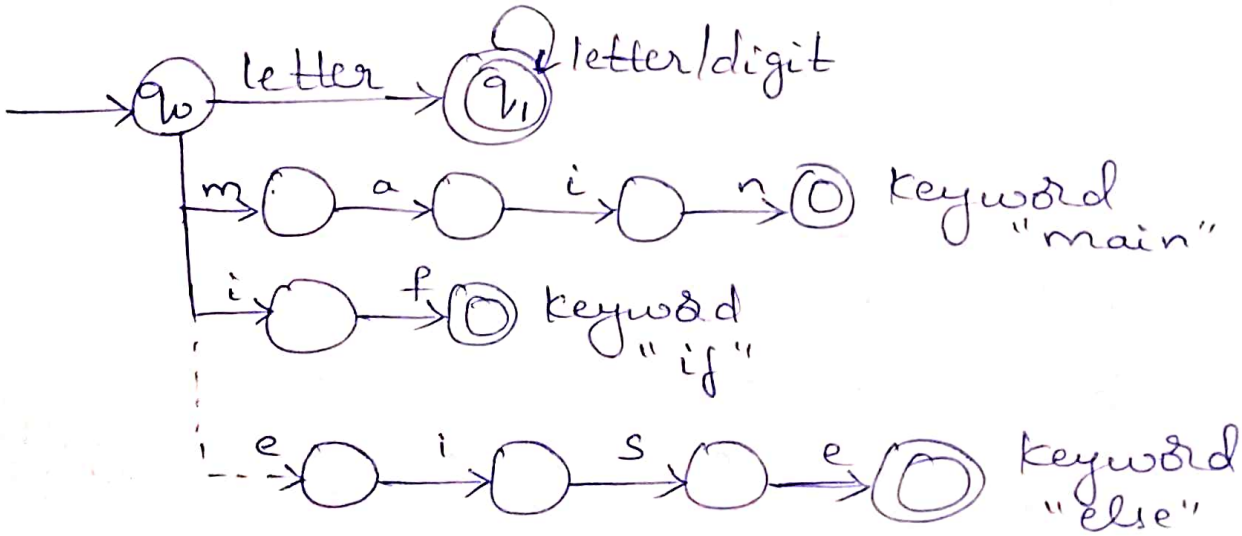
when the expression is specified using regular Expression, the corresponding finite automata generated is shown.



letter - [A-Za-z]

digit - [0-9]

⇒ DFA for Identifiers



⇒ DFA for Keywords

* Finding Patterns

The regular Expression technology is found to be useful in finding words corresponding to some patterns to be searched which are defined vaguely.

* To Scan a Very large no. of webpagesand detect addresses.
* TO create mailing list.
* To classify the business by their location and to answer the queries.
* To search files in the system based on a pattern

UNIT - 3

Regular Grammars

Grammar Contains 4 tuples

{V, T, P, S}

V – Variables
T – terminals
P – Production rule
S – Start Symbol

S → aSb | ∈ ⟶ Entire it is a production

a & b → terminals

Variable.

Eg:- * S → aSb | ∈ [There are 2 diff. productions one is aSb other is ∈]

Considering one production.

S → aSb [in place of S Substitute aSb]

S → aaSbb

from here we got aabb

now again Substitute aSb in place of S

S → aaaSbbb

Now if S is ∈ then we ~~would~~ will get aaabbb

⇒ if ∈ is placed instead of S, it is known as terminating.

⇒ if aSb is placed instead of S then the no. of a's and b's will be increasing and also no. of a's = no. of b's.

So, the language generated by this grammar is $a^n b^n$. [i.e., no. of a's = no. of b's]

* $S \to SS$
$S \to aSb$
$S \to bSa$
$S \to \epsilon$
} Given Grammar

The 4 productions can also be written as

$S \to SS | aSb | bSa | \epsilon$

cont. consider $S \to SS$ production.

$S \to SS$ [Substitute $S = aSb$]

$S \to \underline{aSb}aSb$ [again Sub. $S = aSb$]

$S \to a\underline{aSb}ba\underline{aSb}b$ [Sub. $S = bSa$]

$S \to aa\underline{bSa}bbaa\underline{bSa}bb$ [Sub. $S = \epsilon$] i.e terminating

$S \to \underline{aababbaababb}$

6 a's, 6 b's [no. of a's = no. of b's]

So, the language generated by this grammar is $(ab + ba)^*$

diff. b/w the above 2 eg's :-

1st eg :- no. of a's = no. of b's, & a should follow b and b should a. (condition)

2nd eg :-

no. of a's = no. of b's but there is no condition that a should follow b and b should follow a.