

Method 5 (Transition Graph)

Definition:

A Transition graph (TG), is a collection of the followings Finite number of states, at least one of which is start state and some (maybe none) final states.

Finite set of input letters (Σ) from which input strings are formed.

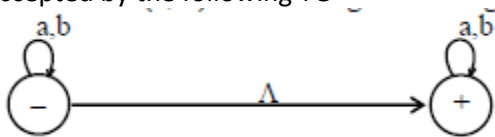
Finite set of transitions that show how to go from one state to another based on reading specified substrings of input letters, possibly even the null string (Λ).

Note

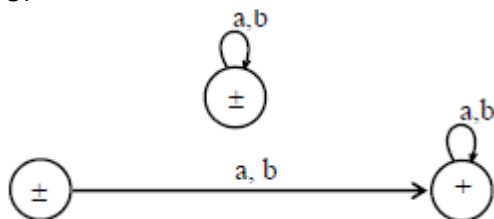
It is to be noted that in TG there may exist more than one paths for certain string, while there may not exist any path for certain string as well. If there exists at least one path for a certain string, starting from initial state and ending in a final state, the string is supposed to be accepted by the TG, otherwise the string is supposed to be rejected. Obviously collection of accepted strings is the language accepted by the TG.

Example

Consider the Language L, defined over $\Sigma = \{a, b\}$ of **all strings including Λ** . The language L may be accepted by the following TG



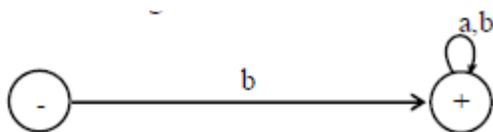
Or



It may be noted that every FA is a TG as well, but the converse may not be true, *i.e.* every TG may not be an FA.

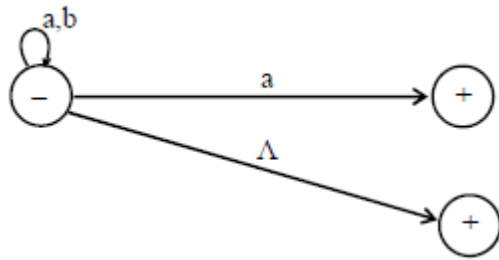
Example

Consider the language L of strings, defined over $\Sigma = \{a, b\}$, **starting with b**. The language L may be expressed by RE $b(a + b)^*$, may be accepted by the following TG



Example

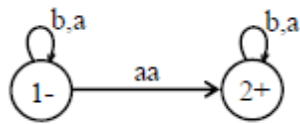
Consider the language L of strings, defined over $\Sigma = \{a, b\}$, **not ending in b**. The language L may be expressed by RE $\Lambda + (a + b)^*a$, may be accepted by the following TG



Example

Consider the Language L of strings, defined over $\Sigma = \{a, b\}$, **containing double a.**

The language L may be expressed by the following regular expression $(a+b)^* (aa) (a+b)^*$. This language may be accepted by the following TG

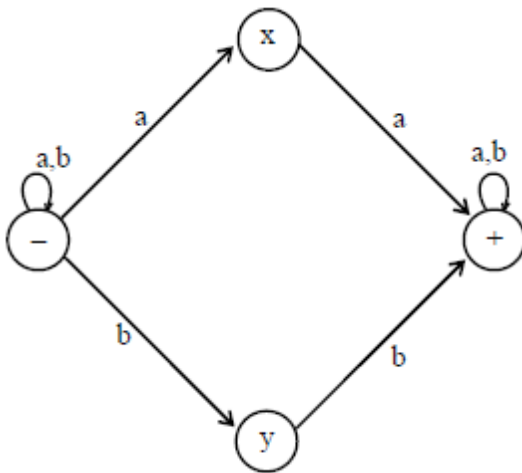


Example

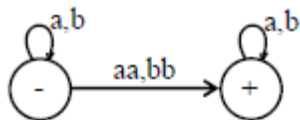
Consider the language L of strings, defined over $\Sigma = \{a, b\}$, **having double a or double b.**

The language L can be expressed by RE $(a+b)^* (aa + bb) (a+b)^*$.

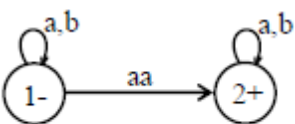
The above language may also be expressed by the following TGs.



OR



OR



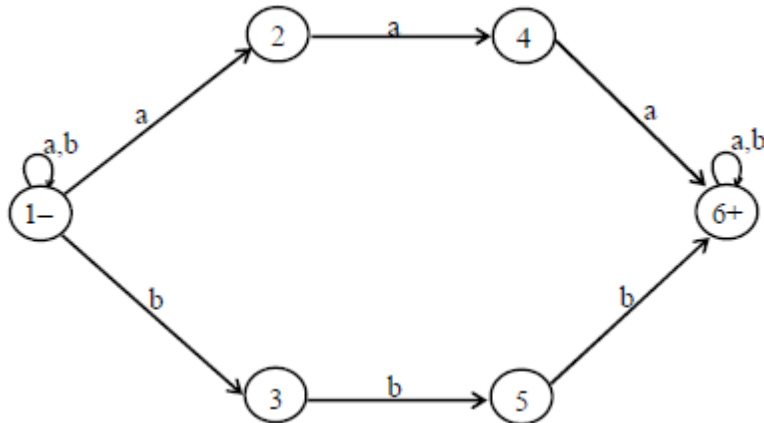
Note

In the above TG if the states are not labeled then it may not be considered to be a single TG

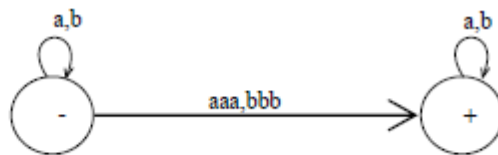
Example

Consider the language L of strings, defined over $\Sigma = \{a, b\}$, **having triple a or triple b**. The language L may be expressed by RE $(a+b)^* (aaa + bbb) (a+b)^*$

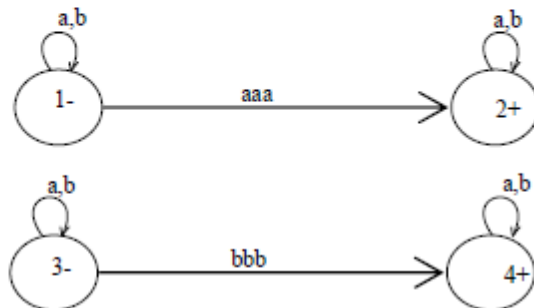
This language may be accepted by the following TG



OR



OR

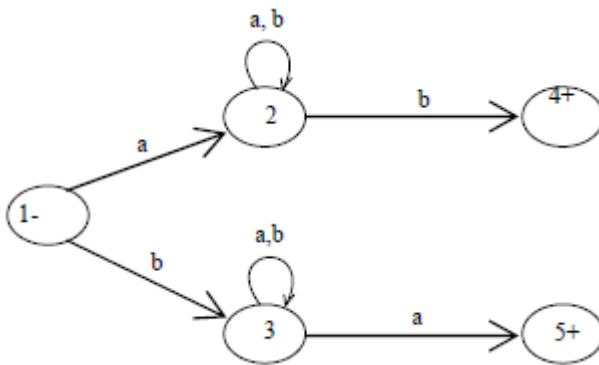


Example

Consider the language L of strings, defined over $\Sigma = \{a, b\}$, **beginning and ending in different letters**.

The language L may be expressed by RE $a(a + b)^*b + b(a + b)^*a$

The language L may be accepted by the following TG



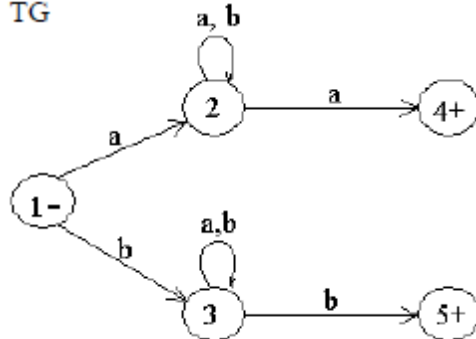
Example

Consider the Language L of strings **of length two or more**, defined over $\Sigma = \{a, b\}$, **beginning with and ending in same letters**.

The language L may be expressed by the following regular expression $a(a + b)^*a + b(a + b)^*b$

This language may be accepted by the following TG

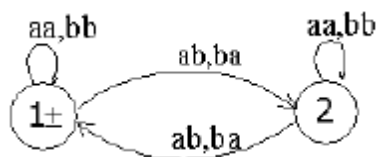
TG



Example

Consider the **EVEN-EVEN** language, defined over $\Sigma = \{a, b\}$. As discussed earlier that **EVEN-EVEN** language can be expressed by a regular expression $(aa+bb+(ab+ba)(aa+bb)^*(ab+ba))^*$

The language **EVEN-EVEN** may be accepted by the following TG



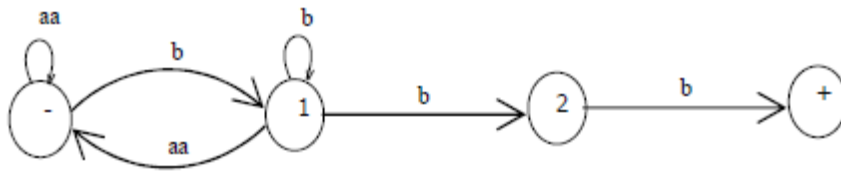
Example

Consider the language L, defined over $\Sigma = \{a, b\}$, in which **a's occur only in even clumps and that ends in three or more b's**. The language L can be expressed by its regular expression $(aa)^*b(b^*+(aa(aa)^*b)^*)bb$

OR

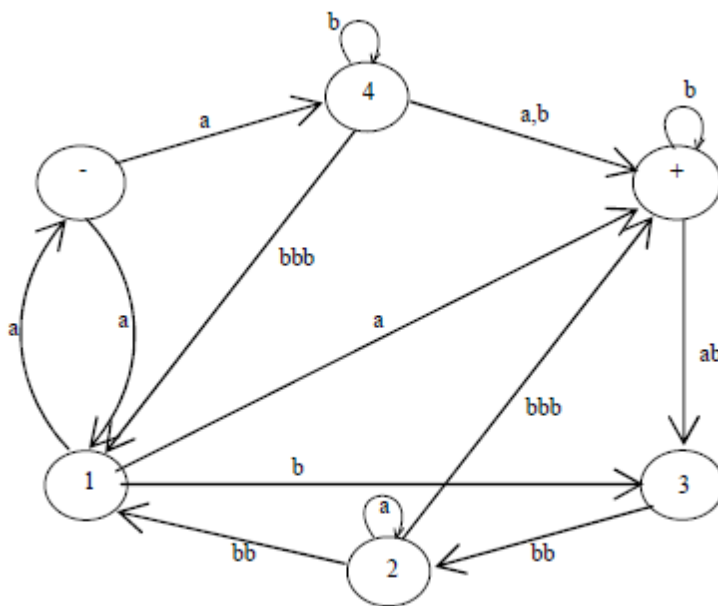
$(aa)^*b(b^*+(aa)^+b)^*)bb$.

The language L may be accepted by the following TG



Example

Consider the following TG



Consider the string abbbabbbabba. It may be observed that the above string traces the following three paths,

(using the states)

(a)(b) (b) (b) (ab) (bb) (a) (bb) (a)

(-)(4)(4)(+)(+)(3)(2)(2)(1)(+)

(a)(b) ((b)(b)) (ab) (bb) (a) (bb) (a)

(-)(4)(+)(+)(+)(3)(2)(2)(1)(+)

(a)((b) (b)) (b) (ab) (bb) (a) (bb) (a)

(-) (4)(4)(4)(+) (3)(2)(2)(1)(+)

Which shows that all these paths are successful, (*i.e.* the path starting from an initial state and ending in a final state).

Hence the string abbbabbbabba is accepted by the given TG.

Generalized Transition Graphs

A generalized transition graph (GTG) is a collection of three things

Finite number of states, at least one of which is start state and some (maybe none) final states.

Finite set of input letters (Σ) from which input strings are formed.

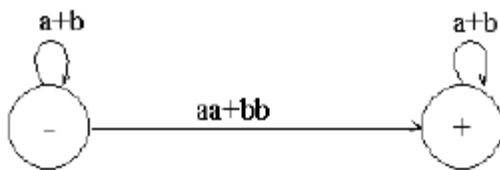
Directed edges connecting some pair of states labeled with regular expression.

It may be noted that in GTG, the labels of transition edges are corresponding regular expressions

Example

Consider the language L of strings, defined over $\Sigma = \{a, b\}$, containing **double a or double b**. The language L can be expressed by the following regular expression $(a+b)^* (aa + bb) (a+b)^*$

The language L may be accepted by the following GTG.

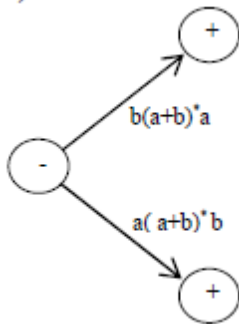


Example

Consider the language L of strings of, defined over $\Sigma = \{a, b\}$, **beginning and ending in different letters**.

The language L may be expressed by RE $a(a+b)^*b + b(a+b)^*a$

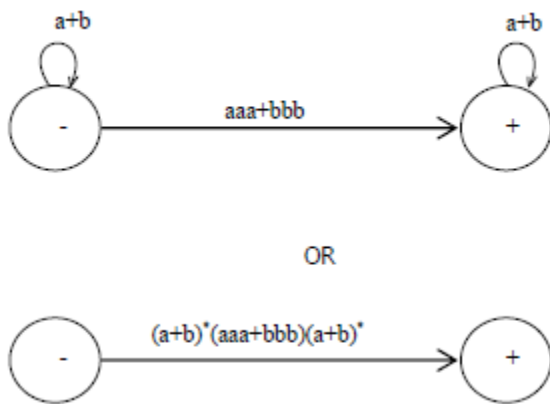
The language L may be accepted by the following GTG



Example

Consider the language L of strings, defined over $\Sigma = \{a, b\}$, **having triple a or triple b**. The language L may be expressed by RE $(a+b)^* (aaa + bbb) (a+b)^*$

This language may be accepted by the following GTG



Nondeterminism

TGs and GTGs provide certain relaxations *i.e.* there may exist more than one path for a certain string or there may not be any path for a certain string, this property creates **nondeterminism** and it can also help in differentiating TGs or GTGs from FAs. Hence an FA is also called a Deterministic Finite Automaton (DFA).

Kleene's Theorem

If a language can be expressed by
FA or
TG or
RE

then it can also be expressed by other two as well.

It may be noted that the theorem is proved, proving the following three parts

Kleene's Theorem Part I

If a language can be accepted by an FA then it can be accepted by a TG as well.

Kleene's Theorem Part II

If a language can be accepted by a TG then it can be expressed by an RE as well.

Kleene's Theorem Part III

If a language can be expressed by a RE then it can be accepted by an FA as well.

Proof(Kleene's Theorem Part I)

Since every FA can be considered to be a TG as well, therefore there is nothing to prove.

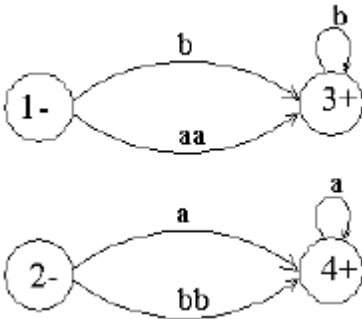
Proof(Kleene's Theorem Part II)

To prove part II of the theorem, an algorithm consisting of different steps, is explained showing how a RE can be obtained corresponding to the given TG. For this purpose the notion of TG is changed to that of GTG *i.e.* the labels of transitions are corresponding REs.

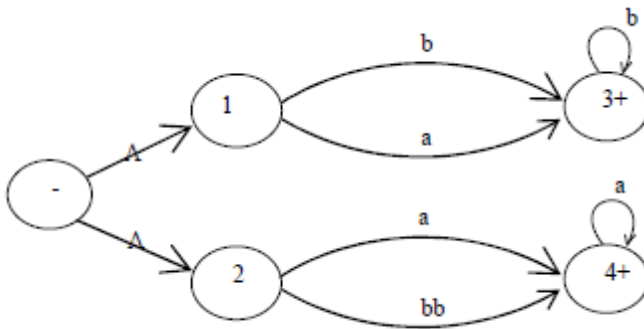
Basically this algorithm converts the given TG to GTG with one initial state along with a single loop, or one initial state connected with one final state by a single transition edge. The label of the loop or the transition edge will be the required RE.

Step 1 If a TG has more than one start states, then introduce a new start state connecting the new state to the old start states by the transitions labeled by Λ and make the old start states the non-start states. This step can be shown by the following example

Example



The above TG can be converted to

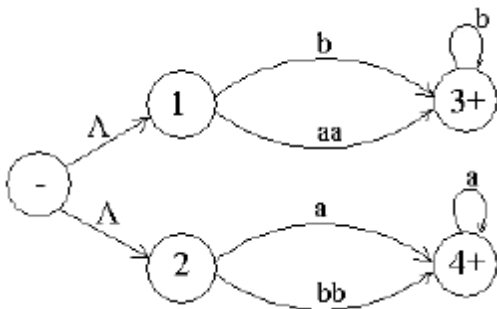


Step 2:

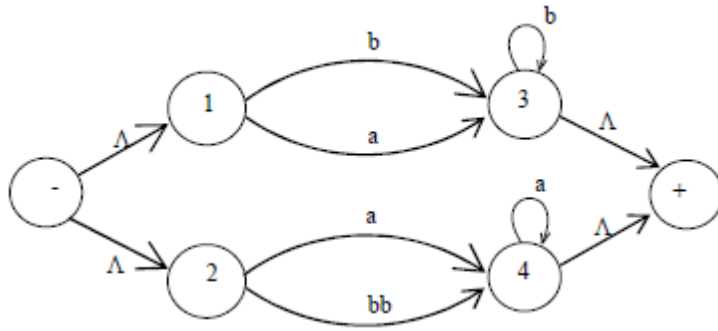
If a TG has more than one final states, then introduce a new final state, connecting the old final states to the new final state by the transitions labeled by Λ .

This step can be shown by the previous example of TG, where the step 1 has already been processed

Example



The above TG can be converted to

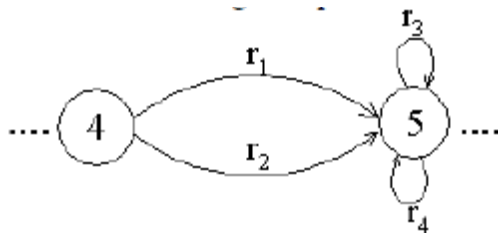


Step 3:

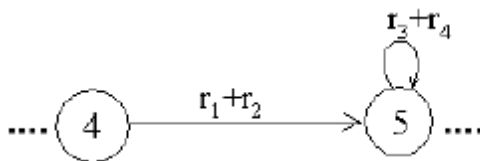
If a state has two (more than one) incoming transition edges labeled by the corresponding REs, from the same state (including the possibility of loops at a state), then replace all these transition edges with a single transition edge labeled by the sum of corresponding REs.

This step can be shown by a part of TG in the following example

Example

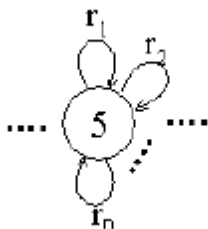


The above TG can be reduced to

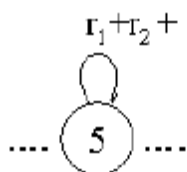


Note

The step 3 can be generalized to any finite number of transitions as shown below

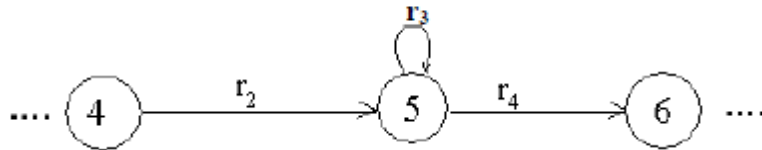


The above TG can be reduced to

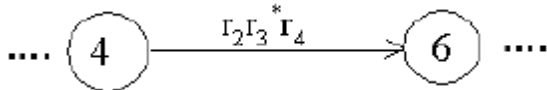


Step 4 (bypass and state elimination)

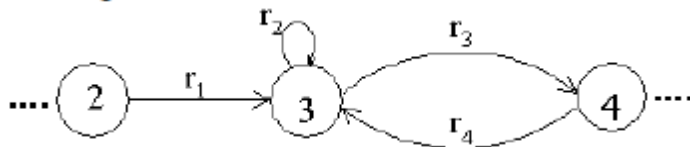
If three states in a TG, are connected in sequence then eliminate the middle state and connect the first state with the third by a single transition (include the possibility of circuit as well) labeled by the RE which is the concatenation of corresponding two REs in the existing sequence. This step can be shown by a part of TG in the following example



To eliminate state 5 the above can be reduced to

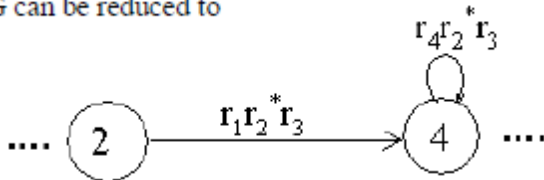


Consider the following example containing a circuit



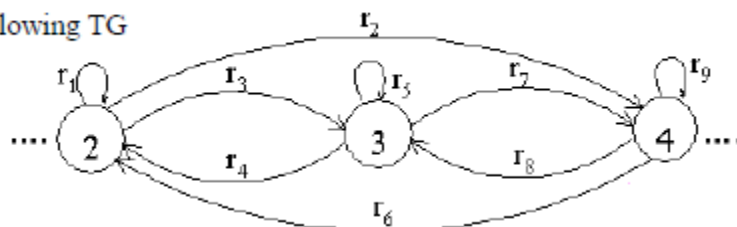
To eliminate state 3 the above TG can be reduced to

TG can be reduced to

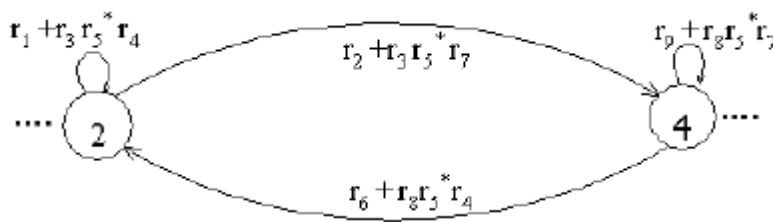


Example


Consider a part of the following TG



To eliminate state 3 the above TG can be reduced to



To eliminate state 4 the above TG can be reduced to

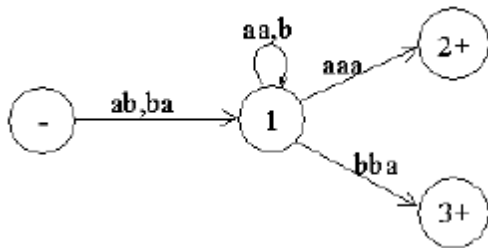
$$(r_1 + r_3 r_5^* r_4) + (r_2 + r_3 r_5^* r_7)(r_9 + r_8 r_5^* r_7)^*(r_6 + r_8 r_5^* r_4)$$


Note

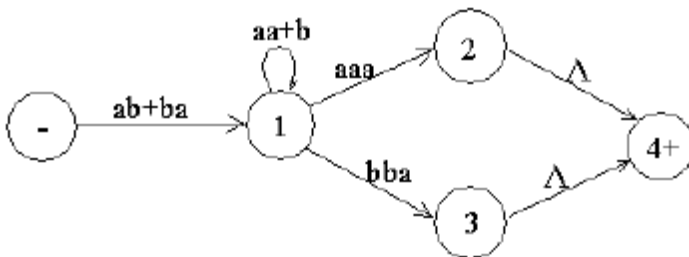
It is to be noted that to determine the RE corresponding to a certain TG, four steps have been discussed. This process can be explained by the following particular examples of TGs

Example

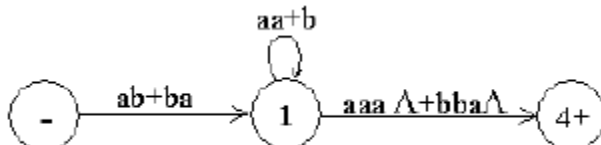
Consider the following TG



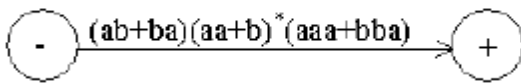
To have single final state, the above TG can be reduced to the following



To eliminate states 2 and 3, the above TG can be reduced to the following



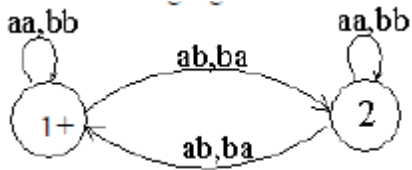
To eliminate state 1 the above TG can be reduced to the following



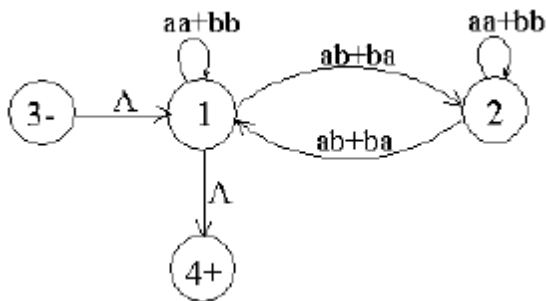
Hence the required RE is $(ab+ba)(aa+b)^*(aaa+bba)$

Example

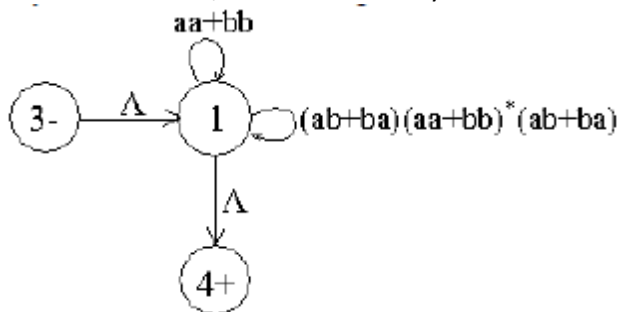
Consider the following TG, accepting EVEN-EVEN language



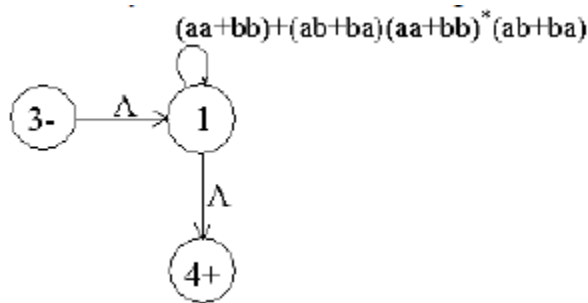
It is to be noted that since the initial state of this TG is final as well and there is no other final state, so to obtain a TG with single initial and single final state, an additional initial and a final state are introduced as shown in the following TG



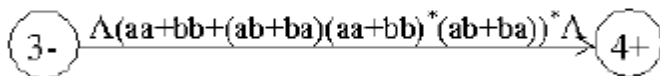
To eliminate state 2, the above TG may be reduced to the following



To have single loop at state 1, the above TG may be reduced to the following



To eliminate state 1, the above TG may be reduced to the following



Hence the required RE is $(aa+bb+(ab+ba)(aa+bb)^*(ab+ba))^*$

Kleene's Theorem Part III

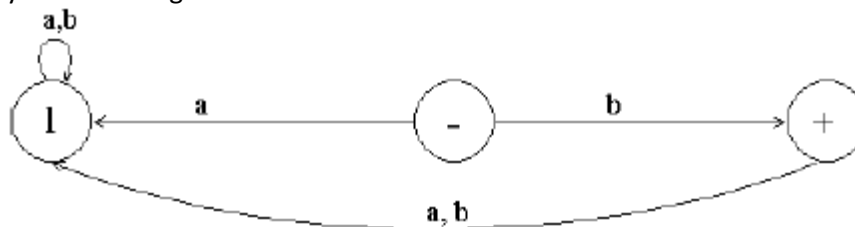
Statement:

If the language can be expressed by a RE then there exists an FA accepting the language.

As the regular expression is obtained applying addition, concatenation and closure on the letters of an alphabet and the Null string, so while building the RE, sometimes, the corresponding FA may be built easily, as shown in the following examples

Example

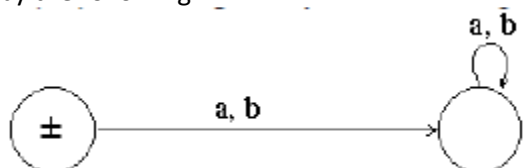
Consider the language, defined over $\Sigma = \{a,b\}$, **consisting of only b**, then this language may be accepted by the following FA



which shows that this FA helps in building an FA accepting only one letter

Example

Consider the language, defined over $\Sigma = \{a,b\}$, **consisting of only Λ**, then this language may be accepted by the following FA

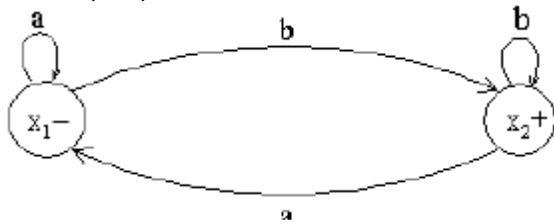


As, if r_1 and r_2 are regular expressions then their sum, concatenation and closure are also regular expressions, so an FA can be built for any regular expression if the methods can be developed for building the FAs corresponding to the sum, concatenation and closure of the regular expressions along with their FAs. These three methods are explained in the following discussion

Method1 (Union of two FAs): Using the FAs corresponding to r_1 and r_2 an FA can be built, corresponding to $r_1 + r_2$. This method can be developed considering the following examples

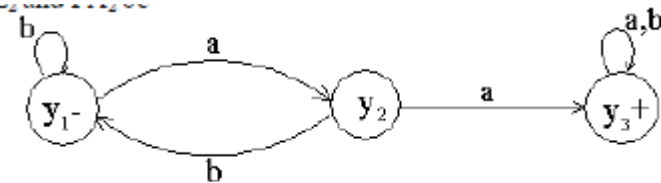
Example

Let $r_1 = (a+b)^*b$ defines L_1 and the FA1 be



and $r_2 = (a+b)^*aa(a+b)^*$ defines L_2

and FA2 be

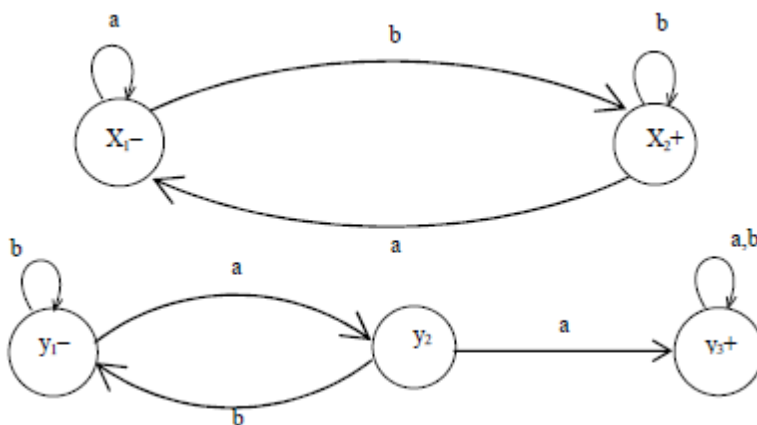


Let FA3 be an FA corresponding to $r_1 + r_2$, then the initial state of FA3 must correspond to the initial state of FA1 and the initial state of FA2.

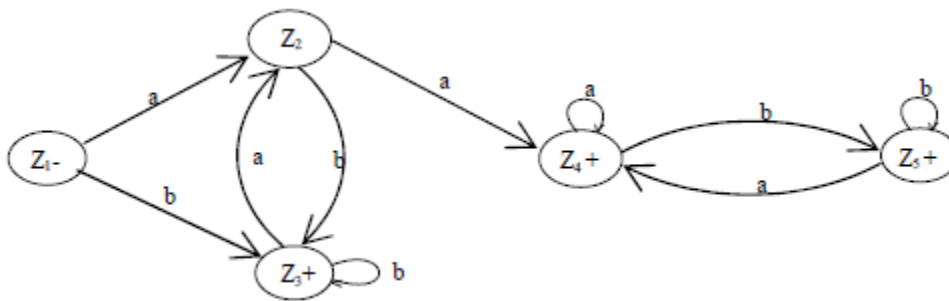
Since the language corresponding to $r_1 + r_2$ is the union of corresponding languages L_1 and L_2 , consists of the strings belonging to L_1 or L_2 or both, therefore a final state of FA3 must correspond to a final state of FA1 or FA2 or both.

Since, in general, FA3 will be different from both FA1 and FA2, so the labels of the states of FA3 may be supposed to be z_1, z_2, z_3, \dots , where z_1 is supposed to be the initial state. Since z_1 corresponds to the states x_1 or y_1 ,

so there will be two transitions separately for each letter read at z_1 . It will give two possibilities of states either z_1 or different from z_1 . This process may be expressed in the following transition table for all possible states of FA3.



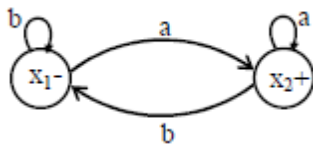
Old States	New States after reading	
	a	b
$z_1 \equiv (x_1, y_1)$	$(x_1, y_2) \equiv z_2$	$(x_2, y_1) \equiv z_3$
$z_2 \equiv (x_1, y_2)$	$(x_1, y_3) \equiv z_4$	$(x_2, y_1) \equiv z_3$
$z_3 \equiv (x_2, y_1)$	$(x_1, y_2) \equiv z_2$	$(x_2, y_1) \equiv z_3$
$z_4 \equiv (x_1, y_3)$	$(x_1, y_3) \equiv z_4$	$(x_2, y_3) \equiv z_5$
$z_5 \equiv (x_2, y_3)$	$(x_1, y_3) \equiv z_4$	$(x_2, y_3) \equiv z_5$



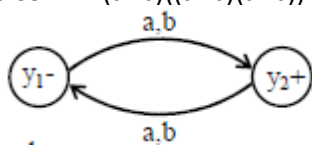
RE corresponding to the above FA may be $r_1 + r_2 = (a+b)^*b + (a+b)^*aa(a+b)^*$.

Example

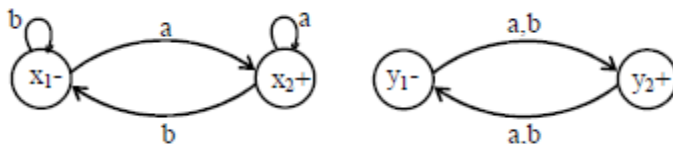
Let $r_1 = (a+b)^*a$ and the corresponding FA1 be



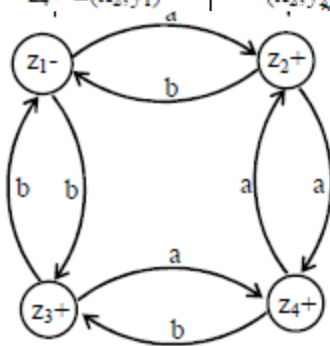
also $r_2 = (a+b)((a+b)(a+b))^*$ or $((a+b)(a+b))^*(a+b)$ and FA2 be



FA corresponding to $r_1 + r_2$ can be determined as

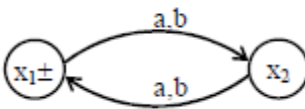


Old States	New States after reading	
	a	b
$z_1 \equiv (x_1, y_1)$	$(x_2, y_2) \equiv z_2$	$(x_1, y_2) \equiv z_3$
$z_2 \equiv (x_2, y_2)$	$(x_2, y_1) \equiv z_4$	$(x_1, y_1) \equiv z_1$
$z_3 \equiv (x_1, y_2)$	$(x_2, y_1) \equiv z_4$	$(x_1, y_1) \equiv z_1$
$z_4 \equiv (x_2, y_1)$	$(x_2, y_2) \equiv z_2$	$(x_1, y_2) \equiv z_3$

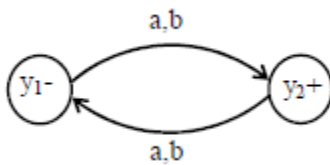


Example

Let $r_1 = ((a+b)(a+b))^*$ and the corresponding FA1 be



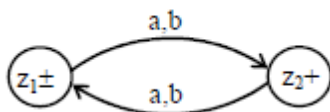
also $r_2 = (a+b)((a+b)(a+b))^*$ or $((a+b)(a+b))^*(a+b)$ and FA2 be



FA corresponding to r_1+r_2 can be determined as

Old States	New States after reading	
	a	b
$z_1 \equiv (x_1, y_1)$	$(x_2, y_2) \equiv z_2$	$(x_2, y_2) \equiv z_2$
$z_2 \equiv (x_2, y_2)$	$(x_1, y_1) \equiv z_1$	$(x_1, y_1) \equiv z_1$

Hence the required FA will be as follows

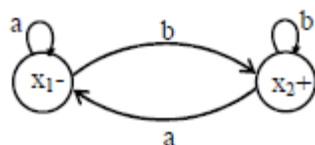


Method2 (Concatenation of two FAs):

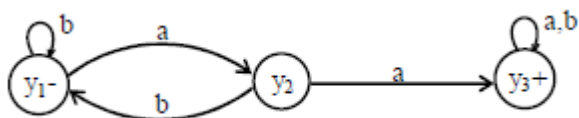
Using the FAs corresponding to r_1 and r_2 , an FA can be built, corresponding to r_1r_2 . This method can be developed considering the following examples

Example

Let $r_1 = (a+b)^*b$ defines L_1 and FA1 be



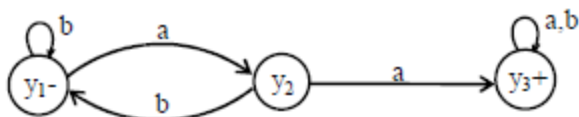
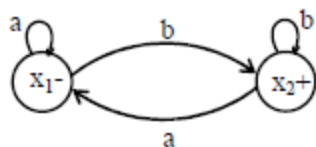
and $r_2 = (a+b)^*aa(a+b)^*$ defines L_2
and FA2 be



Let FA3 be an FA corresponding to r_1r_2 , then the initial state of FA3 must correspond to the initial state of FA1 and the final state of FA3 must correspond to the final state of FA2. Since the language corresponding to r_1r_2 is the concatenation of corresponding languages L_1 and L_2 , consists of the strings obtained, concatenating the strings of L_1 to those of L_2 , therefore **the moment a final state of first FA is entered, the possibility of the initial state of second FA will be included as well.**

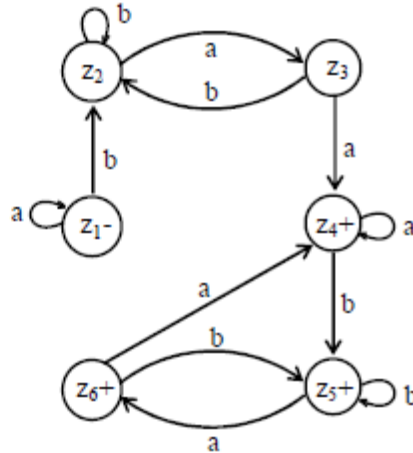
Since, in general, FA3 will be different from both FA1 and FA2, so the labels of the states of FA3 may be supposed to be z_1, z_2, z_3, \dots , where z_1 stands for the initial state. Since z_1 corresponds to the states x_1 , so there will be two transitions separately for each letter read at z_1 . It will give two possibilities of states which correspond to either z_1 or different from z_1 . This process may be expressed in the following transition table for all possible states of FA3

Hence the required FA will be as follows



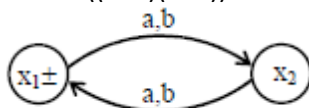
Old States	New States after reading	
	a	b
$Z_1 \equiv X_1$	$X_1 \equiv Z_1$	$(X_2, Y_1) \equiv Z_2$
$Z_2 \equiv (X_2, Y_1)$	$(X_1, Y_2) \equiv Z_3$	$(X_2, Y_1) \equiv Z_2$
$Z_3 \equiv (X_1, Y_2)$	$(X_1, Y_3) \equiv Z_4$	$(X_2, Y_1) \equiv Z_2$
$Z_4 \equiv (X_1, Y_3)$	$(X_1, Y_3) \equiv Z_4$	$(X_2, Y_1, Y_3) \equiv Z_5$
$Z_5 \equiv (X_2, Y_1, Y_3)$	$(X_1, Y_2, Y_3) \equiv Z_6$	$(X_2, Y_1, Y_3) \equiv Z_5$
$Z_6 \equiv (X_1, Y_2, Y_3)$	$(X_1, Y_3) \equiv Z_4$	$(X_2, Y_1, Y_3) \equiv Z_5$

Hence the required FA will be as follows

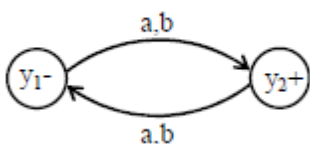


Example

Let $r_1 = ((a+b)(a+b))^*$ and the corresponding FA1 be



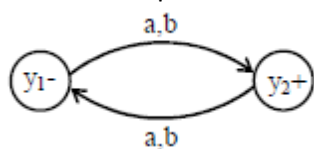
also $r_2 = (a+b)((a+b)(a+b))^*$ or $((a+b)(a+b))^*(a+b)$ and FA2 be



FA corresponding to r_1r_2 can be determined as

Old States	New States after reading	
	a	b
$z_1 \equiv (x_1, y_1)$	$(x_2, y_2) \equiv z_2$	$(x_2, y_2) \equiv z_2$
$z_2 \equiv (x_2, y_2)$	$(x_1, y_1) \equiv z_1$	$(x_1, y_1) \equiv z_1$

Hence the required FA will be as follows



Method3: (Closure of an FA)

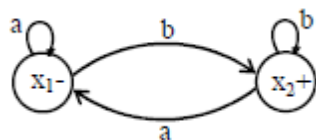
Building an FA corresponding to r^* , using the FA corresponding to r .

It is to be noted that if the given FA already accepts the language expressed by the closure of certain RE, then the given FA is the required FA. However the method, in other cases, can be developed considering the following examples

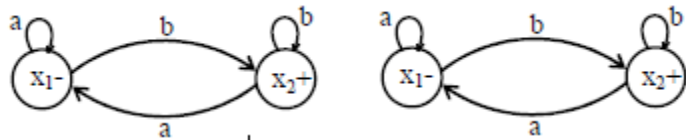
Closure of an FA, is same as concatenation of an FA with itself, except that the initial state of the required FA is a final state as well. Here the initial state of given FA, corresponds to the initial state of required FA and a non final state of the required FA as well.

Example

Let $r = (a+b)^*b$ and the corresponding FA be

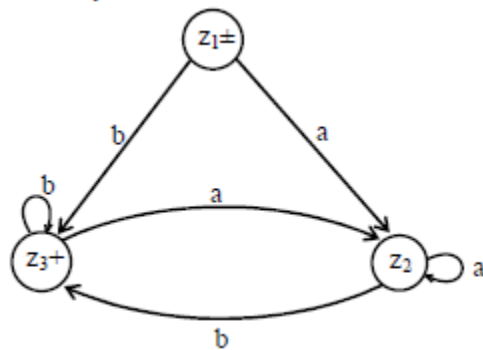


then the FA corresponding to r^* may be determined as under



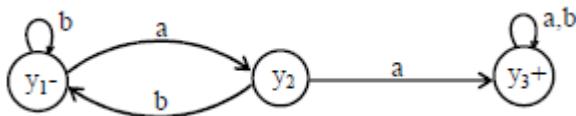
Old States	New States after reading	
	a	b
Final $z_1 \equiv x_1$	$x_1 \equiv z_2$	$(x_2, x_1) \equiv z_3$
Non-final $z_2 \equiv x_1$	$x_1 \equiv z_2$	$(x_2, x_1) \equiv z_3$
$z_3 \equiv (x_2, x_1)$	$x_1 \equiv z_2$	$(x_2, x_1) \equiv z_3$

The corresponding transition diagram may be as under



Example

Let $r = (a+b)^*aa(a+b)^*$ and the corresponding FA be



then the FA corresponding to r^* may be determined as under

Old States	New States after reading	
	a	b
Final $z_1 \equiv y_1$	$y_2 \equiv z_3$	$y_1 \equiv z_2$
Non-Final $z_2 \equiv y_1$	$y_2 \equiv z_3$	$y_1 \equiv z_2$
$z_3 \equiv y_2$	$(y_3, y_1) \equiv z_4$	$y_1 \equiv z_2$
$z_4 \equiv (y_3, y_1)$	$(y_3, y_1, y_2) \equiv z_5$	$(y_3, y_1) \equiv z_4$
$z_5 \equiv (y_3, y_1, y_2)$	$(y_3, y_1, y_2) \equiv z_5$	$(y_3, y_1) \equiv z_4$

The corresponding transition diagram may be as under

