Recap Definition of TG continued …

Definition: A Transition graph (TG), is a collection of the followings

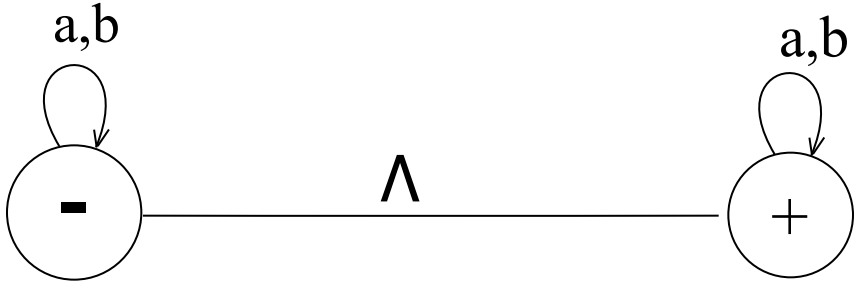
1) Finite number of states, at least one of which is start state and some (maybe none) final states.
2) Finite set of input letters (Σ) from which input strings are formed.
3) Finite set of transitions that show how to go from one state to another based on reading specified substrings of input letters, 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

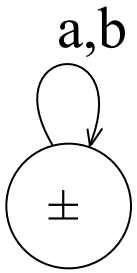
- Consider the Language L , defined over Σ = {a, b} of **all strings including Λ.** The language L may be accepted by the following TG



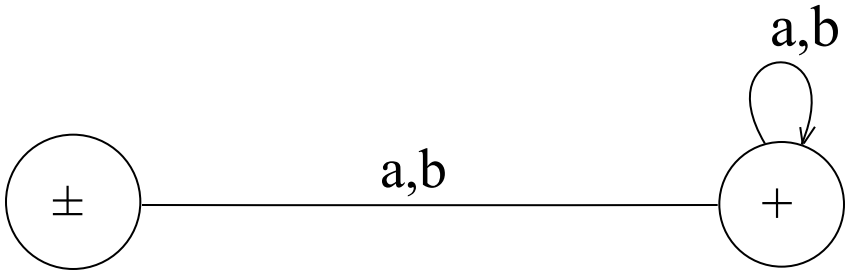
The language L may also be accepted by the following TG

Example Continued ...

TG_1



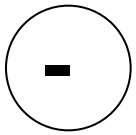
TG_2



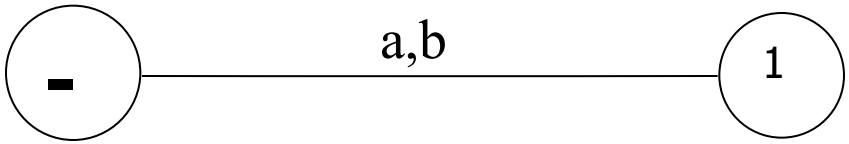
Example

- Consider the following TGs

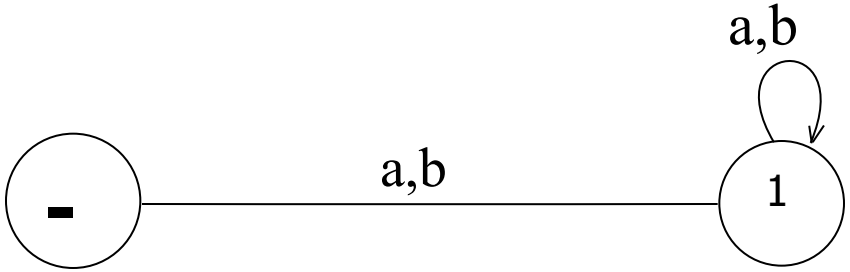
TG_1



TG_2



TG_3



Example Continued …

- It may be observed that in the first TG, no transition has been shown. Hence this TG does not accept any string, defined over any alphabet. In TG_2 there are transitions for a and b at initial state but there is no transition at state 1. This TG still does not accept any string. In TG_3 there are transitions at both initial state and state 1, but it does not accept any string.

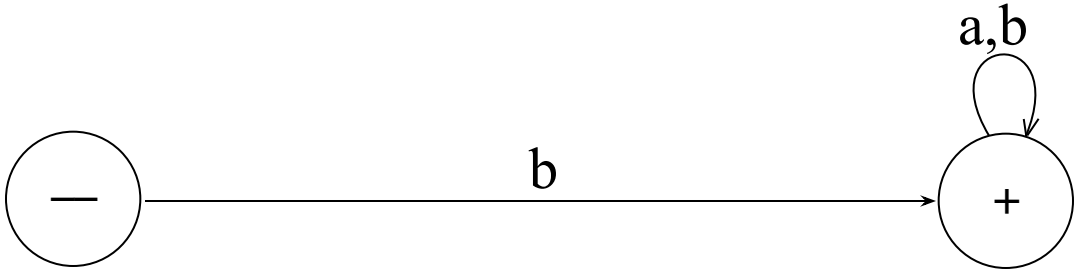
Example Continued …

Thus none of TG_1, TG_2 and TG_3 accepts any string, *i.e.* these TGs accept empty language. It may be noted that TG_1 and TG_2 are TGs but not FA, while TG_3 is both TG and FA as well.

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.

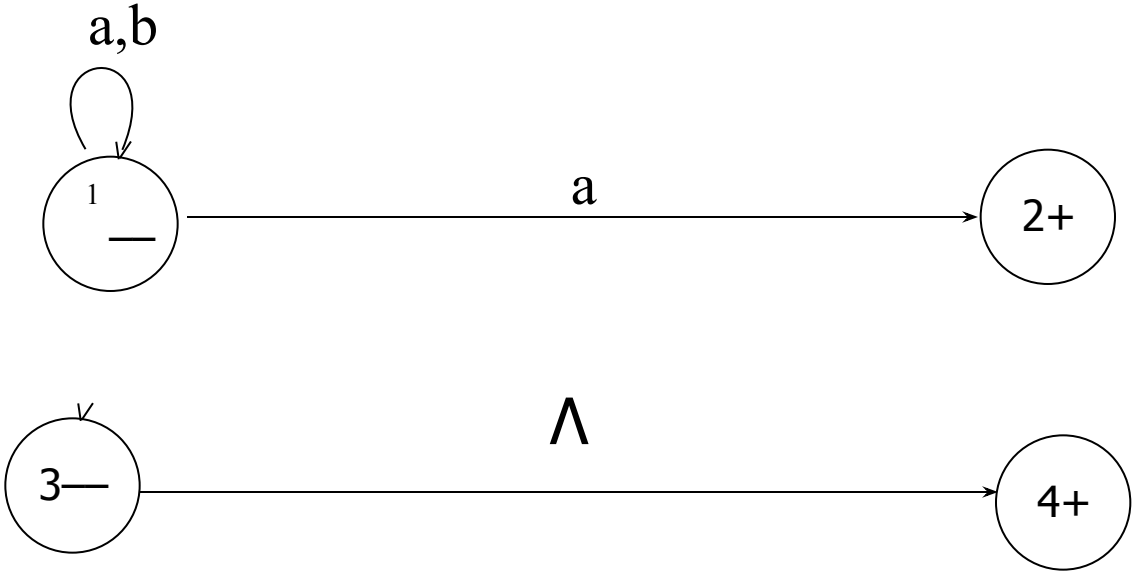
Example

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



TASK

- Build a TG accepting the language of strings, defined over Σ={a, b}, **ending in b.**

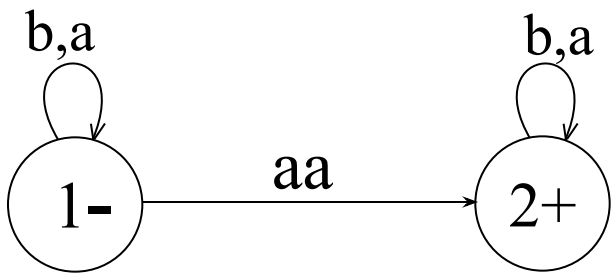
Example

Consider the Language L of strings, defined over Σ = {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 Continued ...



Example

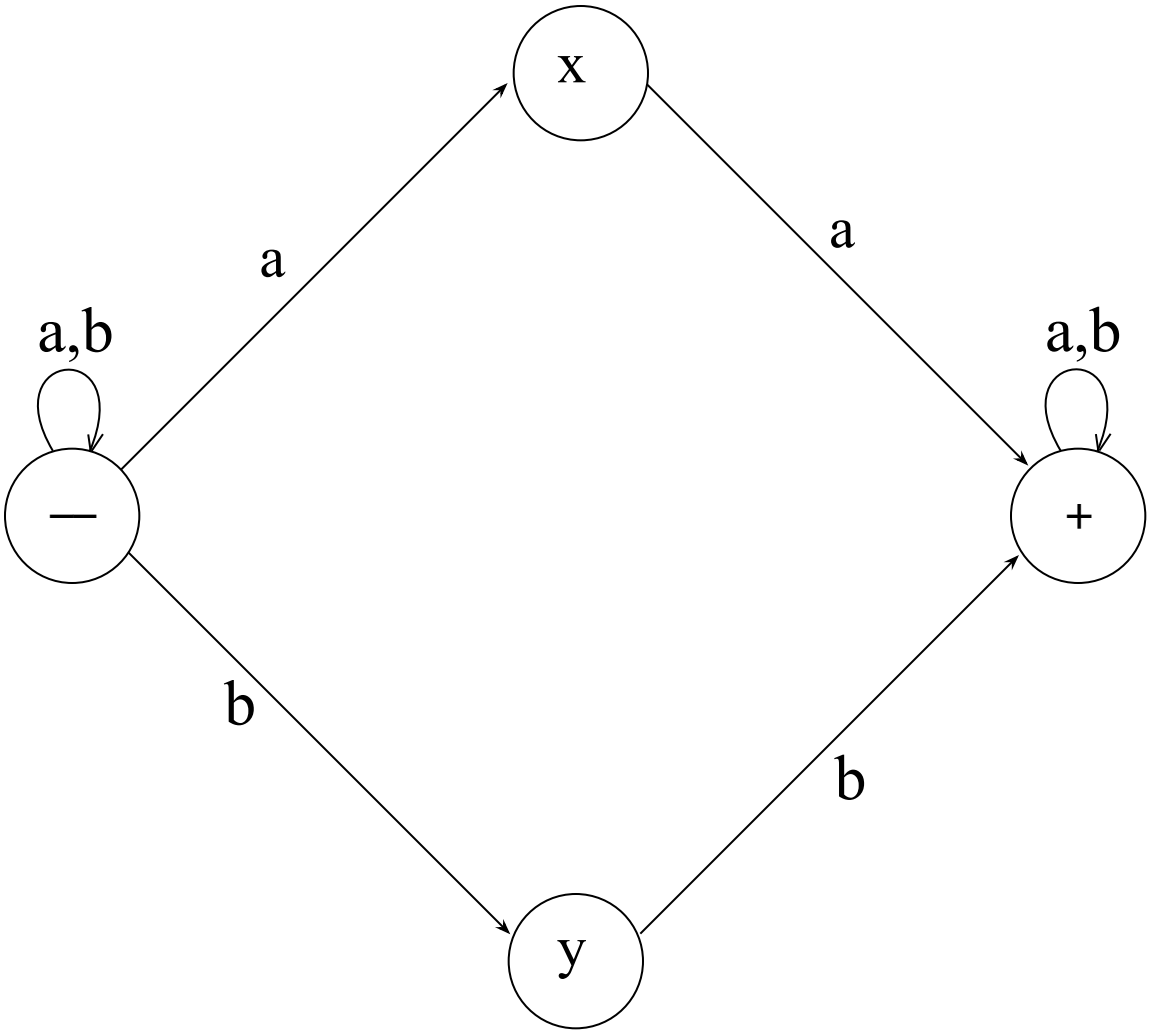
Consider the language L of strings, defined over Σ={a, b}, **having double a or double b**.

The language L can be expressed by RE

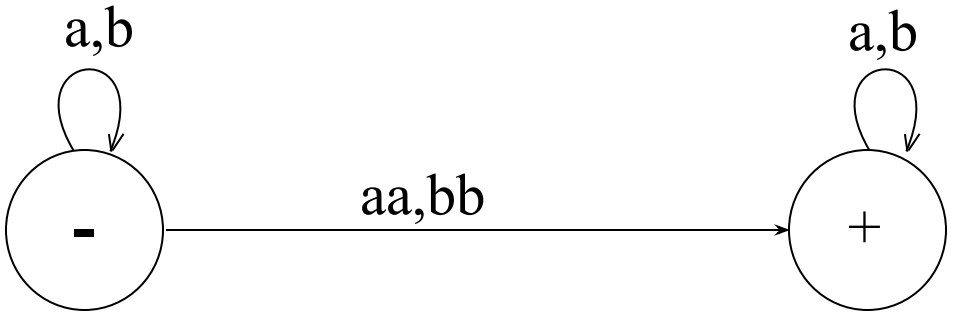
$(a+b)^* (aa + bb) (a+b)^*$.

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

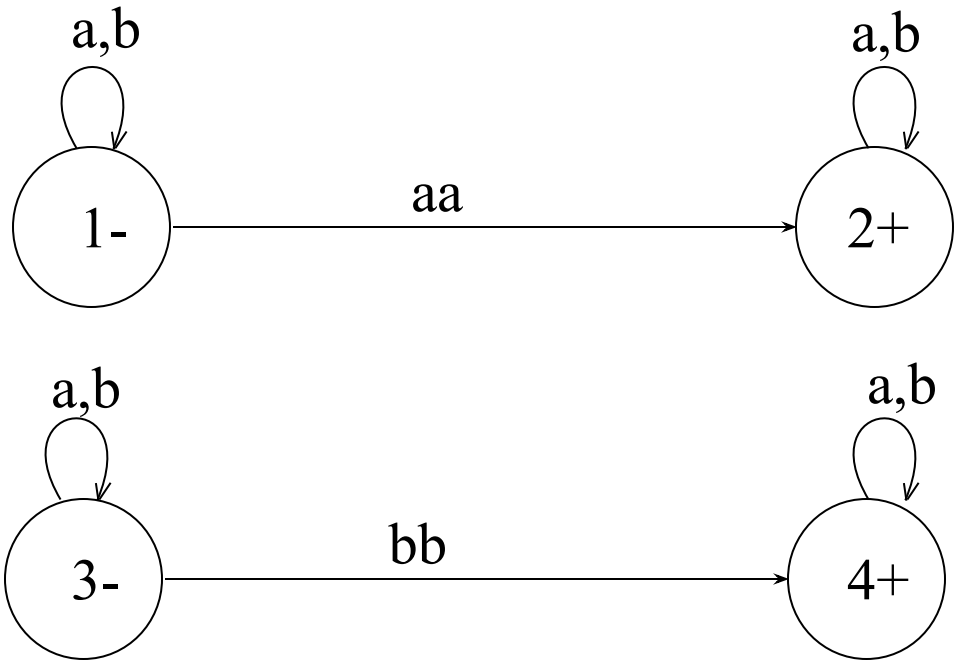
Example continued ...



OR



OR

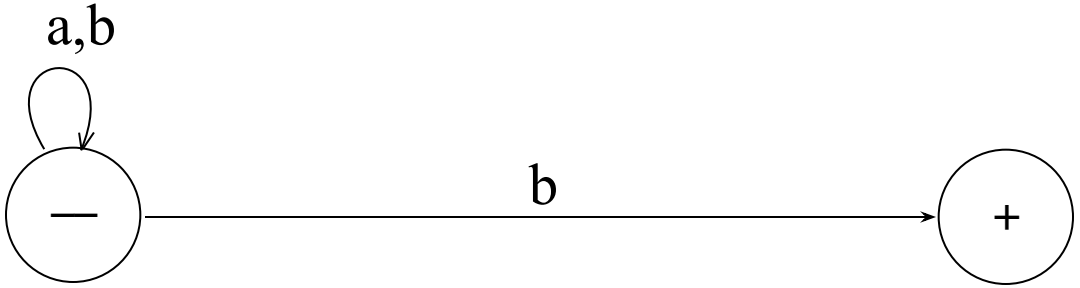


Note

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

Task Solution

- Build a TG accepting the language L of strings, defined over Σ={a, b}, **ending in b.**
- **Solution** The language L may be expressed by RE $(a + b)^*b$, may be accepted by the following TG



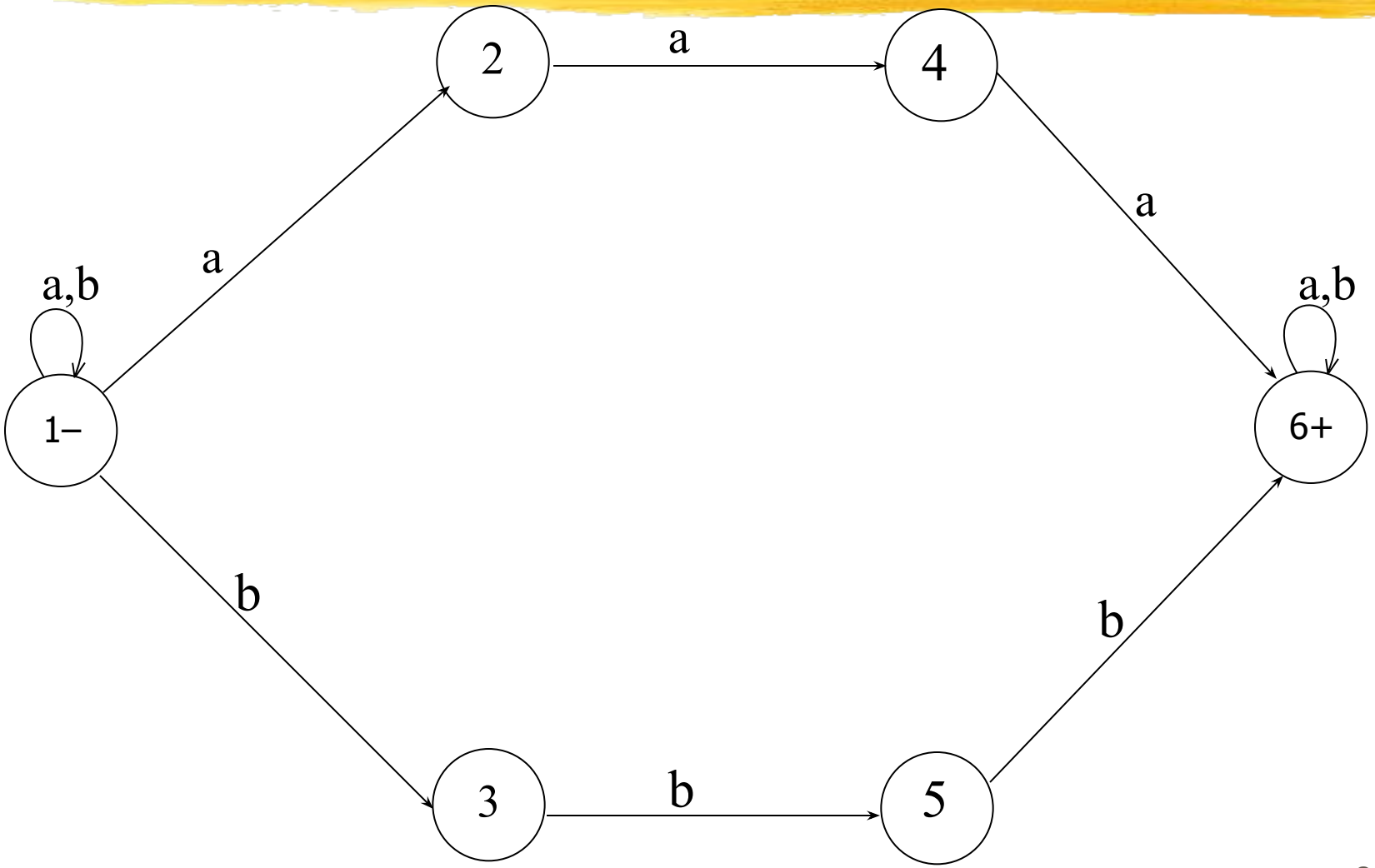
Example

Consider the language L of strings, defined over Σ={a, b}, **having triple a or triple b.** The language L may be expressed by RE

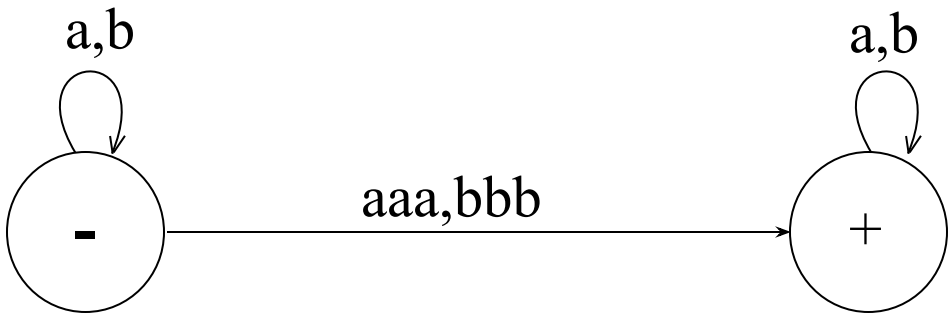
$$(a+b)^* (aaa + bbb) (a+b)^*$$

This language may be accepted by the following TG

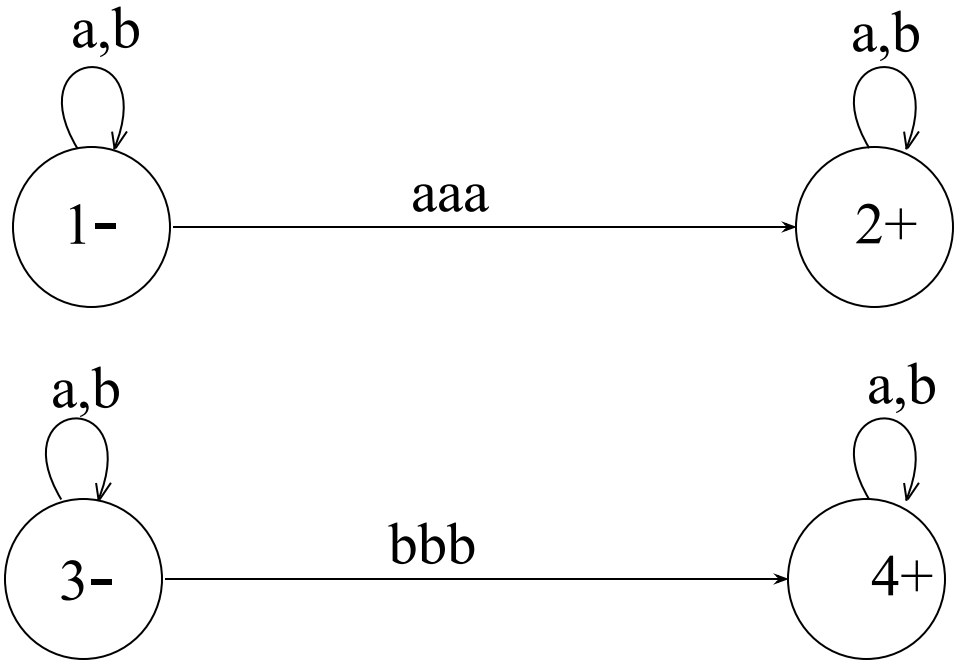
Example Continued ...



OR



OR



Example

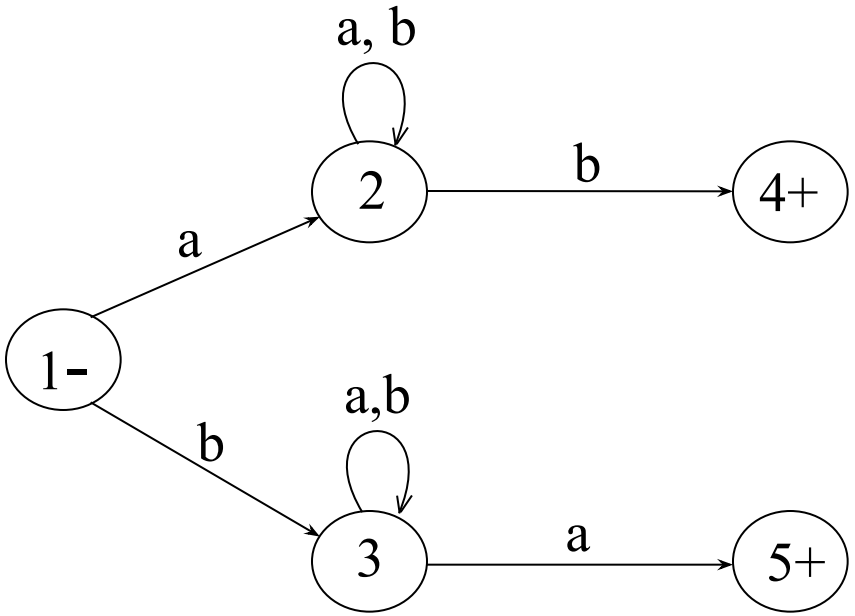
Consider the language L of strings, defined over Σ = {a, b}, **beginning and ending in different letters**.

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 continued ...



Example

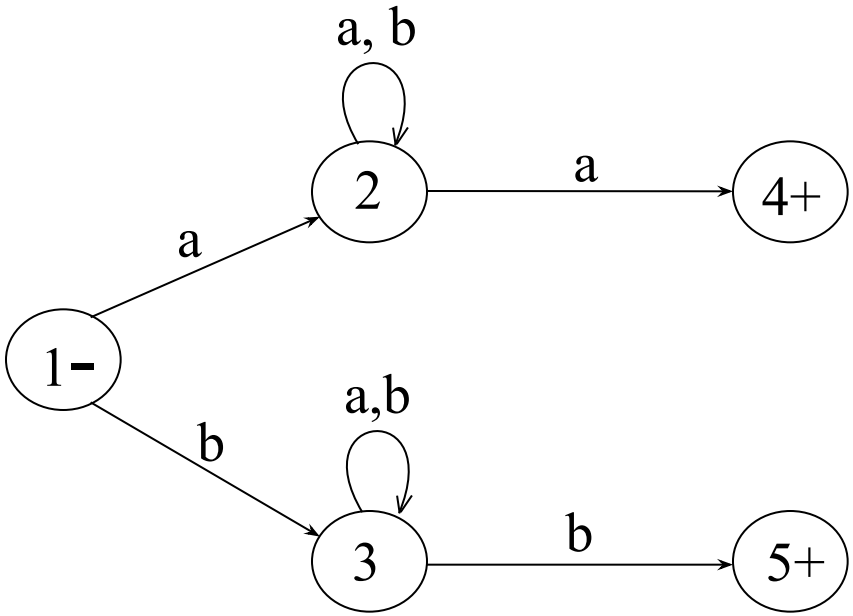
- Consider the Language L of strings **of length two or more**, defined over Σ = {a, b}, **beginning with and ending in same letters.**

  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

Example Continued ...



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

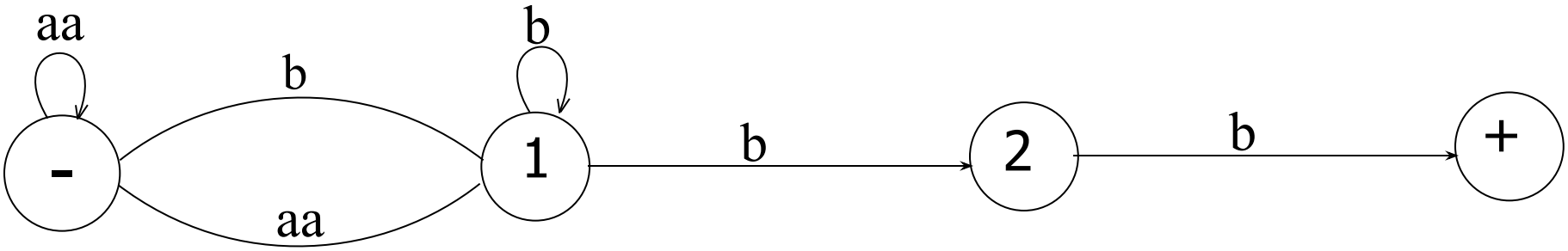
  $(aa)^*b(b^*+(aa(aa)^*b)^*)\,bb$

  OR

  $(aa)^*b(b^*+(\,(aa)^+b)^*)\,bb$

  The language L may be accepted by the following TG

Example Continued ...

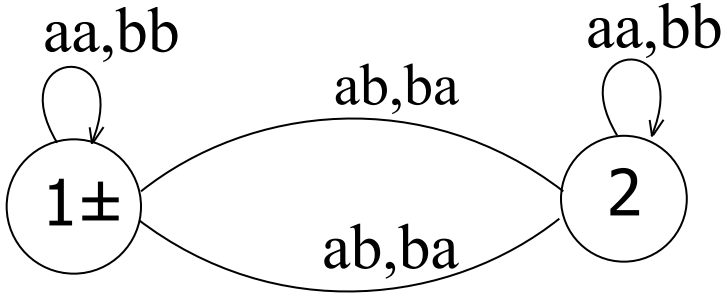


Example

- Consider the **EVEN-EVEN** language, defined over Σ={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 continued ...



Recap lecture 9

- **TGs accepting the languages:containing aaa or bbb, beginning and ending in different letters, beginning and ending in same letters, EVEN-EVEN, a's occur in even clumps and ends in three or more b's, example showing different paths traced by one string, Definition of GTG**

Task Solution …

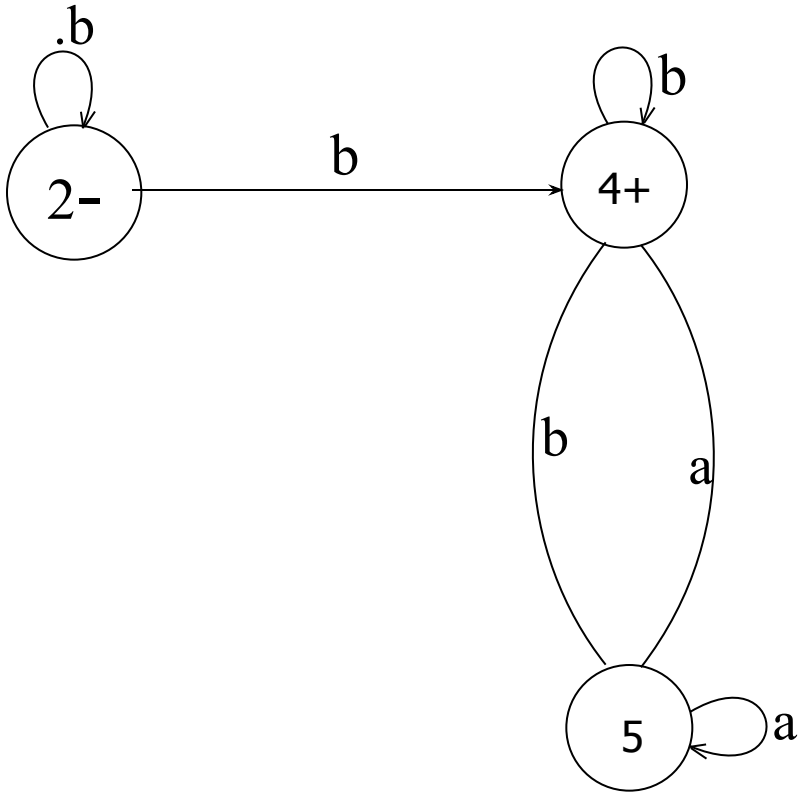
Build a TG accepting the language L of strings, defined over Σ={a, b}, **beginning with and ending in the same letters**

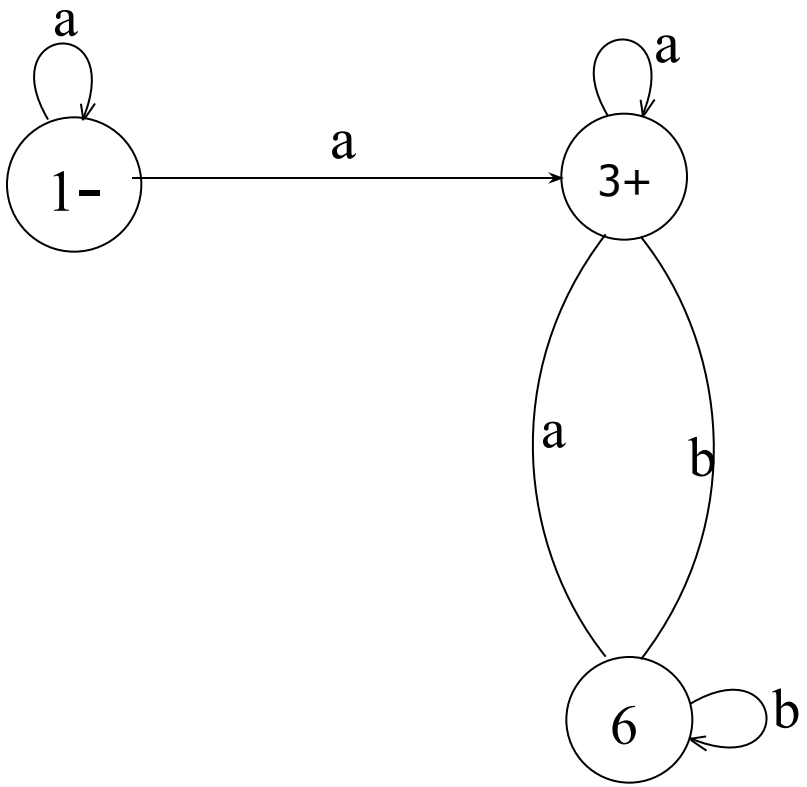
Solution:The language L may be expressed by

$a+b+a(a + b)^*a + b(a + b)^*b$

The language L may be accepted by the following TG

Task Solution





Generalized Transition Graphs

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

1) Finite number of states, at least one of which is start state and some (maybe none) final states.
2) Finite set of input letters (Σ) from which input strings are formed.
3) 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

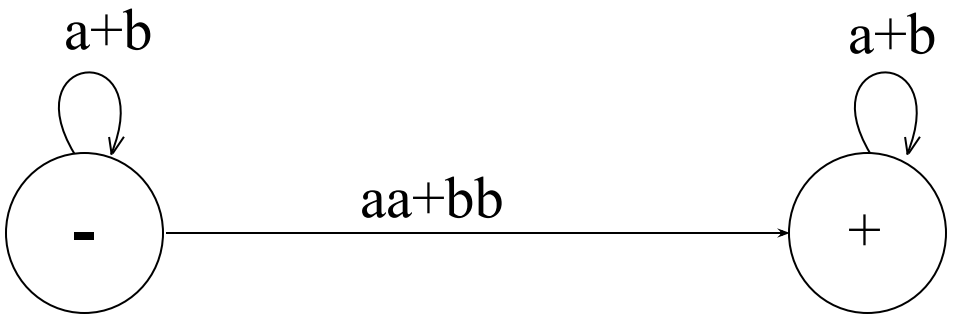
Example

Consider the language L of strings, defined over Σ={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 continued ...



Example

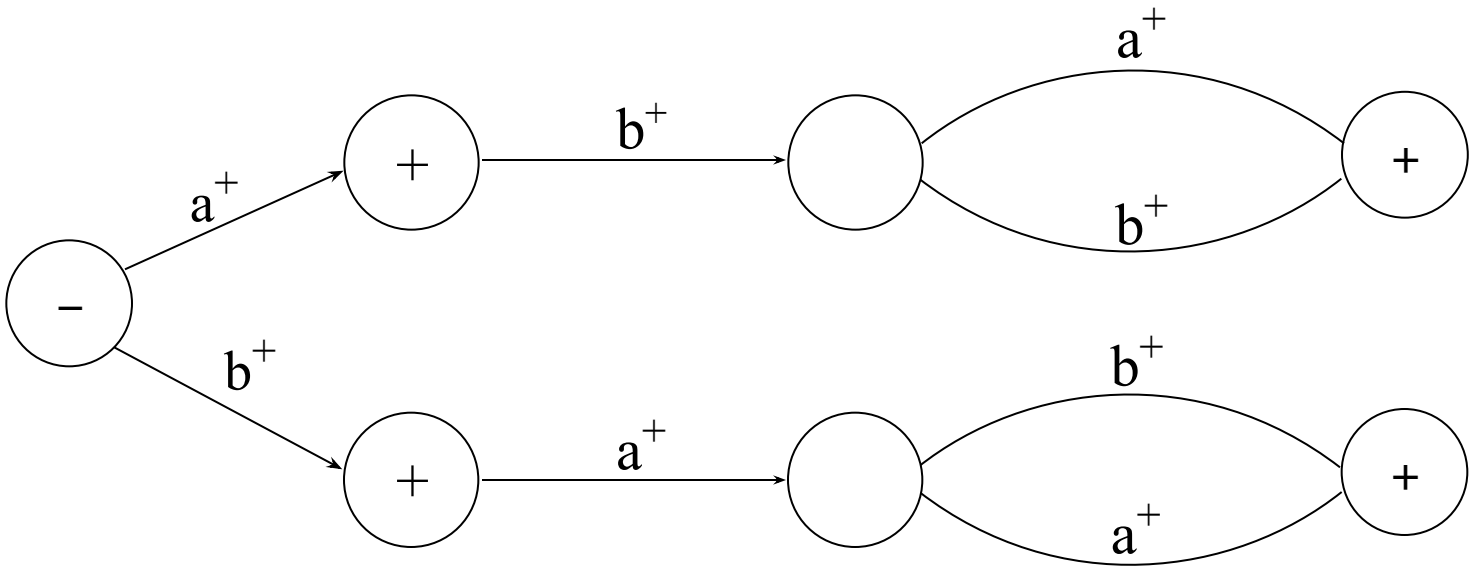
- Consider the Language L of strings, defined over Σ = {a, b}, **beginning with and ending in same letters.**

  The language L may be expressed by the following regular expression

  $$(a+b)+ \ a(a + b)^*a + b(a + b)^*b$$

  This language may be accepted by the following GTG

Example



Example

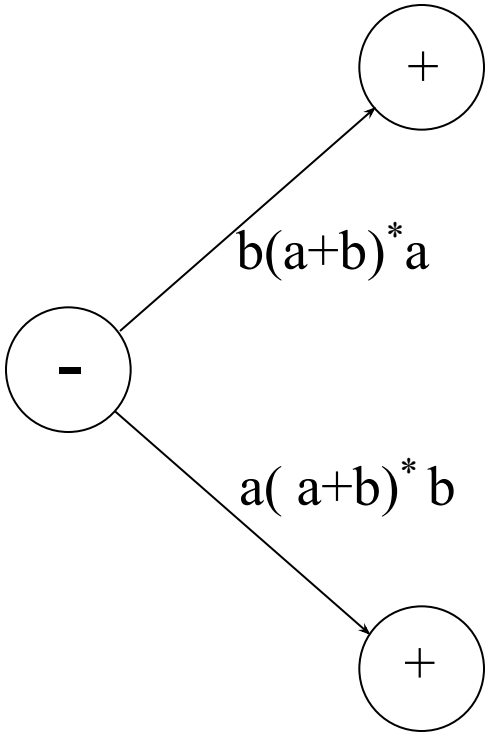
Consider the language L of strings of, defined over $\Sigma = \{a, b\}$, **beginning and ending in different letters**.

The language L may be expressed by RE

$$a(a + b)^*b + b(a + b)^*a$$

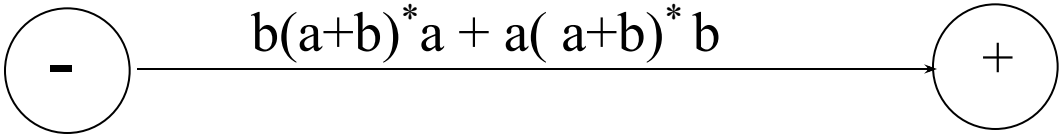
The language L may be accepted by the following GTG

Example Continued …



The language L may be accepted by the following GTG as well

Example Continued ...



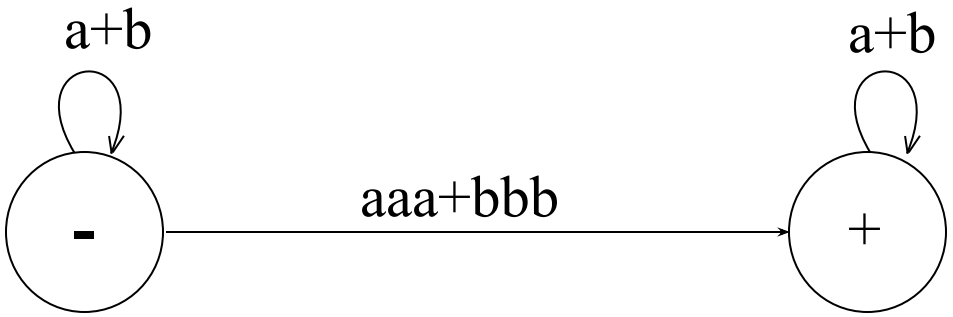
Example

Consider the language L of strings, defined over Σ={a, b}, **having triple a or triple b.** The language L may be expressed by RE

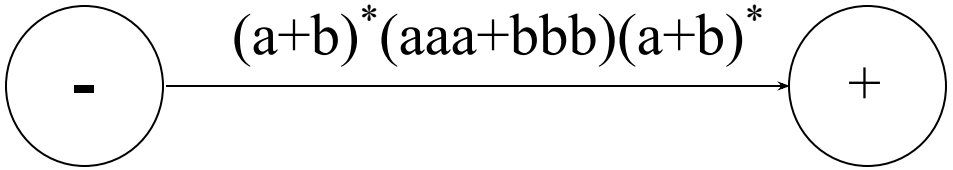
$$(a+b)^* (aaa + bbb) (a+b)^*$$

This language may be accepted by the following GTG

Example Continued ...



OR



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

1. FA or
2. TG or
3. 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 continued ...

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.

Kleene's Theorem continued …

Proof(Kleene's Theorem Part I)

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

Kleene's Theorem continued …

Proof(Kleene's Theorem Part II)

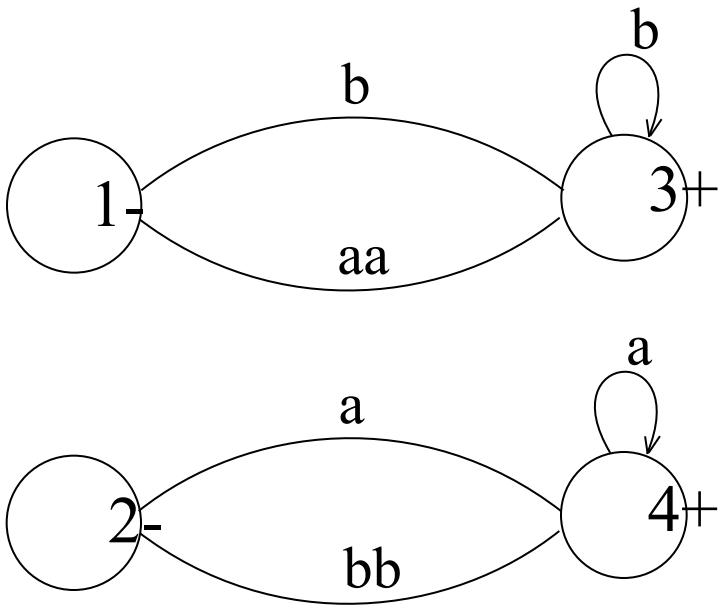
To prove part II of the theorem, an algorithm consisting of different steps, is explained showing how a RE can be obtained corresponding to the given TG. For this purpose the notion of TG is changed to that of GTG *i.e.* the labels of transitions are corresponding REs.

Kleene's Theorem part II continued …

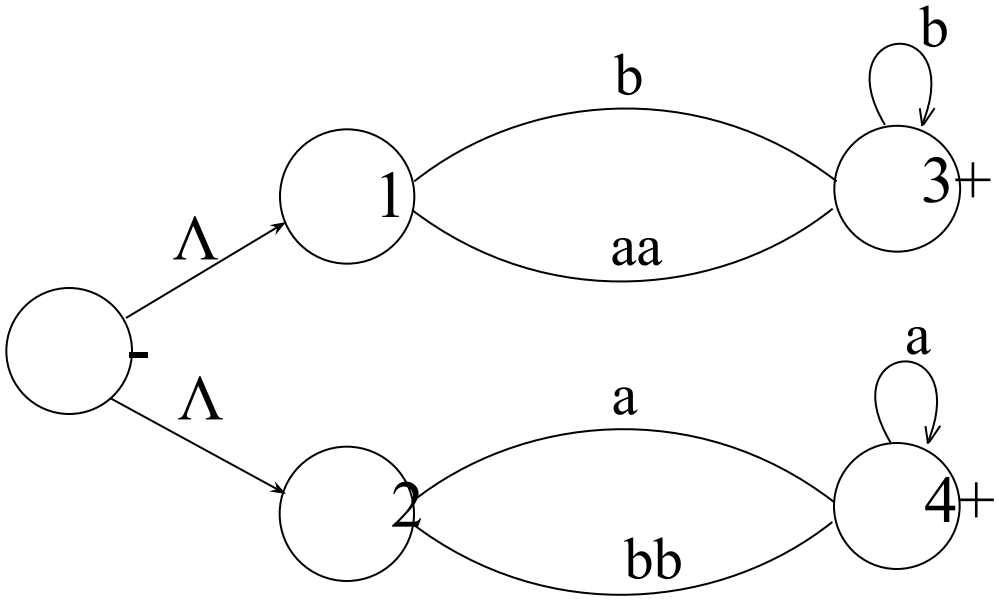
Basically this algorithm converts the given TG to GTG with one initial state along with a single loop, or one initial state connected with one final state by a single transition edge. The label of the loop or the transition edge will be the required RE.

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



Example Continued ...



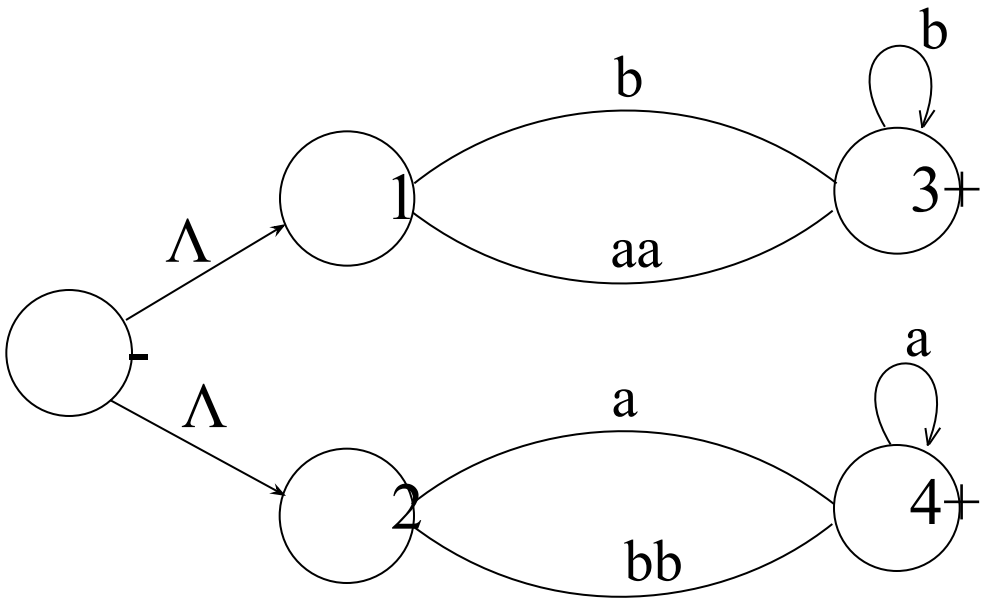
Kleene's Theorem part II continued ...

Step 2:

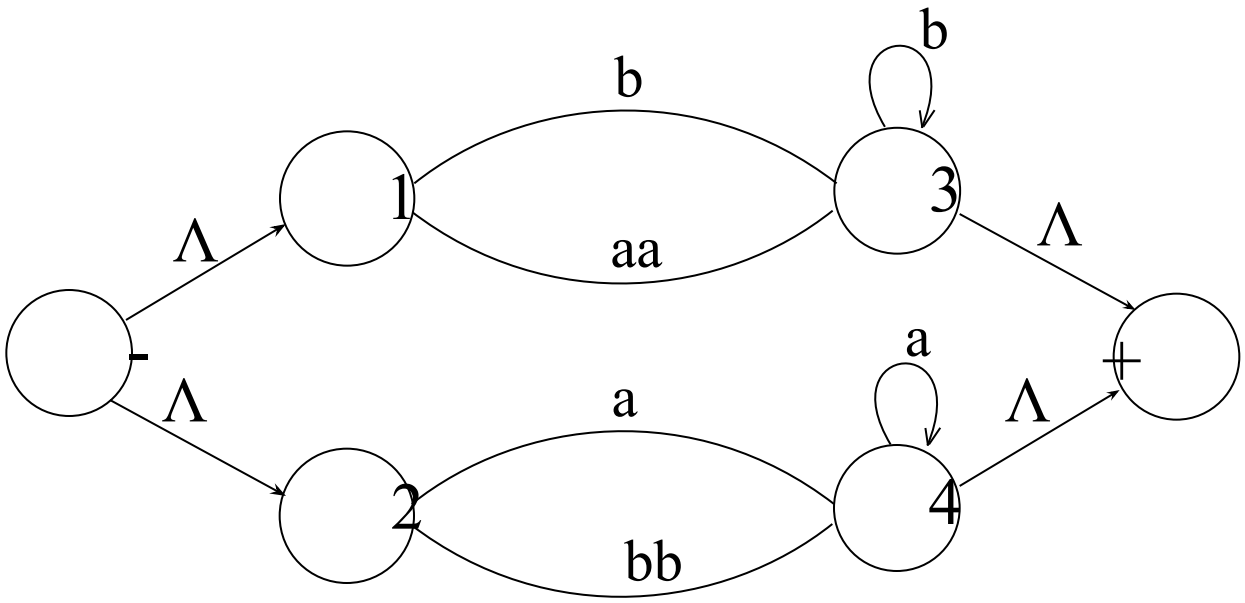
If a TG has more than one final states, then introduce a new final state, connecting the old final states to the new final state by the transitions labeled by Λ.

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

Example



Example continued ...



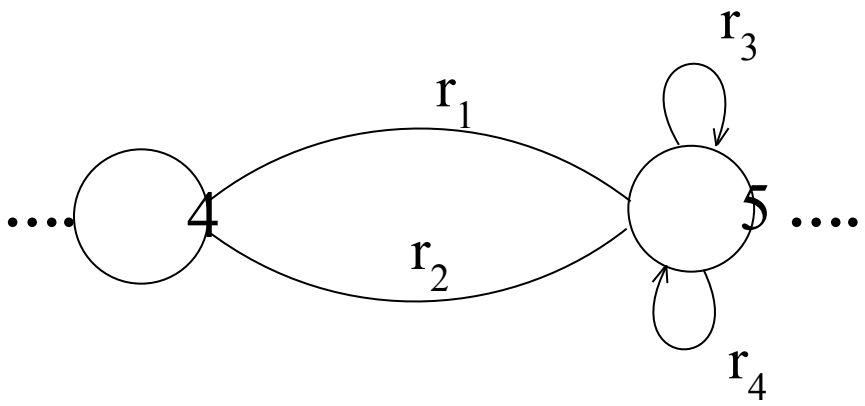
Kleene's Theorem part II continued …

Step 3:

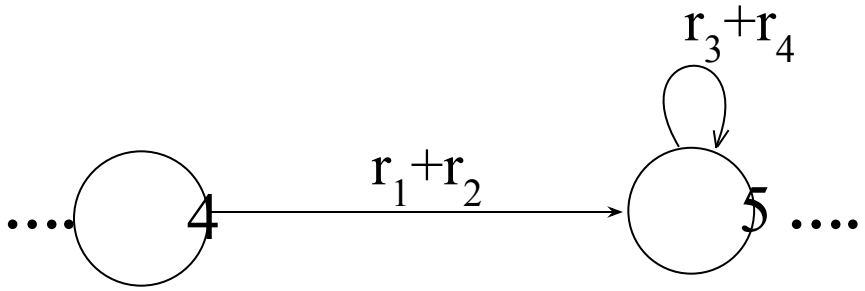
If a state has two (more than one) incoming transition edges labeled by the corresponding REs, from the same state (including the possibility of loops at a state), then replace all these transition edges with a single transition edge labeled by the sum of corresponding REs.

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

Example

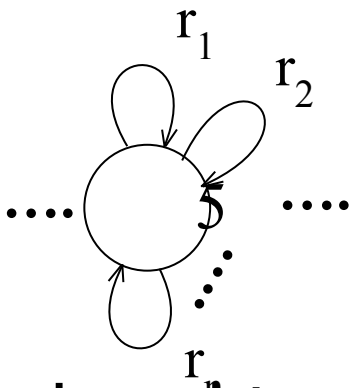


The above TG can be reduced to

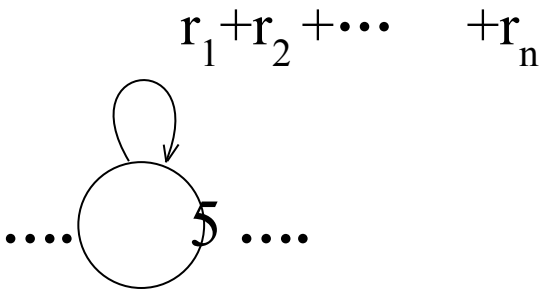


Note

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



The above TG can be reduced to

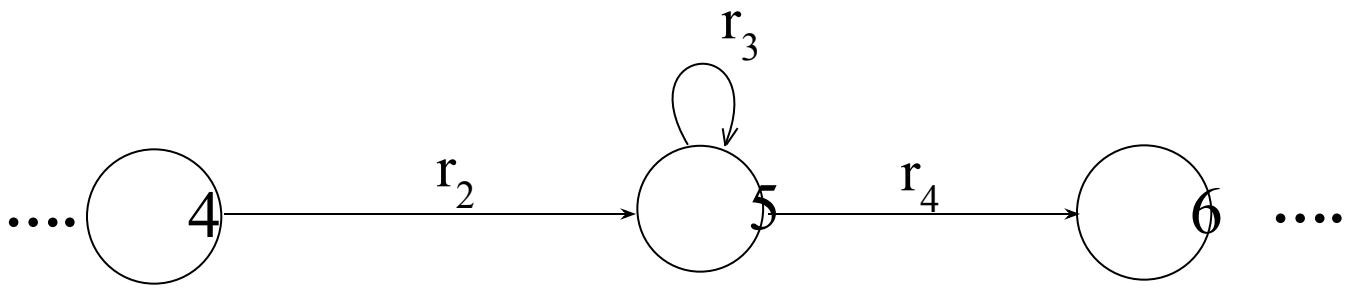


Kleene's Theorem part II continued ...

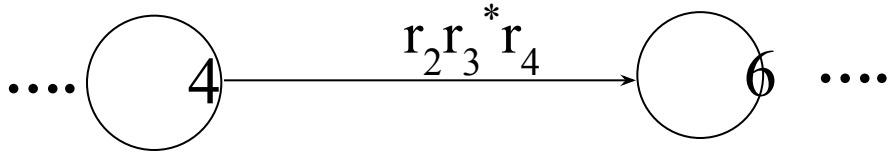
Step 4 (bypass and state elimination)

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

Example



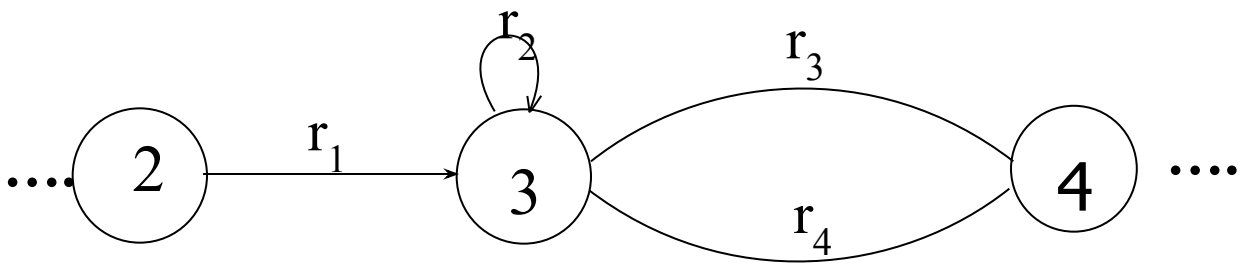
To eliminate state 5 the above can be reduced to



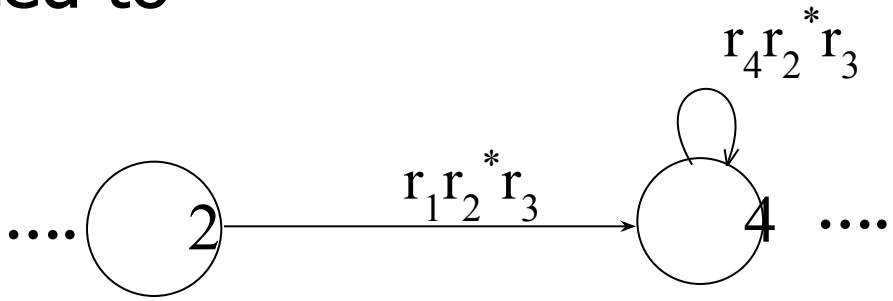
Consider the following example containing a circuit

Example

Consider the part of a TG, containing a circuit at a state, as shown below

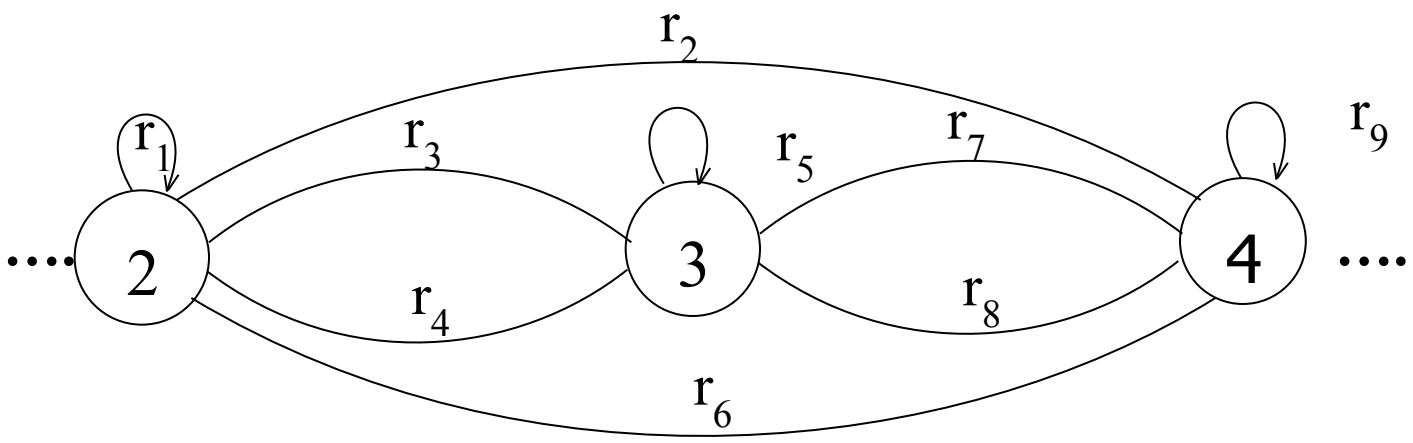


To eliminate state 3 the above TG can be reduced to

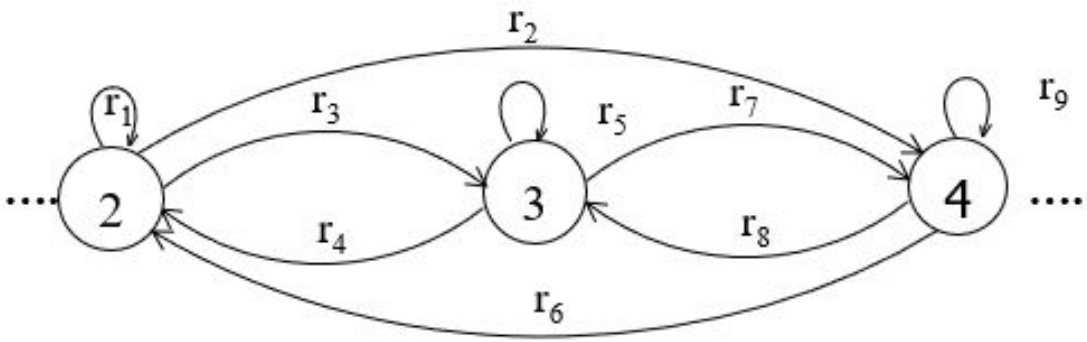


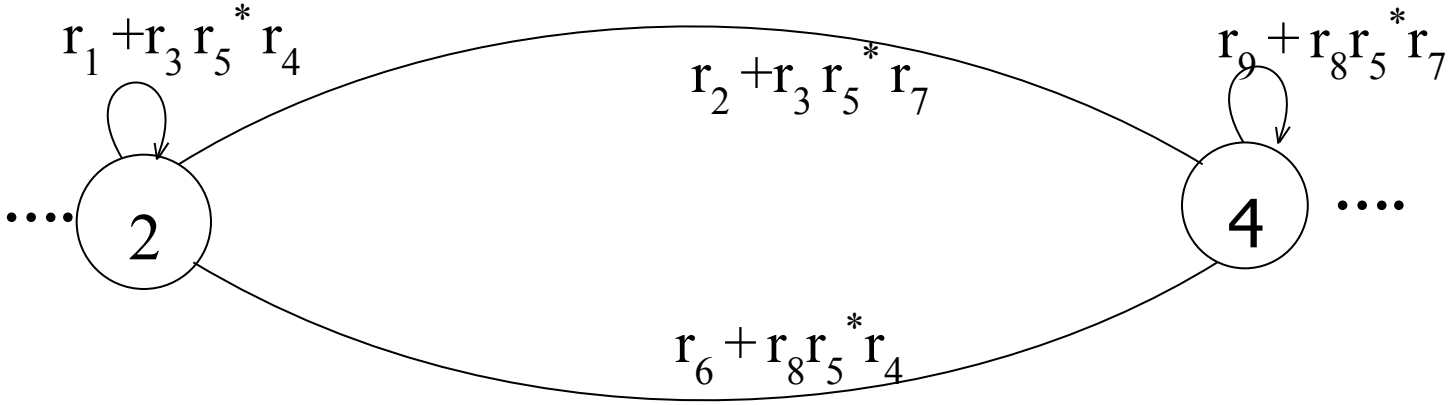
Example

Consider a part of the following TG



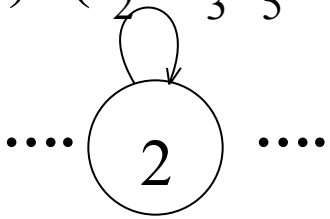
To eliminate state 3 the above TG can be reduced to





To eliminate state 4 the above TG can be reduced to

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

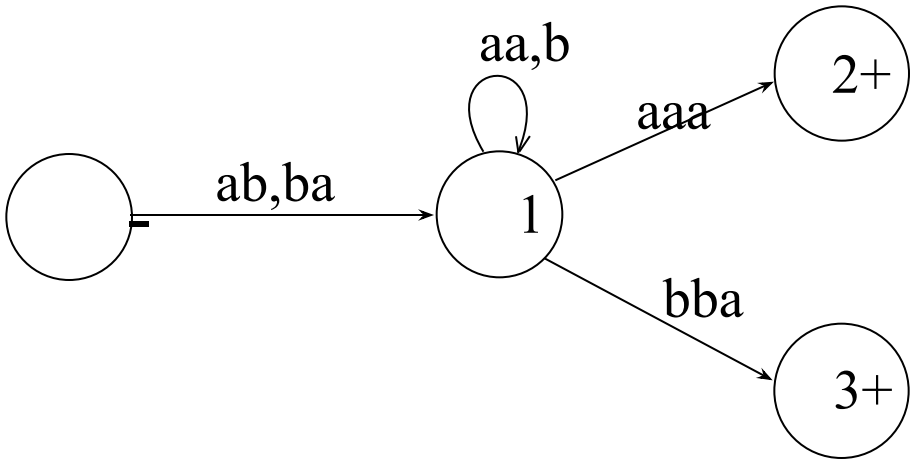


Note

- 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

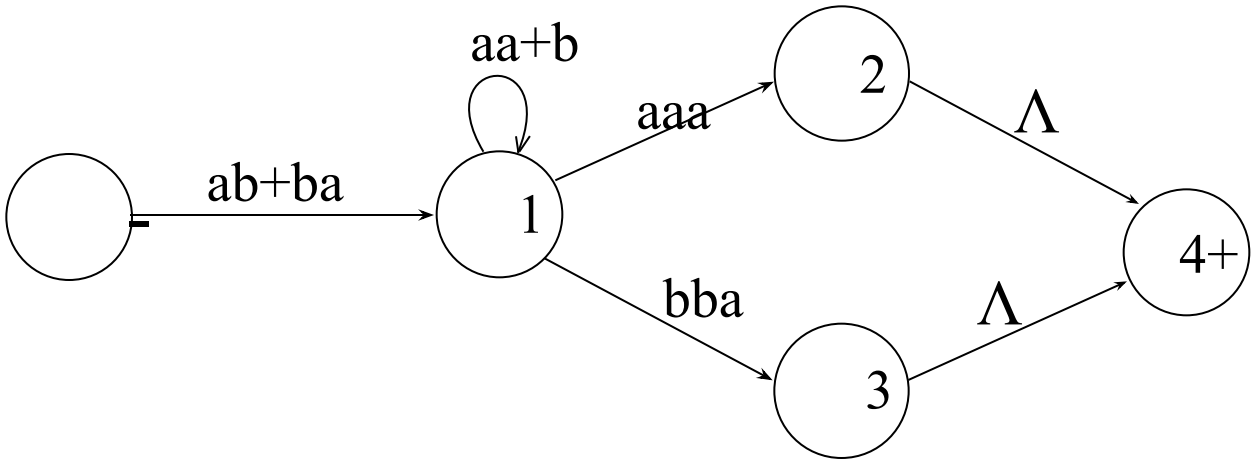
Example

- Consider the following TG

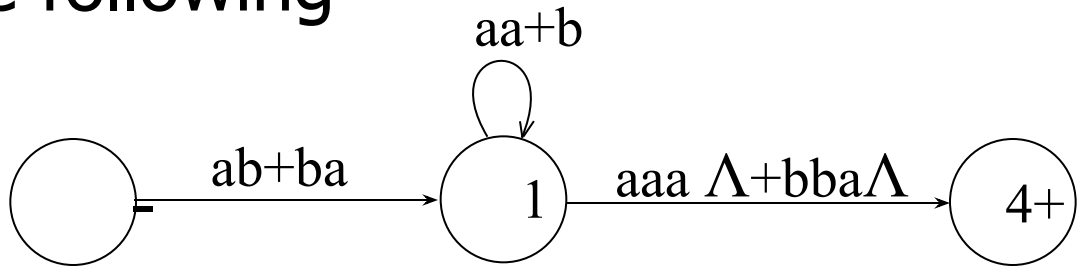


To have single final state, the above TG can be reduced to the following

Example continued ...



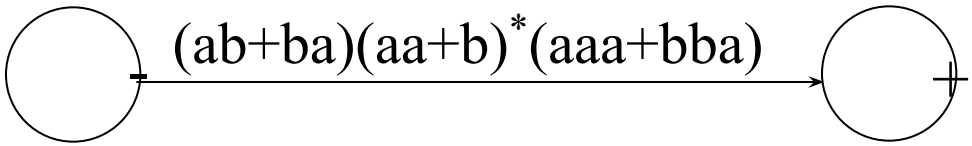
To eliminate states 2 and 3, the above TG can be reduced to the following



The above TG can be reduced to the following

Example continued …

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



Hence the required RE is

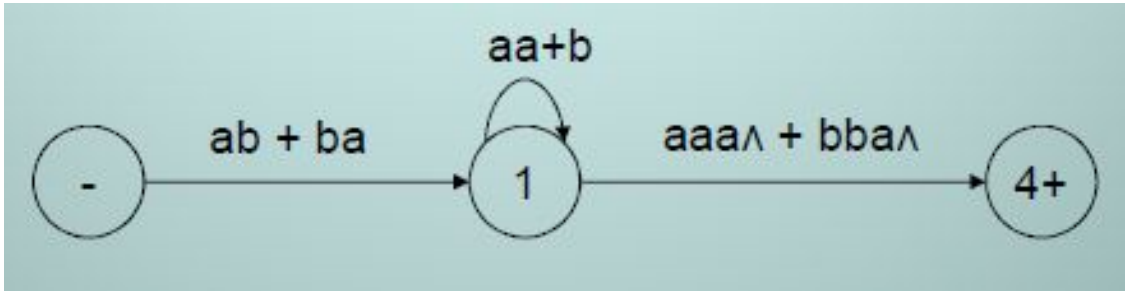
$(ab+ba)(aa+b)^*(aaa+bba)$

Summing Up

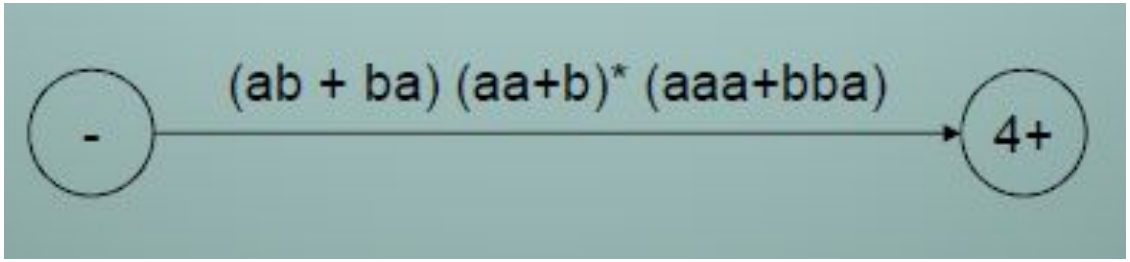
- Definition of GTG, examples of GTG accepting the languages of strings:containing aa or bb, beginning with and ending in same letters, beginning with and ending in different letters, containing aaa or bbb,
- Nondeterminism, Kleene's theorem (part I, part II, part III), proof of Kleene's theorem part I

Kleene's Theorem

- To eliminate state 2 and 3, the above TG cab be reduced to



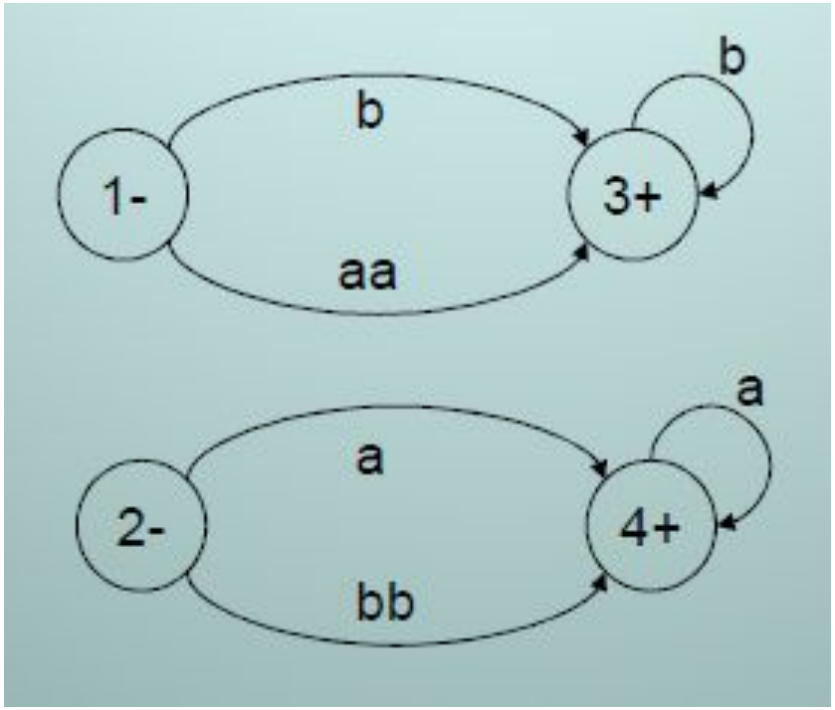
- To eliminate state 1 the above TG can be reduced to



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

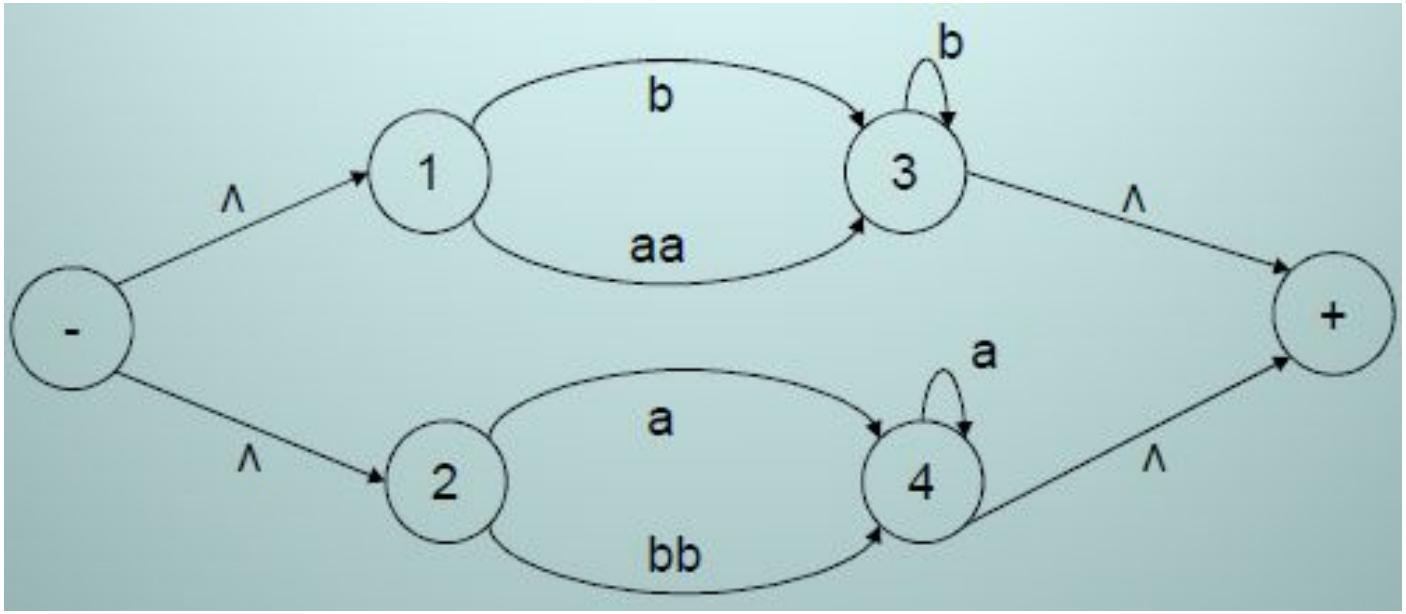
Kleene's Theorem

- Example:



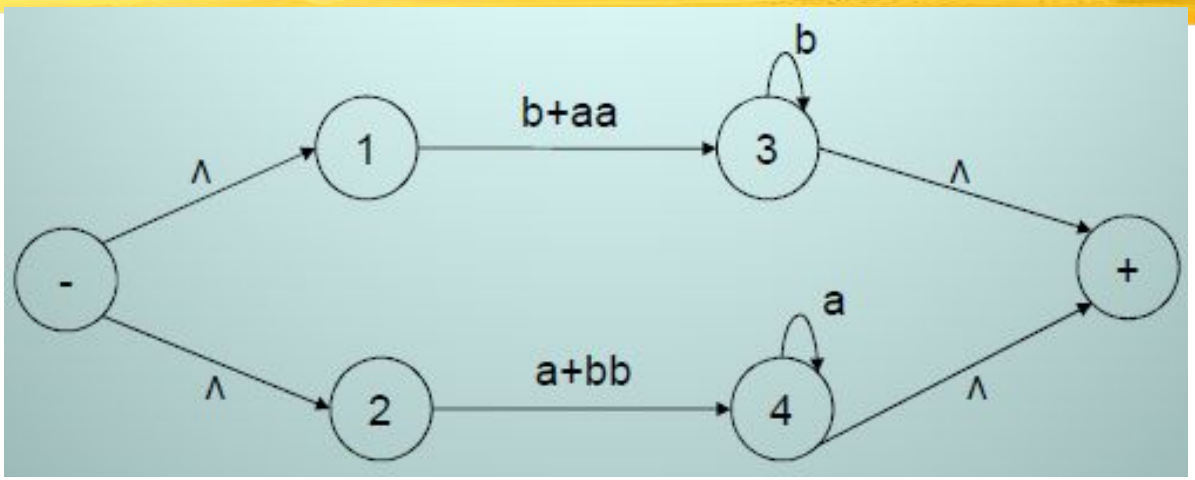
  - To have single initial and single final state the above TG can be reduced to the following

Kleene's Theorem

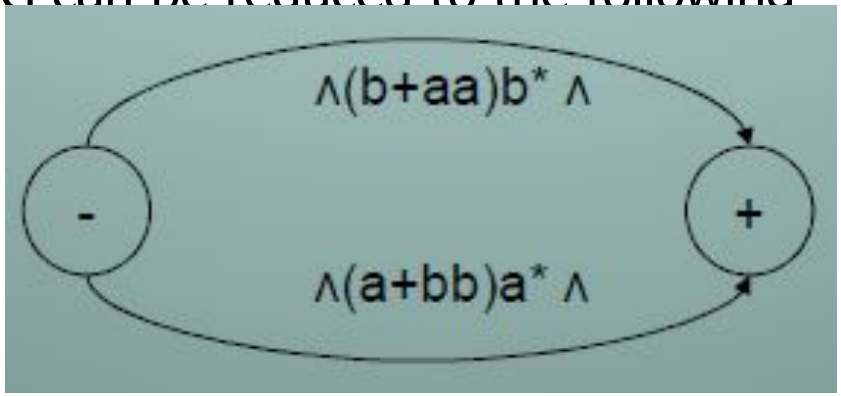


- To obtain single transition edge between 1 and 3; 2 and 4, the above TG can be reduced to the following

Kleene's Theorem

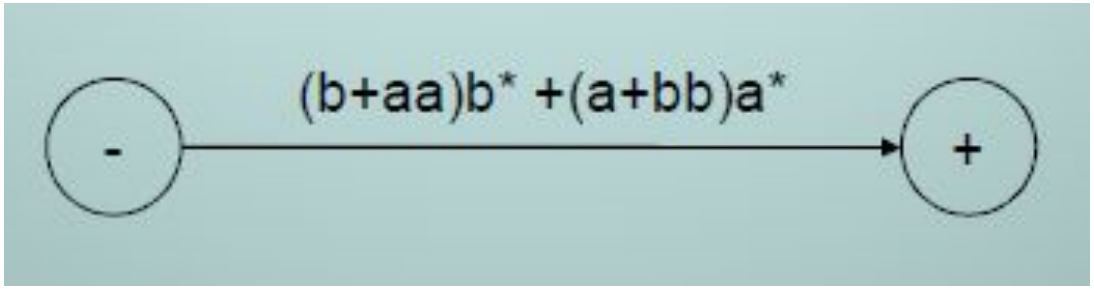


- To obtain single transition edge between 1 and 3; 2 and 4, the above TG can be reduced to the following



Kleene's Theorem

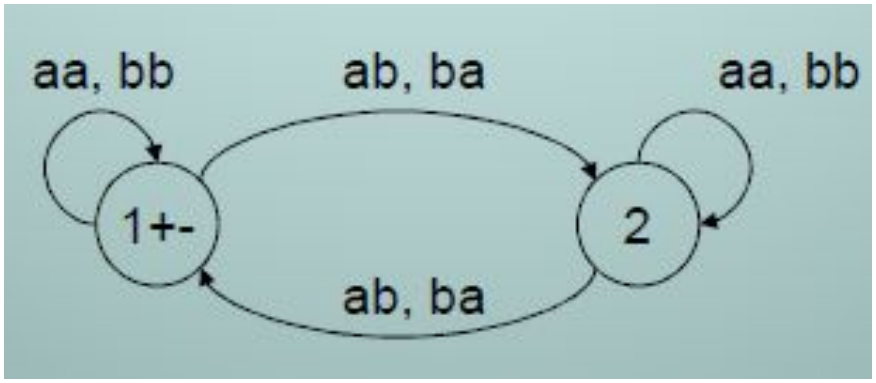
- To connect the initial state with the final state by single transition edge, the above TG can be reduced to the following



- Hence the required RE is (b+aa)b* +(a+bb)a*

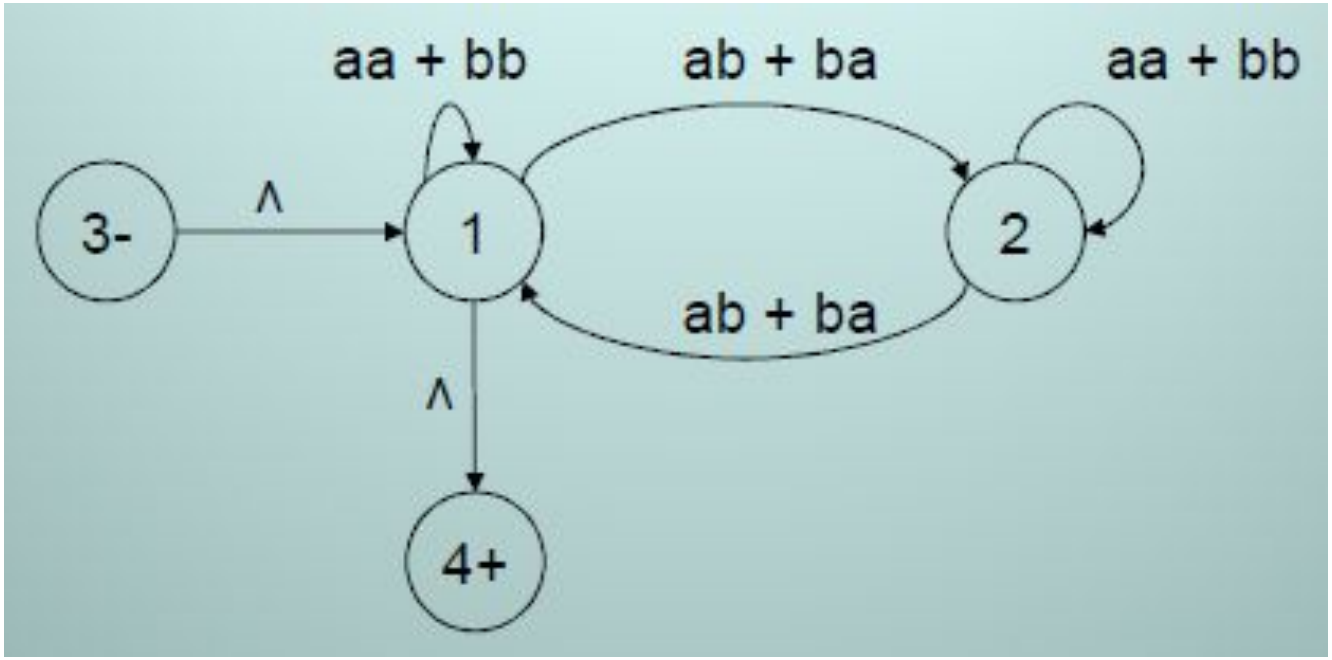
Kleene's Theorem

- Example:
  - Consider the following TG accepting Even-Even Language



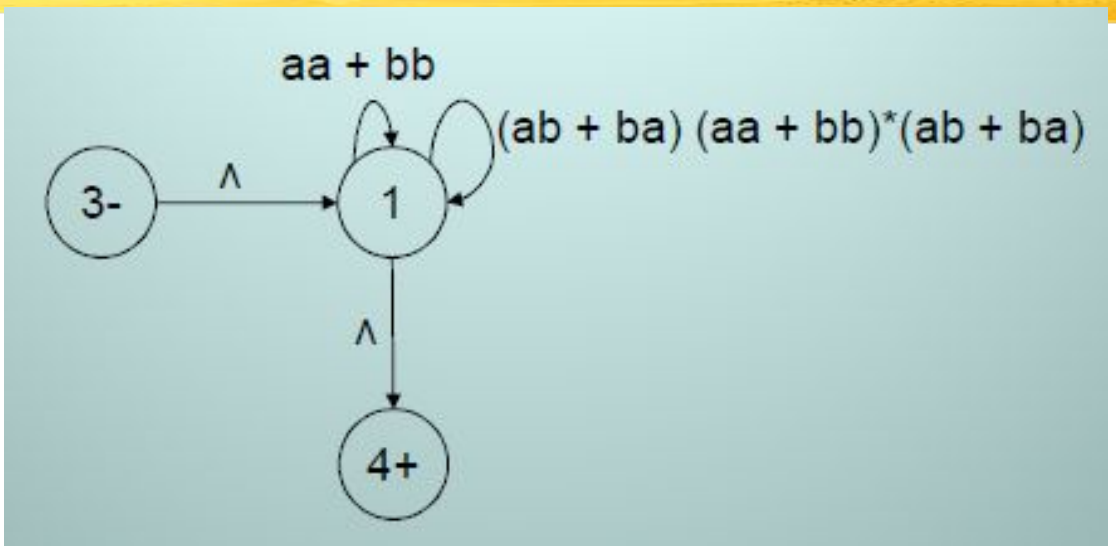
  - To have single initial and single final state the above TG can be reduced to the following

Kleene's Theorem

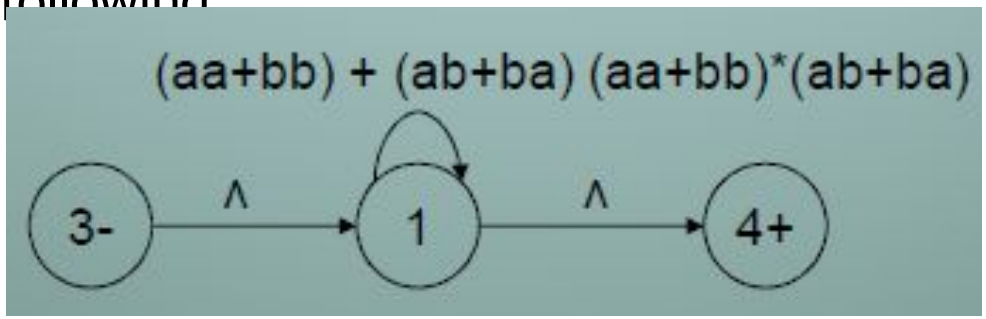


- To eliminate state 2 the above TG can be reduced to the following

Kleene's Theorem

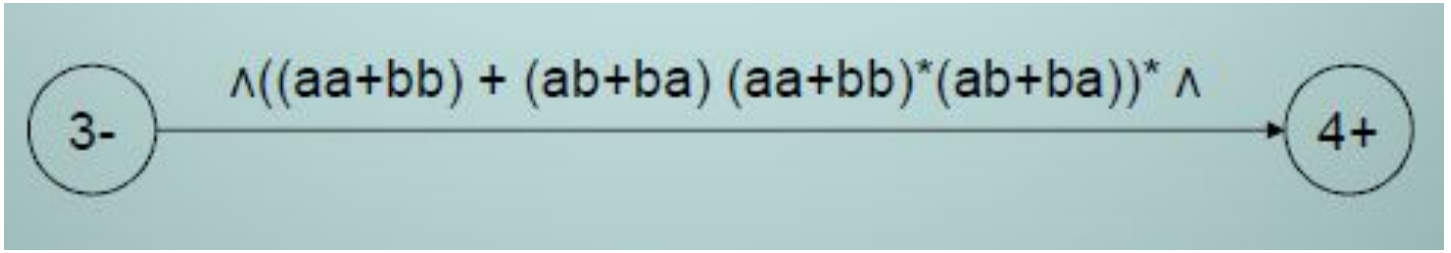


- To have single loop at state 1 the above TG can be reduced to the following



Kleene's Theorem

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

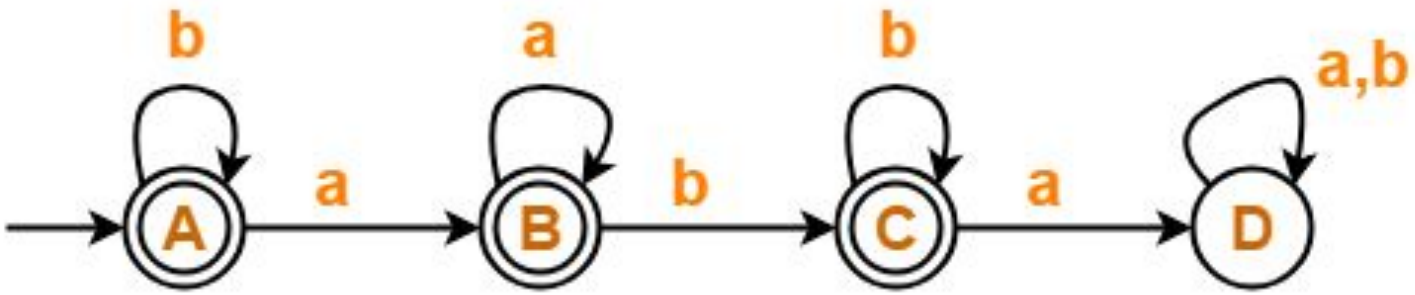


- Hence the required RE is ((aa+bb) + (ab, ba) (aa, bb)*(ab, ba))*

Summing Up

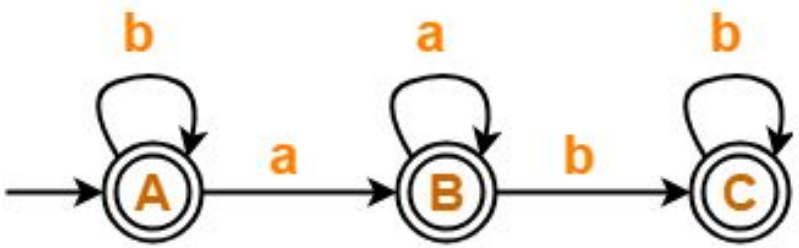
- proof of Kleene's theorem part II (method with different steps), particular examples of TGs to determine corresponding Res.

Find regular expression for the following DFA-



- State D is a dead state as it does not reach to any final state.
- So, we eliminate state D and its associated edges.

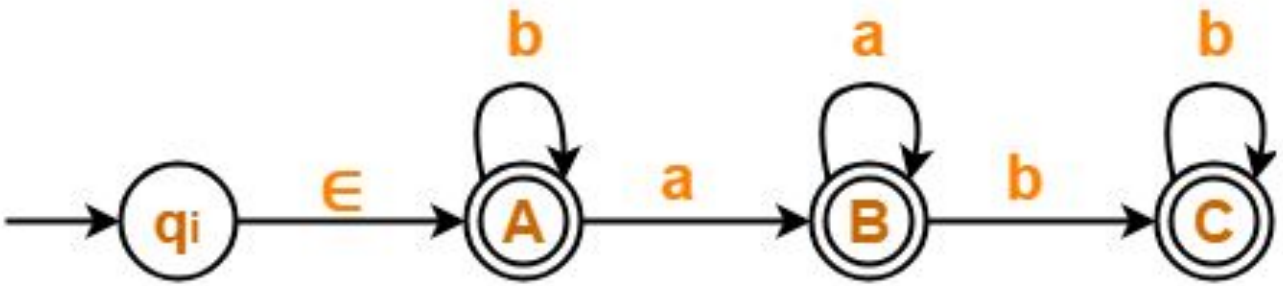
The resulting DFA is-



Step-02:

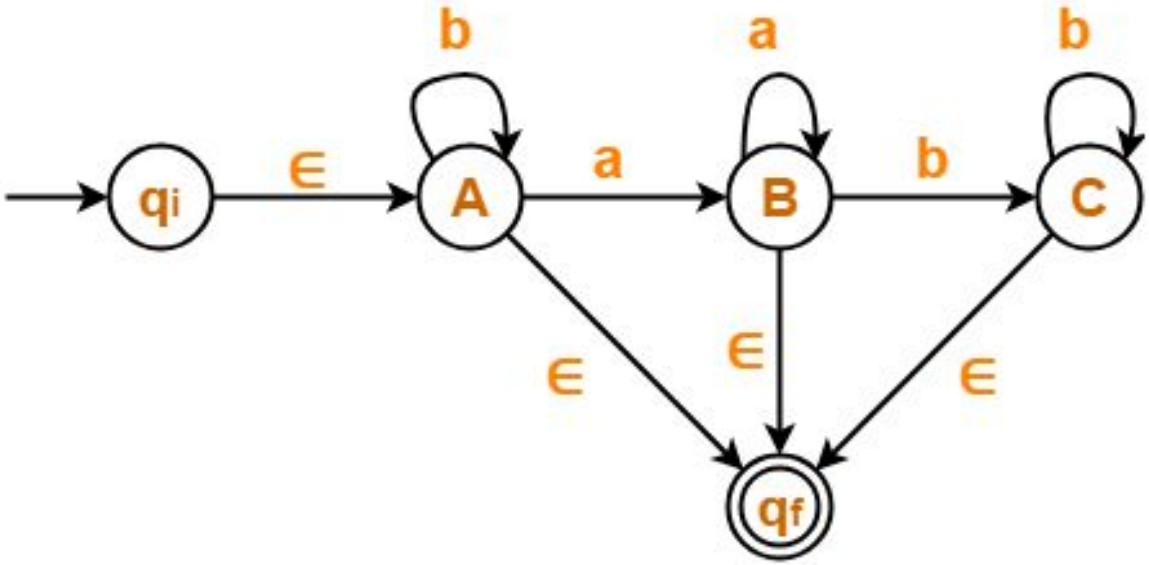
- Initial state A has an incoming edge (self loop).
- So, we create a new initial state q_i.

The resulting DFA is-



- There exist multiple final states.
- So, we convert them into a single final state.

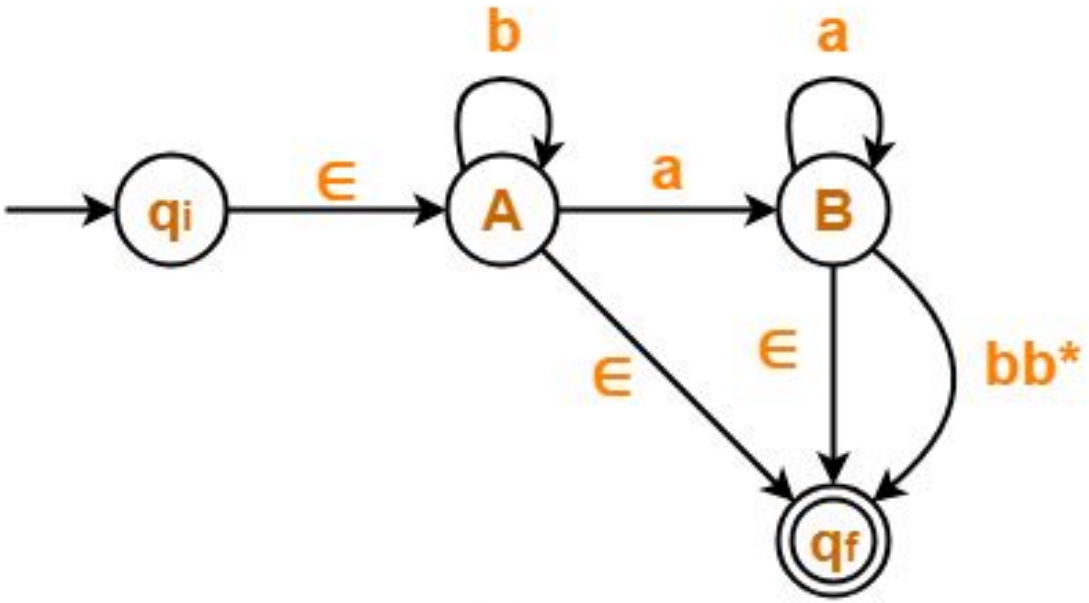
The resulting DFA is-

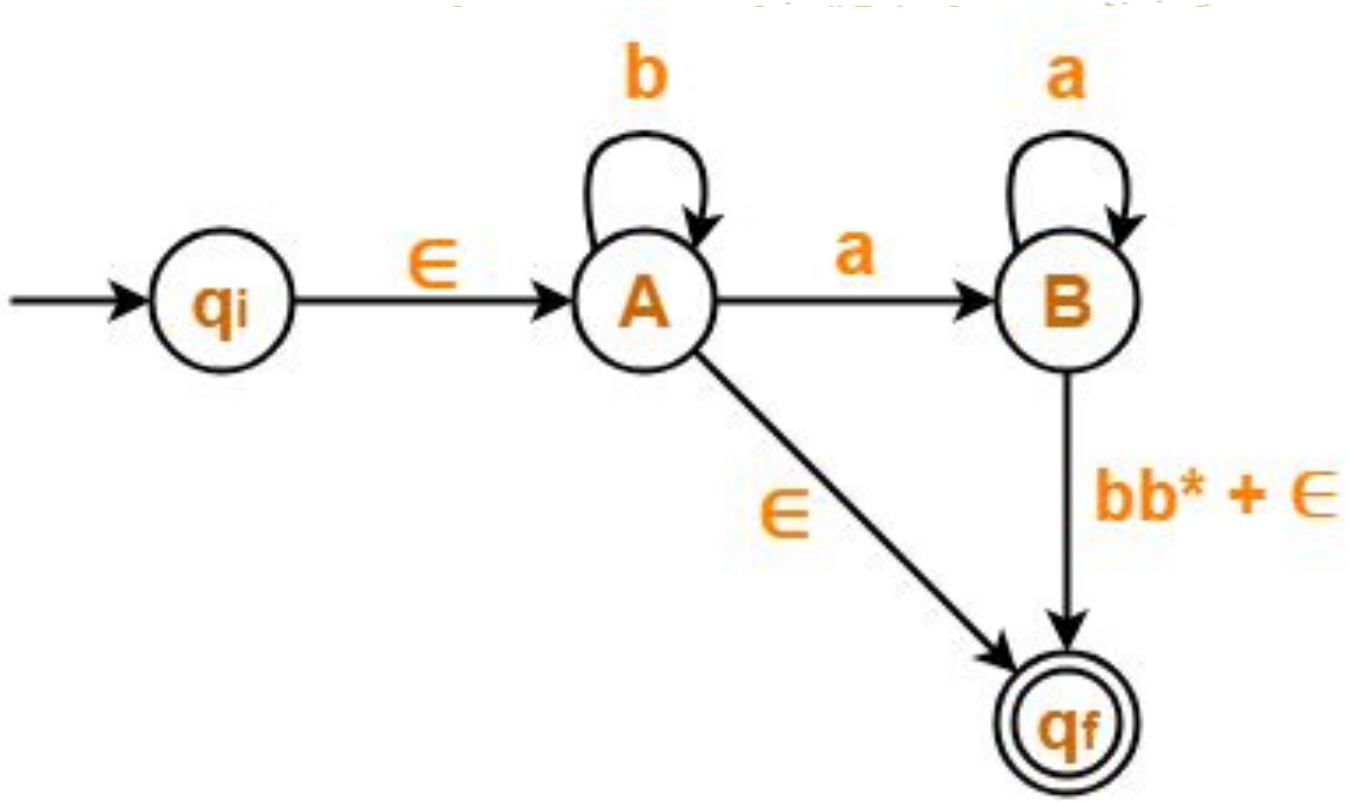


First, let us eliminate state C.

- There is a path going from state B to state q_f via state C.
- So, after eliminating state C, we put a direct path from state B to state q_f having cost b.b*.∈ =

Eliminating state C, we get-



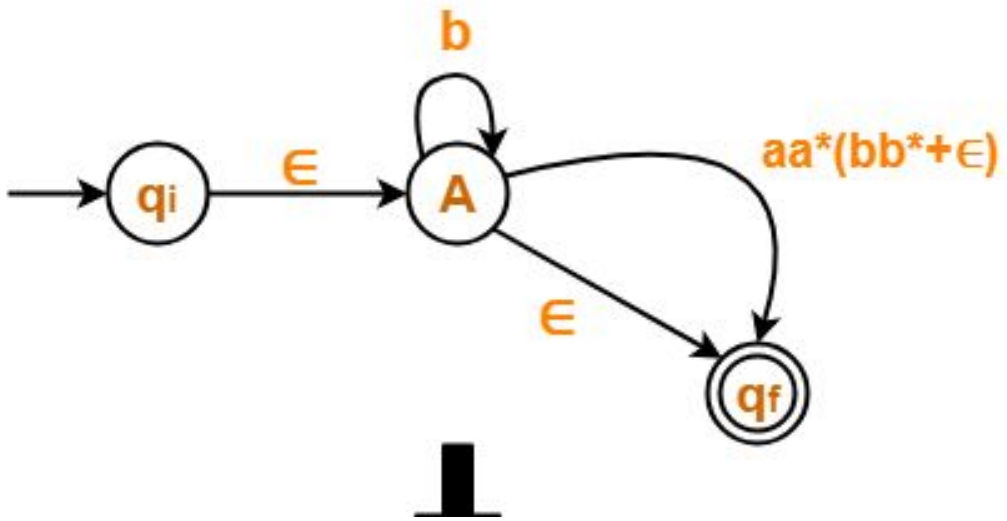

b
a
qi
∈
A
a
B
∈
bb* + ∈
qf

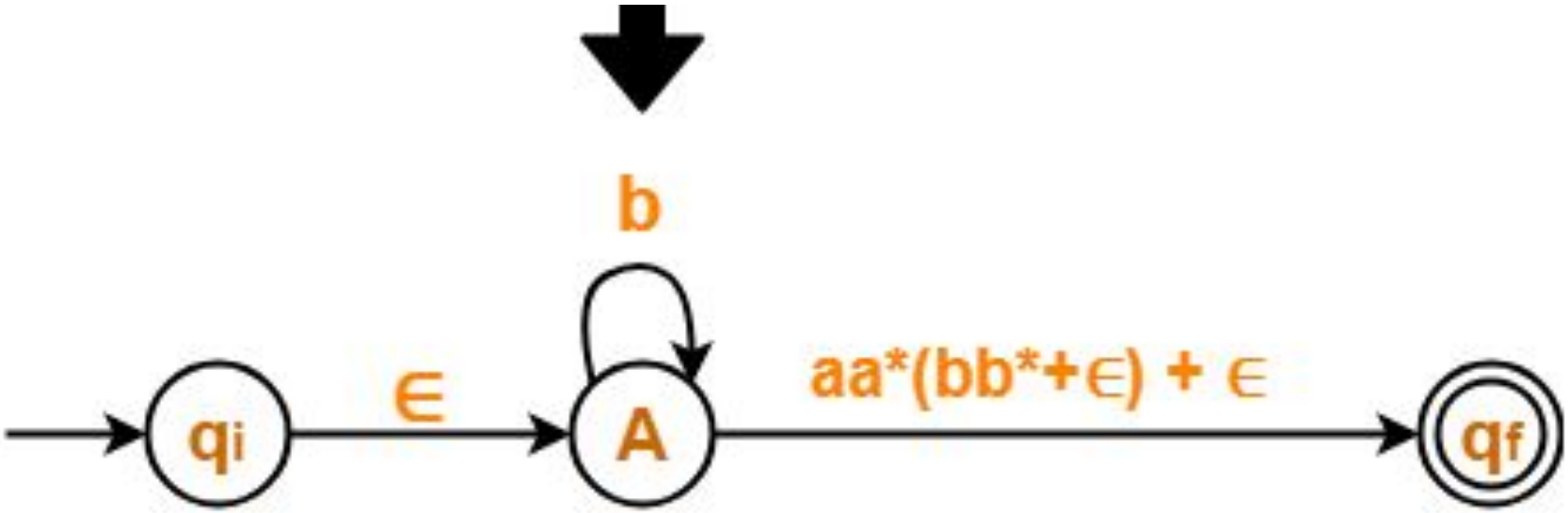
Step-05:

Now, let us eliminate state B.

- There is a path going from state A to state q_f via state B.
- So, after eliminating state B, we put a direct path from state A to state q_f having cost a.a*.(bb*+∈) = aa*(bb*+∈)

Eliminating state B, we get-




b
qi
∈
A
aa*(bb*+∈) + ∈
qf

Now, let us eliminate state A.

- There is a path going from state q_i to state q_f via state A.
- So, after eliminating state A, we put a direct path from state q_i to state q_f having cost $\epsilon.b^*.(a(bb^*+\epsilon)+\epsilon) = b^*(aa^*(bb^*+\epsilon)+\epsilon)$

Eliminating state A, we get-

Regular Expression = b*(aa*(bb*+ϵ)+ϵ)

Summing Up

- proof of Kleene's theorem part II (method with different steps), particular examples of TGs to determine corresponding Res.