Deterministic Finite Automata (DFA) (Lecture 4)

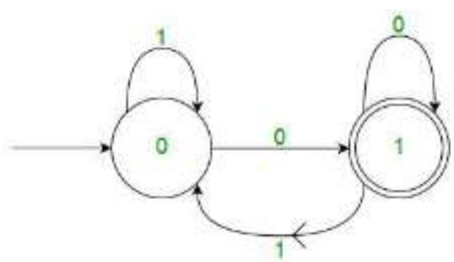
DFA consists of 5 tuples {Q, ∑, q, F, δ}.

- 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.

In a DFA

- For a particular input character, the machine goes to one state only.
- A transition function is defined on every state for every input symbol.
- Also in DFA null (or ε) move is not allowed, i.e., DFA cannot change state without any input character.

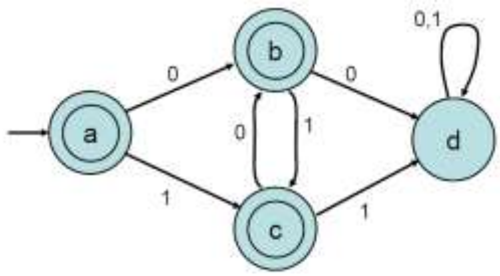
For example, below DFA with ∑ = {0, 1} accepts all strings ending with 0.



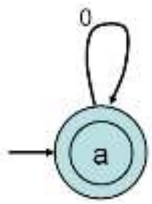
- Note: One important thing to note is, ***there can be many possible DFAs for a pattern***. A DFA with minimum number of states is generally preferred.
- Language that is accepted by some FAs are known as Regular language.
- The two concepts: REs and Regular language are essentially same i.e. (for) every regular language can be developed by (there is) a RE, and for every RE there is a Regular Language.

Example

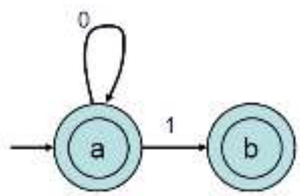
L = {w∈ {0, 1}* | w does not contain either 00 or 11 as a substring}.



- State d is a trap state= a non-accepting state that you cannot leave.
- Sometimes we will omit some arrows; by convention, they go to a trap state.
- L = {w | all nonempty blocks of 1s in w have odd length}.
- E.g., ε, or 100111000011111, or any number of 0s.
- Initial 0s don't matter, so start with:

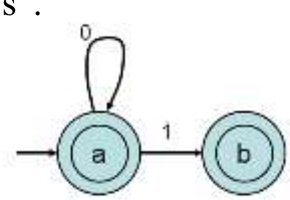


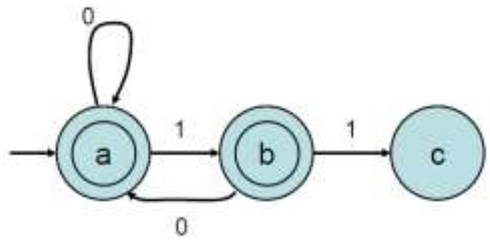
- Then 1 also leads to an accepting state, but it should be a different one, to "remember" that the string ends in one 1.



L = { w | all nonempty blocks of 1s in w have odd length }.

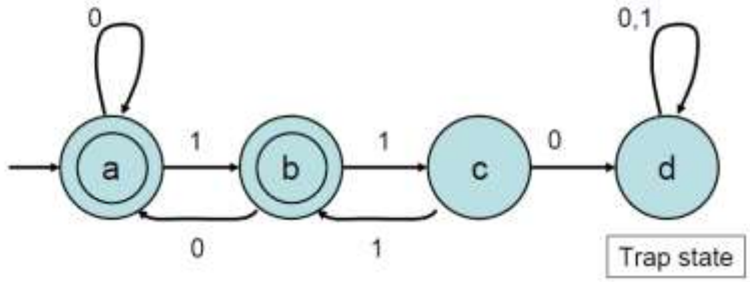
- From b:
  - 0 can return to a, which can represent either ε, or any string that is OK so far and ends with 0.
  - 1 should go to a new non accepting state, meaning "the string ends with two 1s".





– Note: c is not a trap state---we can accept some extensions.

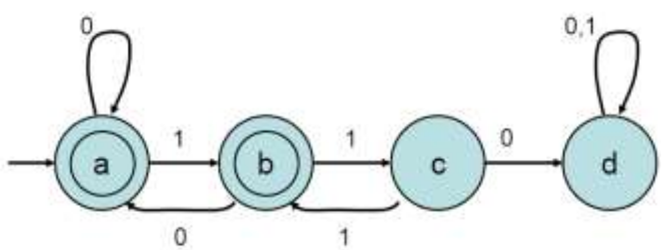
L = {w | all nonempty blocks of 1s in w have odd length}.



From c:

– 1 can lead back to b, since future acceptance decisions are the same if the string so far ends with any odd number of 1s.
    - Reinterpret b as meaning, "ends with an odd number of 1s".
    - Reinterpret c as "ends with an even number of 1s".
– 0 means we must reject the current string and all extensions.

L = { w | all nonempty blocks of 1s in w have odd length }.



- Meanings of states (more precisely):

    a: Either ε, or contains no bad block(even block of 1s followed by 0) so far and ends with 0.

    b: No bad block so far, and ends with odd number of 1s.

c: No bad block so far, and ends with even number of 1s.

d: Contains a bad block.

Example

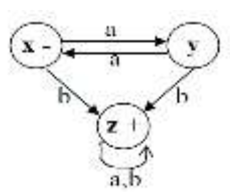
- L = EQ = {w | w contains an equal number of 0s and 1s}.
- No FA recognizes this language.
- Idea :
  - Machine must "remember" how many 0s and 1s it has seen, or at least the difference between these numbers.
  - Since these numbers (and the difference) could be anything, there cannot be enough states to keep track.
  - Therefore, the machine will sometimes get confused and give a wrong answer.

Example

If Σ={a,b}, states={x,y,z} Rules of transition:

1. from state x and input a go to state y.
2. from state x and input b go to state z.
3. from state y and input a go to state x.
4. from state y and input b go to state z.
5. from state z and any input stay at the state z.

Let x be the start state and z be the final state.



Transition Diagram

- The FA above will accept all strings that have the letter b in them and no other strings.
- The language associated with (or accepted by) this FA is the one defined by the **regular expression**: $a^*b(a+b)^*$

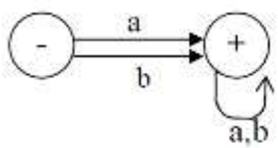
- The set of all strings that do leave us in a final state is called the language defined by the FA. The word abb is accepted by this FA, but The word aaa is not.

	a	B
x -	y	Z
y	x	Z
z +	z	Z

Transition Table

Example

The following FA accept all strings from the alphabet {a,b} except Λ.



The regular expression is: $(a+b)(a+b)^{*}=(a+b)^{+}$

Example

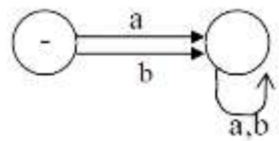
The following FA accept all words from the alphabet {a,b}.

The regular expression is $(a+b)^{*}$

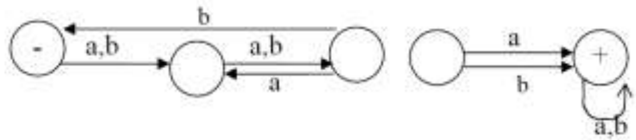
- **Note**: every language that can be accepted by an FA can be defined by a regular expression and every language that can be defined by a regular expression can be accepted by some FA.

FA that accepts no language will be one of the two types:

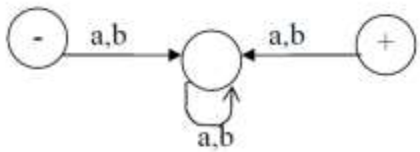
1. FA that have no final states. Like the following FA:



2. FA in which the final states cannot be reached. Like the following FA:

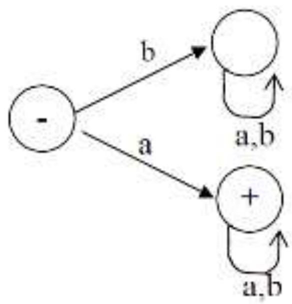


Or Like the following FA:



Example

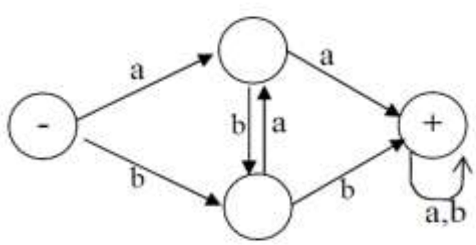
The following FA accept all strings from the alphabet {a,b} that start with a.



The regular expression is a $(a+b)^*$

Example

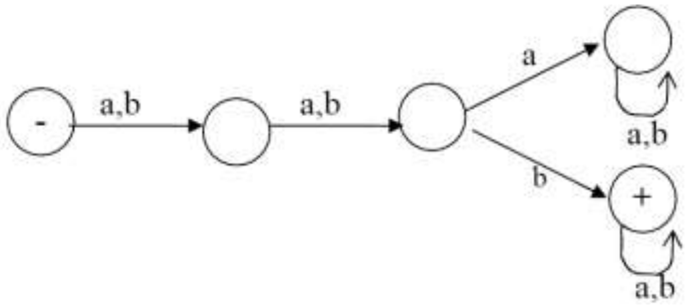
The following FA accept all strings from the alphabet **{a,b}** with double letter.



The regular expression is: $\mathbf{(a+b)^*(aa+bb)\ (a+b)^*}$

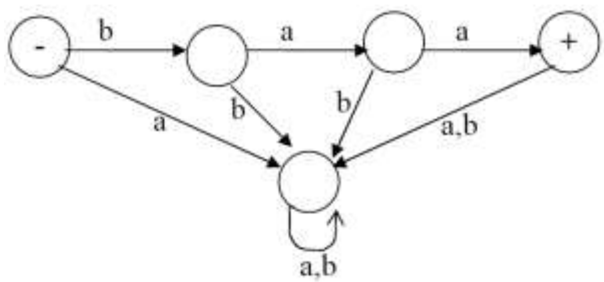
Example

The following FA accepts the language defined by the regular expression: **$(a+b)\ (a+b)\ b\ (a+b)^*$**



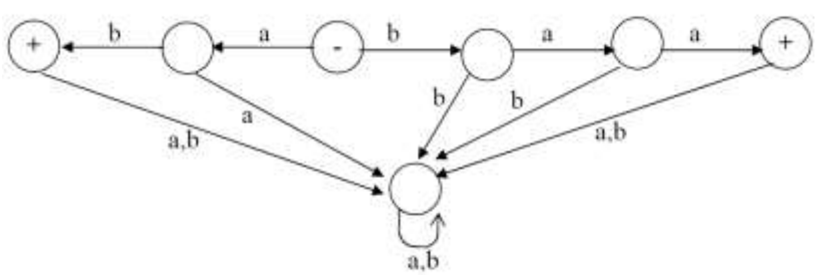
Example

The following FA accepts only the word baa.



Example

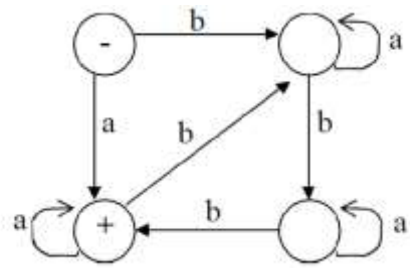
The following FA accepts the words baa and ab.



Example

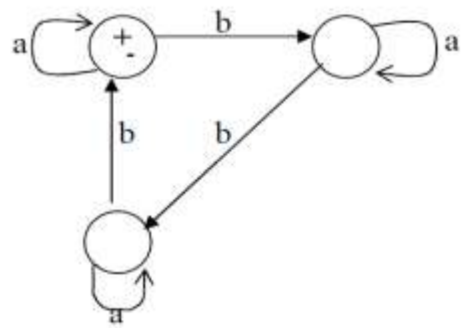
The following FA accepts the language defined by the regular expression: **$(a+ba^*ba^*b)^+$**

a,bbb,aaaabbbbbbbbbbbb



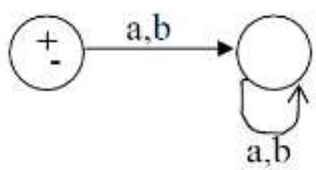
Example

The following FA accepts the language defined by the regular expression: $\mathbf{(a+ba^*ba^*b)^*}$



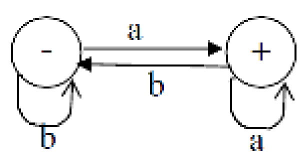
Example

The following FA accepts only the word Λ.



Example

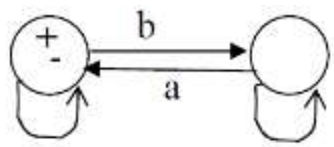
The following FA accepts all words from the alphabet **{a,b}** that end with a.



The regular expression for this language is: $\mathbf{(a+b)^*a}$

Example

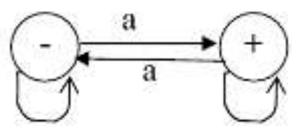
The following FA accepts all words from the alphabet **{a,b}** that do not end in b and accept Λ.



The regular expression for this language is: **$(a+b)^*a + \Lambda$**

Example

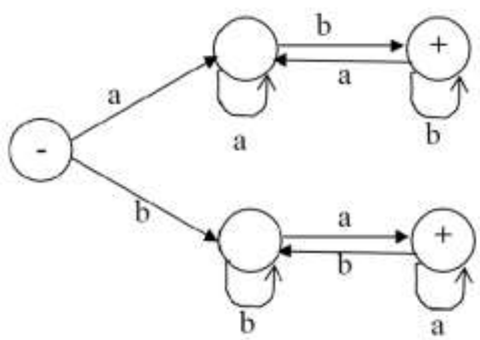
The following FA accepts all words from the alphabet **{a,b}** with an odd number of **a**'s.



The regular expression for this language is: **$b^*a\ (b^*ab^*ab^*)^*$**

Example

The following FA accepts all words from the alphabet **{a,b}** that have different first and last letters.

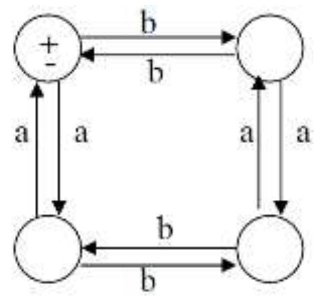


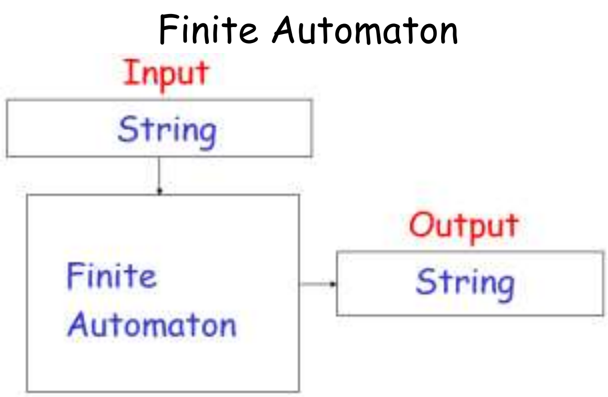
The regular expression for this language is: **$a(a+b)^*b + b(a+b)^*a$**

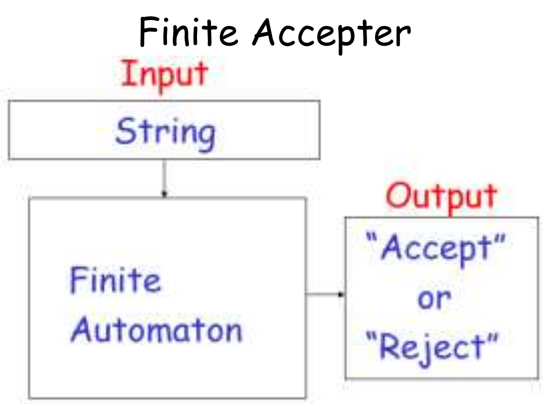
Example

The following FA accepts the language defined by the regular expression (even-even): **$[aa+bb+(ab+ba)(aa+bb)^*(ab+ba)]^*$**

Aa OR bb OR ab OR aaab





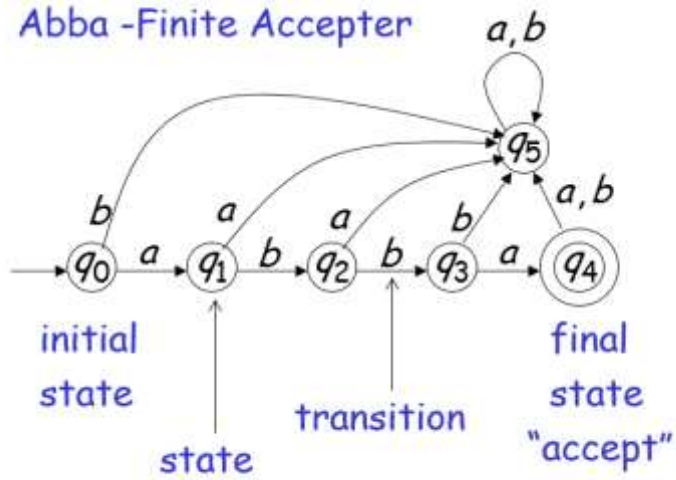


Transition Graph Lecture 4-1

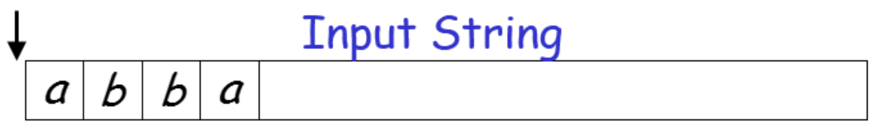
A Transition Graph (TG) is a collection of three things:

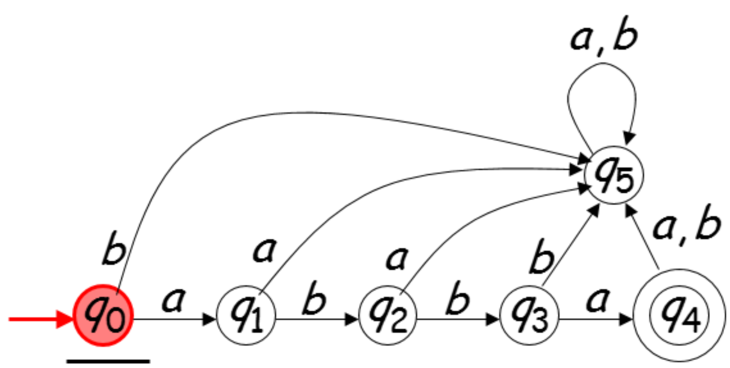
1. A finite set of states, at least one of which is designed as the start state (-) and some (may be none) of which are designed as final states (+).
2. An alphabet Σ of possible input letters from which input string are formed.
3. A finite set of transitions that show how to go from one state to another based on reading specified substrings of input letters (possibly even the null string Λ).

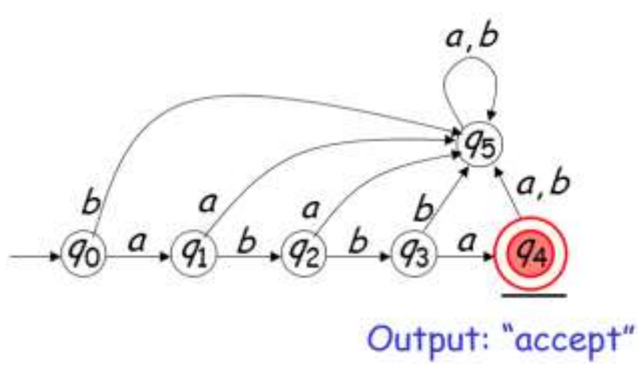
Transition Graph



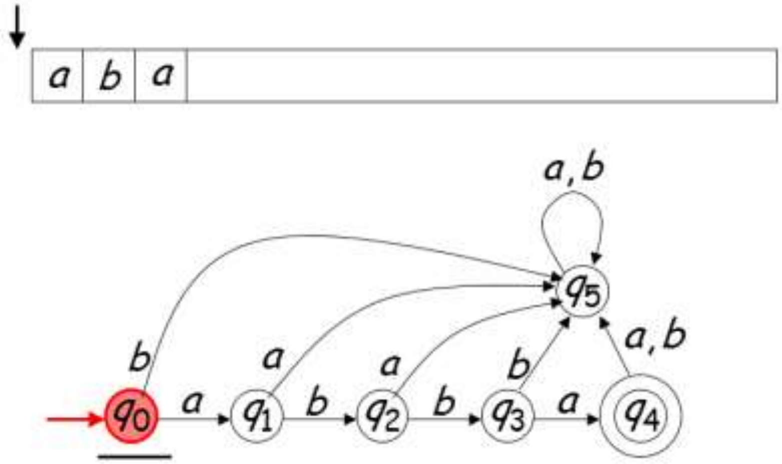
Initial Configuration



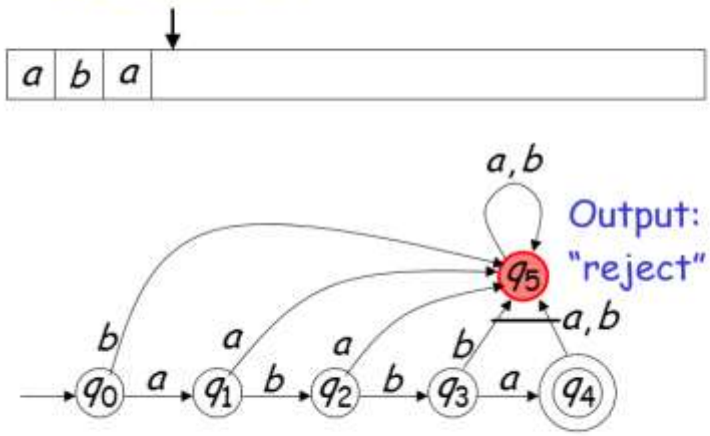




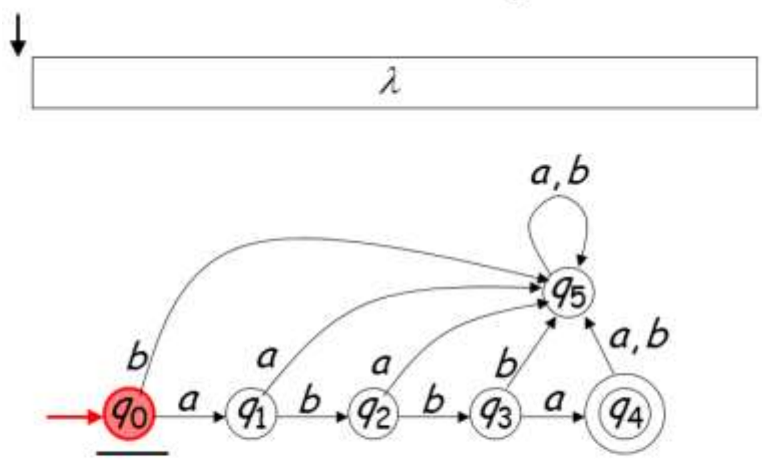
Ex2: Rejection

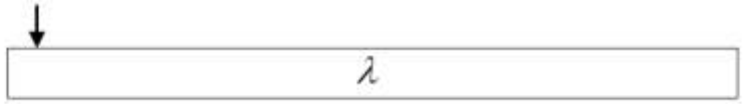


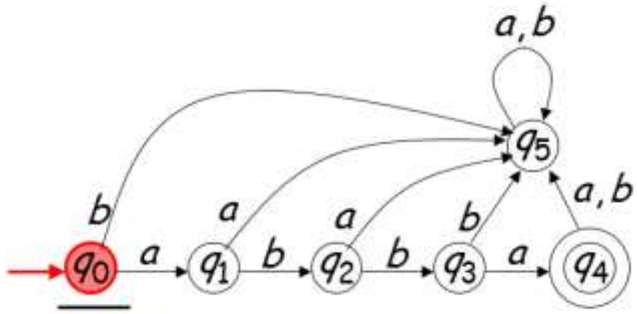
Input finished



Ex 3 : Another Rejection

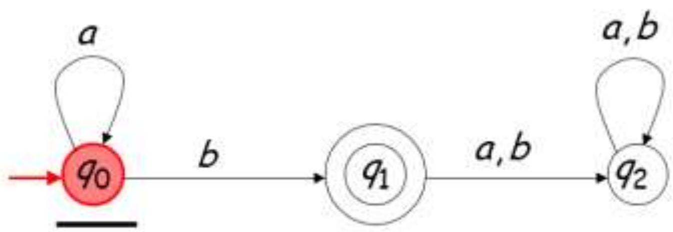




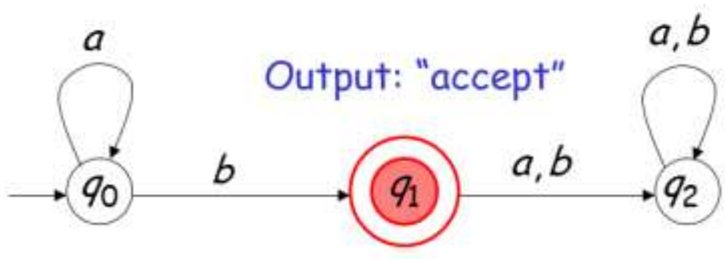


Output:
"reject"

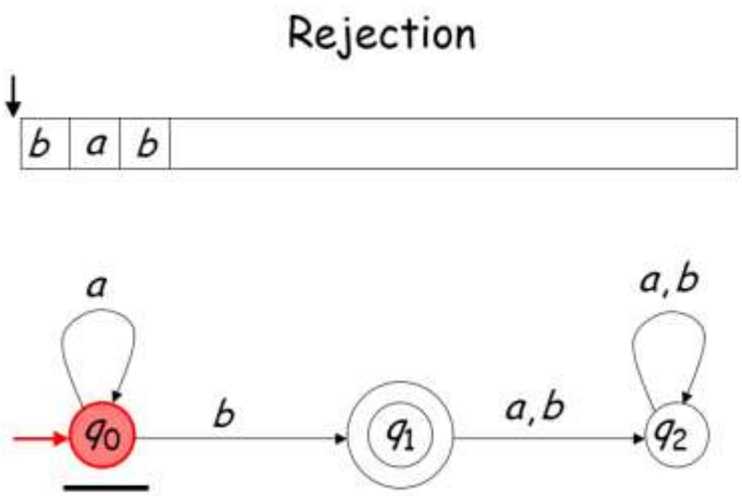
Another Example

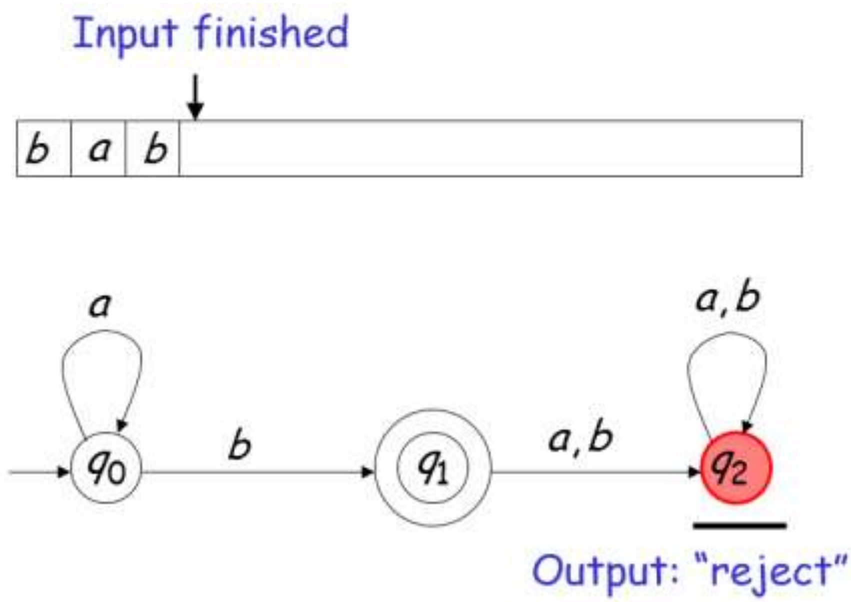


Input finished



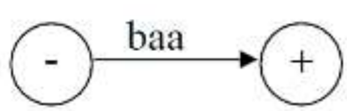
Output: "accept"





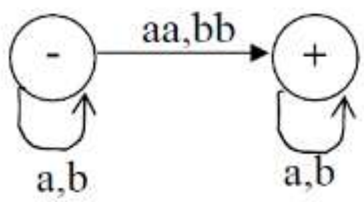
Example

The following TG accepts the word baa.



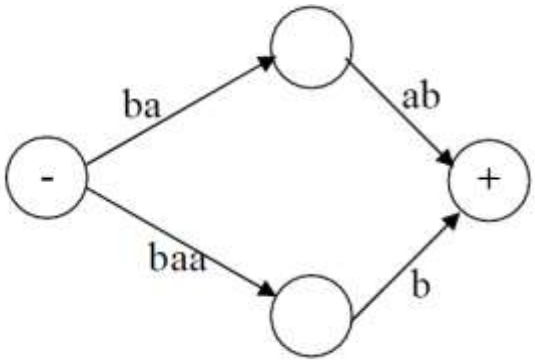
Example

The following TG accepts the words with double letters.



Example

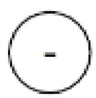
The following TG accepts the word baab in two different ways.



- **Note**: In TG, some words have several paths accept them while in FA there is only one
- **Note**: every FA is also a TG.

Example

The following TG accept nothing.

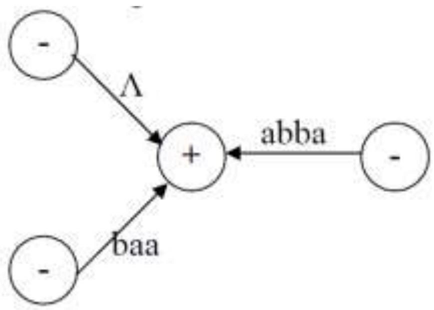


Example

The following TG accept Λ.

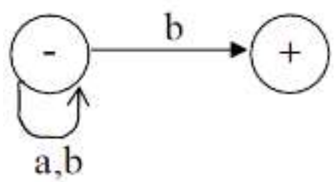
Example

The following TG accept the words {Λ, baa, abba}.



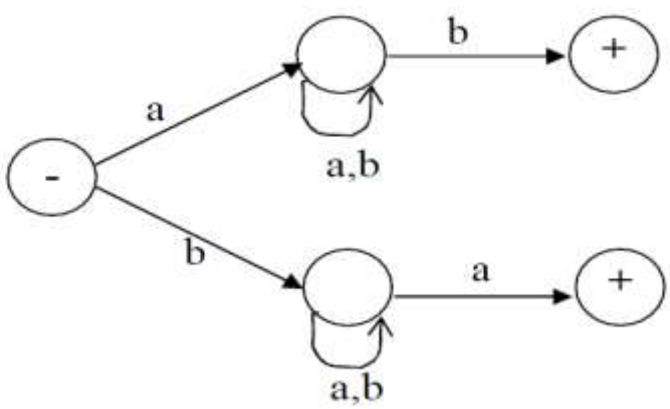
Example

The following TG accept all words that end with b.



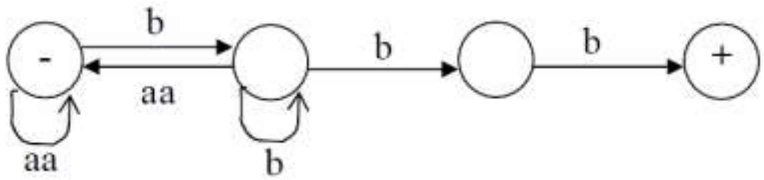
Example

The following TG accept all words that have different first and last letters.



Example

The following TG accept all words in which a's occur in even clumps only and end in three or more b's.



Example

The following TG for even-even.