DFA (Deterministic finite automata)

Formal definition of DFA:

An DFA can be represented by a 5-tuple (Q, ∑, δ, q0, F) where

- **Q** is a finite set of states.
- ∑ is a finite set of symbols called the alphabets.
- **δ** is the transition function
- **q0** is the initial state from where any input is processed (q0 ∈ Q) *(DFA have must only one initial state)*
- **F** is a set of final state/states of Q (F ⊆ Q) *(DFA have one final state, but it also possible that it have more than 1 final states)*

DFA have must the following properties:

- No empty transition
- **No. of arrows** on each state **depend upon alphabets** *(it means if there is 1 alphabets in language like ∑ = {a}, then there must be only one arrow on each state and if there are two alphabets in language like ∑ = {a, b}, then there must be two arrows on each state and so on…)*

OR

- In DFA, state can read only one letter at a time. *(It means that every state has strictly one transition for each alphabet. Suppose there are two alphabets in the language L={a,b}, then each state has strictly had two transitions. One for alphabet "a" and one transition for alphabet "b")*

- Reject state sometimes.

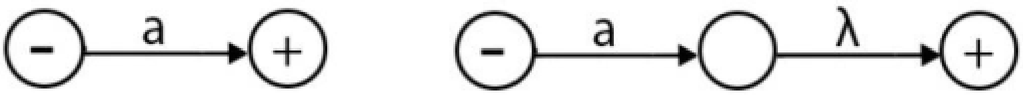
No empty transition means?

It means that null (empty) transition is not allowed in DFA.

Let's clears it with example:

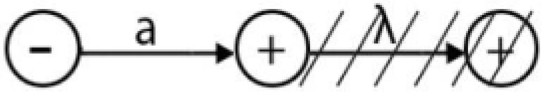
We have Regular Expression **"a"** if ∑ = {a , b}

NFA

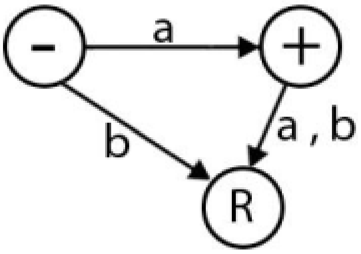


Both above diagrams have same meaning in NFA, because NFA allow empty transition.

DFA



But DFA don’t allow empty transition.



Reject state means?

A state which not allow us to reach **final state.**

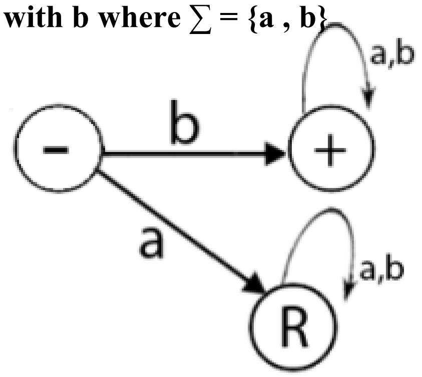
Simply we can say that on this particular transition **letter will be skip / ignore.**

In the above DFA it will only read letter “a”. ***Letter “b” & “a,b” will be reject.***

Examples of DFA:

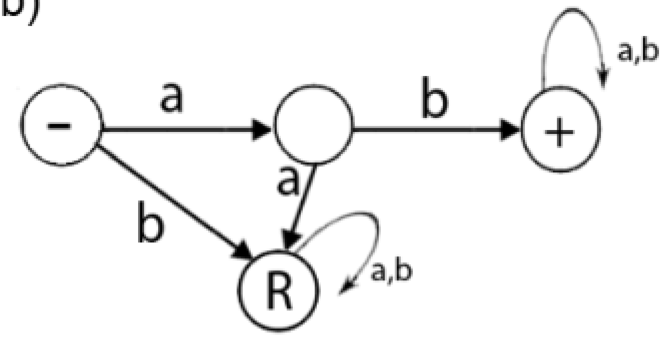
1. Draw DFA that starts with b where ∑ = {a , b}

R.E = b (a+b)*



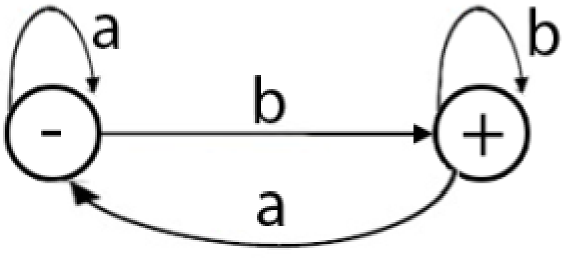
2. Draw DFA that starts with ab where ∑ = {a , b}

R.E = ab (a+b)*



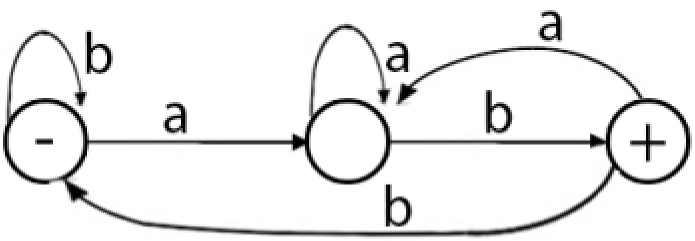
3. Draw DFA that ends with b where ∑ = {a , b}

RE = (a+b)* b



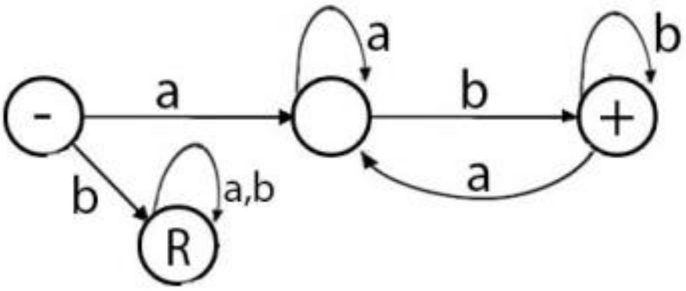
4. Draw DFA that ends with ab where ∑ = {a , b}

RE = (a+b)* ab



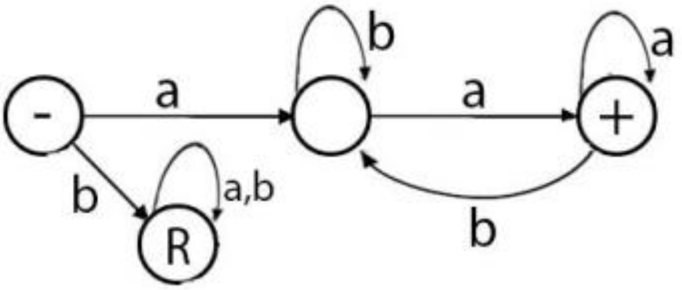
5. Draw DFA that starts with a & end with b where ∑ = {a , b}

RE = a (a+b)* b



6. Draw DFA that starts & end with a where ∑ = {a , b}

RE = a (a+b)* a



Difference between NFA & DFA:

NFA	DFA
NFA is a generalize type of Finite Automata	DFA is a special type of Finite Automata
NFA have redundant transition. (It means that the transition from a state can be to multiple next states for each input symbol. Hence it is called non-deterministic)	DFA have no redundant transition. (It means that the transition from a state is to a single particular next state for each input symbol. Hence it is called deterministic)
NFA allows empty string transitions.	DFA don't allows empty string transitions.
NFA requires less space**.**(small set of strings)	DFA requires more space**.**(Large set of stings)
Regular expression can be easily converted to NFA.	Conversion of regular expression to DFA is complex.
NFA can't be easily implemented	DFA can be easily implemented