

## Convert the NFA into DFA

For each NFA we can find a DFA accepting the same language. The number of states of the DFA could be exponential in the number of states of the NFA, but in practice this worst case occurs rarely.

**Algorithm:** Constructing a deterministic finite automaton from a nondeterministic one.

**Input:** An NFA  $N$ .

**Output:** A DFA  $D$  accepting the same language.

**Method:** Each state of  $D$  is a set of states which  $N$  could be in after reading some sequence of input symbols. Thus  $D$  is able to simulate all possible moves  $N$  can make on a given input string. The initial state of  $D$  is the set consisting of  $S_0$ , the initial state of  $N$ , together with all states of  $N$  that can be reached from  $S_0$  by means of  $\epsilon$ -transition only. The accepting states of  $D$  are those sets of states that contain at least one accepting states of  $N$ .

Let us define the function  $\epsilon$ -CLOSURE ( $s$ ) to be the set of states of  $N$  built by applying the following rules:-

1-  $S$  is added to  $\epsilon$ -CLOSURE ( $s$ ).

2- If  $t$  is in  $\epsilon$ -CLOSURE ( $s$ ), and there is an edge labeled  $\epsilon$  from  $t$  to  $u$ , then  $u$  is added to  $\epsilon$ -CLOSURE ( $s$ ) if  $u$  is not already there. **Rule 2** is repeated until no more states can be added to  $\epsilon$ -CLOSURE ( $s$ ).

Thus,  $\epsilon$ -CLOSURE ( $s$ ) is just the set of states that can be reached from  $S$  on  $\epsilon$ -transition alone. If  $T$  is a set of states, then  $\epsilon$ -CLOSURE ( $T$ ) is just the **union** over all states  $S$  in  $T$  of  $\epsilon$ -CLOSURE ( $S$ ).

The computation of  $\epsilon$ -CLOSURE ( $T$ ) is a typical process of searching a graph for nodes reachable from a given set of nodes. In this case the nodes of  $T$  are the given set, and the graph consists of the  $\epsilon$ - labeled edge of the transition diagram only. A simple algorithm to compute  $\epsilon$ -CLOSURE ( $T$ ) is shown in Fig.14

```

Begin
  Push all states in T onto STACK;
   $\epsilon$ -CLOSURE (T):=T;
  While STACK not empty do
    Pop S, the top element of STACK, off of stack;
    For each state t with an edge from S to t labeled  $\epsilon$  do
      If t is not in  $\epsilon$ -CLOSURE (T) then
        Begin
          Add t to  $\epsilon$ -CLOSURE (T);
          Push t onto STACK;
        End
      End
    End
  End
End

```

Fig.14: Computation of  $\epsilon$ -CLOSURE

The states of **D** and their transition are constructed as follows. Initially, let  $\epsilon$ -CLOSURE (**S**<sub>0</sub>) be a state of **D**. This state is the start state of **D**. We assume each state of **D** is initially "unmarked". Then perform the algorithm of Fig.15.

```

While there is an unmarked state x= (S1, S2... Sn) of D do
  Begin
    Mark x;
    For each input symbol a do
      Begin
        Let T be the set of states to which there is a transition on 'a'
        from some state Si in x;
        y: =  $\epsilon$ -CLOSURE (T);
        If y has not yet been added to the set of states of D then
          Make y an "unmarked" state of D;
          Add a transition from x to y labeled a if not already
          present
        End
      End
    End
  End
End

```

Fig.15: The Subset Construction

**Example** Convert the NFA shown in Fig.16 into DFA

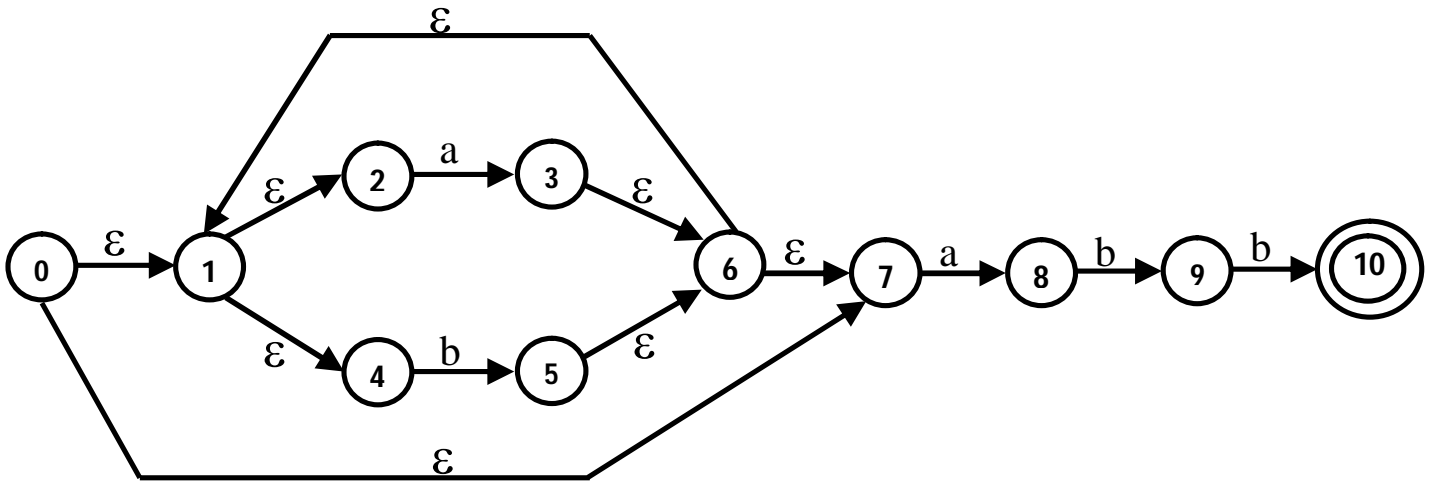


Fig.16: NFA N for  $(a \mid b)^*abb$ .

Fig.16 shows NFA N accepting the language  $(a \mid b)^*abb$ . The initial state of the equivalent DFA is  $\epsilon$ -CLOSURE (0), which is  $A = \{0, 1, 2, 4, 7\}$ , **since these are exactly the states reachable from state 0 via a path in which every edge is labeled  $\epsilon$** . Note that 0 is reached from itself by such a path, the path having no edges.

The algorithm of Fig.15 tell us to set x to A and compute T, the set of states of N having transitions on **a** from members of A. Among the states 0, 1, 2, 4 and 7, only 2 and 7 have such transition, to 3 and 8, so

$$y = \epsilon\text{-CLOSURE}(\{3, 8\}) = \{1, 2, 3, 4, 6, 7, 8\}$$

Let us call this set B. Among the states in A, only 4 has a transition on **b** to 5, so the DFA has a transition on b from A to

$$C = \epsilon\text{-CLOSURE}(\{5\}) = \{1, 2, 4, 5, 6, 7\}$$

Then we have two transitions:-

$$A \xrightarrow{a} B$$

$$A \xrightarrow{b} C$$

We continue this algorithm for state B and C.

1- For set B

a- If the input is a symbol "a", then only state 2 and 7 give a transition on symbol "a" to state 3 and 8, therefore  $\epsilon\text{-CLOSURE}(\{3,8\})=B$ , then we have this transition

$$B \xrightarrow{a} B$$

b- If the input is a symbol "b", then only state 4 and 8 give a transition on symbol "b" to state 5 and 9, therefore

$$\epsilon\text{-CLOSURE}(\{5, 9\}) = \{1, 2, 4, 5, 6, 7, 9\}$$

Let us call this set D. Because a state D not added to the state of DFA, then we add it, with transition:-

$$B \xrightarrow{b} D$$

2- For set C

a- If the input symbol is "a", then only state 2 and 7 have transition on "a" to state 3 and 8, therefore

$\epsilon\text{-CLOSURE}(\{3, 8\}) = B$ , then the transition is:

$$C \xrightarrow{a} B$$

b- If the input symbol is "b", then only state 4 gives a transition on b to state 5, therefore

$\epsilon\text{-CLOSURE}(5) = C$ , then the transition is:-

$$C \xrightarrow{b} C$$

3- For set D:-

a- If the input symbol is "a", then only state 2 and 7 gives a transition on "a" to state 3 and 8, therefore

$\epsilon\text{-CLOSURE}(\{3, 8\}) = B$ , then the transition is

$$D \xrightarrow{a} B$$

b- If the input symbol is "b" then only state 4 and 9 gives a transition on "b" to state 5, and 10, therefore:-

$$\epsilon\text{-CLOSURE}(\{5, 10\}) = \{1, 2, 4, 5, 6, 7, 10\}$$

Let us call this E, then the transition:-

$$D \xrightarrow{b} E$$

4- For set E

a- If the input symbol is "a", then only state 2 and 7 given a transition on "a" to state 3 and 8, therefore:-

$\varepsilon$ -CLOSURE ( $\{3, 8\}$ ) = B, then the transition is

$E \xrightarrow{a} B$

b- If the input symbol is "b", then only state 4 gives a transition on "b" to state 5, therefore

$\varepsilon$ -CLOSURE (5) = C, then the transition is:-

$E \xrightarrow{b} C$

The transition table for NFA is shown in Fig.17, and the DFA equivalent to NFA of Fig.16 is shown in Fig.18. The final state is E because it contain the state 10 which is the final state for NFA in Fig.16.

State	Input	
	a	b
(Start) A	B	C
B	B	D
C	B	C
D	B	E
(Accept) E	B	C

Fig.17: Transition Table

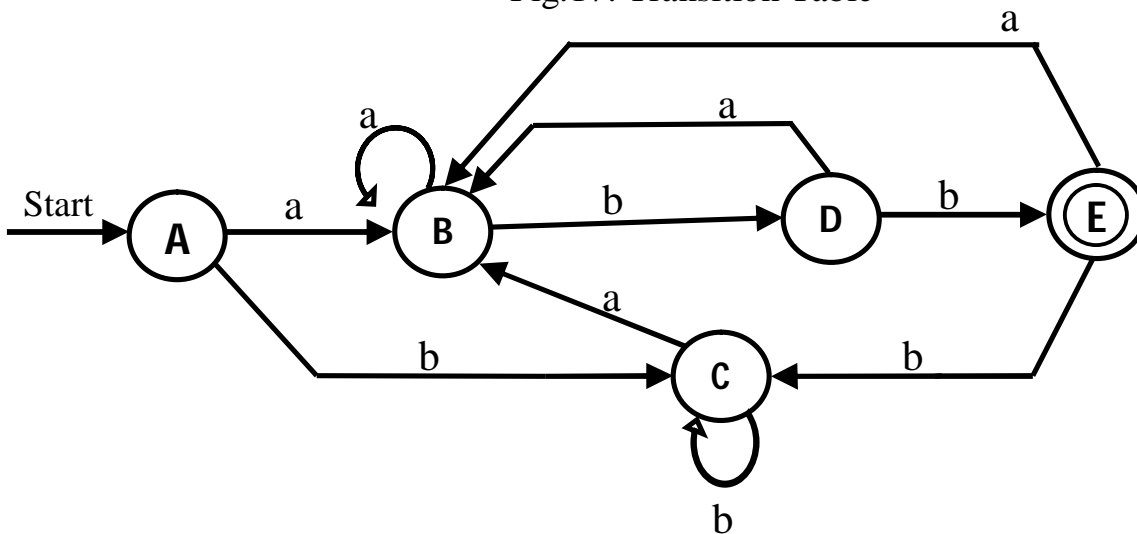


Fig.18: DFA accepting  $(a \mid b)^*abb$ .

## Minimizing the Number of states of a DFA

The DFA of Fig (18) constructed from the NFA of Fig (16) is not the smallest possible. In particular, we show Fig (12), another DFA with only four states that also accept  $(a \mid b)^*abb$ . Part of the reason for this nonminimality of the subset construction lies in the fact that we did not include in the algorithm, for converting NFA to DFA, some obvious state identifications that could be made.

There are two methods for minimizing the number of states and transitions:-

### 1-Empirical Method:

This method minimized the number of states and transitions according to the following steps:-

**a-** We merge two states in DFA in one state if they have the same important states. The important state is the state with **no  $\epsilon$ -transition**

**b-** We merge two states if they either both include or both exclude accepting states of the NFA.

**Example:** Minimize the DFA shown in Fig.19 according to the Empirical method.

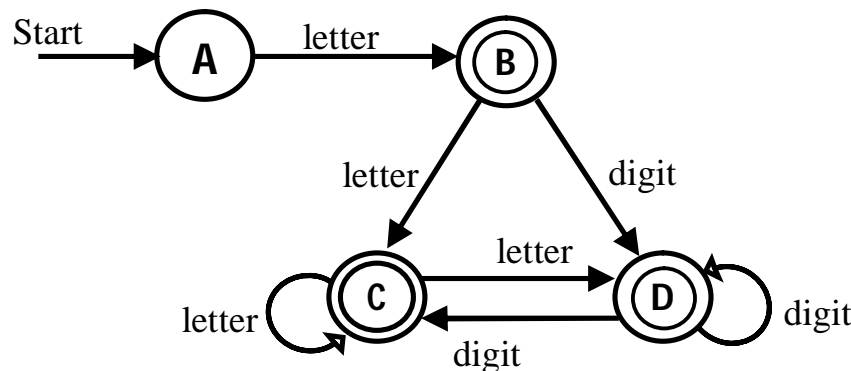


Fig.19: DFA accepting  $\text{letter}(\text{letter} \mid \text{digit})^*$

$A = \{0\}$ ,  $B = \{1, 2, \underline{8}, \underline{3}, \underline{5}\}$ ,  $C = \{4, 7, \underline{8}, 2, \underline{3}, \underline{5}\}$ ,  $D = \{6, 7, \underline{8}, 2, \underline{3}, \underline{5}\}$

Final is state **8**.

From Fig.19 we know that the language that NFA are shown in Fig.20

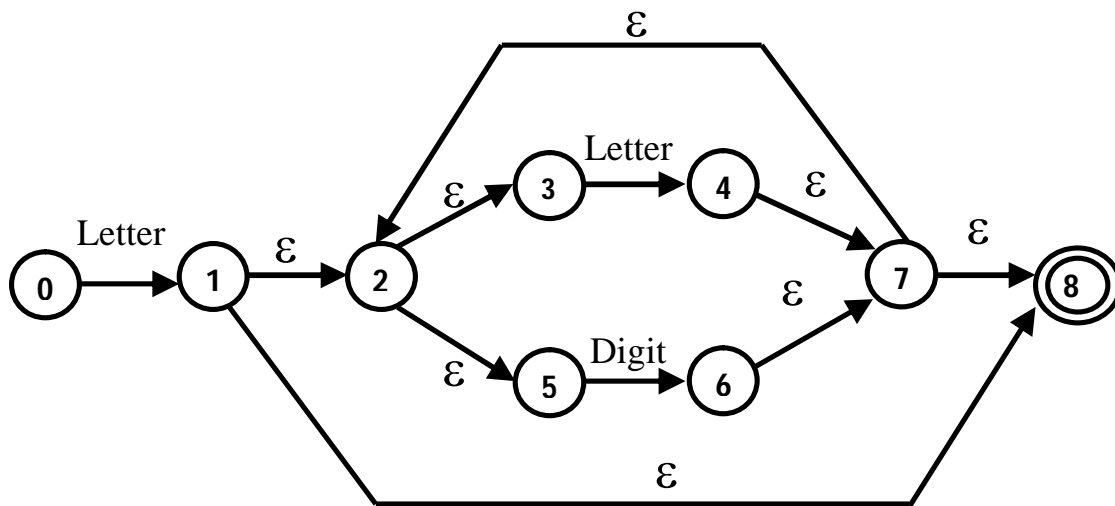
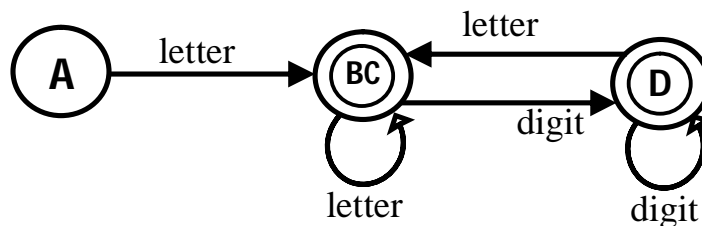


Fig.20: NFA accepting letter (letter | digit)\*

The important states in Fig.20 are {0, 3, 5}. We see that state **B** and **C** contain the same important and contain the final state (8), therefore we can merge these two states into one state named **BC**, and then the result is



BC = {1, 2, 3, 4, 5, 6, 7, 8}

After that we look to the state BC and state D and see that they have the same important state and the same final state, therefore we can merge these states into another state called BCD then the minimized DFA is shown in Fig.21.

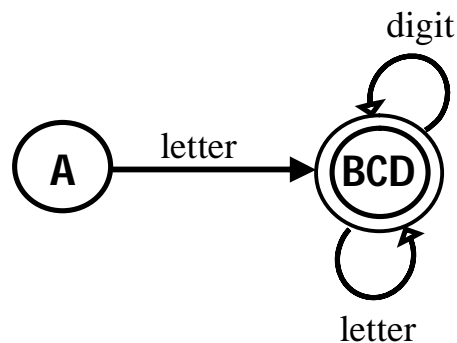


Fig.21: NFA accepting letter (letter | digit)\*

## **2- Formal Method**

In this method we don't need to experience for deciding which of the state that are legal for merging, instead of state we must use the following algorithm:-

**Algorithm:** - Minimizing the number of states of a DFA

**Input:** - A DFA  $M$  with set of states  $S$ , inputs  $\Sigma$ , transitions defined for all states and inputs, initial state  $S_0$ , and set the final state  $F$ .

**Output:** - A DFA  $M'$  accepting the same language as  $M$  and having as few states as possible.

**Method:-**

1- We construct a partition  $T$  of the set of states Initially,  $T$  consist of two groups, the final states  $F$ , and the non final states  $S-F$ . Then we construct a new partition  $T_{new}$  by the procedure of Fig (22).  $T_{new}$  will always be a refinement of  $T$ , meaning that  $T_{new}$  consists of the groups of  $T$ , each split into one or more pieces. If  $T_{new} = T$ , then no more changes can ever occur, so we terminate this part of the algorithm:

For each group  $G$  of  $T$  do

    Begin

        Partitions  $G$  into subgroups such that two states  $S$  and  $t$  of  $G$  are in the same subgroup iff for all input symbol  $a$ , states  $S$  and  $t$  have transitions on  $a$  to states in the same group of  $T$ ;

        Place all subgroups so formed in  $T_{new}$ ;

    End.

Fig (22). New partition  $T_{new}$

2-When the final partition  $T$  has been constructed in step (1), select a representative for each group, that is, an arbitrary state in the group. The representative will be the states of the reduced DFA  $M'$ . Let  $S$  be a representative state, and suppose on input  $a$  there is a transition of  $M$  from  $s$  to  $t$ . Let  $r$  be the representative of  $t$ 's group ( $r$  may be  $t$ ). Then  $M'$  has a transition from  $S$  to  $r$  on  $a$ . Let the initial state of  $M'$  be the representative of the group containing the initial  $S_0$



of  $M$ , and let the final states of  $M'$  be the representative which are in  $F$ . Note that each group of  $T$  either consists only of states in  $F$  or has no states in  $F$ .

**3-** If  $M'$  has a dead state, that is, a state  $d$  which is not final and with transitions to itself on all input symbol  $a$ , then remove  $d$  from  $M'$ . Also remove any states not reachable from the initial state. Any transitions to  $d$  from other states become undefined.

**Example:-** Minimize The DFA shown in Fig.18

**Solution:** The initial partition of  $T$  consists of two groups:

- 1- The final states:- which contains (E)
- 2- The non final states:- which contains (ABCD).

To construct  $T_{\text{new}}$  by the algorithm of Fig (22),

**(1)** We first consider (E), since this consists of one state, it cannot be further split, so we place (E) in  $T_{\text{new}}$ .

**(2)** Now consider (ABCD) and do the following:-

**(a)** If the input is  $a$ , then each of these states goes to B, so they could all be placed in one group as far as input  $a$  is consumed.

**(b)** If the input is  $b$ , then A, B, and C go to members of the group (ABCD) of  $T$ , while D goes to E, a member of another group. Thus, in  $T_{\text{new}}$  (ABCD) must be split into groups (ABC) and (D). The new value of  $T$  is thus (ABC)(D) (E).

Constructing the next  $T_{\text{new}}$ , we given have no splitting on input  $a$ , but (ABC) must be split into two groups (AC) (B), since on input  $b$  A and C each go to C, while B goes to D, a member of a group of  $T$  different from that of C. Thus the next value of  $T$  is (AC) (B) (D) (E). When we construct  $T_{\text{new}}$  from this, we cannot split any of the groups consisting of a single state. The only possibility is that (AC) could be split. But A and C go to the same state, B, on input  $a$ , and they go to the same state, C, on input  $b$ . Hence,  $T_{\text{new}}=T$ , and the final partition  $T$  from step (1) of algorithm is (AC) (B) (D) (E).

Let us now choose representatives. Of course B, D and E represent the group containing only themselves. Let us choose A to represent the group (AC). Then the transition table for the reduce automaton is shown in Fig.23.

	State	Input	
		a	b
(Start)	A	B	A
	B	B	D
	D	B	E
(Accept)	E	B	A

Fig.23: Reduced automaton

For example, E goes to C on input b in the automata of Fig.18. **Since A is the representative of the group for C**, we have a transition from E to A on input b in Fig.23. A similar change has taken place in entry for A on input b, and all other transitions are copied from Fig.18.

There is no dead state in Fig.23, and all states are reachable from the initial state A. Fig.24 shows the minimized DFA.

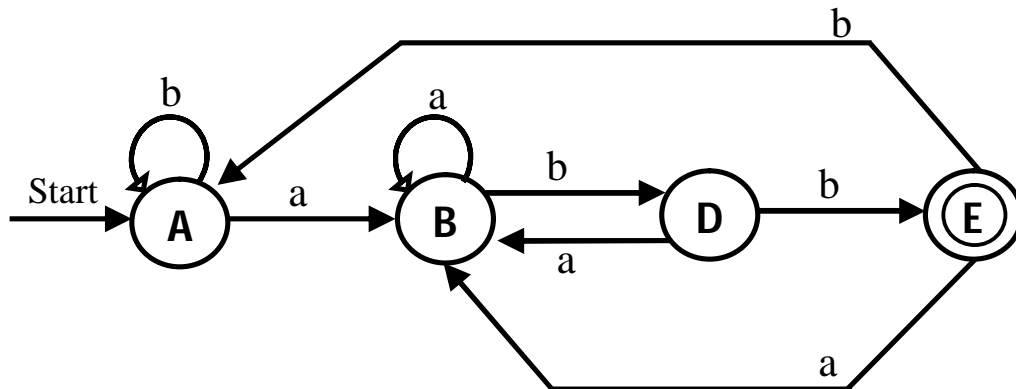


Fig.24: Minimized DFA accepting  $(a \mid b)^*abb$ .