Minimizing the Number of states of a DFA

The DFA of Fig (16) constructed from the NFA of fig (14) is not the smallest possible. In particular, we show in fig (10), another DFA with only four states that also accept \((a \mid b)^*abb\). Part of the reason for this non-minimality 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 into one state if they have the same important states. The important state is the state with no \(\varepsilon\)-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 (17) according to the Empirical method.

![Diagram](image-url)
Fig (17) DFA accepting letter (letter | digit)*
A= \{0\}, B= \{1, 2, 3, 5\}, C= \{4, 7, 8, 2, 3, 5\}, D= \{6, 7, 8, 2, 3, 5\}
Final is state 8.
From fig (17) we know that the language that NFA are shown in fig (18)

Fig (18) NFA accepting letter (letter | digit)*

The important states in Fig (18) are \{0, 3, 5\}. We see that state B and C contain the same important state and contain the final state (8), therefore we can merge these two states into one state named BC, then the result is

BC= \{1, 2, 3, 4, 5, 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 (19).

![DFA diagram](https://via.placeholder.com/150)

**Fig (19) DFA accepting letter (letter | digit)***

### 1- Formal Method

In this method we don’t need an experience for deciding which of the state that are legal for merging, instead of that 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 states \( F \).

**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 \( T_{\text{new}} \) by the procedure of fig (20). \( T_{\text{new}} \) will always be a refinement of \( T \), meaning that \( T_{\text{new}} \) consists of the groups of \( T \), each split
into one or more pieces. If \( T_{\text{new}} \neq T \), we replace \( T \) by \( T_{\text{new}} \) and repeat the procedure of Fig. 20. if \( T_{\text{new}} = T \), 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 to states in the same group of \( T \);

Place all subgroups so formed in \( T_{\text{new}} \);

End.

Fig (20) Construction of \( T_{\text{new}} \)

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 (16)

**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 (20),

(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 concerned.

(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 again 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 \textit{a}, and they go to the same state, C, on input \textit{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 (21).

\begin{center}
\begin{tabular}{|l|l|l|}
\hline
State & Input & \textit{a} & \textit{b} \\
\hline
A & B & A \\
B & B & D \\
D & B & E \\
E & B & A \\
\hline
\end{tabular}
\end{center}

Fig (21) Reduced automaton

For example, E goes to C on input \textit{b} in the automata of fig (16). Since A is the representative of the group for C, we have a transition from E to A on input \textit{b} in fig (21). A similar change has taken place in entry for A on input \textit{b}, and all other transitions are copied from fig (16). There is no dead state in fig (21), and all states are reachable from the initial state A. Fig (22) shows the minimized DFA.

Fig (22): Minimized DFA accepting (a | b)*abb.