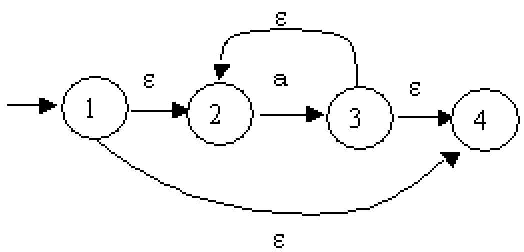
Converting NFAs to DFAs

Note: to view the symbols on this page correctly, you should be using a PC with the Windows TrueType Symbol font loaded.

Converting a nfa to a dfa

Defn: The **e-closure of a state** is the set of all states, including S itself, that you can get to via e-transitions. The e-closure of state S is denoted: $\overline{S}$

Example:



The e-closure of state 1: $\overline{1}$ = { 1, 2, 4 }

The e-closure of state 3: $\overline{3}$ = { 3, 2, 4 }

Defn: The **e-closure of a set of states** S_1, ... S_n is
$\overline{S}_1$ ï¿½ $\overline{S}_2$ ï¿½ ... ï¿½ $\overline{S}_n$.

Example: The e-closure for above states 1 and 3 is

{ 1, 2, 4 } ï¿½ { 3, 2, 4 } = { 1, 2, 3, 4 }

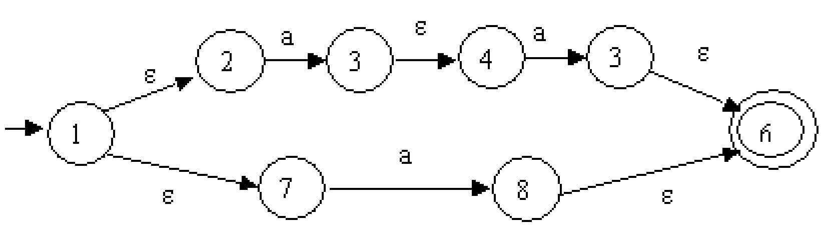
To construct a dfa from a nfa:

Step 1: Let the start state of the dfa be formed from the e-closure of the start state of the nfa.

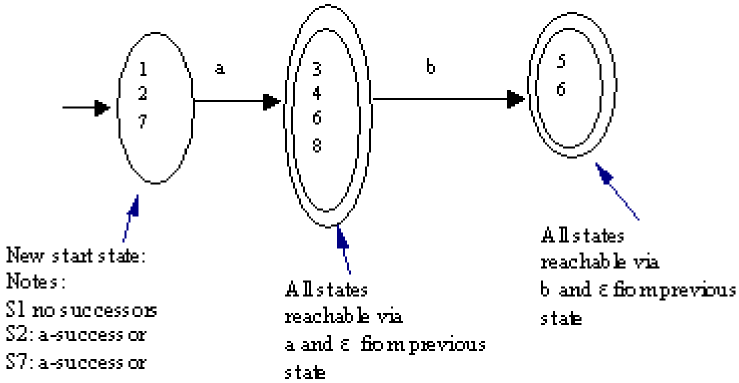
Subsequent steps: If S is any state that you have previously constructed for the dfa and it is formed from say states t_1, ... , t_r of the nfa, then for any symbol x for which at least one of the states t_1, ... , t_r has a x-successor, the x-successor of S is the e-closure of the x-successors of t_1, ... , t_r.

Any state of the dfa which is formed from an accepting state, among others, of the nfa becomes an accepting state.

Example 1: To convert the following nfa:

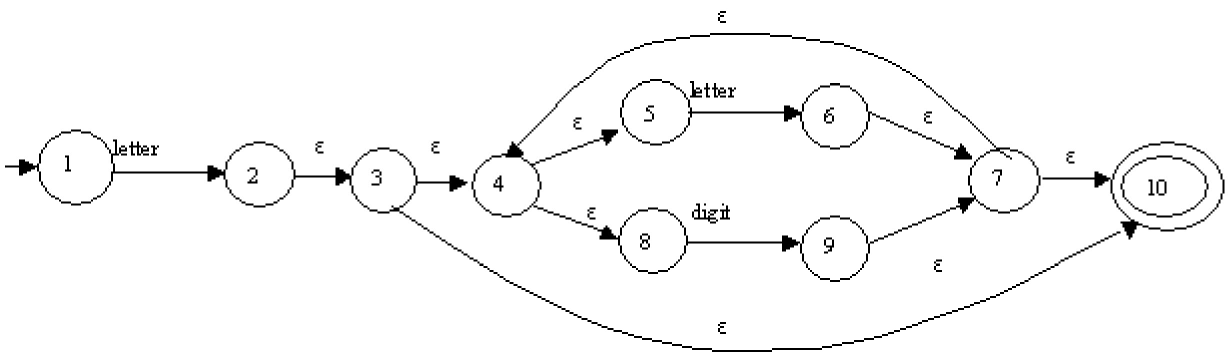


we get:

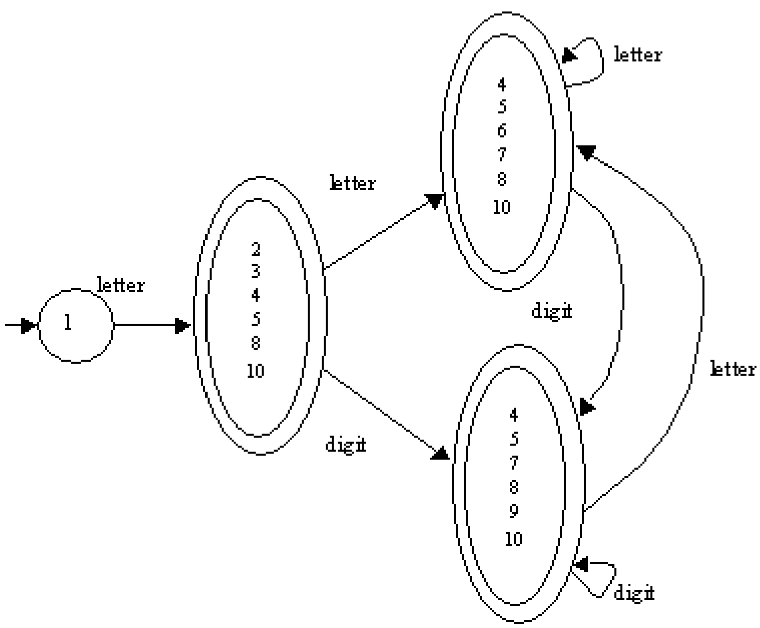



```
This constructs a dfa that has no epsilon-transitions and a
single accepting state.

Example 2:  To convert the nfa for an identifier to a dfa
```




```
we get:
```

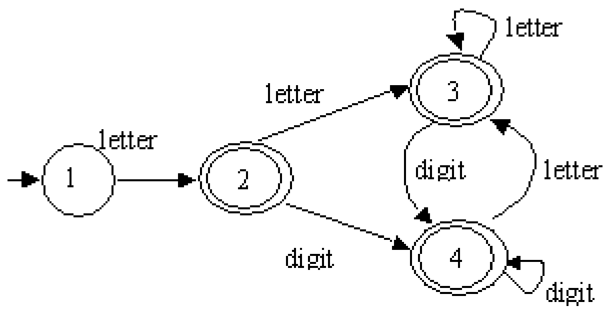


Minimizing the Number of States in a DFA

```
Step 1:  Start with two sets of states
         (a) all the accepting states, and
         (b) all the nonaccepting states
```

```
Subsequent steps:
         Given the sets of states S1, ... Sr consider
         each state S and each symbol x in turn.  If any
         member of S has a x-successor and this x-successor
         is in say S', then unless all the members of S
         have x-successors that are in S', split up S into
         those members whose x-successors are in S' and the
         others (which don't have x-successors in S').
```

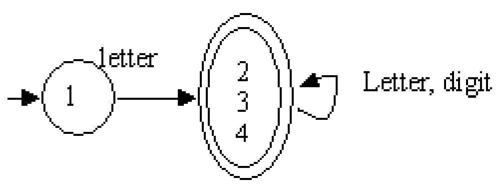
Example 1. Consider the dfa we constructed for an identifier (with renumbered states):



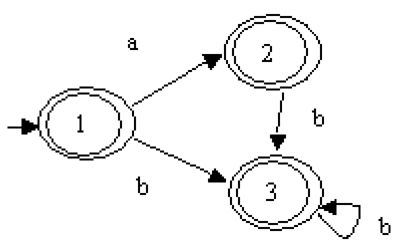
The sets of states for this dfa are:

S1 Nonaccepting states:	S2 Accepting states:	
1	2 3 4	All states in S2 have the successors letter-successor and digit-successor, and the successor states are all in the set of states S2.

Combine all the states of S2 to get:



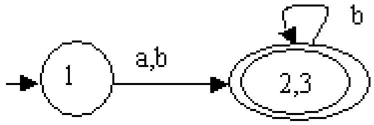
Example 2. Consider the dfa:



All of the states (1, 2, and 3) are accepting states and all their successors are also accepting states, but state 1 has an a-successor whereas states 2 and 3 do not.

So, we split the set of accepting states into two sets S1 and S2 where:
S1 consists of state 1, and
S2 consists of states 2, 3

to get:


b
1
a,b
2,3