Deterministic Finite Automata[1]

A *deterministic finite automaton* (DFA for short) is a special case of a non-deterministic finite automaton in which

1. No state has an Î -transition, i.e. a transition on input Î
2. for each state *s* and input symbol *a*, there is at most one edge labeled a leaving *s*.

A deterministic finite automaton has at most one transition from each state on any input. If we are using a transition table to represent the transition function of a DFA, then each entry in the transition table is a single state. As a consequence, it is very easy to determine whether a deterministic finite automaton accepts an input string, since there is at most one path from the start state labeled by that string. The following algorithm shows how to simulate the behavior of a DFA on an input string.

Input. An input string *x* terminated by and end-of-file character **eof**. A DFA *D* with start state s_0 and set of accepting states *F*.

Output. The answer "yes" if *D* accepts x; "no" otherwise.

Method. Apply the algorithm given below to the input string *x*. The function *move*(*s*,*c*) gives the state to which there is a transition from state *s* on input character *c*. The function *nextchar* returns the next character of the input string *x*.

```
s:=s0;
c := nextchar;
while c ≠ eof do
    s := move(s,c);
    c := nextchar
end;
if x is in F then
    return "yes"
else
    return "no";
```

[1] Excepted from Aho, Sethi, and Ullman *Compilers, principles, techniques, and tools*. Addison-Wesley, 1986, ISBN 0-201-10088-6

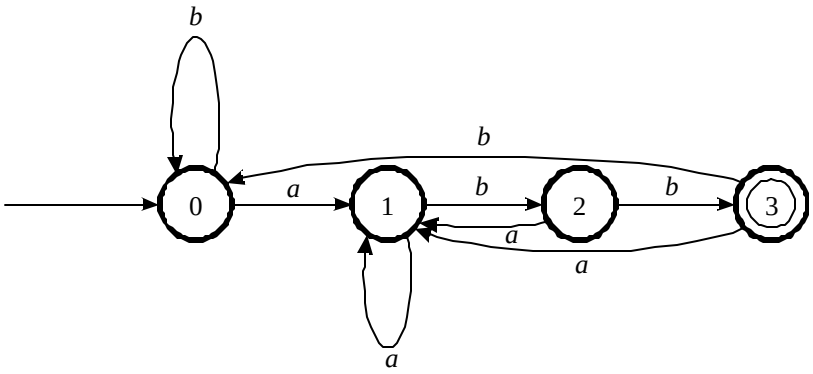

Figure 1. DFA accepting (*a*|*b*)**abb*

State	Input Symbol	
	a	***b***
0	1	0
1	1	2
2	1	3
3	1	0

Table 1. Transition function for the DFA of Figure 1.

1. $S = \{0,1,2,3\}$
2. Table 1 shows the transition function move for the DFA of Figure 1.
3. $\Sigma = \{a,b\}$
4. $s_0 = 0$
5. $F = \{3\}$