Algorithm 3.2. (*Subset construction.*) Constructing a DFA from an NFA.

Input. An NFA N.

Output. A DFA D accepting the same language.

Method. Our algorithm constructs a transition table *Dtran* for D. Each DFA state is a set of NFA states and we construct *Dtran* so that D will simulate "in parallel" all possible moves N can make on a given input string.

We use the operations in Fig. 3.24 to keep track of sets of NFA states (s represents an NFA state and T a set of NFA states).

OPERATION	DESCRIPTION
ϵ-*closure*(s)	Set of NFA states reachable from NFA state s on ϵ-transitions alone.
ϵ-*closure*(T)	Set of NFA states reachable from some NFA state s in T on ϵ-transitions alone.
move(T, a)	Set of NFA states to which there is a transition on input symbol a from some NFA state s in T.

Fig. 3.24. Operations on NFA states.

Before it sees the first input symbol, N can be in any of the states in the set ϵ-*closure*(s_0), where s_0 is the start state of N. Suppose that exactly the states in set T are reachable from s_0 on a given sequence of input symbols, and let a be the next input symbol. On seeing a, N can move to any of the states in the set *move*(T, a). When we allow for ϵ-transitions, N can be in any of the states in ϵ-*closure*(*move*(T, a)), after seeing the a.

```
initially, ε-closure(s₀) is the only state in Dstates and it is unmarked;
while there is an unmarked state T in Dstates do begin
      mark T;
      for each input symbol a do begin
            U := ε-closure(move(T, a));
            if U is not in Dstates then
                  add U as an unmarked state to Dstates;
            Dtran[T, a] := U
      end
end
```

Fig. 3.25. The subset construction.

We construct *Dstates*, the set of states of D, and *Dtran*, the transition table for D, in the following manner. Each state of D corresponds to a set of NFA