CONVERSION FROM NFA TO DFA USING SUBSET CONSTRUCTION METHOD – SCHEDULED ON 17.12.07 4th HOUR

Going Deterministic: Subset Construction

Using Thompson's construction, we can build an NFA from a regular expression, we can then employ *subset construction* to convert the NFA to a DFA. Subset construction is an algorithm for constructing the deterministic FA that recognizes the same language as the original nondeterministic FA. Each state in the new DFA is made up of a set of states from the original NFA. The start state of the DFA will be the start state of NFA. The alphabet for both automata is the same.

Subset construction algorithm

In: NFA N
Out: DFA D
Method: Construct transition table Dtran. Each DFA state is a set of NFA states. Dtran simulates in parallel all possible moves N can make on a given string.

Operations to keep track of sets of NFA states

ε_closure(s)
    set of states reachable from state s via ε
ε_closure(T)
    set of states reachable from any state in set T via ε
move(T,a)
    set of states to which there is an NFA transition from states in T on symbol a

Subset construction

```
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
```

Computation of ε –closure

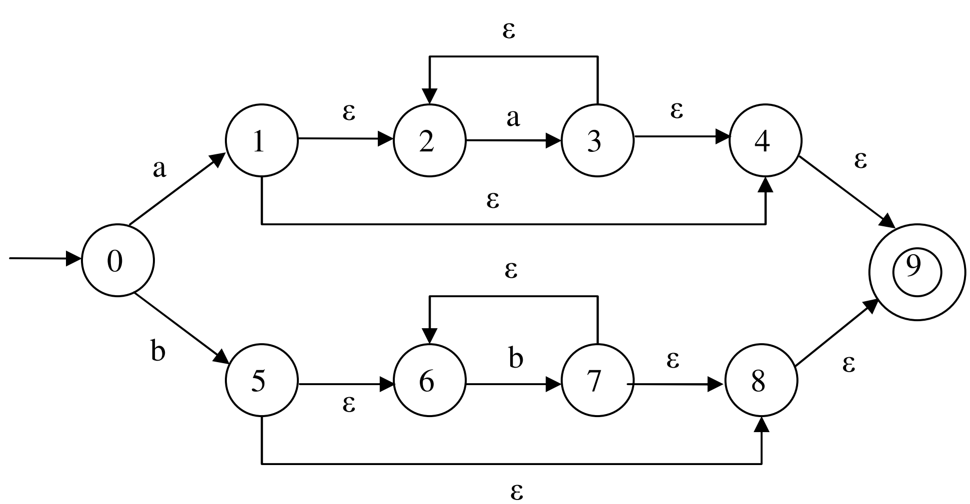
```
push all states in T onto stack;
initialize ε-closure(T) to T;
while stack is not empty do begin
        pop t, the top element from the stack;
        for each state u with an edge from t to u labeled ε do
                if u is not in ε-closure(T) do begin
                        add u to ε-closure(T)
                        push u onto stack
                end
end
```

Example

NFA N accepting the language aa*|bb* is shown in the figure below.

Step1:

The start state of the equivalent DFA is ε-closure(0), which is A={0}, since these are exactly the states reachable from 0 via a path in which every edge is labeled ε. ε-closure(0) is computed using the algorithm **Computation of ε –closure.**



The input symbol alphabet is {a, b}.

State A is added to Dstates and it is unmarked. See whether any unmarked states are there in Dstates. A is there. Mark it. Compute (ε-closure(move(A,x)) for each input alphabet x.

Step 2:

move(A,a) ={ } – this is nothing but the states that are reached from some state in A on input symbol a.
For example in A we have {0}. {0} on input symbol a moves to state 1. so move(A, a)={1}.
(ε-closure(move(A,a)) => ε-closure({1})={1,2,4,9} = B

If state B is not in Dstates then add to it as unmarked state.

move(A,b) ={ } – this is nothing but the states that are reached from some state in A on input symbol b.
For example in A we have {0}. {0} on input symbol b moves to state 5. so move(A, b)={5}.
(ε-closure(move(A,b)) => ε-closure({1})={5,6,8,9} = C

If state C is not in Dstates then add to it as unmarked state.

Step 2 is repeated there are no unmarked states.

Continuing like this we get the following

move(B,a)={3}
(ε-closure(move(B,a)) => ε-closure({3})={2,3,4,9} = D

move(B,b)={}
(ε-closure(move(B,b)) => ε-closure({})= -

move(C,a)={}
(ε-closure(move(C,a)) => ε-closure({})= -

move(C,b)={7}
(ε-closure(move(C,b)) => ε-closure({7})={6,7,8,9} = E

move(D,a)={3}
(ε-closure(move(D,a)) => ε-closure({3})={2,3,4,9} = C

move(E,a)={}
(ε-closure(move(E,a)) => ε-closure({})= -

move(E,b)={7}
(ε-closure(move(E,b)) => ε-closure({7})= ={6,7,8,9} = E

Transition table for DFA is given below

STATE	INPUT SYMBOL	
	a	b
A	B	C
B	D	-
C	-	E
D	D	-
E	-	E