lec 4:minimize DFA

1.Finite State Automata with ε moves:

If the definition of a NFSA is altered , so that moves from one state to another can be accomplished without any input we say that the automaton has ε-moves .More formally , a NFSA M=(Q , ∑ , t ,S,F) has ε-moves if t, instead of being a function Q x ∑→2Q , is defined as a function

$Q \times (\sum \cup \{\varepsilon\}) \rightarrow 2Q$.

$t\text{-}(Q, \varepsilon)=R(Q)$

$t\text{-}(Q, \sum)=R(t(R(Q), \sum))$

Convert NFSA with empty move into NFSA without empty move.

If there is NFSA with empty move M = (Q, ∑, q0 , t , F) then there is FSA without empty move M- = (Q- , ∑, q0 , t- , F-)

Define $t : Q\times(\sum \cup \{\varepsilon\}) \rightarrow Q$

By

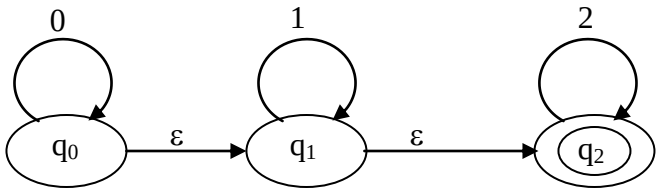
$t'(K,\varepsilon)=R(K)$ or ε-closure(K)

$t'(K,a)=R(t(R(K),a))$

The set of final states F- is $F \cup \{q\}$ if $R(q) \in F$

Example:

Convert the NFSA with empty move into without empty move.



t'(q0,0)=R(t(R(q0),0))=R(t({q0,q1,q2},0))={q0,q1,q2}

t'(q0,1)=R(t(R(q0),1))= R(t({q0,q1,q2},1))={q1,q2}

t'(q0,2)= R(t(R(q0),2))= R(t({q0,q1,q2},2))={q2}

t'(q1,0)=R(t(R(q1),0))=R(t({q1,q2},0))={}

t'(q1,1)=R(t(R(q1),1))= R(t({q1,q2},1))={q1,q2}

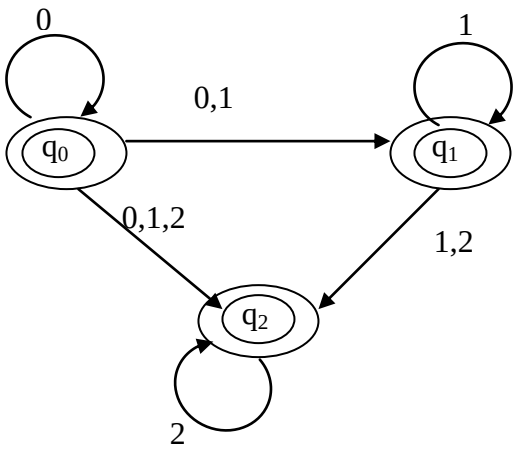
t'(q1,2)= R(t(R(q1),2))= R(t({q1,q2},2))={q2}

t'(q2,0)=R(t(R(q2),0))=R(t({q2},0))={}

t'(q2,1)=R(t(R(q2),1))= R(t({q2},1))={}

t'(q2,2)= R(t(R(q2),2))= R(t({q2},2))={q2}

F-={q0,q1,q2}



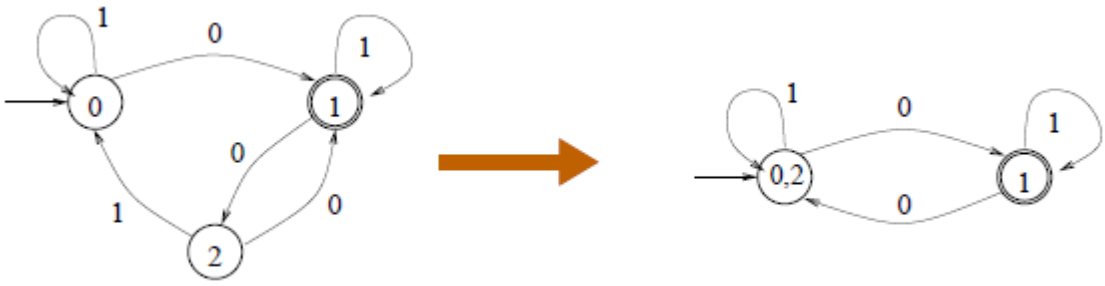
2. Minimization of Finite Automata

We now consider the following problem: for a given DFA A, and an equivalent DFA with a minimum number of states.

We start with two examples. Consider the automaton on the left

You can see that it is not possible to ever visit state 2. States like this are called unreachable. We can simply remove them from the automaton without changing its behavior. (This will be, indeed, the frst step in our minimization algorithm.) In our case, after removing state 2, we get the automaton on the right.

As it turns out, however, removing unreachable states is not su_cient. The next example is a bit more subtle (see the automaton on the left):



Let us compare what happens if we start processing some string w from state 0 or from state 2. I claim that the result will be always the same. (For now I didn't specify what exactly I mean by \the result". It will become clear soon.) This can be seen as follows. If w = λ, we will stay in either of the two states and λ will not be accepted. If w starts with 0, in both cases we will go to state 1 and both computations will be now identical. Similarly, if w starts with 1, in both cases we will go to state 0 and both computations will be identical. So no matter what w is, either we accept w in both cases or we reject w in both cases. Intuitively, from the point of view of the \future", it does not matter whether we start from state 0 or state 2. Therefore the transitions into state 0 can be redirected into state 2 and vice versa, without changing the outcome of the computation. Alternatively, we can combine states 0 and 2 into one state.

We now formalize the above idea. Let A be a finite automaton. We will say that w ∈ $\sum^*$distinguishes between two states p; q ∈ Q if either $\delta(p,w) \in F \ \&\ \delta(q,w) \notin F$, or $\delta(p,w) \notin F \ \&\ \delta(q,w) \in F$ Two states p; q ∈ Q are called distinguishable iff there is a word that distinguishes between them. States that are indistinguishable will also be sometimes called equivalent

algorithm

The algorithm for marking distinct states follows the above (recursive) definition. Create a table Distinct with an entry for each pair of states. Table cells are initially blank.

(1) For every pair of states (p; q)

If p is final and q is not, or vice versa,

Set Distinct(p; q) to be ε.

(2) Loop until there is no change in the table contents

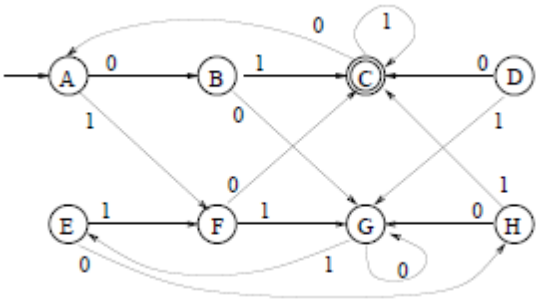
For each pair of states (p; q) and each character a in the alphabet:

if Distinct(p; q) is empty and Distinct (δ(p; a); δ (q; a)) is not empty

Set Distinct(p; q) to be a.

(3) Two states p and q are distinct iff Distinct(p; q) is not empty

Example: We apply our algorithm to the automaton given below:



Step 1: We have one unreachable state, state D. We remove this state and proceed to Step 2.

Step 2: We _rst mark pairs of _nal and non-_nal states, getting the following tableau:

B						
C	X	X				
E			X			
F			X			
G			X			
H			X			
	A	B	C	E	F	G

In the _rst iteration we examine all unmarked pairs. For example, for pair A;B, we get $\delta(A; 1) = F$ and $\delta(B; 1) = C$, and the pair C; F is marked, so we mark A;B too. After doing it for all pairs, we get the following tableau.

B	X					
C	X	X				
E		X	X			
F	X	X	X	X		
G		X	X		X	
H	X		X	X	X	X
	A	B	C	E	F	G

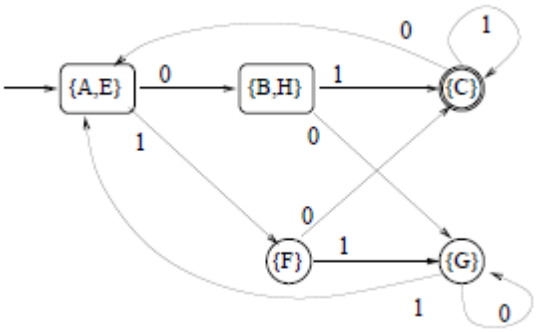
In the next iteration, we examine the remaining pairs. For example, we will mark pair A;G because $\delta(A; 0) = B$ and $\delta(G; 0) = G$, and the pair B;G is marked. When we're done we get the following tableau

B	X					
C	X	X				
E		X	X			
F	X	X	X	X		
G	X	X	X	X	X	
H	X		X	X	X	X
	A	B	C	E	F	G

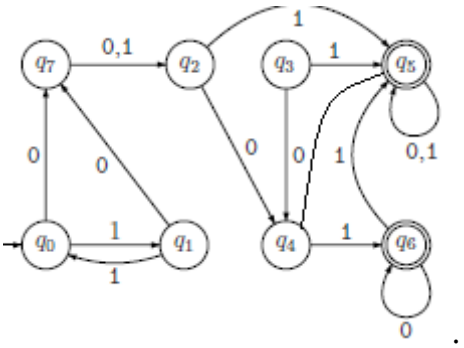
One more iteration will be executed now, but no new distinguishable pairs will be discovered. So we are

done with Step 2 now.

Step 3: We group the states into the equivalence classes. Since A;E are equivalent and B;H are equivalent,the classes are: $\{A, E\}, \{B, H\}, \{C\}, \{F\}, \{G\}$. The minimal automaton A^ is:



example: minimize the flowing DFA



step (1):

q_0								
q_1								
q_2								
q_3								
q_4								
q_5	ϵ	ϵ	ϵ	ϵ	ϵ			
q_6	ϵ	ϵ	ϵ	ϵ	ϵ			
q_7						ϵ	ϵ	
	q_0	q_1	q_2	q_3	q_4	q_5	q_6	q_7

(Note, that for a pair of states (qi; qj) we need only a single entry since (qj ; qi) is equivalent, and we do not need to consider pair on the diagonal of the form (qi; qi).)

q_0								
q_1								
q_2	1	1						
q_3	1	1						
q_4	0	0	0	0				
q_5	ϵ	ϵ	ϵ	ϵ	ϵ			
q_6	ϵ	ϵ	ϵ	ϵ	ϵ			
q_7			1	1	0	ϵ	ϵ	
	q_0	q_1	q_2	q_3	q_4	q_5	q_6	q_7

After one iteration of step (2)

q_0								
q_1								
q_2	1	1						
q_3	1	1						
q_4	0	0	0	0				
q_5	ϵ	ϵ	ϵ	ϵ	ϵ			
q_6	ϵ	ϵ	ϵ	ϵ	ϵ			
q_7	1	1	1	1	0	ϵ	ϵ	
	q_0	q_1	q_2	q_3	q_4	q_5	q_6	q_7

After the second iteration of step (2)

Third iteration of step (2) makes no changes to the table, so we halt. The cells (q0; q1), (q2; q3) and (q5; q6) are still empty, so these pairs of states are not distinct. Merging them produces the following simpler DFA recognizing the same language.