| *#Now that we have the NFA, time to get a DFA out of it*  *#Powerset construction involves tracking which NFA states are currently active, and representing this entire set with a single DFA state*   *#Same as NFA, except it can have multiple accepting states*  *#Of course, NFAs can actually have multiple accepting states as well,* *#But we pretended they can have only one during Thompson's construction in order to make things easier (and without loss of generality)* class DFA {  State start  State[] accept   State[] states   Edge[] transitions }  #We will need to a subroutine to get the epsilon closure of an arbitrary #NFA state  #The epsilon closure of a state S, to remind you, is the set of all states #that can be reached from S  #By taking epsilon transitions only, i.e. without consuming any input  #They are very relevant to Powerset Construction because those are the #states you can be in simultaneously subroutine Epsilon-Closure of   Input: State s   Input: Edge[] transitions   Output: Set<State> eps    *#Epsilon Closures are very easy once you recursively define them as follows*   *#A state R is in the epsilon closure of a state S if*  *#*  *#1- S is R*  *#*  *#2- R can be reached via an epsilon transition from any state in the epsilon closure of S*   *#Let's translate that to code*  eps = {S}   new\_states\_to\_be\_added = True   while new\_states\_to\_be\_added {  new\_states\_to\_be\_added = False   For all states T in eps {  For all transitions (T,R,{}) {  if next state R is not already in eps {  add R to eps  new\_states\_to\_be\_added = True  }  }  }  }  return eps  subroutine Powerset-Construction of   Input: NFA   Output: DFA     dfa-start = Epsilon-Closure(nfa.start,nfa.transitions)  dfa-accept = []  dfa-states = {}  dfa-transitions = []   superstates-with-unknown-transitions = {dfa-start}   while superstates-with-unknown-transitions is not empty {  superstate = superstates-with-unknown-transitions.pop()  add superstate to dfa-states   if nfa.accept in superstate {  dfa-accept.push(superstate)  }   superstate-transition-table = new map   for all edges (a,b,chars) in nfa.transitions {  if a in superstate and chars is not {} {  if chars in superstate-transition-table {  superstate-transition-table[chars].union(Epsilon-Closure(b))  }   else {  superstate-transition-table[chars] = Epsilon-Closure(b)  }   }  }   for all (chars,next-superstate) in superstates-transition-table {  dfa-transitions.push(new Edge(superstate,next-superstate,chars))   if next-superstate is not already in dfa-states {  add it to superstates-with-unknown-transitions  }   }   }    return new DFA(dfa-start,  dfa-accept,  dfa-states as list,  dfa-transitions) |
| --- |