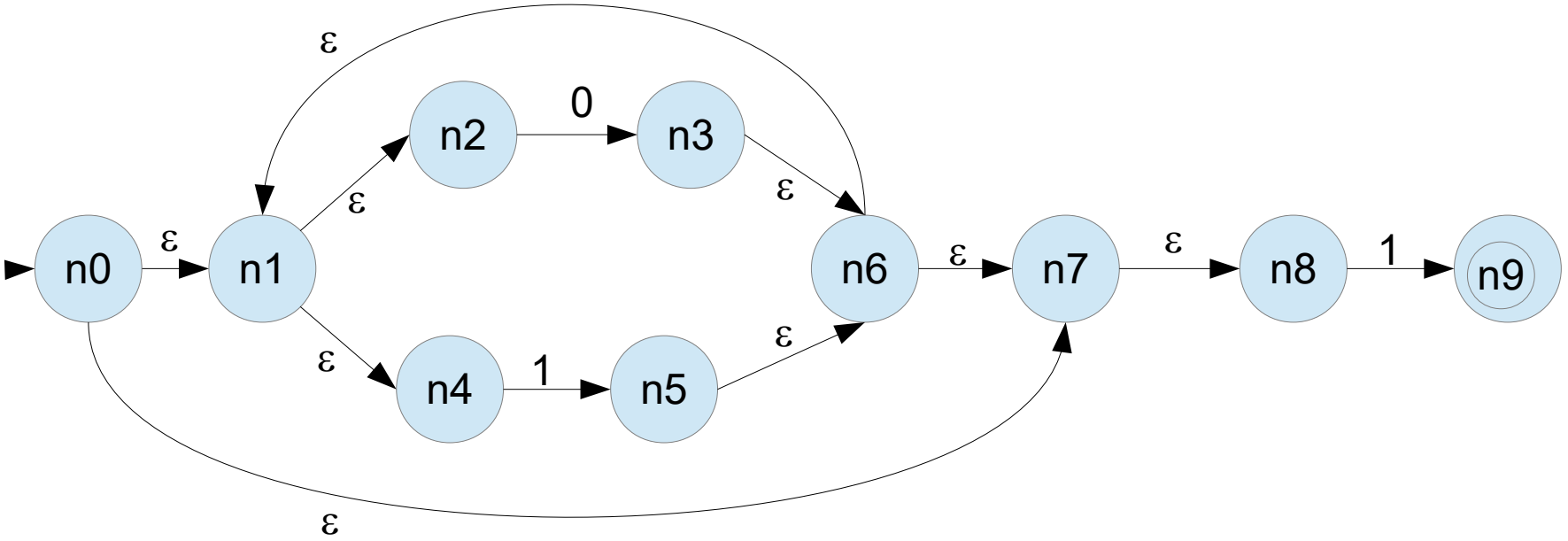
# Subset construction

- Given an NFA, we want to build a matching DFA
- Each char consumed in the NFA, together with possible null transitions, can lead to any one of a set of states
- Each state in the DFA correspond to the set of NFA states that are reachable by a specific input pattern (with null transitions)
- For N states in NFA, there are 2^N possible subsets, thus theoretically 2^N possible states in the DFA
- We'll try and build the DFA for just the usable subsets

# Example: (0|1)*1

- Reachable states from n2 consuming 0 then null transitions are n3, n6, n7, n8, n1, n2, n4



# Algorithm setup

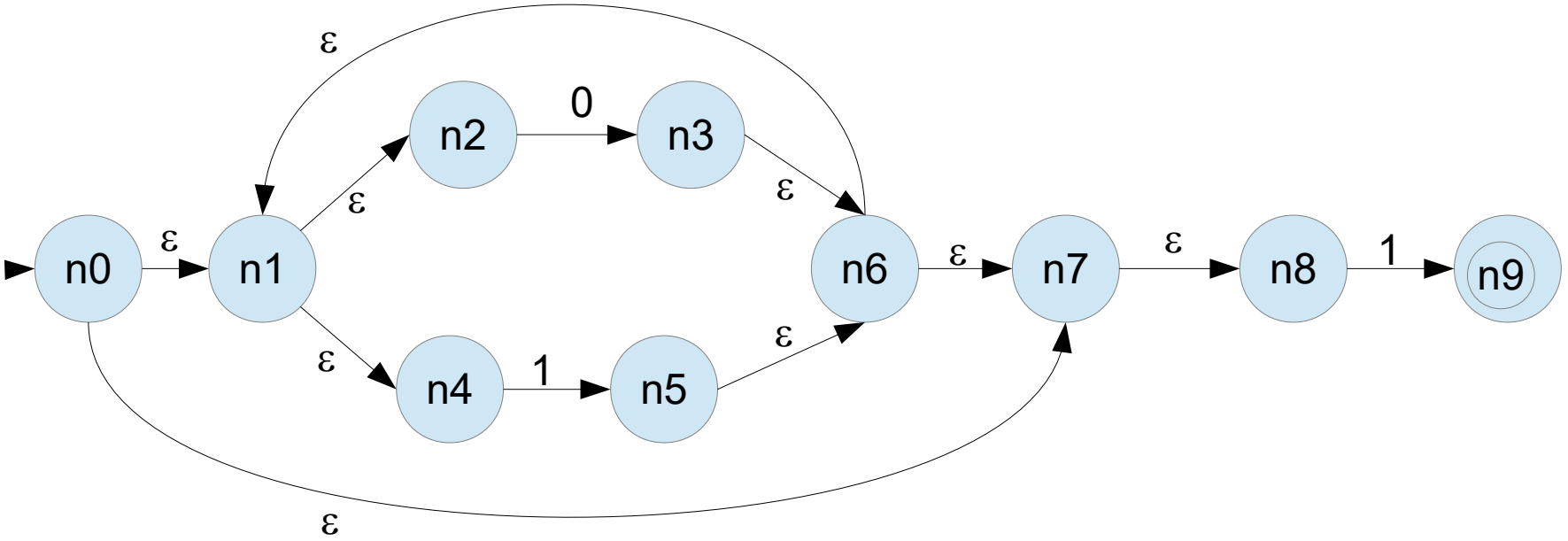
- Given the NFA and its start state n0, we'll gradually build up a set of DFA states, each one representing some subset of the NFA states
- The DFA's start state, q0, will represent n0 plus all the NFA states reachable by null transitions from n0
- Q will be the set of DFA states we build
- T[q][c] will represent the DFA transition table: given state q and input c it tells us what the next state will be
- ToDo will be a set of DFA states we need to process

# The subset construction algorithm

```
add q0 to Q, and to ToDo
while ToDo isn't empty:
    remove the next state, q, from ToDo
    for each character, c, in the alphabet:
        tmp = set of reachable states in NFA from q
        T[q][c] = tmp
        if tmp is not yet in Q then add it to Q and ToDo
```

# Example: (0|1)*1

- Recall our example from Thompson's construction



# Example setup

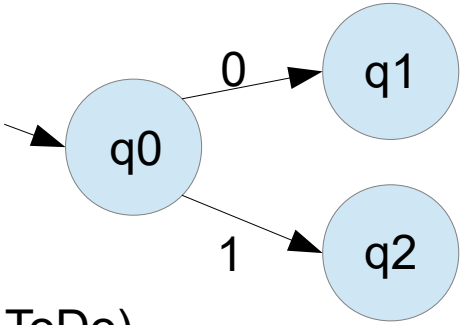
- DFA's start state, q0, is set of reachable states in NFA, i.e. q0 = { n0, n1, n2, n4, n7, n8 }
- Add q0 to Q, i.e. Q = { q0 }
- Add q0 to ToDo list, i.e. ToDo = { q0 }
- Now we'll start making passes through the while loop until the ToDo list is empty (some passes will add new DFA states to Q and the ToDo list)
- Remember each DFA state, qi, will be identified/labelled by a subset of NFA states

# While loop first pass, q0

- q = q0
- First pass through for loop, c=0
- tmp = states reachable from q using c then null transitions
- tmp = { n3, n6, n7, n8, n1, n4 } ... I'm going to call this q1
- T[q0][0] = q1, add q1 to Q and ToDo
- Second pass through for loop, c = 1
- tmp = { n5, n9, n6, n7, n8, n1, n2, n4 } ... call it q2
- T[q0][1] = q2, add q2 to Q and ToDo

# While loop second pass, q1

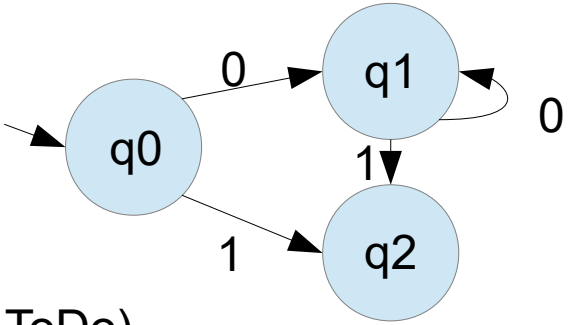
- Q so far is { q0, q1, q2 }
- ToDo is currently { q1, q2 }




Next while loop pass, tmp = q1 (remove from ToDo)
On char 0 the reachable states from q1 are
{ n1, n2, n3, n4, n6, n7, n8}
This is the same as q1
T[q1][0] = q1, and no need to add to Q or ToDo
On char 1 the reachable states from q1 are
{ n1, n2, n4, n5, n6, n7, n8, n9 }
This is the same as q2
T[q1][1] = q2, and no need to add to Q or ToDo

# While loop second pass, q1

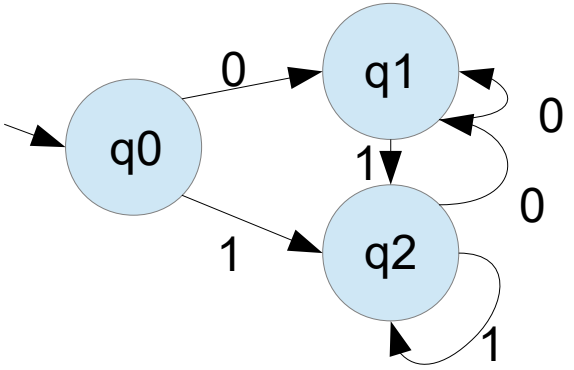
- Q so far is { q0, q1, q2 }
- ToDo is currently {  q2 }




Next while loop pass, tmp = q2 (remove from ToDo)
On char 0 the reachable states from q2 are
  { n1, n2, n3, n4, n6, n7, n8}
This is the same as q1
T[q2][0] = q1, and no need to add to Q or ToDo
On char 1 the reachable states from q2 are
  { n1, n2, n4, n5, n6, n7, n8, n9 }
This is the same as q2
T[q2][1] = q2, and no need to add to Q or ToDo

# Wrapup

- Q so far is { q0, q1, q2 }
- ToDo is currently empty




We're done!

Got lucky and produced a minimal DFA this time, for most larger NFAs
You'll wind up with sets of equivalent states and a DFA that is much larger
Than it needs to be

So .... next time is on to Hopcroft's algorithm for DFA minimization

# Whoops! reachability

- We've mentioned getting “all the states reachable from x” on some input
- Assuming we've got a graph (matrix or edge-list) representation of our NFA, then for each node we can look up which transitions exist from any given node
- Our reachability algorithm would involve creating a ToDo list and a Reachable list, and processing one element at a time until the ToDo list was empty
- Get a node from the ToDo list, check if any valid transitions exist to nodes not already in the Reachable list. If any are found, then add those nodes to the ToDo and Reachable lists