Equivalence of DFAs and NDFAs

- every language accepted by a DFA is accepted by an NDFA
  - a DFA is (essentially) an NDFA – NDFA does not require multiple or ε-transitions, and for $\delta(q,a) = q'$, $\partial(q,a) = \{q'\}$
- every language accepted by an NDFA is accepted by a DFA

  Theorem 3.2. *Every language that is accepted by an NFA is accepted by a DFA.*

  - proof idea: give an algorithm for constructing an equivalent DFA from an NDFA (then prove the algorithm correct)

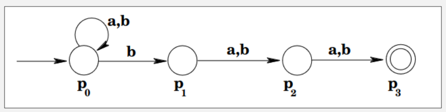
Equivalence of DFAs and NDFAs

Theorem 3.2. *Every language that is accepted by an NFA is accepted by a DFA.*

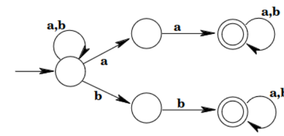
- let NFA $N = (P,\Sigma,p_0,\partial,F)$ and DFA $D = (Q,\Sigma,q_0,\delta,F)$
- idea – the states of the DFA D correspond to sets of states in the NDFA N
- q_0 corresponds to $\partial^*(p_0,\varepsilon)$
- repeatedly
  - find a state q that has been added to D but whose out-transitions have not yet been added
  - for each input symbol a, look at all of N's states that can be reached from any one of the $p_1, p_2, \ldots, p_n$ corresponding to q by consuming a (include ε-transitions)
    - add state $q' = \partial^*(p_1,a) \cup \ldots \cup \partial^*(p_n,a)$ if not already present
    - add transition $\delta(q,a) = q'$ to D
- accepting states of D are those corresponding to at least one of N's

NFA to DFA

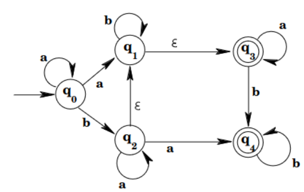
- q_0 corresponds to $\partial^*(p_0,\varepsilon)$
- repeatedly
  - find a state q that has been added to D but whose out-transitions have not yet been added
  - for each input symbol a, look at all of N's states that can be reached from any one of the $p_1, p_2, \ldots, p_n$ corresponding to q by consuming a (include ε-transitions)
    - add state $q' = \partial^*(p_1,a) \cup \ldots \cup \partial^*(p_n,a)$ if not already present
    - add transition $\delta(q,a) = q'$ to D
- accepting states of D are those corresponding to at least one of N's



2. Give a DFA that accepts the language accepted by the following NFA.



3. Give a DFA that accepts the language accepted by the following NFA. (Be sure to note that, for example, it is possible to reach both q_1 and q_3 from q_0 on consumption of an a, because of the ε-transition.)



Finite-State Automata and Regular Languages

Theorem 3.3. *Every language generated by a regular expression can be recognized by an NFA.*

- proof idea
  - definition of regular expression is recursive, so utilize proof by induction
- proof sketch
  - simplest regular expressions are Φ, ε, *a*
    - build NFA for each of these
  - regular expression operators are |, •, *
    - build NFA for each of these