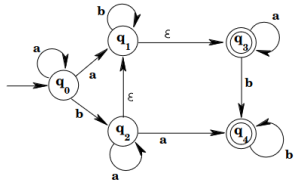
3. Give a DFA that accepts the language accepted by the following NFA. (Be sure to note that, for example, it is possible to reach both q_1 and q_3 from q_0 on consumption of an a, because of the ε-transition.)



Finite-State Automata and Regular Languages

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

- proof idea(s)
  - constructive proof – give an algorithm for building an NFA for a given regular expression
  - definition of regular expression is recursive, so utilize proof by induction
- proof sketch
  - simplest regular expressions are Φ, ε, *a*
    - build an NFA for each of these
  - regular expression operators are |, •, *
    - build an NFA for each of these

Finite-State Automata and Regular Languages

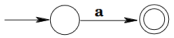
Theorem 3.3. *Every language generated by a regular expression can be recognized by an NFA.*

- simplest regular expressions are Φ, ε, *a*

Consider the regular expression Φ. $L(\Phi) = \{\}$. Here is a machine that accepts $\{\}$:

Consider the regular expression ε. $L(\varepsilon) = \{\varepsilon\}$. Here is a machine that accepts $\{\varepsilon\}$:

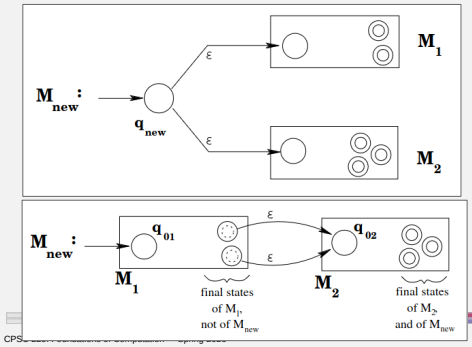
Consider the regular expression a. $L(a) = \{a\}$. Here is a machine that accepts $\{a\}$:



Finite-State Automata and Regular Languages

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

- regular expression operators are |, •, *



Finite-State Automata and Regular Languages

Complete the proof of Theorem 3.3 by showing how to modify a machine that accepts $L(r)$ into a machine that accepts $L(r^*)$.

Using the construction described in Theorem 3.3, build an NFA that accepts $L((ab \mid a)^*(bb))$.

Show that for any DFA or NFA, there is an NFA with exactly one final state that accepts the same language.

Finite-State Automata and Regular Languages

Theorem 3.4. *Every language that is accepted by a DFA or an NFA is generated by a regular expression.*

- a strategy (not a proof or algorithm)
  - if the DFA/NFA has more than one final state, build an equivalent NFA with a single final state
  - repeatedly replace sequences of transitions with single transitions labeled with the corresponding regular expressions until only start and final states remain
    - replace self loops
    - replace simple loops with self-loops, dropping no longer needed transitions and states
      - "simple loop" is a cycle where the middle state(s) are not final and have only a single transition in and out (excluding self loops)
    - replace simple and length 2 paths with transitions, dropping no longer needed transitions and states
      - "simple path" is a path where the middle state(s) are not final and have only a single transition in and out (excluding self loops)
    - combine parallel transitions
  - read the result