Algorithmic DFA Minimization

Based on slides of Aaron Deever

Computing the Minimal Finite Automaton

- Equivalent States
  - $M = (Q, \Sigma, \delta^*, q_0, F)$
  - Two states, $p, q \in Q$ are said to be <u>indistinguishable</u> if
    - For all strings $x \in \Sigma^*$
      - If $\delta^*(p, x)$ is an accepting state then $\delta^*(q, x)$ is an accepting state
      - If $\delta^*(p, x)$ is not an accepting state then $\delta^*(q, x)$ is not an accepting state

Computing the Minimal Finite Automaton

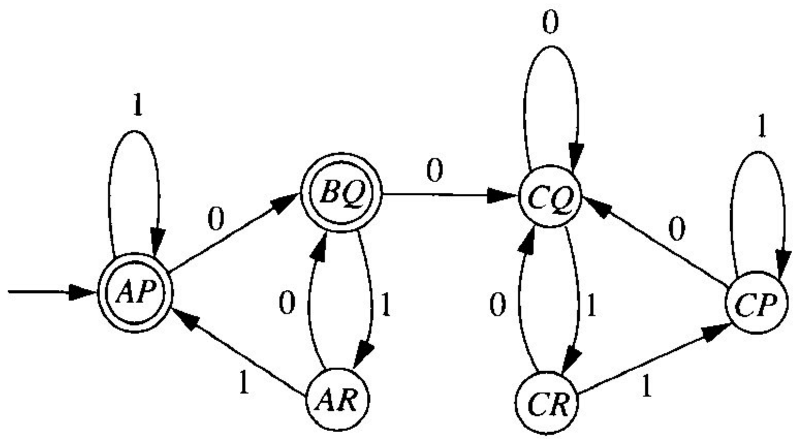
- Equivalent States
  - $M = (Q, \Sigma, \delta^*, q_0, F)$

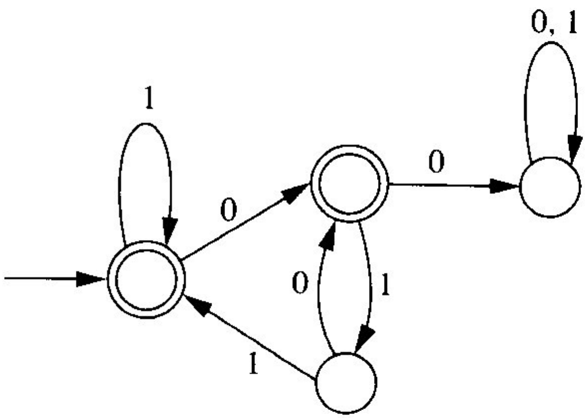
- If two states are not <u>indistinguishable</u>, they are said to be <u>distinguishable</u>.
- There is a string z such that
  - δ^* (p, z) is an accepting state and δ^* (q, z) is a non-accepting state OR
  - δ^* (p, z) is a non-accepting state and δ^* (q, z) is an accepting state

Computing the Minimal Finite Automaton

- Equivalent States
  - In building a minimal DFA, indistinguishable states can be combined.

Minimal Finite Automaton



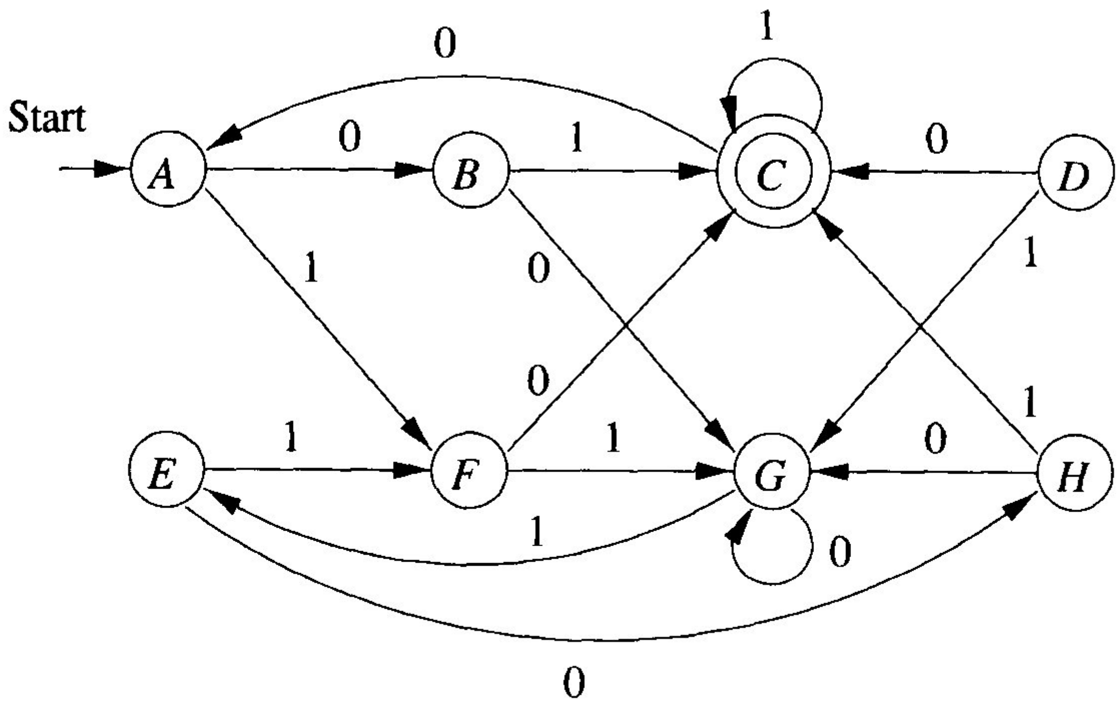


Original DFA

Minimal DFA

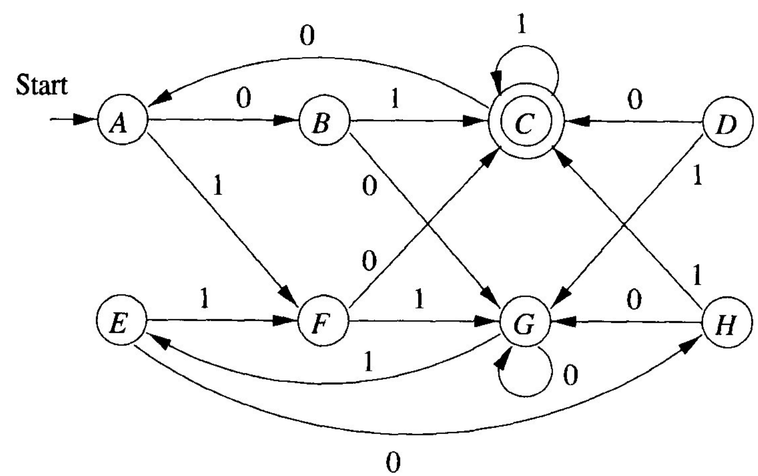
Minimal Finite Automaton

- Example



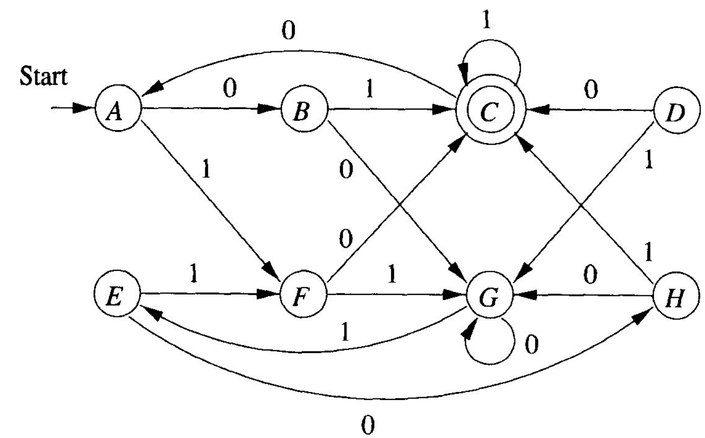
Minimal Finite Automaton

- Example:
  - States C and G are distinguishable
    - One is accepting, one is not
  - States A and G are distinguishable
    - $\delta^*(A, 01) = C$ (accepting)
    - $\delta^*(G, 01) = E$ (non-accepting)



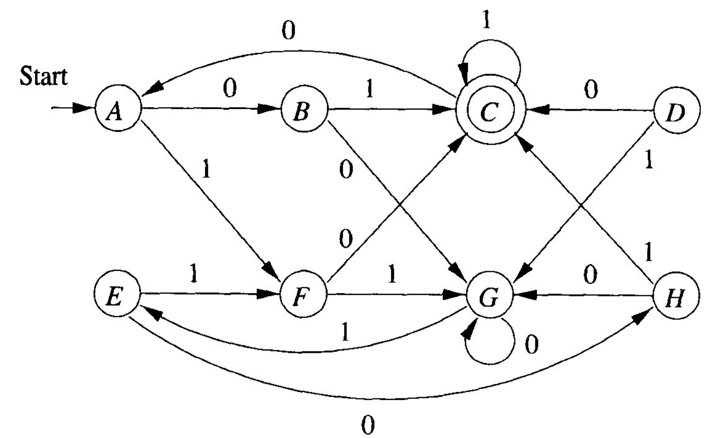
Minimal Finite Automaton

- Example:
  - States B and H are equivalent
    - B and H both non-accepting
    - δ^* (B, 1) = δ^* (H, 1) = C
      - δ^*(B, 1x) = δ^* (H, 1x) for any x
    - δ^* (B, 0) = δ^* (H, 0) = G
      - δ^* (B, 0x) = δ^* (E, 0x) for any x
    - So for any x, δ^* (B, x) and δ^* (H, x) will either both be accepting or both be non-accepting.



Minimal Finite Automaton

- Example:
  - States A and E are equivalent
    - A and E both non-accepting
    - δ^* (A, 1) = δ^* (E, 1) = F
      - δ^* (A, 1x) = δ^* (E, 1x) for any x
    - δ^* (A, 0) = B, δ^* (E, 0) = H
      - B and H are equivalent
      - δ^* (A, 0x) and δ^* (E, 0x) will either both be accepting or both be non-accepting.



Minimal Finite Automaton

- Algorithm to find distinguishable states:
  - Consider pairs {p,q}
  - For each pair we will determine whether p is distinguishable from q
  - Said another way, for each pair {p,q} we will determine if p is not equivalent to q.

Minimal Finite Automaton

- Iterative algorithm
  - Initialization:
    - If p is accepting and q is non-accepting then {p,q} is distinguishable
  - General Case:
    - For some pair {p,q} if
      - $\delta^*(p,a) = r$ and $\delta^*(q,a) = s$ and
      - {r,s} is distinguishable then
      - {p,q} is distinguishable

Minimal Finite Automaton

- Let's take a look at this general case:
  - If $r = \delta^* (p,a)$ and $s = \delta^* (q,a)$ are distinguishable, then there is a string x such that $\delta^* (r,x)$ is accepting and $\delta^* (s,x)$ is not, or vice-versa
  - Then for x, $\delta^* (p,ax)$ is accepting and $\delta^* (q,ax)$ is not, or vice-versa.
  - We found a string, ax such that $\delta^* (p,ax)$ is accepting and (q,ax) is not (or vice-versa), thus {p,q} are distinguishable

Minimal Finite Automaton

- This algorithm can be visualized by using a table with each table cell representing a pair of states. A mark in a table cell indicates that the two states of the pair are distinguishable.

Minimal Finite Automaton

- Table for determining distinguishable states

B							
C							
D							
E							
F							
G							
H							
	A	B	C	D	E	F	G

Minimal Finite Automaton

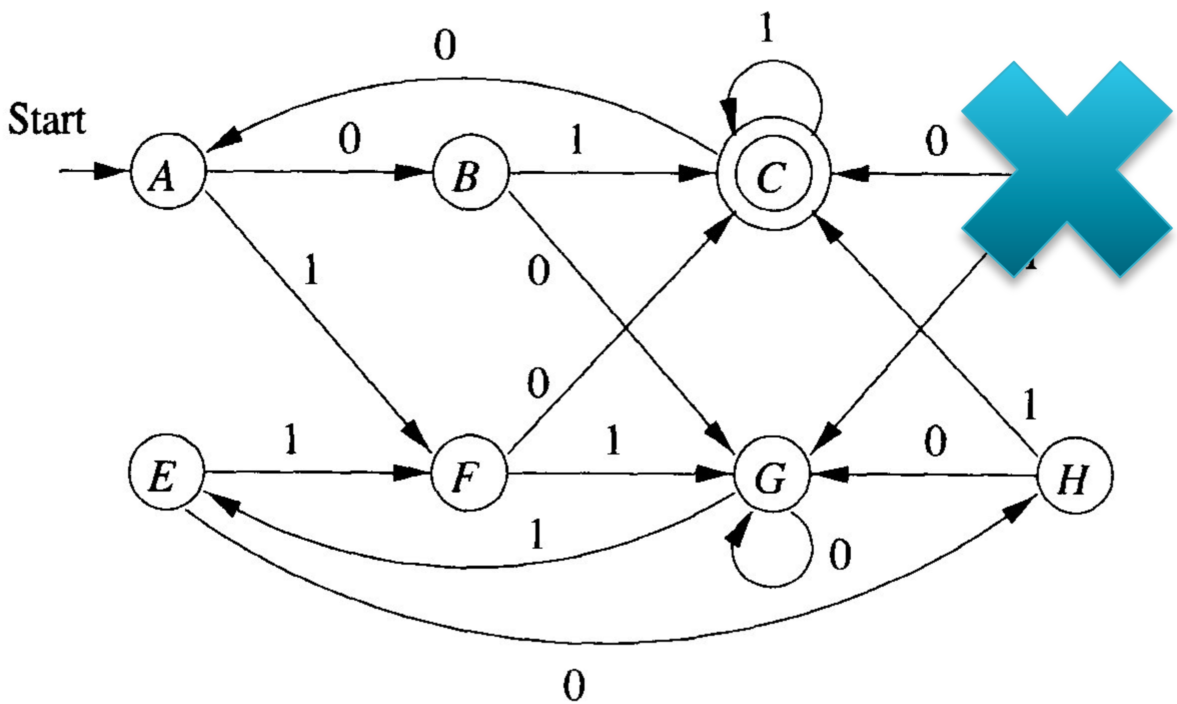
- Restatement of algorithm
  - First remove all states that are unreachable from the start state.

  - For all pairs {p,q} such that p is accepting and q is not, mark the equivalent cell in the table.
  - Consider each pair {p,q} not yet marked.
    - Determine $r = \delta^*(p,a)$ and $s = \delta^*(q,a)$ for each a in Σ.
    - If {r,s} is marked, then mark {p,q}
  - Repeat until no further cells are marked during an entire iteration of the algorithm
    - (one iteration considers all unmarked pairs of states)

Minimal Finite Automaton

- Example

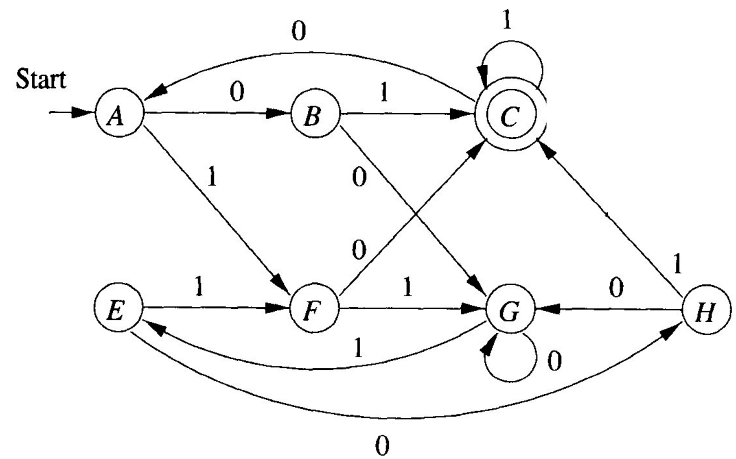
D is unreachable



Minimal Finite Automaton

- Let's try on our example

B	✖					
C	✖	✖				
E		✖	✖			
F	✖	✖	✖	✖		
G	✖	✖	✖	✖	✖	
H	✖		✖	✖	✖	✖
	A	B	C	E	F	G



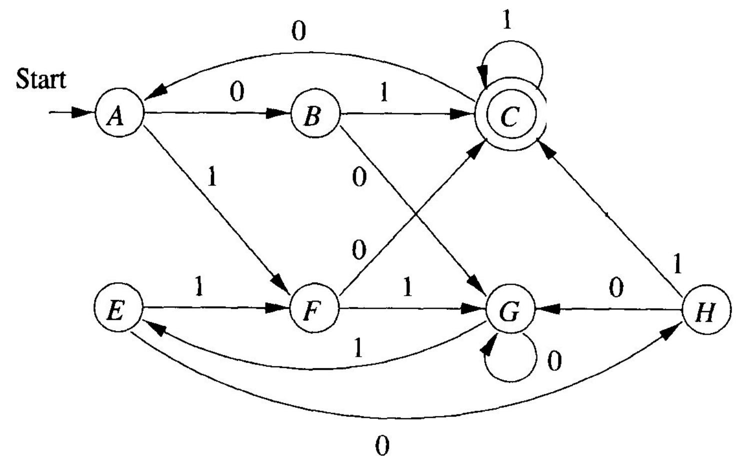
Minimal Finite Automaton

- Once the table is complete
  - All unmarked cells correspond to state pairs that are not distinguishable, i.e. they are equivalent
  - Combine equivalent states into one
  - Transitions from equivalent states should map to equivalent states

Minimal Finite Automaton

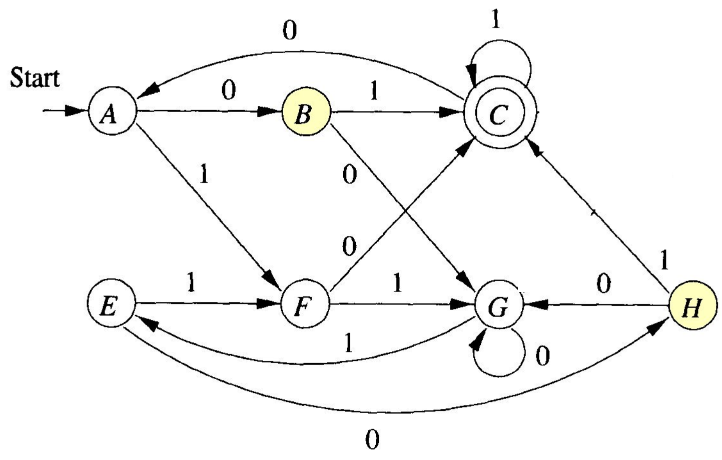
- A and E are equivalent
- B and H are equivalent

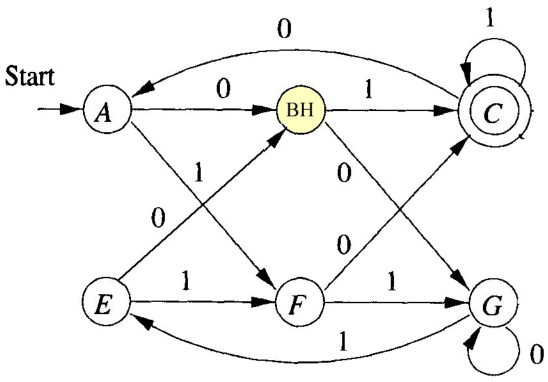
	A	B	C	E	F	G
B	✖					
C	✖	✖				
E		✖	✖			
F	✖	✖	✖	✖		
G	✖	✖	✖	✖	✖	
H	✖		✖	✖	✖	✖



Minimal Finite Automaton

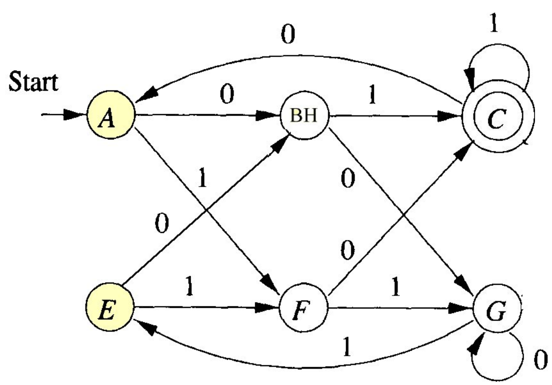
- Combine H and B

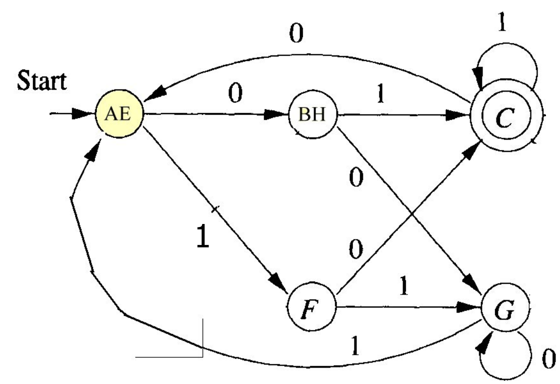




Minimal Finite Automaton

- Combine E and A





Minimal Finite Automaton

- What have we done?
  - Defined the notion of equivalent states
  - Developed an algorithm to determine which states in a DFA are equivalent
  - Combined equivalent states to create a DFA with minimal number of states.

  - Given 2 specifications of regular languages, do the specifications describe the same language?
    - Create a minimal DFA for each language
    - Compare the minimal DFAs on a state by state basis.