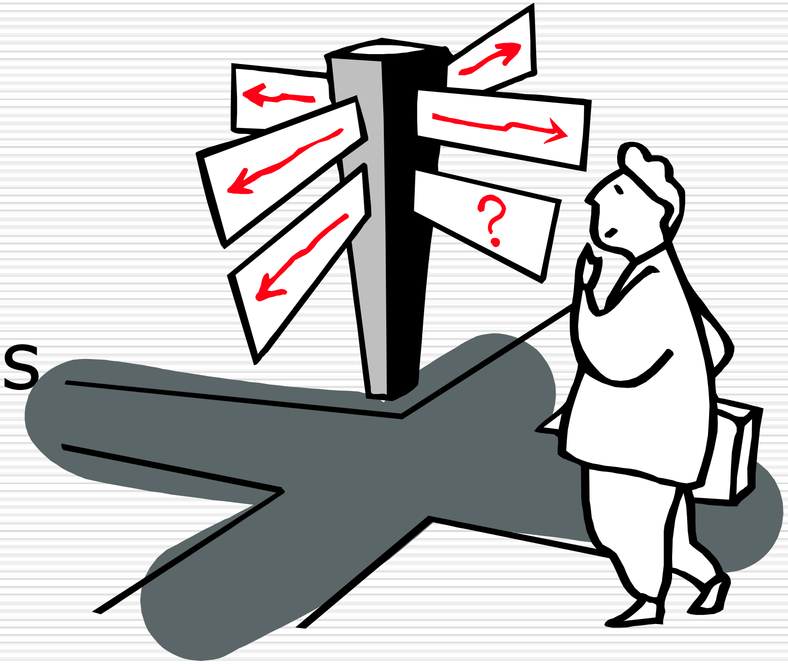
1.3 Properties of Regular Languages

- Pumping Lemma
- Closure properties
- Decision properties
- Minimization of DFAs



Closure Properties

- Certain operations on regular languages are guaranteed to produce regular languages
  - Union: $L \cup M$
  - Intersection: $L \cap M$
  - Complement: $\overline{L}$
  - Difference: $L - M$
  - Reversal: $L^R = \{w^R \mid w \in L\}$
  - Closure: L^*
  - Concatenation: LM
  - Homomorphism:
    $h(L) = \{h(w) \mid w \in L, h \text{ is a homomorphism}\}$
  - Inverse homomorphism:
    $h^{-1}(L) = \{w \mid h(w) \in L, h \text{ is a homomorphism}\}$

Closure under Regular Operators

- $L = L(R_1)$, $M = L(R_2)$, then by definition
  - $L \cup M = L(R_1 + R_2)$
  - $LM = L(R_1 R_2)$
  - $L^* = L(R^*)$

Closure under Complement

- if L is a regular language over Σ, so is $\overline{L} = \Sigma^* - L$

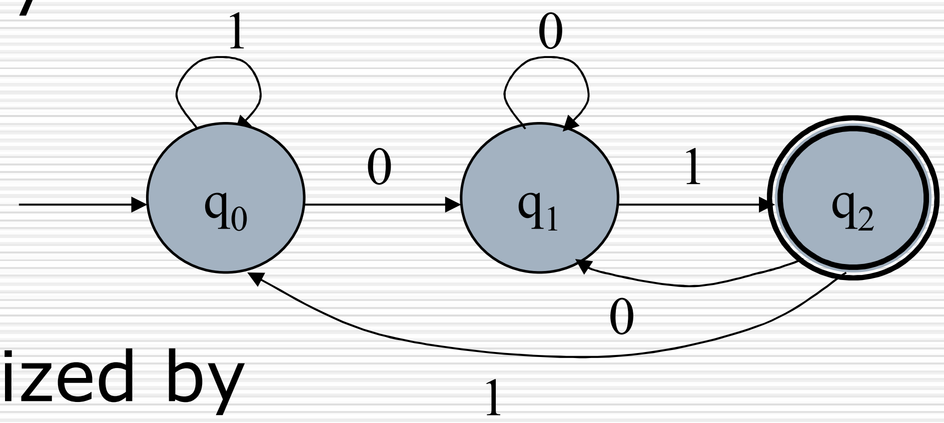
Proof. Let L be recognized by an DFA

$$A = (Q, \Sigma, \delta, q_0, F)$$

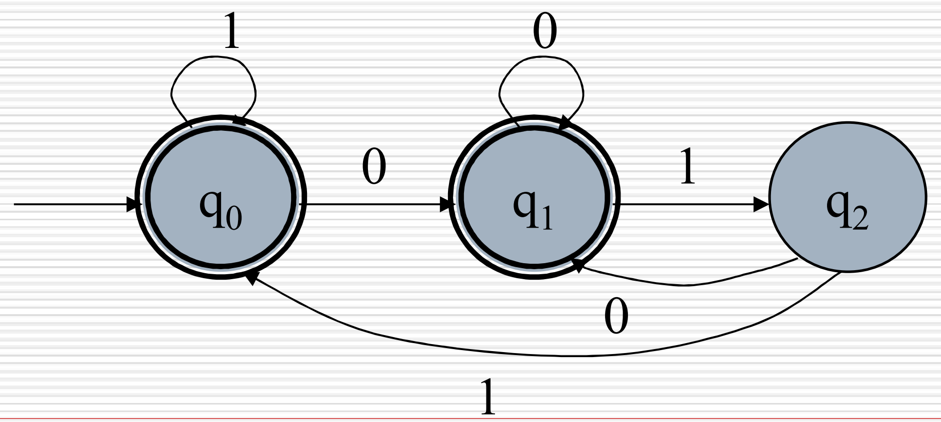
Construct B as $(Q, \Sigma, \delta, q_0, Q-F)$, now $\overline{L} = L(B)$

Example

- L is recognized by



- Then $\overline{L}$ is recognized by



Closure under Intersection

- If L and M are regular languages, so is $L \cap M$

Proof 1. By DeMorgan's Law, $L \cap M = \overline{\overline{L} \cup \overline{M}}$. we already know that regular languages are closed under complement and union.

Closure under Intersection

Proof 2. Let L be recognized by an DFA

$A_L = (Q_L, \Sigma, \delta_L, q_L, F_L)$

And M be recognized by an DFA

$A_M = (Q_M, \Sigma, \delta_M, q_M, F_M)$

We can cross product the two DFAs as:

$A_{L\cap M} = (Q_L x Q_M, \Sigma, \delta_{L\cap M}, (q_L, q_M), F_L x F_M)$

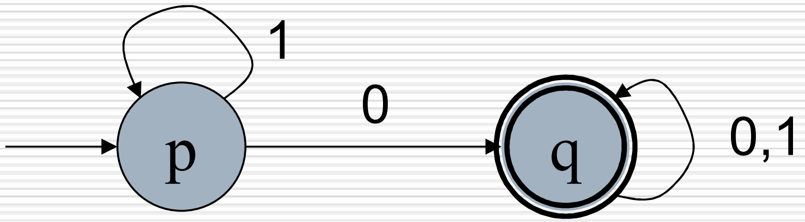
Where

$\delta_{L\cap M}((p, q), a) = (\delta_L(p, a), \delta_M(q, a))$

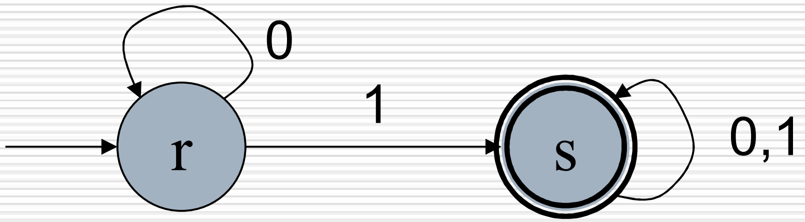
Then $L\cap M$ is recognized by $A_{L\cap M}$.

Example

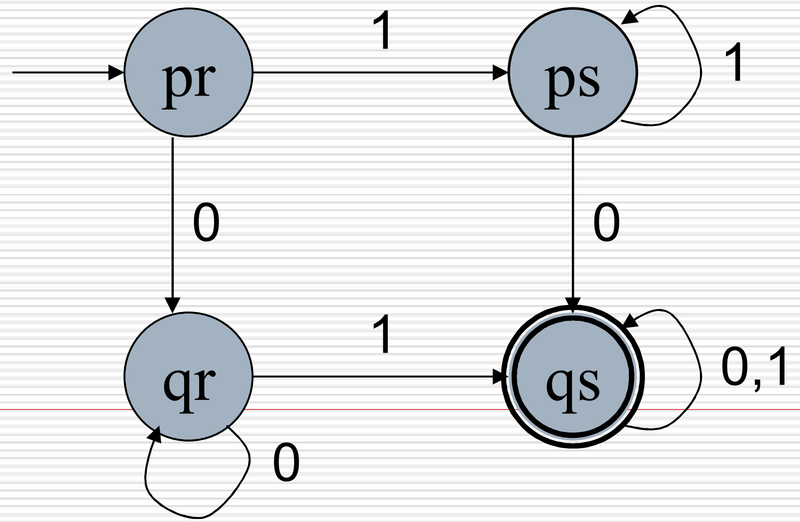
- L:



- M:



- L∩M:



Closure under Difference

- If L and M are regular languages, then so is L - M

Proof. Observe that $L - M = L \cap \overline{M}$. We already know that regular languages are closed under complement and intersection.

Closure under Reversal

- If L is a regular language, so is L^R

Proof 1. Let L be recognized by an FA A, turn A into an FA recognizing L^R, by

- Reversing all arcs
- Making the old start state the new sole accepting state
- Creating a new start state q_0, with $\delta(q_0, \varepsilon)=F$ (the old accepting states)

Closure under Reversal

Proof 2. Let L be described by a regex E.

We shall construct a regex E^R such that $L(E^R) = L^R$.

We proceed by a structural induction on E.

- E is ε, $\varnothing$, a, then $E^R = E$
- $E = F + G$, then $E^R = F^R + G^R$
- $E = FG$, then $E^R = G^R F^R$
- $E = (F)^*$, then $E^R = (F^R)^*$

Homomorphism

- A homomorphism on Σ_1 is a function h: $\Sigma_1^* \rightarrow \Sigma_2^*$, where Σ_1 and Σ_2 are alphabets.
- Let $w = a_1 a_2 \ldots a_n$, then

  $$h(w) = h(a_1)h(a_2)\ldots h(a_n)$$

  and $h(L) = \{h(w) \mid w \in L\}$
- Example: Let h: $\{0,1\}^* \rightarrow \{a,b\}^*$ be defined by $h(0)=ab$, $h(1)=\varepsilon$. Then
  - $h(0011) = abab$
  - $h(L(10^*1)) = L((ab)^*)$

Closure under Homomorphism

- If L is a regular language over Σ, and h is a homomorphism on Σ, then $h(L)$ is regular

Proof. Let L be described by a regex E. We claim that $L(h(E)) = h(L)$

- E is ε, $\emptyset$, then $h(E)=E$, $L(h(E)) = L(E) = h(L(E))$
- E is a, then $L(E)=\{a\}$, $L(h(E)) = \{h(E)\} = \{h(a)\} = h(L(E))$
- $E = F+G$, then $L(h(E)) = L(h(F+G)) = L(h(F)+h(G)) = L(h(F))\cup L(h(G)) = h(L(F))\cup h(L(G)) = h(L(F)\cup L(G)) = h(L(F+G)) = h(L(E))$
- $E = FG$, then $L(h(E)) = L(h(FG)) = L(h(F)h(G)) = L(h(F))L(h(G)) = h(L(F))h(L(G)) = h(L(F)L(G)) = h(L(FG)) = h(L(E))$
- $E = F^*$, then $L(h(E)) = L(h(F^*)) = L(h(F)^*) = L(h(F))^* = h(L(F))^* = h(L(F)^*) = h(L(F^*)) = h(L(E))$

Inverse Homomorphism

- Let h: $\Sigma_1^* \rightarrow \Sigma_2^*$ be a homomorphism, and $L \subseteq \Sigma_2^*$, then define

  $$h^{-1}(L) = \{w \in \Sigma_1^* \mid h(w) \in L\}$$

Example

- Let $h: \{a,b\}^* \rightarrow \{0,1\}^*$ be defined by $h(a)=01$, $h(b)=10$. If $L = L((00+1)^*)$, then $h^{-1}(L) = L((ba)^*)$.

Claim: $h(w) \in L$ if and only if $w=(ba)^n$

Proof. If $w=(ba)^n$, then $h(w)=(1001)^n \in L$;

if $h(w) \in L$, and assume w not in $L((ba)^*)$, then four possible cases for w.

- w begins with a. Then h(w) begins with 01, not in L
- w ends with b. Then h(w) ends with 10, not in L
- w = xaay. Then h(w) = u0101v, not in L
- w = xbby. Then h(w) = u1010v, not in L

Closure under Inverse Homom.

- Let h: $\Sigma_1^* \rightarrow \Sigma_2^*$ be a homomorphism, and $L \subseteq \Sigma_2^*$ is regular, then $h^{-1}(L)$ is regular.

Proof. Let L is recognized by an FA

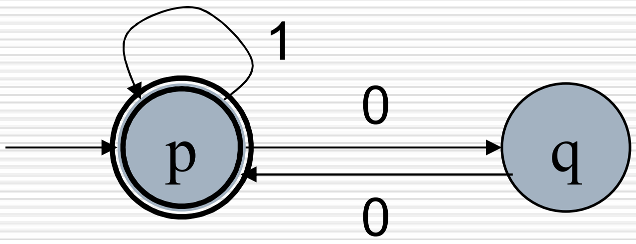
$$A = (Q, \Sigma_2, \delta, q_0, F)$$

We construct an FA $B = (Q, \Sigma_1, \gamma, q_0, F)$, where $\gamma(q, a) = \delta(q, h(a))$.

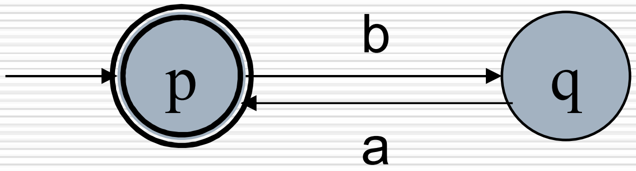
$h^{-1}(L)$ is recognized by B.

Example

- $\Sigma_1 = \{a, b\}, \Sigma_2 = \{0, 1\}$
- $h(a) = 01, h(b) = 10$
- L:



- $h^{-1}(L)$:



1.3 Properties of Regular Languages

- Pumping Lemma
- Closure properties
- Decision properties
- Minimization of DFAs

Decision Properties

- Given a representation (e.g. RE, FA) of a regular language, what can we tell about L?
  - Membership: Is string w in L?
  - Emptiness: Is L = ∅?
  - Finiteness: Is L a finite language?
    - Note that every finite language is regular (why?), but a regular language is not necessarily finite.

Emptiness

- Given an FA for L, L is not empty if and only if at least one final state is reachable from the start state in FA.
- Alternatively, given a regex E for L, we can use the following to test if $L(E) = \emptyset$:
  - $E=F+G$, $L(E)=\emptyset$ if and only if $L(F)$ and $L(G)$ are empty
  - $E=FG$, $L(E)=\emptyset$ if and only if either $L(F)$ or $L(G)$ is empty
  - $E=F^*$, $L(E)$ is not empty

Finiteness

- Given a DFA for L, eliminate all states that are not reachable from the start state and all states that do not reach an accepting state.
- Test if there are any cycles in the remaining DFA; if so, L is infinite, if not, then L is finite.

Equivalence of States

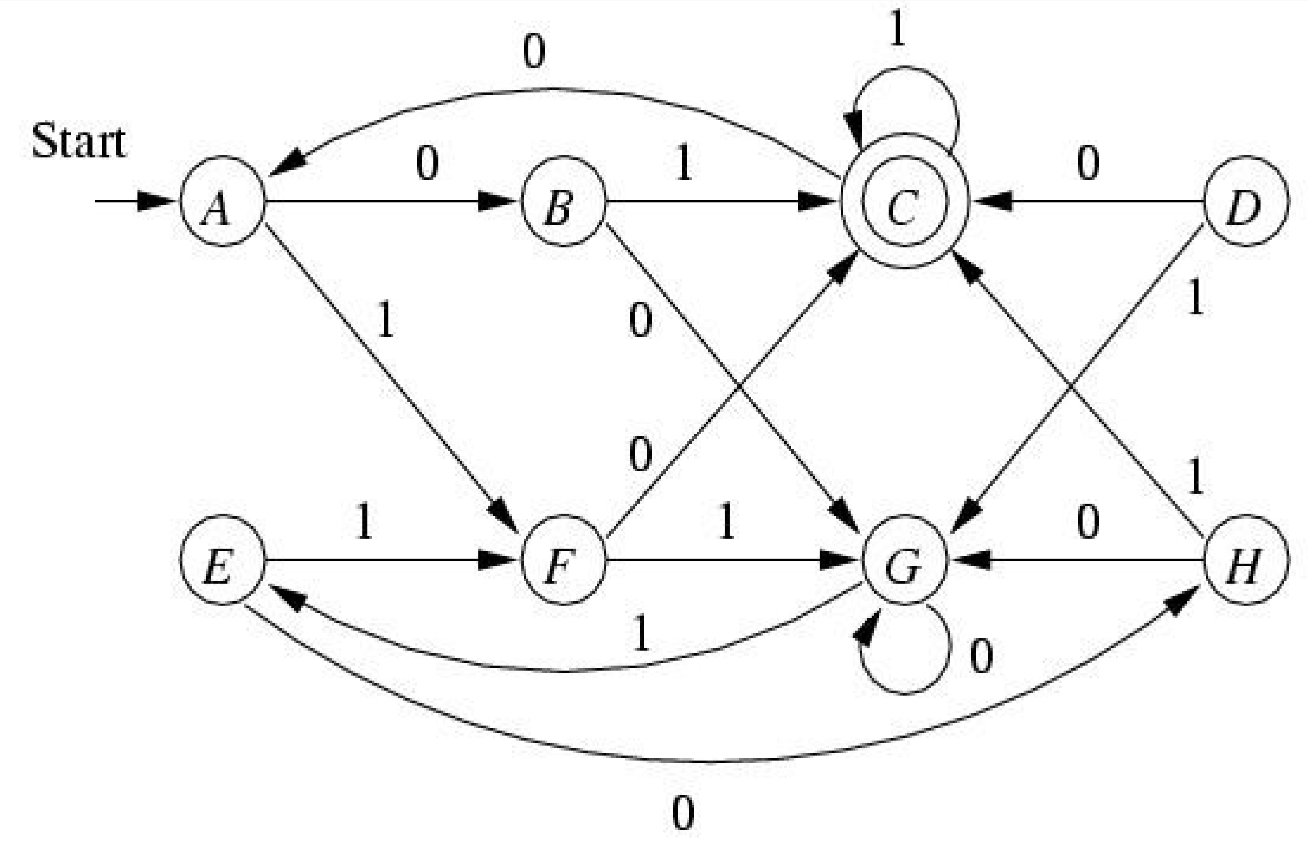
Let $A = (Q, \Sigma, \delta, q_0, F)$, and $p, q \in Q$. We define

- $p \equiv q$ (p and q are equivalent) $\Leftrightarrow$ $\forall w \in \Sigma^*$, $\delta^*(p,w) \in F$ iff $\delta^*(q,w) \in F$
- Otherwise, p and q are distinguishable $\Leftrightarrow$ $\exists w \in \Sigma^*$, $\delta^*(p,w) \in F$ and $\delta^*(q,w) \notin F$, or vice versa
- "$\equiv$" is an equivalence relation

Compute Equivalence of States

- Initially, all pairs of states are in relation "≡"; remove the pairs of distinguishable states inductively as the following
  - Basis: any non-accepting state is distinguishable from any accepting state. ($w=\varepsilon$)
  - Induction: p and q are distinguishable if there is some input symbol a such that $\delta(p, a)$ is distinguishable from $\delta(q, a)$.

Example



A ≡ E

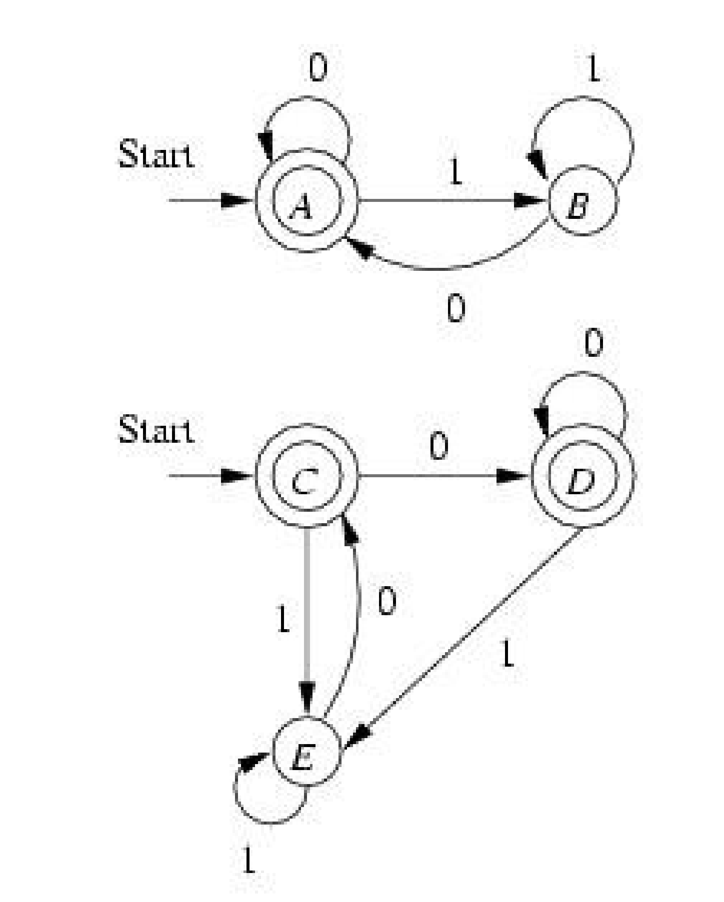
B ≡ H

D ≡ F

Equivalence of Reg. Languages

- Let L and M be two regular languages, to test if L = M?
  - Convert L and M to DFA representations
  - Compute the equivalence relation "≡" on all the states of the two DFAs together.
  - If the two start states are equivalent, then L = M, else L ≠ M.

Example



A ≡ C

B ≡ E

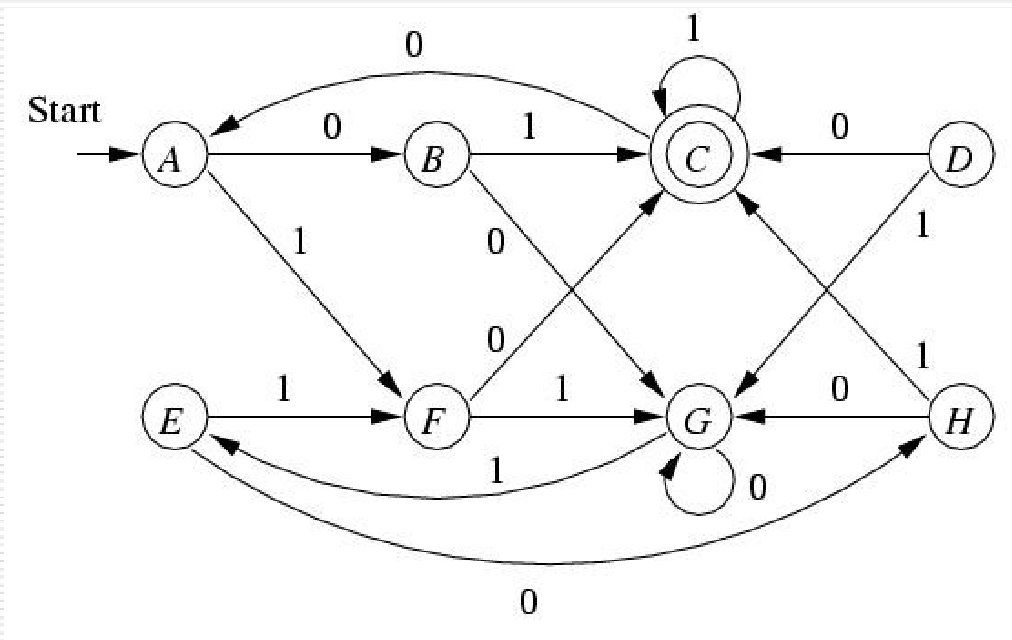
D ≡ C

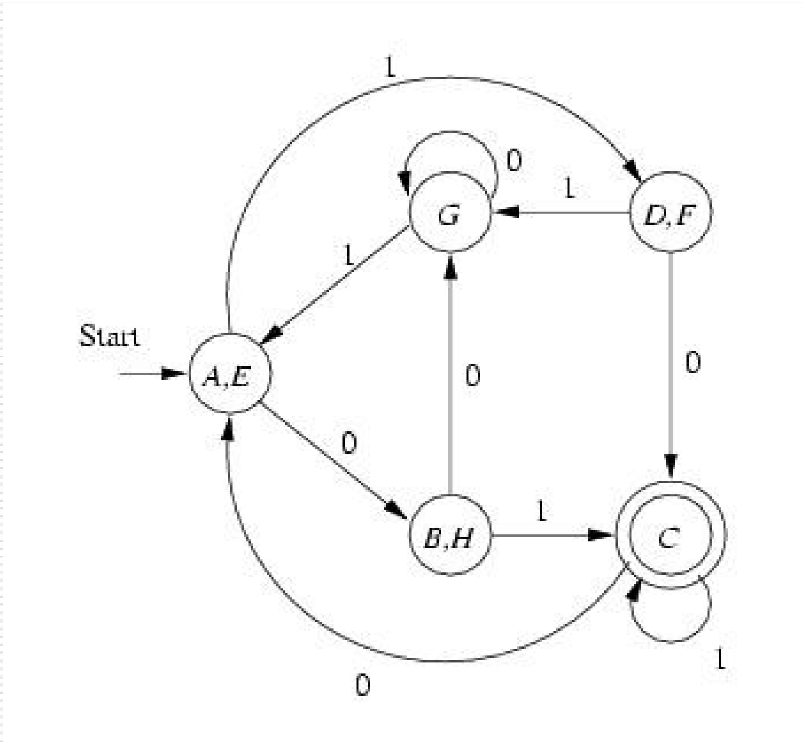
Minimization of DFAs

- Equivalence relation "≡" partitions the states into groups, where all the states in one group are equivalent.
- We can minimize the DFA by merging all equivalent states into one state, and merging the transitions between the states into the transitions between groups.

Example

- ({A, E}, {B, H}, {C}, {D, F}, {G})





Why the Minimization Can't Be Beaten?

- Suppose we have a DFA A, and we minimize it to construct a DFA M. Yet there is another DFA N with fewer states than M and L(N)=L(M)=L(A). Proof by contradiction that this can't happen:
  - Compute the equivalence relation "≡" on the states of M and N together.
  - Start states of M and N are equivalent because L(M)=L(N).
  - If p, q are equivalent, then their successors on any one input symbol are also equivalent. Since neither M not N could have an inaccessible state, every state of M is equivalent to at least one state of N.

Why the Minimization Can't Be Beaten? (Cont'd)

- Since N has fewer states than M, there are two states of M that are equivalent to the same state of N, and therefore equivalent to each other.
- But M was designed so that all its states are distinguishable from each other.
- We have a contradiction, so the assumption that N exists is wrong.
- In fact (stronger), there must be a 1-1 correspondence between the states of any other minimum-state N and the DFA M, showing that the minimum-state DFA for A is **unique** up to renaming of the states.

Summary of Chap. 1

- Finite Automata perform simple computations that read the input from left to right and employ a finite memory.
- The languages recognized by FA are the regular languages.
- Nondeterministic Finite Automata may have several choices at each step.
- NFAs recognize exactly the same languages that FAs do.

Summary of Chap. 1

- Regular expressions are languages built up from the operations union, concatenation, and star.
- Regular expressions describe exactly the same languages that FAs (and NFAs) recognize.
- Some languages are not regular. This can be proved using the Pumping Lemma.

Summary of Chap. 1

- The regular languages are closed under union, concatenation, star, and some other operations.
- Any FA has a unique minimum-state equivalent DFA.