Equivalence and DFA Minimization

Homework

- Homework #1 returned
- Homework #2 Due today
- Homework #3
  - Exercise 4.1.1b pg 129
  - Exercise 4.1.1d pg 129
  - Exercise 4.3.2 pg 153
  - Exercise 4.3.4 pg 154
  - Exercise 4.4.2 (a,b) pg 164

Homework & Exams

- Our first exam is next Monday 10/6
- Homework #3 is due
  - Next Wednesday 10/8
  - Problem session this Wednesday 10/1
    - Please come with questions!!!

Before We Start

- Any questions?

Plan for today

- Minimization of DFAs

Languages

- Recall.
  - What is a language?
  - What is a class of languages?

Regular Languages

- What we know about regular languages
  - Described using regular expressions
    - Set operations of union, concatenation, Kleene Star
  - Kleene Theorem
    - A language is regular iff there exists a finite automata that accepts the language

Minimal Finite Automata

- Motivation
  - Consider the question:
    - Do two finite automata accept the same language?
  - To answer, we introduce the Minimal Finite Automata (MFA)
    - Given a DFA, create a new DFA with the minimal number of states possible that accepts the same language.

Minimal Finite Automata

- Motivation
  - Consider the question:
    - Do two finite automata accept the same language?
  - Answer
    - We can generate the MFA for each DFA, then compare the MFAs on a state by state basis.

Minimal Finite Automata

- Plan
  - Equivalent states of a DFA
  - Devise an algorithm (based on equivalent states) that creates a minimal DFA from an DFA
  - Some examples

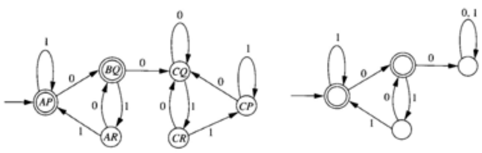
Minimal Finite Automata

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

Minimal Finite Automata

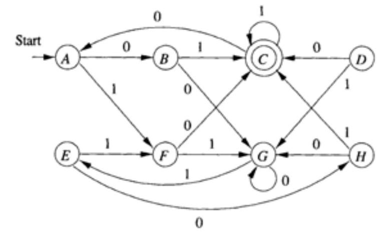
- Equivalent States
  - In building a MFA, equivalent states can be combined.

Minimal Finite Automata



Minimal Finite Automata

- Example



Minimal Finite Automata

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

Minimal Finite Automata

- Example:
  - States B and H are equivalent
    - δ (B, 1) = δ (H, 1) = C
      - $\hat{\delta}$ (B, 1x) = $\hat{\delta}$ (H, 1x) for any x
    - δ (B, 0) = δ (H, 0) = G
      - $\hat{\delta}$ (B, 0x) = $\hat{\delta}$ (E, 0x) for any x
    - So for any x, $\hat{\delta}$ (B, x) and $\hat{\delta}$ (H, x) will either both be accepting or both not be accepted.

Minimal Finite Automata

- Example:
  - States A and E are equivalent
    - δ (A, 1) = δ (E, 1) = F
      - $\hat{\delta}$ (A, 1x) = $\hat{\delta}$ (E, 1x) for any x
    - δ (A, 0) = B, δ (E, 0) = H
      - B and H are equivalent
      - $\hat{\delta}$ (A, 0x) and $\hat{\delta}$ (E, 0x) will either both be accepting or both be non-accepting.

Minimal Finite Automata

- Recursive 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 Automata

- Recursive algorithm
  - Base case:
    - If p is accepting and q is non-accepting then {p,q} is distinguishable
  - Induction
    - For some pair {p,q} if
      - δ (p,a) = r and δ (q,a) = s and
      - {r,s} is distinguishable then
      - {p,q} is distinguishable

Minimal Finite Automata

- Let's take a look at this induction step
  - If r = δ (p,a) and s = δ (q,a) are distinguishable, then there is a string x such that δ (r,x) is accepting and δ(s,x) is not, or visa-versa
  - Then for x, δ (p,ax) is accepting and δ (q,ax) is not, or visa-versa.
  - We found a string, ax such that δ (p,ax) is accepting and (q,ax) is not (or visa-versa), thus {p,q} are distinguishable

Minimal Finite Automata

- This algorithm is sometime best 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 Automata

- Distinguishable table

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

Minimal Finite Automata

- Restatement of algorithm
  - 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 = δ (p,a) and s = δ (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 iteration of the algorithm

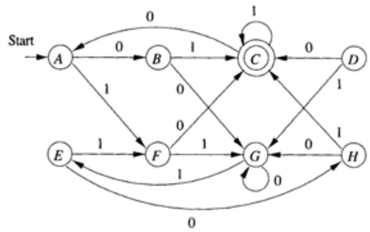
Minimal Finite Automata

- Example

δ (A, 0) = B	δ (A, 1) = F
δ (B, 0) = G	δ (B, 1) = **C**
δ (C, 0) = A	δ (C, 1) = **C**
δ (D, 0) = **C**	δ (D, 1) = G
δ (E, 0) = H	δ (E, 1) = F
δ (F, 0) = **C**	δ(F, 1) = G
δ (G, 0) = G	δ(G, 1) = E
δ (H, 0) = G	δ(H, 1) = **C**

Minimal Finite Automata

- Example



Minimal Finite Automata

- Let's try on our example

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

Minimal Finite Automata

- Once our 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 Automata

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

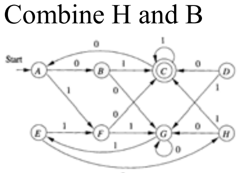
E and A are equivalent

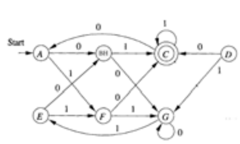
H and B are equivalent

D and F are equivalent

Minimal Finite Automata

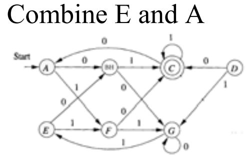
- Combine H and B

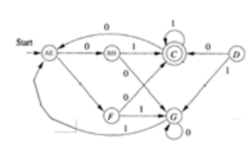




Minimal Finite Automata

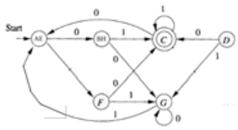
- Combine E and A

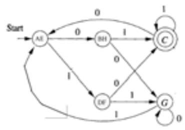




Minimal Finite Automata

- Combine D and F





Minimal Finite Automata

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

Minimal Finite Automata

- Let's revisit the question:
  - Given 2 specifications of regular languages, do the specifications describe the same language.
    - Create a MFA for each language
    - Compare the MFAs on a state by state basis.

For the mathematically minded

- Let's go back to our Discrete Math
  - Relation
    - Defines relationship between objects
    - Usually given as an ordered pair,
      - (x, y) where $x,y \in$ some Set
  - Equivalence relation
    - Reflective: (a, a)
    - Symmetric: if (a,b) then (b,a)
    - Transitive: if (a,b) and (b,c) then (a,c)

For the mathematically minded

- Equivalence relations
  - The nice thing about equivalence relations
    - It partitions the elements of your set into a number of distinct and disjoint subsets.
    - Each subset is called an equivalence class

For the mathematically minded

- MFA and Equivalence Classes
  - State equivalence can be shown to be an equivalence relation on a language.
  - This relation partitions the strings of L into a number of equivalence classes.
  - Each equivalence class corresponds to a state in the MFA.

Minimal Finite Automata

- Questions?

- Let’s take a break