CS 581: Introduction to the Theory of Computation

Lecture 1

James Hook
Portland State University
hook@cs.pdx.edu

http://www.cs.pdx.edu/~hook/cs581f10/

Welcome!

Contact Information

- Jim Hook
- Office: FAB 120-05
- Phone: 503 725 5540
- Email: hook@cs.pdx.edu
- Office hours: Tuesdays, 2 - 4pm (no arrivals after 3:30 please) or by appointment
- TA: TBA

Assumptions:

1. Students have been exposed to the concepts of
   1. regular expressions,
   2. context free grammars, and
   3. programming in a general purpose language.
2. They have applied these concepts to solve problems such as lexical analysis, parsing, and code generation.
3. Students are familiar with discrete mathematics, including sets, sequences, induction and elementary graph theory.

Course Objectives

Introduce students to the classic results in theoretical computer science that classify problems according to the machines that can solve them and the resources required by those machines. This includes basic results relating to computable functions, decidability, and complexity theory.

Master basic proof techniques used in these results including induction, diagonalization, and reduction.

Illuminate the relationship between logic and computation.

Collaboration Policy

Unless explicitly instructed otherwise, please hand in solutions that you prepared individually without directly consulting other sources or notes.

Never represent the work of others as your own work.

Collaboration Policy (cont)

You may meet with other students to discuss homework problems, but please discard all notes from these sessions.

- Do not consult notes from discussions with other students or other solutions when preparing your solution.
- Do not provide other students with access to your solution.

Collaboration Policy (cont)

- If you require resources other than the book to solve a problem please identify those resources with proper citations (but, as for collaborations, set the source aside and do not consult it directly when preparing your solution).
- When selecting other resources, give priority to original sources, texts, and lecture notes.
- Do not consult sample solutions specific to the problems assigned.

Collaboration Policy (cont)

- No exam problems are to be discussed until all students have handed in their exams.
- Students are responsible to keep their exam answers to themselves. Allowing a solution to be copied is as serious a breach of academic integrity as copying.

Academic Integrity

- Violations of academic integrity will be taken seriously
- There will be an in-class penalty
- I will invoke the appropriate university mechanism

Exams

- There will be two exams:
  - Mid-term, October 27, 2010, in-class
  - Final,
    - Monday, December 6, 2010,
    - 5:30 - 7:20pm,
    - in-class, comprehensive

Grading

- All grading will be on a curve
- After applying the curve to normalize the scales, grades will be combined as follows:
  - 30% Homework
  - 30% Midterm
  - 40% Final

Grading

- Assigning letter grades
  - First cut
    - Greater than mean plus standard deviation is an A
    - Less than mean minus standard deviation is a C
  - Refinement
    - Look for clusters; if grades differ by an insignificant amount, round to the higher grade
    - Assign A-, B, B- based on clusters
    - B- and C are functionally equivalent; neither can be applied towards graduation
  - Discretion
    - I will use my judgment to make minor adjustments

End of Course Mechanics

How hard is a problem?

- Does “oo” occur in “Hook”?
- Is 17 prime?
- Is there a winning strategy for tic-tac-toe?
- Will foo.c ever dereference a null pointer?

Static (finite) problems are uniformative

- All these have yes or no answers that I can build into a program
  - Does “oo” occur in “Hook”?
  - Is 17 prime?
  - Is there a winning strategy for tic-tac-toe?
  - Will foo.c ever dereference a null pointer?
- How do we generalize these to problems too big to just memorize?

Problems to Languages 1

- Does "oo" occur in "Hook"?
  - { x | "oo" occurs in x }
- Is 17 prime?
  - {n | n is prime}
- Is there a winning strategy for tic-tac-toe?
  - No obvious generalization; problem too specific
- Will foo.c ever dereference a null pointer?
  - {p.c | p.c does not dereference a null pointer}

Problems to Languages:
Solving problem = language membership

- Does "oo" occur in "Hook"?
  - Is "Hook" in the language { x | "oo" occurs in x }
- Is 17 prime?
  - Is 17 in the set {n | n is prime} ?
- Will foo.c ever dereference a null pointer?
  - Is foo.c in the set of C programs that do not dereference a null pointer?

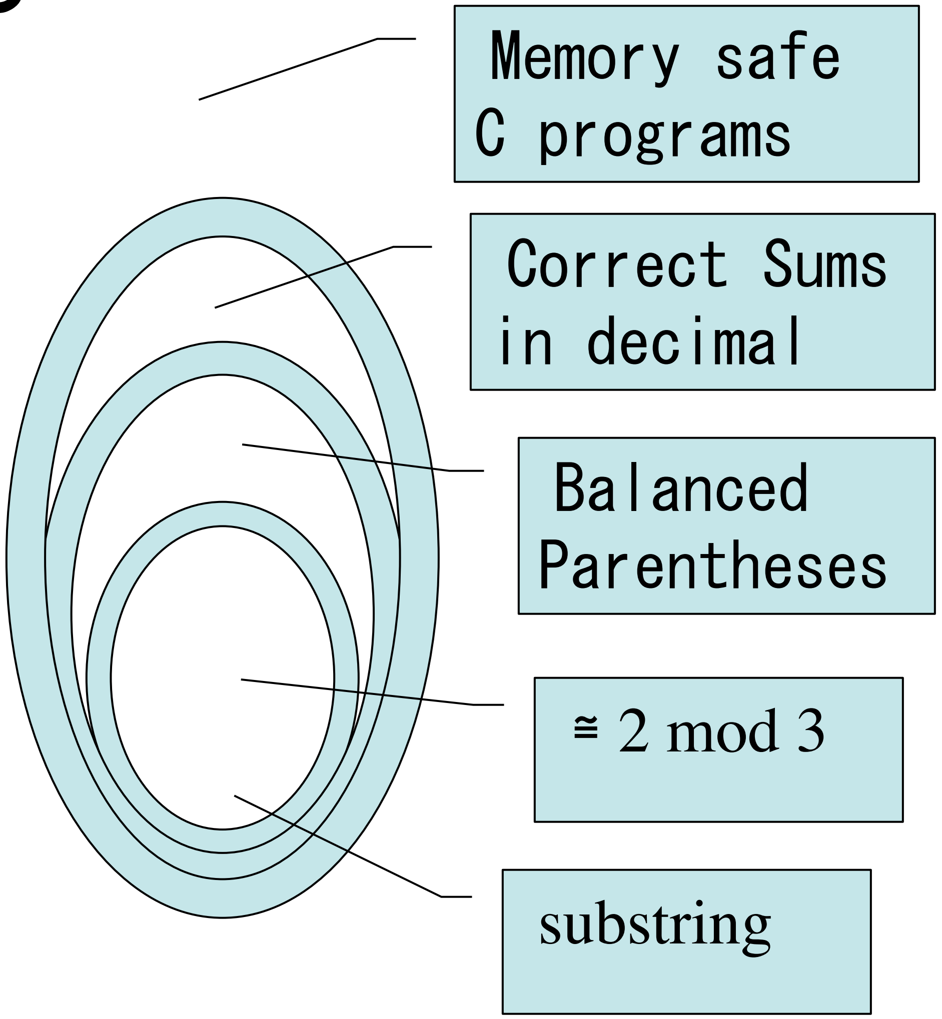
More Languages

- Binary numbers congruent to 2 mod 3
  - {10,101,1000,1011,...}
- Correct sums in decimal
  {<x,y,z> | x+y = z}
  - {<1,1,2>, <2,3,5>, ...}
- Syntactically Correct C programs
- Balanced parenthesis
  - {(), ()(), (()), (()()), ... }

Classifying Problems

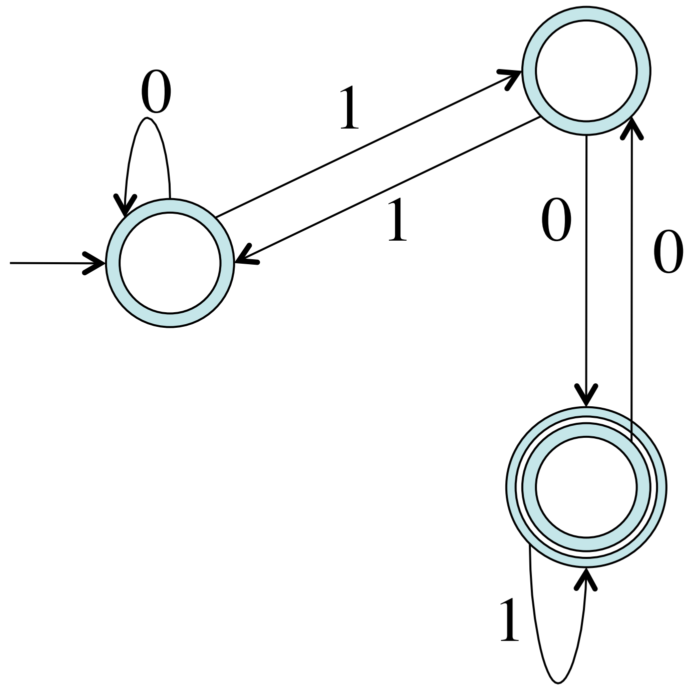
"Map of the world"

- Define classes of problems
  - Machines recognize
  - Grammars generate
- Develop techniques to classify problems by machines



First Model: Finite Automata

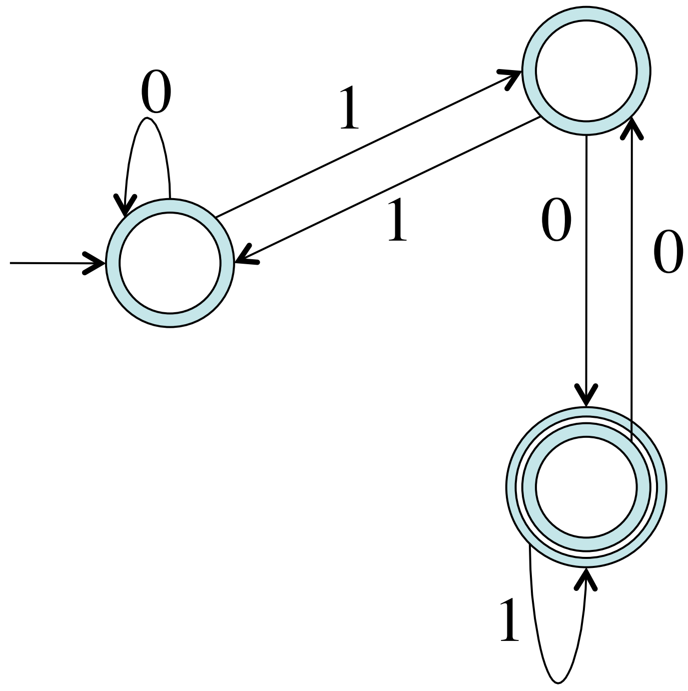
- Cartoon:



- Start State
- Final State
- Transition labels

First Model: Finite Automata

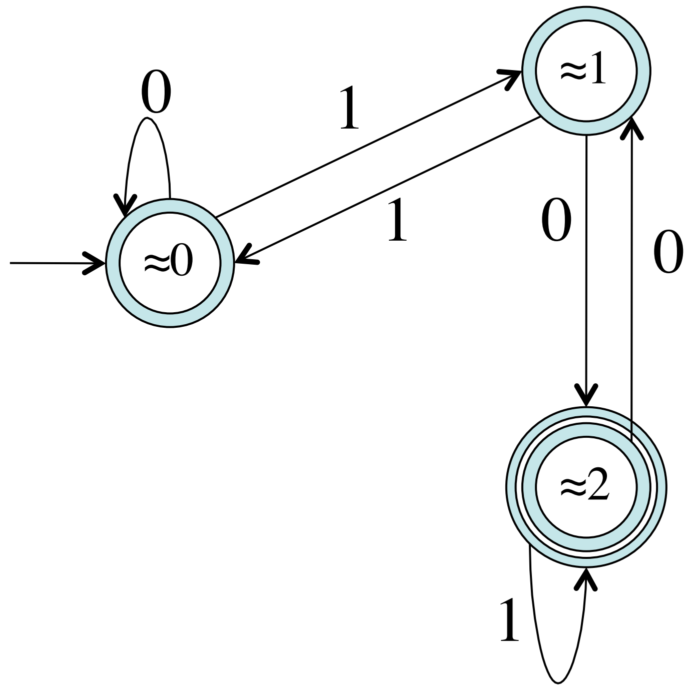
- Cartoon:



- Examples: Which of the following are accepted?
  - 000
  - 10
  - 101
  - 111
  - 100
  - 1011
- What is the language?

First Model: Finite Automata

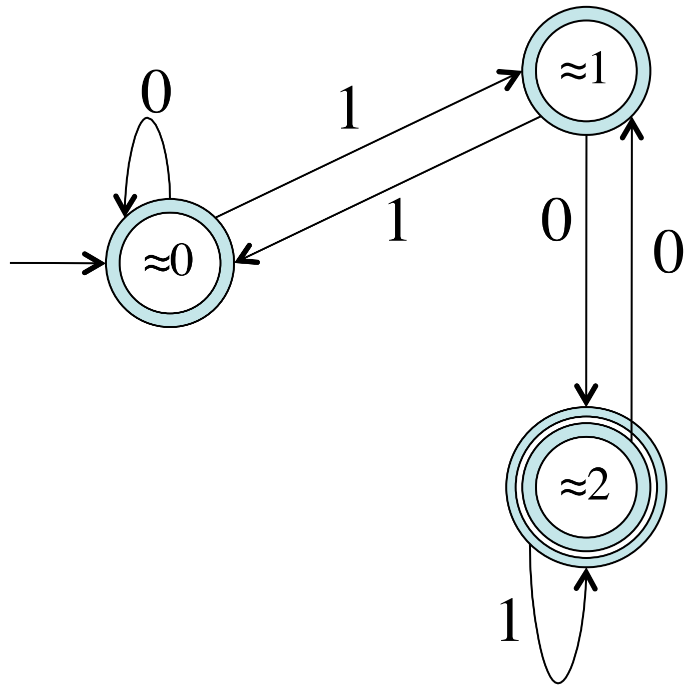
- Cartoon:



•How many states?
•What is the input alphabet

Cartoon to Mathematical Structure

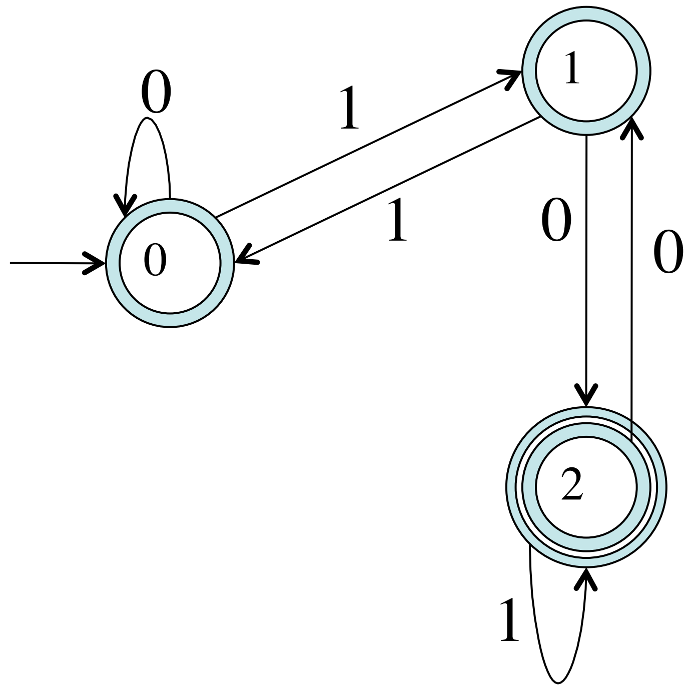
- Cartoon:



A *(deterministic) finite automaton (DFA)* M is a 5-tuple
  M = <Q, Σ, δ, q_0, F>
where
  Q is a finite set (*states*)
  Σ is a finite set (*alphabet*)
  δ: Q×Σ → Q (*transition*)
  q_0 ∈ Q (*initial state*)
  F ⊆ Q (*final states*)

Cartoon to Mathematical Structure

- Cartoon:



$M_{2 \bmod 3} = <Q, \Sigma, \delta, q_0, F>$
where
  $Q = \{0,1,2\}$
  $\Sigma = \{0,1\}$
  $\delta(q,a) = 2*q+a \bmod 3$
  $q_0 = 0$
  $F = \{2\}$

Acceptance

- M *accepts* the string $w = w_1w_2\ldots w_n$ if there exists a sequence of states

  $r_0, r_1, r_2, \ldots, r_n$

  such that

  1. $r_0 = q_0$
  2. $r_{i+1} = \delta(r_i, w_{i+1})$ for all i, $0 <= i < n$
  3. $r_n \in F$

$M_{2 \bmod 3}$ accepts 101

- Calculation:
  - $r_0 = 0$
  - $r_1 = 2*0 + 1 \bmod 3 = 1$
  - $r_2 = 2*1 + 0 \bmod 3 = 2$
  - $r_3 = 2*2 + 1 \bmod 3 = 2$
- Since $r_3 \in F$ (i.e. $2 \in \{2\}$) this satisfies the definition of acceptance for $M_{2 \bmod 3}$

$M_{2 \bmod 3} = \langle Q, \Sigma, \delta, q_0, F\rangle$
where

$Q = \{0,1,2\}$

$\Sigma = \{0,1\}$

$\delta(q,a) = 2*q+a \bmod 3$

$q_0 = 0$

$F = \{2\}$

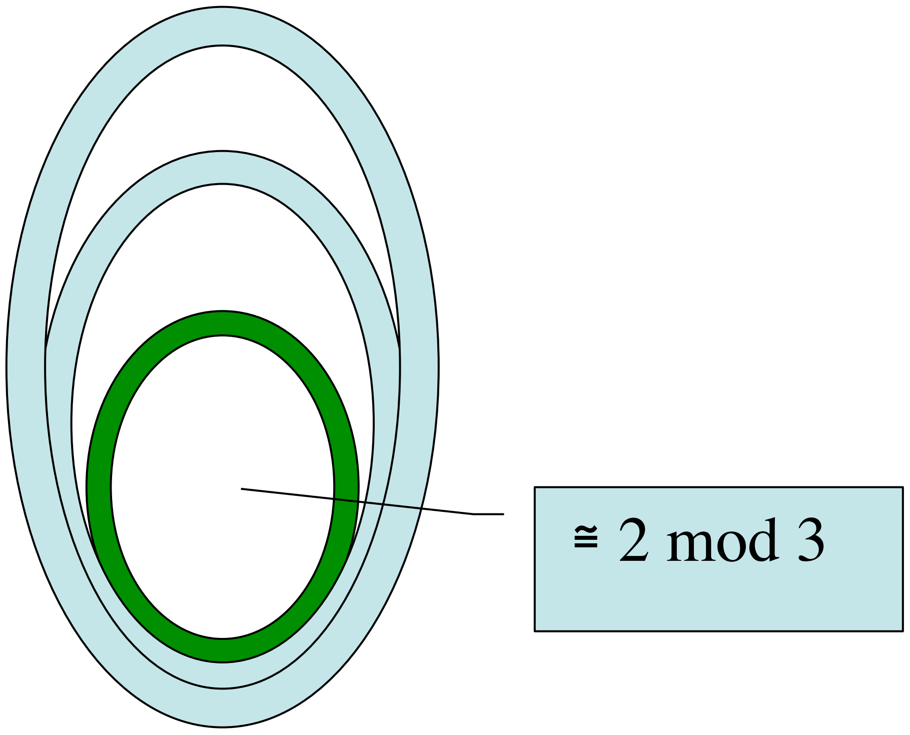
Regular languages

- L(M) = { w | M accepts w }
- A language A is *regular* if A = L(M) for some finite automaton M
- The language
  "binary numbers congruent to 2 mod 3"
  is a regular language because it is $L(M_{2 \bmod 3})$, and $M_{2 \bmod 3}$ is a DFA

Classifying Problems

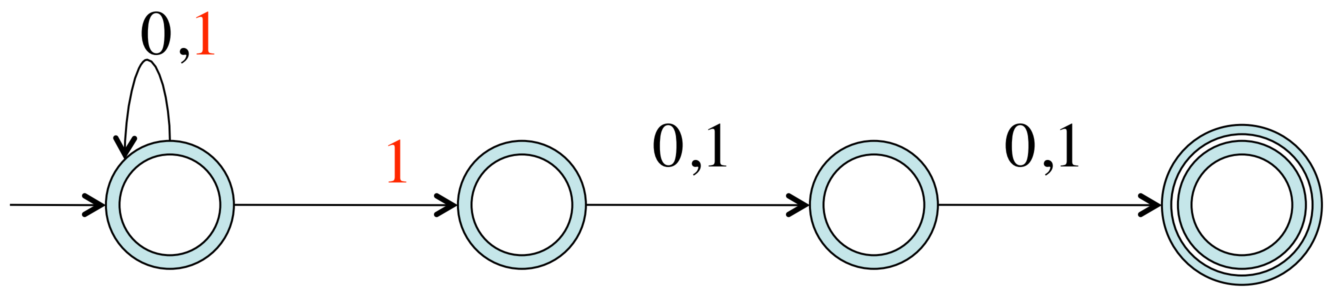
"Map of the world"
  Regular
  Languages

Examples of other
  regular
  languages?



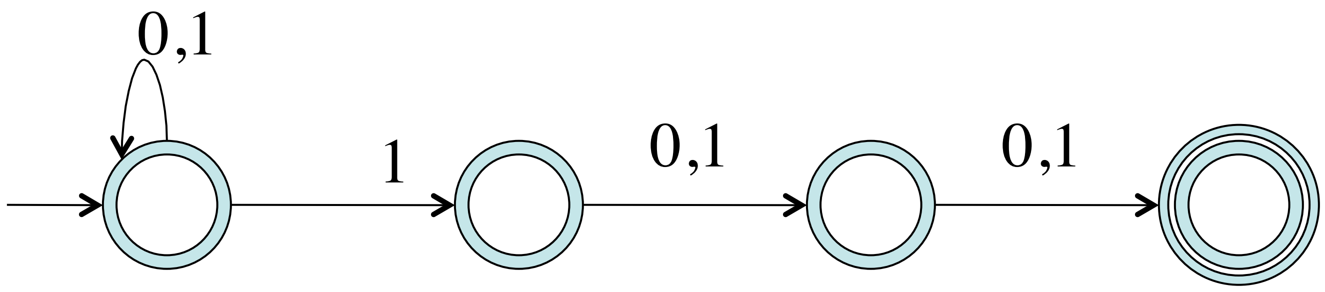
Extending the model

- What if we allow multiple possible transitions on the same symbol?



Extending the model

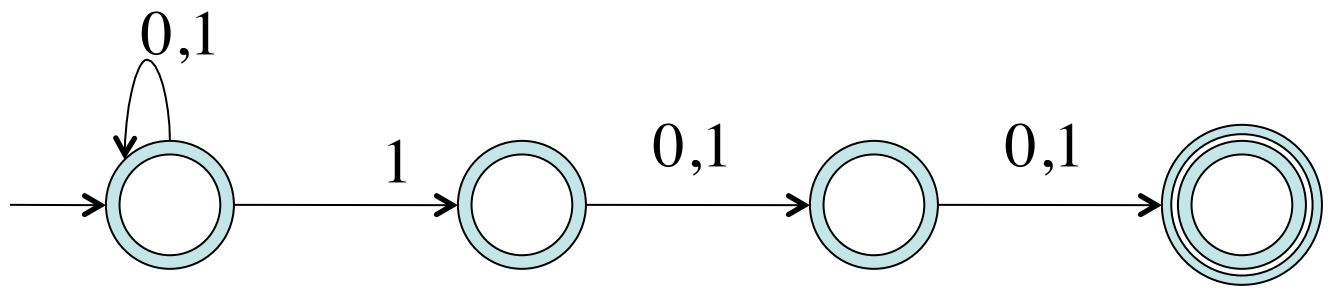
- How should this machine behave?



- Examples:
  - 100
  - 111000
  - 111
  - 0000

Nondeterminism

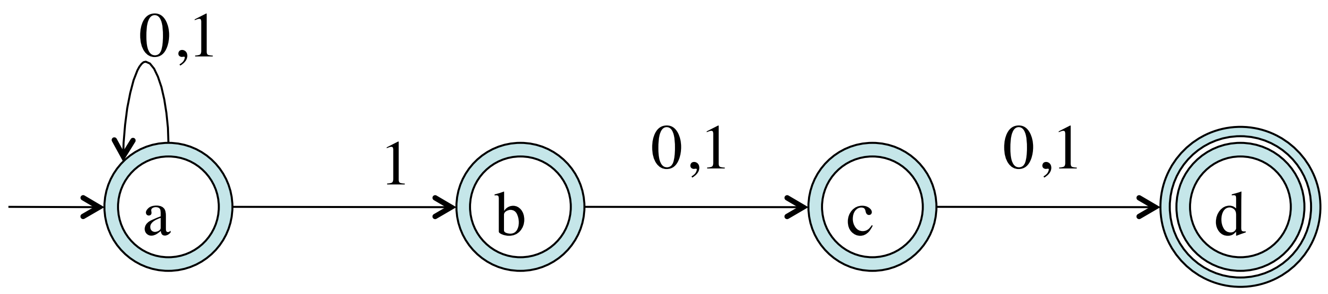
- If we can win we do win



Changing the Model

- A *(~~deterministic~~) nondeterministic finite automaton* (~~*DFA*~~ *NFA*) M is a 5-tuple
- $M = <Q, \Sigma, \delta, q_0, F>$ where
  Q is a finite set (*states*)
- Σ is a finite set (*alphabet*)
- $\delta : Q \times \Sigma \ \to P(Q)$ (*transition*)
- q_0 is in Q (*initial state*)
- F subset of Q (*final states*)

Nondeterminism



M = <Q, Σ, δ, q_0, F>

where

Q = {a,b,c,d}

Σ = {0,1}

q_0 = 0

F = {d}

δ	0	1
a	{a}	{a,b}
b	{c}	{c}
c	{d}	{d}
d	{}	{}

NFA Acceptance

- M *accepts* the string $w = w_1w_2...w_n$ if there exists a sequence of states

  $r_0, r_1, r_2, \ldots , r_n$

  such that

  1. $r_0 = q_0$
  2. $r_{i+1} = \delta(r_i,w_{i+1})$ for all i, $0 <= i < n$
  3. r_n is in F

NFA Acceptance

- M *accepts* the string $w = w_1w_2...w_n$ if there exists a sequence of states

  $r_0, r_1, r_2, \ldots, r_n$

  such that

  1. $r_0 = q_0$
  2. $r_{i+1} \in \delta(r_i, w_{i+1})$ for all i, $0 <= i < n$
  3. r_n is in F

NFA vs. DFA?

- Can NFAs describe more languages than DFAs?

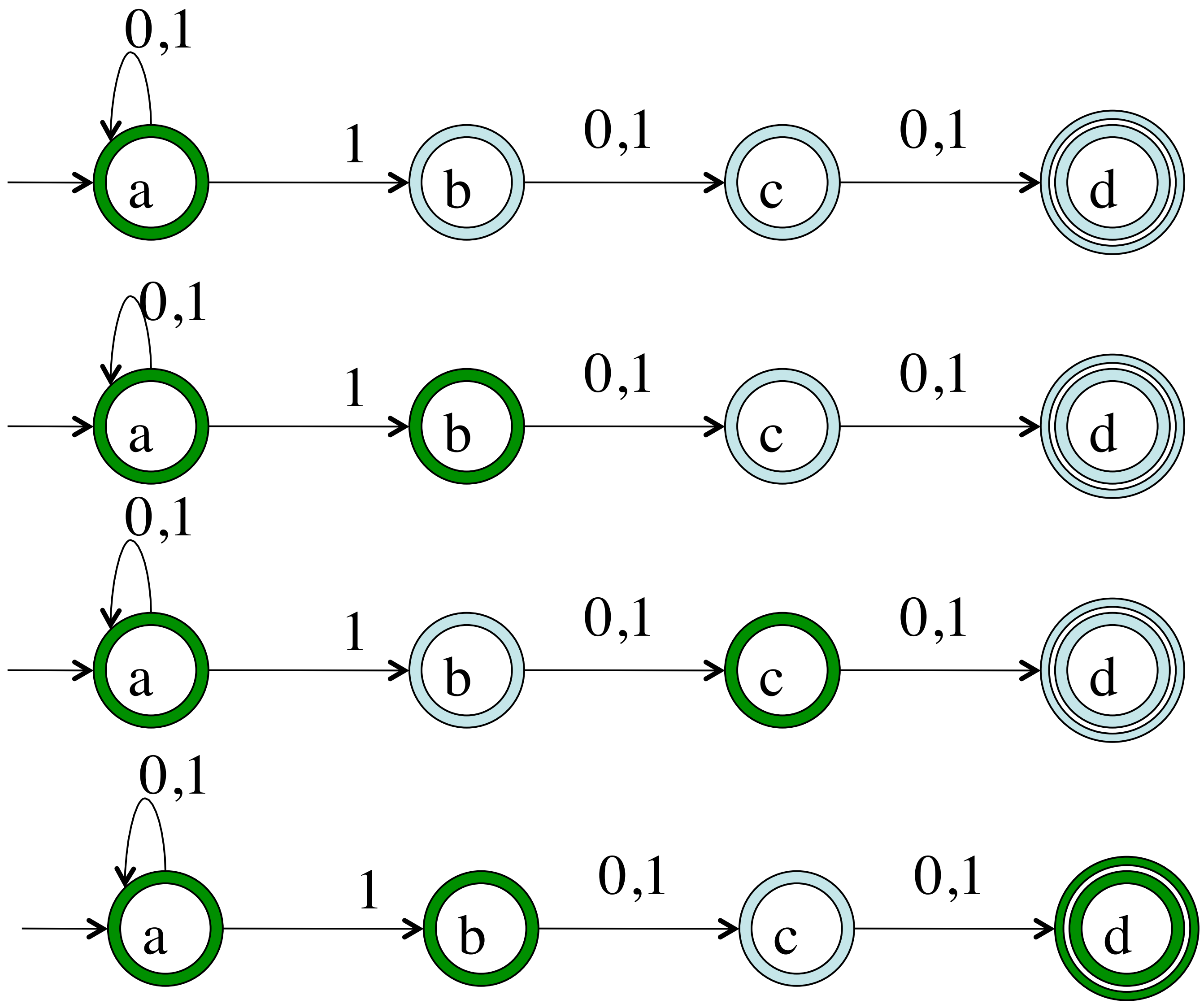
Comparing two models

- To answer the question is X the same as Y we have two strategies
  - No: Show an X that isn't a Y (or v.v.)
    - produce a counter example
  - Yes: Show that:
    - every X is a Y
    - every Y is an X
      - Often we will give constructions (or algorithms) that build an X out of Y (or v.v.)

NFA vs. DFA?

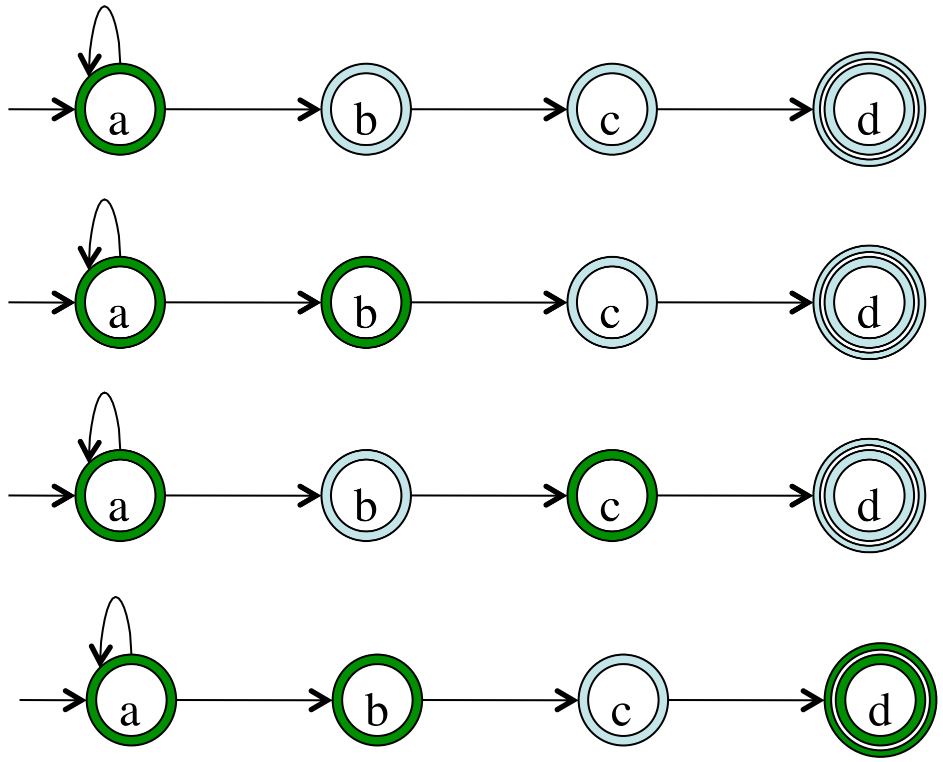
- Can NFAs describe more languages than DFAs?
- Clearly if A = L(M) for a DFA M, then A = L(M) for a similar NFA
  - Why?
- What about the other direction? Can every language recognized by an NFA be recognized by a DFA?

Simulating an NFA on 101



Strategy

- Track the set of states
  - {a}
  - {a,b}
  - {a,c}
  - {a,b,d}



DFA state: set of NFA states

- Since the powerset of a finite set is finite, we can use sets of states of the NFA as states of the DFA that will simulate it
- Each transition will now be deterministic because there is always a well defined set of reachable states
- Which states should be final?

Formalizing

- Claim: If $A = L(M)$ for an NFA M, then $A = L(M')$ for a DFA M'.
- Proof: Given $M = \langle Q, \Sigma, \delta, q_0, F\rangle$, construct $M' = \langle P(Q), \Sigma, \delta', \{q_0\}, F'\rangle$ where

$$\delta'(R,a) = \bigcup_{r\in R} \delta(r,a)$$
$$F' = \{R \mid R \cap F \neq \varnothing\}$$

Are we there yet?

- We have the construction of M′ from M
- We haven′t shown L(M) = L(M′)
- Need
  - $w \in L(M) \Rightarrow w \in L(M')$
  - $w \in L(M') \Rightarrow w \in L(M)$
- Why?

Show $w \in L(M) \Rightarrow w \in L(M')$

- Proof resources?
  - Construction of M′
  - Definition of acceptance for DFAs and NFAs
- Need to show if M can accept w then M′ must accept w
- Is it plausible?

Given $M = \langle Q, \Sigma, \delta, q_0, F \rangle$,
construct $M' = \langle P(Q), \Sigma, \delta', \{q_0\}, F' \rangle$
where $\delta'(R,a) = \bigcup_{r \in R} \delta(r,a)$
$F' = \{R \mid R \cap F \neq \varnothing\}$

M *accepts* the string $w = w_1 w_2 \ldots w_n$ if there exists a sequence of states
$r_0, r_1, r_2, \ldots, r_n$
such that

1. $r_0 = q_0$
2. $r_{i+1} = \delta(r_i, w_{i+1})$ for all i, $0 <= i < n$
3. $r_n \in F$

Show $w \in L(M) \Rightarrow w \in L(M')$

- Basic idea:
  - If M can get from state p to state q reading w,
  - then M′ will move from {p} to a state R containing q on input w

Given $M = \langle Q, \Sigma, \delta, q_0, F \rangle$,
construct $M' = \langle P(Q), \Sigma, \delta', \{q_0\}, F' \rangle$
where $\delta'(R,a) = \bigcup_{r \in R} \delta(r,a)$
$F' = \{R \mid R \cap F \neq \emptyset\}$

M *accepts* the string $w = w_1 w_2 \ldots w_n$ if there exists a sequence of states
$r_0, r_1, r_2, \ldots, r_n$
such that

1. $r_0 = q_0$
2. $r_{i+1} = \delta(r_i, w_{i+1})$ for all i, $0 <= i < n$
3. $r_n \in F$

Notational asside

- "from state p to state q reading w"
- $\hat{\delta}(p, w) = q$
- For DFAs
  - $\hat{\delta}(p, \varepsilon) = p$
  - $\hat{\delta}(p, w_1 w) = \hat{\delta}(\delta(p, w_1), w)$
- For NFAs
  - $\hat{\delta}(p, \varepsilon) = \{p\}$
  - $\hat{\delta}(p, w_1 w) = \cup_{q \in \delta(p, w1)} \hat{\delta}(q, w)$

Show $w \in L(M) \Rightarrow w \in L(M')$

- Basic idea:
  - If $q \in \hat{\delta}(p, w)$
  - then $q \in R$, where $R = \hat{\delta}'(\{p\}, w)$

- Basic idea:
  - If M can get from state p to state q reading w,
  - then M′ will move from {p} to a state R containing q on input w

Show $w \in L(M) \Rightarrow w \in L(M')$

- Claim:
  - If $q \in \hat{\delta}(p, w)$
  - then $q \in R$, where
    $R = \hat{\delta'}(\{p\}, w)$
- Proof: By induction on the length of w
  - Basis: $w = \varepsilon$
    In this case $q = p$, $R=\{p\}$, and the condition holds by definition

Show $w \in L(M) \Rightarrow w \in L(M')$

- Claim:
  - If $q \in \hat{\delta}(p, w)$
  - then $q \in R$, where $R = \hat{\delta}'(\{p\}, w)$
- Step: $w = w_1y$
  Assume the property holds for y to show for w.
  $q \in \hat{\delta}(p, w_1y) = \cup_{r \in \delta(p, w1)} \hat{\delta}(r, y)$
  Hence for some $r' \in \delta(p, w1)$, $q \in \hat{\delta}(r', y)$
  By induction, $q \in R'$, where $R' = \hat{\delta}'(\{r'\}, y)$

Show $w \in L(M) \Rightarrow w \in L(M')$

- Step: $w = w_1 y$
  Assume the property holds for y to show for w.
  $q \in \hat{\delta}(p, w_1 y) = \bigcup_{r \in \delta(p, w1)} \hat{\delta}(r, y)$
  Hence for some $r' \in \delta(p, w1)$, $q \in \hat{\delta}(r', y)$
  By induction, $q \in R'$, where $R' = \hat{\delta}'(\{r'\}, y)$
- Recall we are trying to show
  $q \in R$, where $R = \hat{\delta}'(\{p\}, w_1 y)$
  which is equivalent to
  $q \in R$, where $R = \hat{\delta}'(\delta'(\{p\}, w_1), y)$
  $q \in R$, where $R = \hat{\delta}'(\delta(p, w_1), y)$
- Problem: mismatch between $\{r'\}$ and $\delta(p, w_1)$
- Must generalize claim to apply induction

Show $w \in L(M) \Rightarrow w \in L(M')$

- Claim (generalized):
  - If $q \in \hat{\delta}(p, w)$
  - then $p \in S$ implies $q \in R$, where $R = \hat{\delta}'(S, w)$
- Proof: By induction on the length of w
  - Basis: $w = \varepsilon$
    In this case $q = p$, $R=S$, and the condition holds by reflexivity, $p \in S$ implies $p \in S$

Show $w \in L(M) \Rightarrow w \in L(M')$

- Claim (generalized):
  - If $q \in \delta\hat{}(p, w)$
  - then $p \in S$ implies $q \in R$, where $R = \delta'\hat{}(S, w)$
- Step: $w = w_1 y$
  Assume the property holds for y to show for w.
  $q \in \delta\hat{}(p, w_1 y) = \bigcup_{r \in \delta(p, w1)} \delta\hat{}(r, y)$

- Hence for some $r' \in \delta(p, w1)$, $q \in \delta\hat{}(r', y)$
  By induction, $r' \in S'$ implies $q \in R'$, where
  $R' = \delta'\hat{}(S', y)$

Show $w \in L(M) \Rightarrow w \in L(M')$

- Step: $w = w_1y$
  Assume the property holds for y to show for w.
  $q \in \hat{\delta}(p, w_1y) = \bigcup_{r \in \delta(p, w1)} \hat{\delta}(r, y)$

  Hence for some $r' \in \delta(p, w_1)$, $q \in \hat{\delta}(r', y)$
  By induction, $r' \in S'$ implies $q \in R'$, where $R' = \hat{\delta}'(S', y)$
- Recall we are trying to show
  $p \in S$ implies $q \in R$, where $R = \hat{\delta}'(S, w_1y)$
  rewriting R
  $R = \hat{\delta}'(\delta'(S, w_1), y)$
  $R = \hat{\delta}'(\delta(p, w_1) \cup X, y)$
- Since $p \in S$ implies $r' \in \delta(p, w_1) \cup X$, we can apply induction to conclude $q \in R$, as required

Using the Claim

- By definition of acceptance for M
  - $w \in L(M)$ implies there are states r_0, r_1, … r_n satisfying acceptance conditions
- by the claim
  - there are states of M′ R_0, R_1, …, R_n where $R_0 = \{q_0\}$, $r_i \in R_i$
- since $r_n \in F$ and $r_n \in R_n$, $R_n \cap F \neq \emptyset$
- Hence $R_n \in F'$, establishing the acceptance conditions for M′

Are we there yet?

- We have the construction of M′ from M
- We haven′t shown L(M) = L(M′)
- Need
  - ✓ $w \in L(M) \Rightarrow w \in L(M')$
  - $w \in L(M') \Rightarrow w \in L(M)$

- Why?

$w \in L(M') \Rightarrow w \in L(M)$

- Must show that if the deterministic simulation includes a final state, then the nondeterministic machine could have made choices that reach it
- Claim: $q \in R$, where $R = \hat{\delta}'(S,w)$ implies there exists $p \in S$. $q \in \hat{\delta}(p,w)$
- Proof: By induction on w

$w \in L(M') \Rightarrow w \in L(M)$

- Claim: $q \in R$, where $R = \hat{\delta}'(S,w)$ implies there exists $p \in S$ such that $q \in \hat{\delta}(p,w)$
- Proof: By induction on w
- Basis: $w = \varepsilon$
  In this case R = S. Pick p = q to satisfy the claim

$w \in L(M') \Rightarrow w \in L(M)$

- Claim: $q \in R$, where $R = \delta'\hat{}(S,w)$ implies there exists $p \in S$. $q \in \delta\hat{}(p,w)$
- Step: $w = w_1 y$
  $R = \delta'\hat{}(S,w)$
  $R = \delta'\hat{}(S, w_1 y)$
  $R = \delta'\hat{}(\delta'(S, w_1), y)$
- By induction, there is a p' in $\delta'(S, w_1)$ such that $q \in \delta\hat{}(p',y)$

$w \in L(M') \Rightarrow w \in L(M)$

- Claim: $q \in R$, where $R = \hat{\delta}'(S,w)$ implies there exists $p \in S$ such that $q \in \hat{\delta}(p,w)$
- Step: $w = w_1 y$
- By induction, there is a p' in $\delta'(S, w_1)$ such that $q \in \hat{\delta}'(p',y)$
  By definition, $\delta'(S, w_1) = \cup_{r \in S} \delta(r,a)$
  Hence, for some particular $r \in S$, p' must be in $\delta(r,a)$.
- Take p to be this r.

$w \in L(M') \Rightarrow w \in L(M)$

- Claim: $q \in R$, where $R = \delta'\hat{}(S,w)$ implies there exists $p \in S$ such that $q \in \delta\hat{}(p,w)$
- Step: $w = w_1 y$

- Take p to be this r.
  Verify that p has the property desired as follows:
  $q \in \delta\hat{}(p,w)$
  $q \in \delta\hat{}(p, w_1 y)$
  $q \in \bigcup_{s \in \delta(p,\, w1)} \delta\hat{}(s, y)$

  $q \in \delta\hat{}(r, y) \cup X$ (for some set X, r as previous slide)

But this last line is a consequence of the induction hypothesis, since we have demonstrated $q \in \delta\hat{}(r, y)$

This proves the claim.

Applying the Claim

- To show $w \in L(M') \Rightarrow w \in L(M)$, consider $w \in L(M')$.
- By definition of acceptance this means there is a sequence of states $R_{0,}$ $R_{1,}$ …, R_n satisfying the conditions.
- By definition of M′ and condition 1, $R_0=\{q_0\}$.
- Similarly $R_n \cap F$ is not empty.
- Consider $r_n \in R_n \cap F$, which will satisfy acceptance condition 3
- By repeated application of the claim, select a sequence of $r_i \in R_i$ for each i.
- By the claim these r_i can be selected to satisfy acceptance condition 2 for M. Furthermore, $r_0 = q_{0,}$ satisfying condition 1 for M.
- This establishes $w \in L(M)$, as required

Are we there yet?

- We have the construction of M′ from M
- We haven′t shown L(M) = L(M′)
- Need
  - ✓ – $w \in L(M) \Rightarrow w \in L(M')$
  - ✓ – $w \in L(M') \Rightarrow w \in L(M)$
- Done!

What have we proved?

- If a language is accepted by an NFA then it is regular (accepted by a DFA)

Note

- The development in the book introduces ε-transitions at the same time as nondeterminism
- In this lecture we have omitted ε-transitions
- How does this impact the proofs?
- We will use ε-transitions freely