18.404/6.840 Intro to the Theory of Computation

Instructor: Mike Sipser

TAs:

- Fadi Atieh, Damian Barabonkov,
- Alex Dimitrakakis, Thomas Xiong,
- Abbas Zeitoun, and Emily Liu

18.404 Course Outline

Computability Theory 1930s – 1950s

- What is computable… or not?
- Examples:
  program verification, mathematical truth
- Models of Computation:
  Finite automata, Turing machines, …

Complexity Theory 1960s – present

- What is computable <u>in practice</u>?
- Example: factoring problem
- P versus NP problem
- Measures of complexity: Time and Space
- Models: Probabilistic and Interactive computation

Course Mechanics

Zoom Lectures
- Live and Interactive via Chat
- Live lectures are recorded for later viewing

Zoom Recitations
- Not recorded
- Two convert to in-person
- Review concepts and more examples
- Optional unless you are having difficulty
  Participation can raise low grades
- Attend any recitation

Text
- *Introduction to the Theory of Computation*
  Sipser, 3rd Edition US. (Other editions ok but are missing some Exercises and Problems).

Homework bi-weekly – 35%
- More information to follow

Midterm (15%) and Final exam (25%)
- Open book and notes

Check-in quizzes for credit – 25%
- Distinct Live and Recorded versions
- Complete either one for credit within 48 hours
- Initially ungraded; full credit for participation

Course Expectations

Prerequisites

Prior substantial experience and comfort with mathematical concepts, theorems, and proofs. <u>Creativity will be needed for psets and exams.</u>

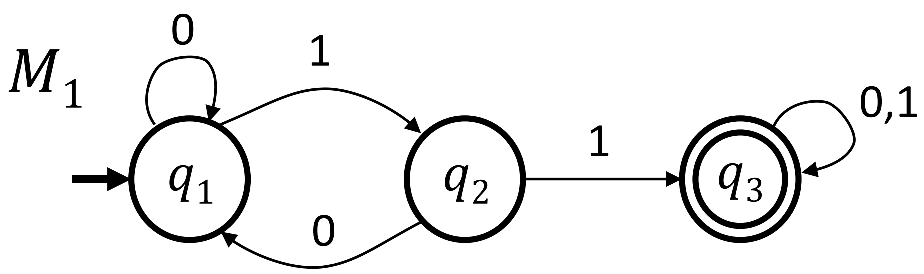
Collaboration policy on homework

- Allowed. But try problems yourself first.
- <u>Write up your own solutions.</u>
- <u>No bibles or online materials.</u>

Role of Theory in Computer Science

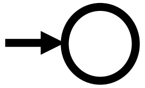
1. **Applications**
2. **Basic Research**
3. **Connections to other fields**
4. **What is the nature of computation?**

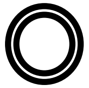
Let's begin: Finite Automata



States: $q_1\ q_2\ q_3$

Transitions: $\xrightarrow{1}$

Start state: 

Accept states: 

Input: finite string
Output: Accept or Reject

Computation process: Begin at start state, read input symbols, follow corresponding transitions, Accept if end with accept state, Reject if not.

Examples: 01101 → Accept
00101 → Reject

M_1 accepts exactly those strings in A where
$A = \{w \mid w \text{ contains substring } 11\}$.

Say that A is the language of M_1 and that M_1 recognizes A and that $A = L(M_1)$.

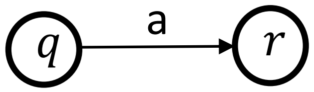
Finite Automata – Formal Definition

Defn: A <u>finite automaton</u> M is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$

Q finite set of states

Σ finite set of alphabet symbols

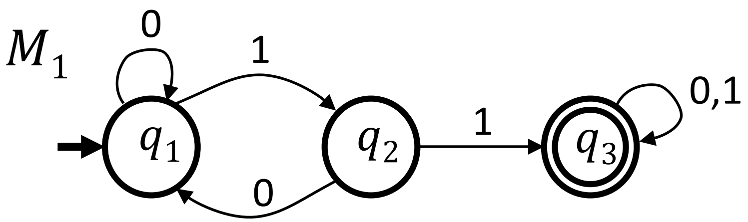
δ transition function $\delta\colon Q \times \Sigma \rightarrow Q$

$\delta\,(q,\ a) \ = \ r$ means 

q_0 start state

F set of accept states

Example:



$M_1 \ = \ (Q, \Sigma, \delta, q_1, F)$

$Q \ = \ \{q_1, q_2, q_3\}$

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

$F \ = \ \{q_3\}$

$\delta =$

	0	1
q_1	q_1	q_2
q_2		
q_3		

Finite Automata – Computation

Strings and languages

- A string is a finite sequence of symbols in Σ
- A language is a set of strings (finite or infinite)
- The empty string ε is the string of length 0
- The empty language ø is the set with no strings

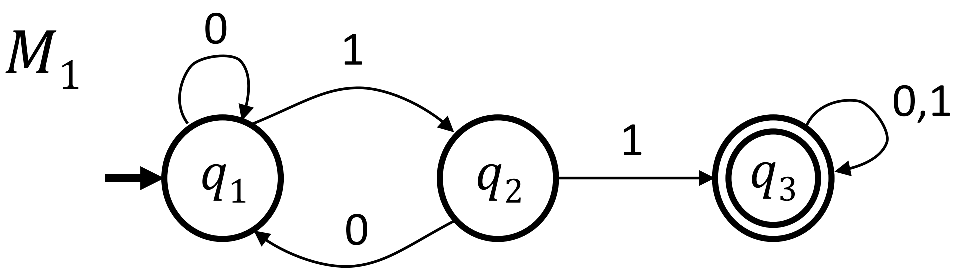
Defn: M accepts string $w = w_1 w_2 \ldots w_n$ each $w_i \in \Sigma$
if there is a sequence of states $r_0, r_1, r_2, , \ldots, r_n \in Q$
where:

- $r_0 = q_0$
- $r_i = \delta(r_{i-1}, w_i)$ for $1 \leq i \leq n$
- $r_n \in F$

- $L(M) = \{w \mid M \text{ accepts } w\}$
- $L(M)$ is the language of M
- M recognizes $L(M)$

Defn: A language is regular if some finite automaton recognizes it.

Regular Languages – Examples



$L(M_1) = \{w|\ w \text{ contains substring } 11\} = A$

<u>Therefore A is regular</u>

More examples:

Let $B = \{w|\ w$ has an even number of 1s$\}$
B is regular (make automaton for practice).

Let $C = \{w|\ w$ has equal numbers of 0s and 1s$\}$
C is <u>not</u> regular (we will prove).

Goal: Understand the regular languages

Regular Expressions

Regular operations. Let A, B be languages:

- Union: $A \cup B = \{w \mid w \in A \text{ or } w \in B\}$
- Concatenation: $A \circ B = \{xy \mid x \in A \text{ and } y \in B\} = AB$
- Star: $A^* = \{x_1 \ldots x_k \mid \text{each } x_i \in A \text{ for } k \geq 0\}$

  Note: $\varepsilon \in A^*$ always

Example. Let $A = \{\text{good, bad}\}$ and $B = \{\text{boy, girl}\}$.

- $A \cup B = \{\text{good, bad, boy, girl}\}$
- $A \circ B = AB = \{\text{goodboy, goodgirl, badboy, badgirl}\}$
- $A^* = \{\varepsilon, \text{good, bad, goodgood, goodbad, badgood, badbad, goodgoodgood, goodgoodbad}, \ldots\}$

Regular expressions

- Built from Σ, members $\Sigma, \emptyset, \varepsilon$ [Atomic]
- By using $\cup, \circ, *$ [Composite]

Examples:

- $(0 \cup 1)^* = \Sigma^*$ gives all strings over Σ
- $\Sigma^* 1$ gives all strings that end with 1
- $\Sigma^* 11 \Sigma^* =$ all strings that contain $11 = L(M_1)$

Goal: Show finite automata equivalent to regular expressions

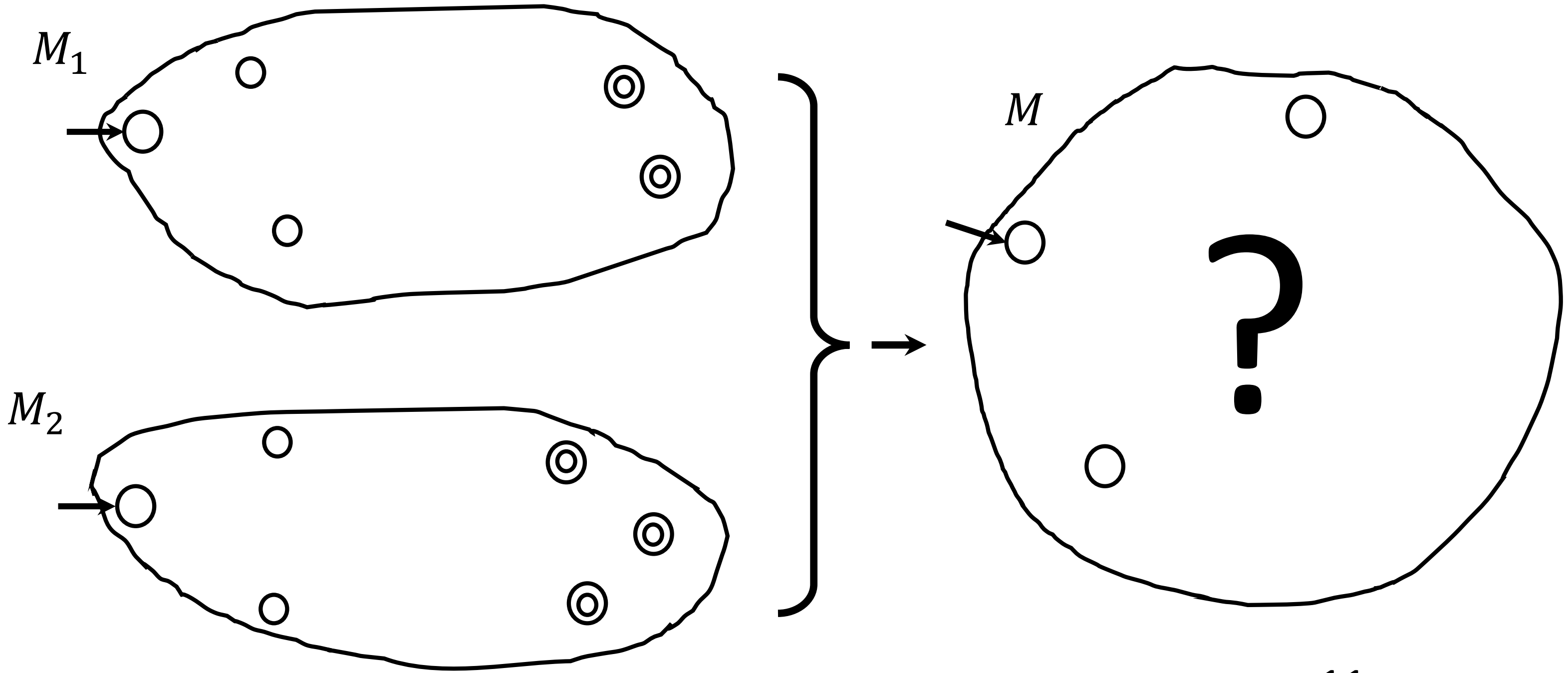
Closure Properties for Regular Languages

Theorem: If A_1, A_2 are regular languages, so is $A_1 \cup A_2$ (closure under $\cup$)

Proof: Let $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize A_1

$M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$

Construct $M = (Q, \Sigma, \delta, q_0, F)$



Components of $\boldsymbol{M}$:

$Q = Q_1 \times Q_2$

$= \{(q_1, q_2) | q_1 \in Q_1 \text{ and } q_2 \in Q_2\}$

$q_0 = (q_1, q_2)$

$\delta((q, r), a) = (\delta_1(q, a), \delta_2(r, a))$

$F = F_1 \times F_2$ NO! [gives intersection]

$F = (F_1 \times Q_2) \cup (Q_1 \times F_2)$

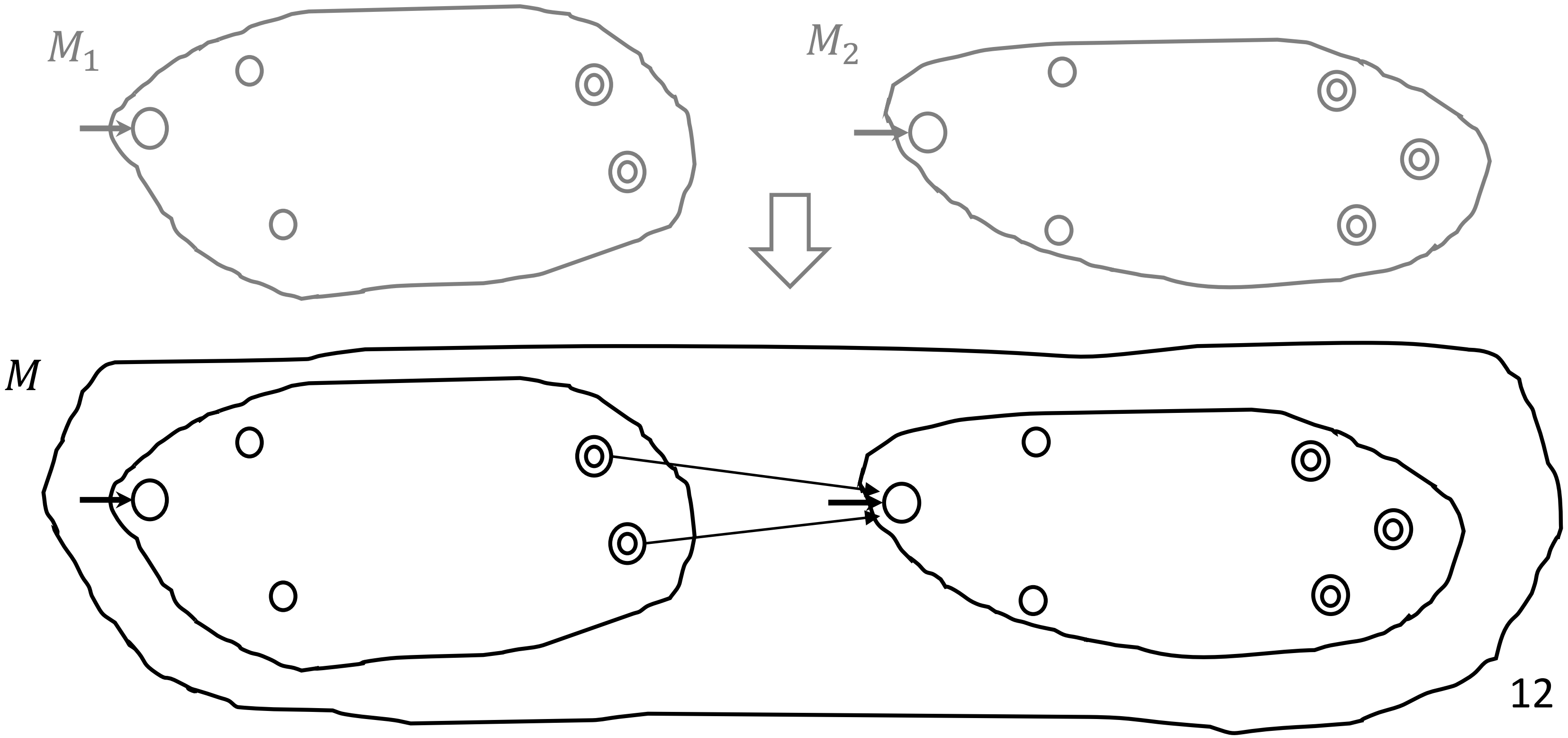
Closure Properties continued

Theorem: If A_1, A_2 are regular languages, so is A_1A_2 (closure under $\circ$)

Proof: Let $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize A_1

$M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$

Construct $M = (Q, \Sigma, \delta, q_0, F)$



M should accept input w
if $w = xy$ where
M_1 accepts x and M_2 accepts y.

Doesn't work: Where to split w?

Quick review of today

1. Introduction, outline, mechanics, expectations
2. Finite Automata, formal definition, regular languages
3. Regular Operations and Regular Expressions
4. Proved: Class of regular languages is closed under $\cup$
5. Started: Closure under $\circ$, to be continued…

18.404/6.840 Lecture 2

Last time: (Sipser §1.1)

- Finite automata, regular languages
- Regular operations $\cup, \circ, *$
- Regular expressions
- Closure under $\cup$

Today: (Sipser §1.2 – §1.3)

- Nondeterminism
- Closure under $\circ$ and $*$
- Regular expressions → finite automata

Goal: Show finite automata equivalent to regular expressions

Problem Sets

- 35% of overall grade

- Problems are hard! Leave time to think about them.

- Writeups need to be clear and understandable, handwritten ok.
  Level of detail in proofs comparable to lecture: focus on main ideas.
  Don't need to include minor details.

- Submit via gradescope (see Canvas) by 2:30pm Cambridge time.
  Late submission accepted (on gradescope) until 11:59pm following day:
    1 point (out of 10 points) per late problem penalty.
  After that solutions are posted so not accepted without S3 excuse.

- Optional problems:
    Don't count towards grade except for A+.
    Value to you (besides the challenge):
    Recommendations, employment (future grading, TA, UROP)

- Problem Set 1 is due in one week.

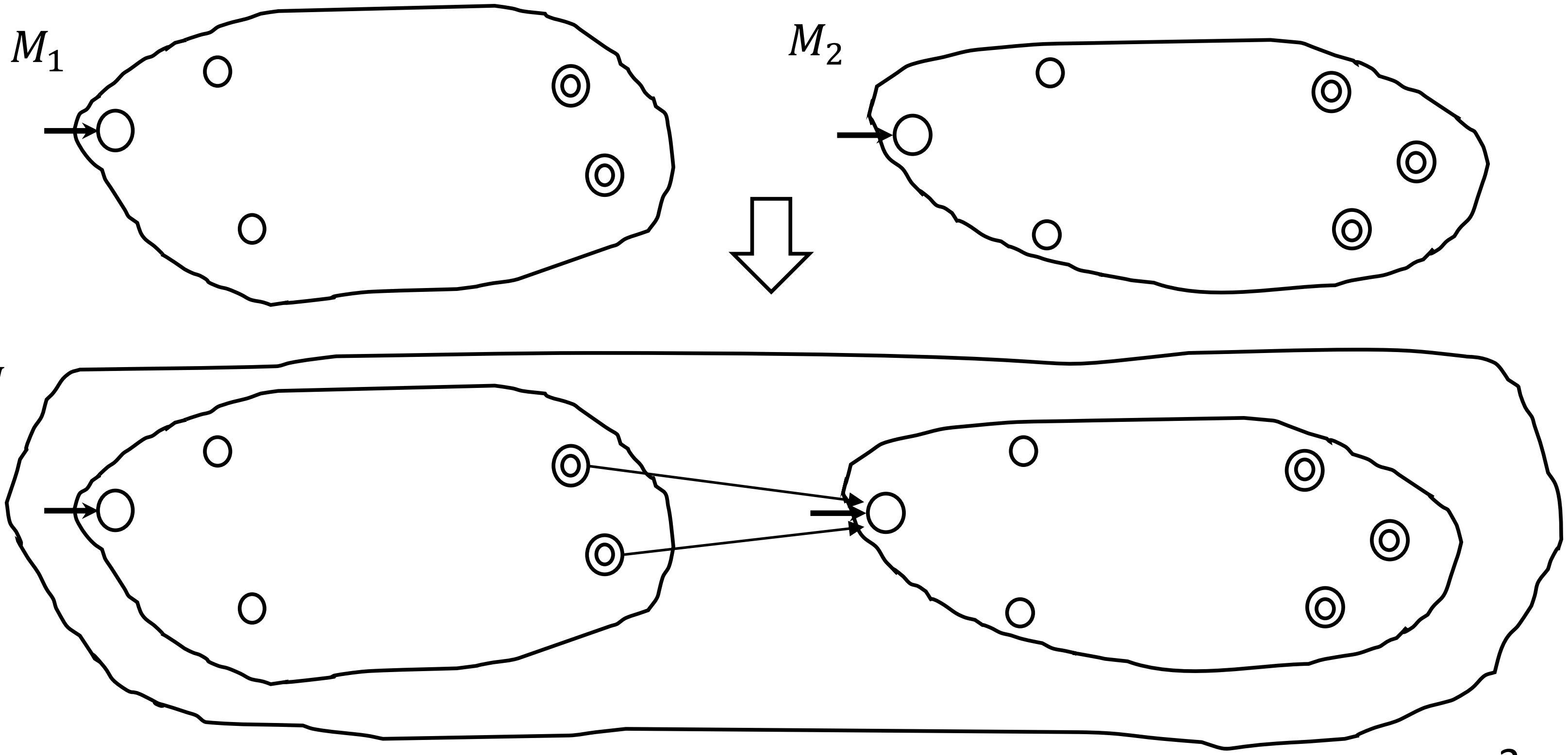
Closure Properties for Regular Languages

Theorem: If A_1, A_2 are regular languages, so is A_1A_2 (closure under $\circ$)

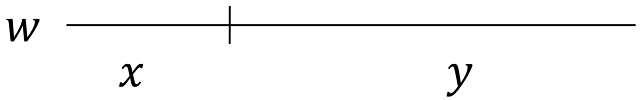
Recall proof attempt: Let $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize A_1

$M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$

Construct $M = (Q, \Sigma, \delta, q_0, F)$



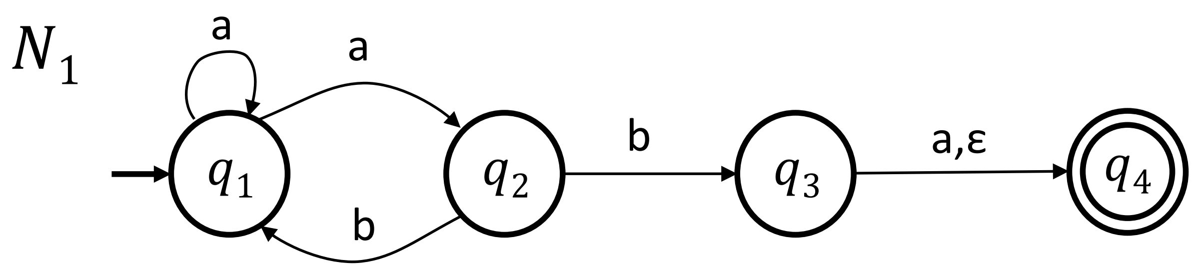
M should accept input w
if $w = xy$ where
M_1 accepts x and M_2 accepts y.



Doesn't work: Where to split w?

Hold off. Need new concept.

Nondeterministic Finite Automata



New features of nondeterminism:

- multiple paths possible (0, 1 or many at each step)
- ε-transition is a "free" move without reading input
- Accept input if <u>some</u> path leads to ◎ accept

Example inputs:

- ab
- aa
- aba
- abb

Check-in 2.1

What does N_1 do on input aab ?

(a) Accept

(b) Reject

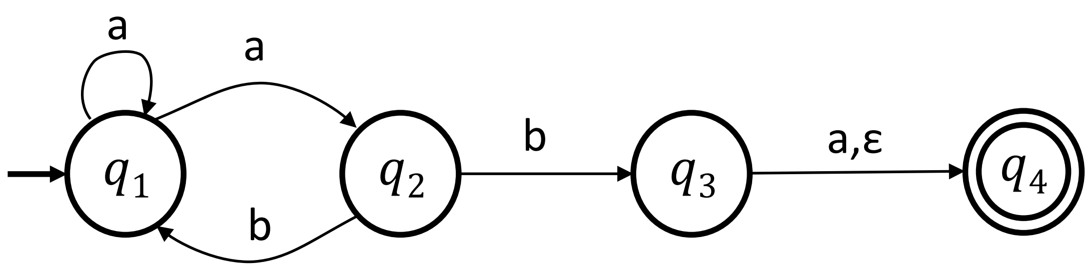
(c) Both Accept and Reject

Nondeterminism doesn't correspond to a physical machine we can build. However, it is useful mathematically.

Check-in 2.1

NFA – Formal Definition

N_1



Defn: A <u>nondeterministic finite automaton (NFA)</u> N is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$

states
alphabet
transition function
start state
accept states

- all same as before except δ
- $\delta: \underbrace{Q \times \Sigma}_{\Sigma \cup \{\varepsilon\}} \rightarrow \mathcal{P}(Q) = \{R | R \subseteq Q\}$ power set
- In the N_1 example: $\delta(q_1, \text{a}) = \{q_1, q_2\}$
  $\delta(q_1, \text{b}) = \emptyset$

Ways to think about nondeterminism:

<u>Computational:</u> Fork new parallel thread and accept if any thread leads to an accept state.

<u>Mathematical:</u> Tree with branches.
Accept if any branch leads to an accept state.

<u>Magical:</u> Guess at each nondeterministic step which way to go. Machine always makes the right guess that leads to accepting, if possible.

Converting NFAs to DFAs

Theorem: If an NFA recognizes A then A is regular

Proof: Let NFA $M = (Q, \Sigma, \delta, q_0, F)$ recognize A

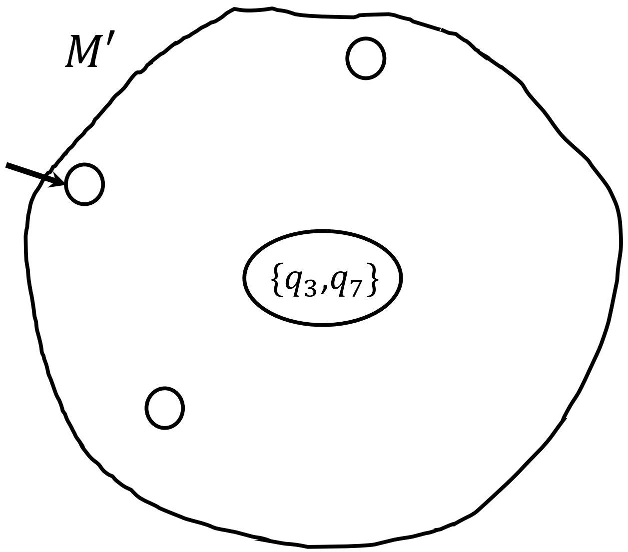
Construct DFA $M' = (Q', \Sigma, \delta', q_0', F')$

(Ignore the

Check-in 2.2

If M has n states, how many states does M' have by this construction?

(a) $2n$
(b) n^2
(c) 2^n



Construction of M':

$Q' = \mathcal{P}(Q)$

$\delta'(R, a) =$

$q_0' = \{q_0\}$

$F' = \{R \in Q' \mid R \text{ intersects } F\}$

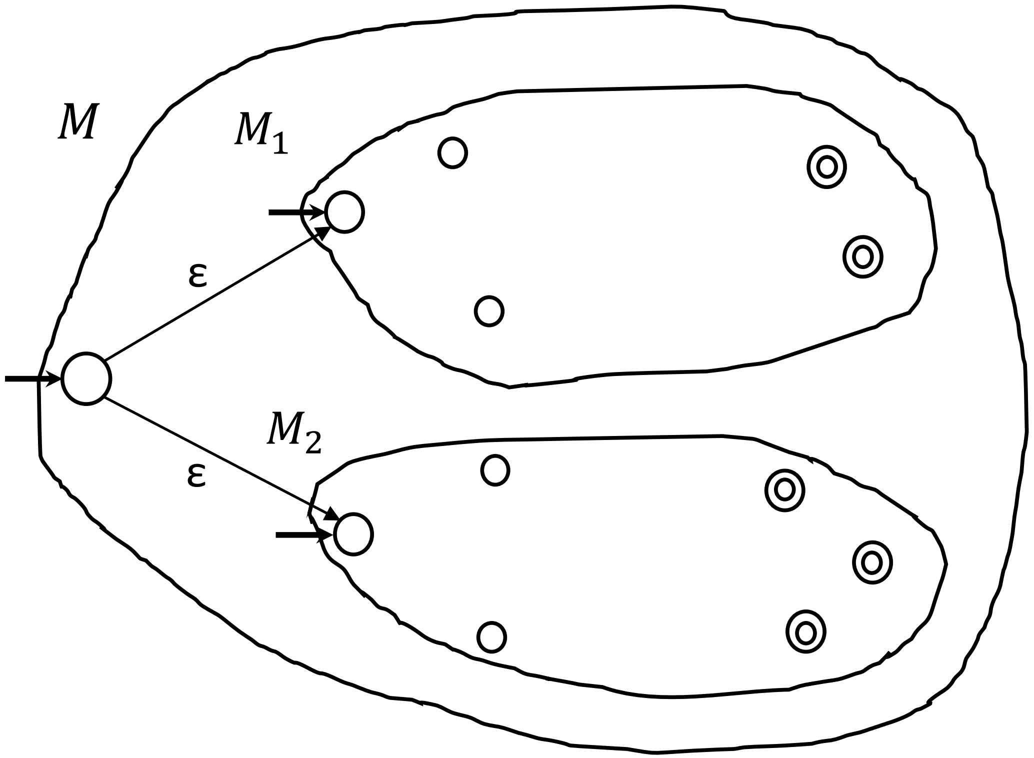
Check-in 2.2

Return to Closure Properties

Recall Theorem:

(The class of regular languages is closed under union)

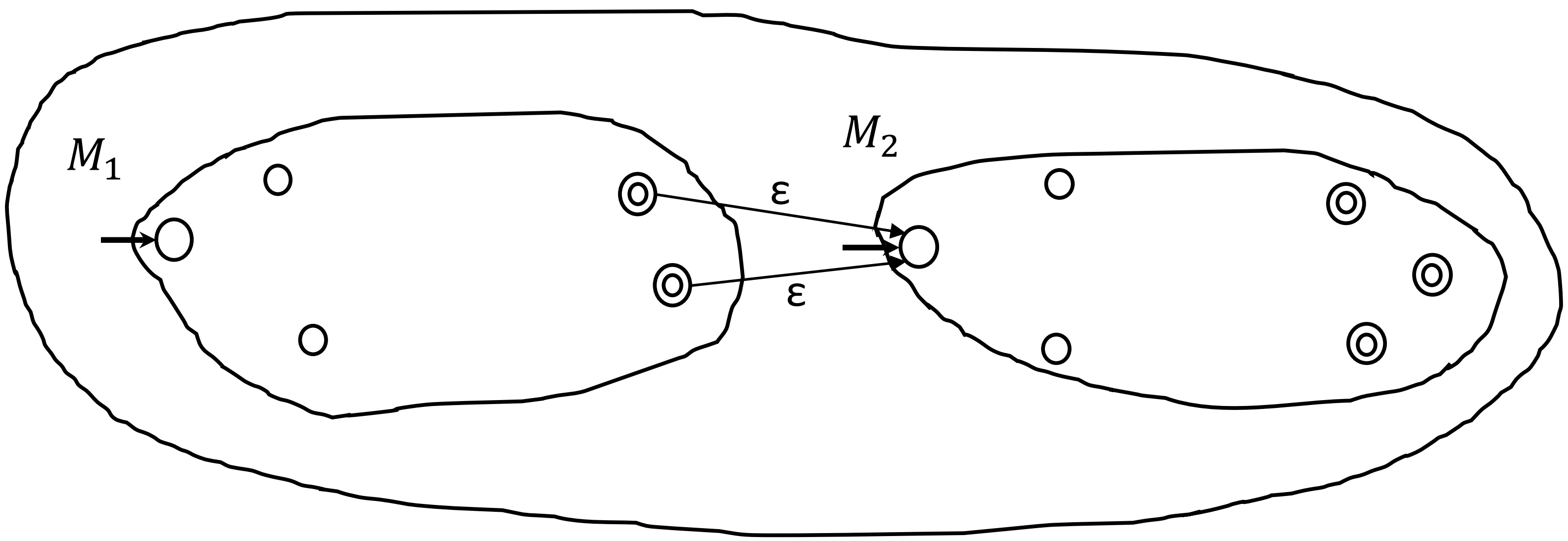
New Proof (sketch):



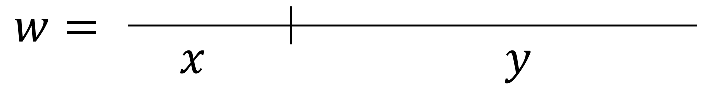
Closure under $\circ$ (concatenation)

Theorem:

Proof sketch:



M should accept input w
if $w = xy$ where
M_1 accepts x and M_2 accepts y.

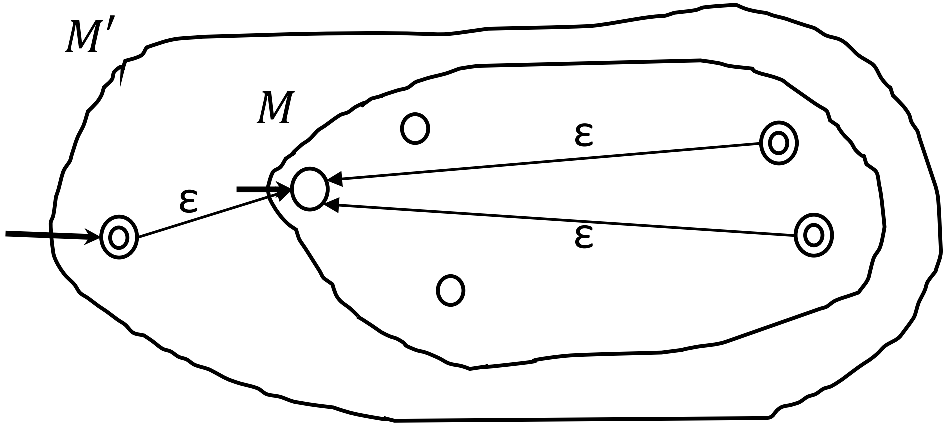


Nondeterministic M' has the option to jump to M_2 when M_1 accepts.

Closure under $*$ (star)

Theorem:

Proof sketch:



accepts ε

Check-in 2.3

If M has n states, how many states does M' have by this construction?

(a) n

(b) $n + 1$

(c) $2n$

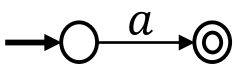
Check-in 2.3

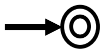
Regular Expressions → NFA

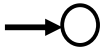
Theorem: If R is a regular expr and $A = L(R)$ then A is regular

Proof: Convert R to equivalent NFA M:

If R is <u>atomic:</u> is:

$R = a$ for $a \in \Sigma$ 

$R = \varepsilon$ 

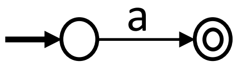
$R = \emptyset$ 

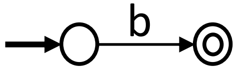
If R is <u>composite:</u>

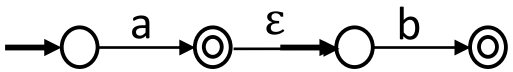
$$\left.\begin{aligned} R &= R_1 \cup R_2 \\ R &= R_1 \circ R_2 \\ R &= R_1^* \end{aligned}\right\}$$

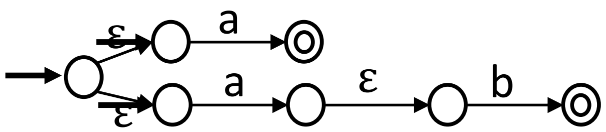
Example:

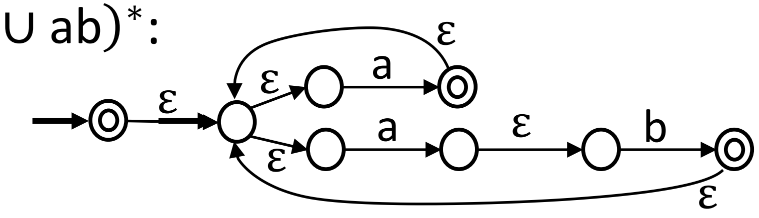
Convert $(\mathrm{a} \cup \mathrm{ab})^*$ to equivalent NFA

a: 

b: 

ab: 

a ∪ ab: 

$(\mathrm{a} \cup \mathrm{ab})^*$: 

Quick review of today

1. Nondeterministic finite automata (NFA)
2. Proved: NFA and DFA are equivalent in power
3. Proved: Class of regular languages is closed under $\circ, *$
4. Conversion of regular expressions to NFA

Check-in 2.4

Recitations start tomorrow online (same link as for lectures).
They are optional, unless you need more help.
You may attend any recitation(s).
Which do you think you'll attend? (you may check several)
(a) 10:00 (b) 11:00 (c) 12:00
(d) 1:00 (e) 2:00 (f) I prefer a different time (please post on piazza, but no promises)

Check-in 2.4

18.404/6.840 Lecture 3

Last time:
- Nondeterminism
- NFA → DFA
- Closure under $\circ$ and $*$
- Regular expressions → finite automata

Today: (Sipser §1.4 – §2.1)
- Finite automata → regular expressions
- Proving languages aren't regular
- Context free grammars

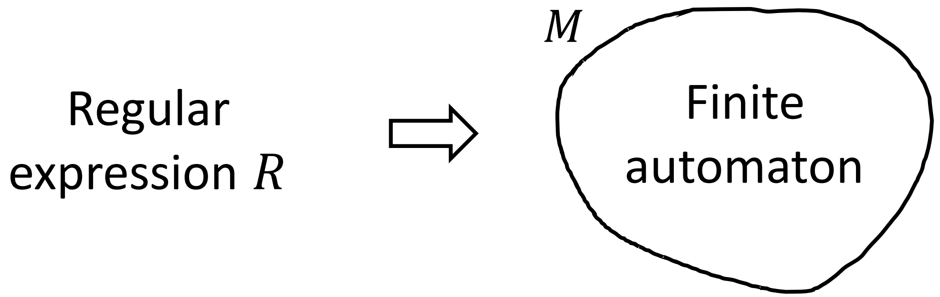
We start counting Check-ins today.
Review your email from Canvas.

Homework due Thursday.

DFAs → Regular Expressions

Recall Theorem: If R is a regular expressipn and $A = L(R)$ then A is regular

Proof: Conversion $R \rightarrow$ NFA $M \rightarrow$ DFA M'



Recall: we did $(\text{a} \cup \text{ab})^*$ as an example

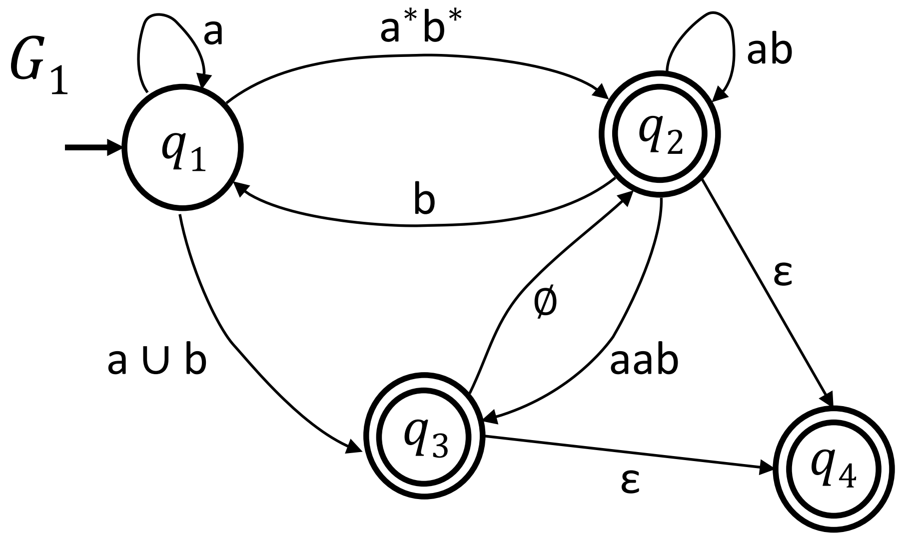
Today's Theorem: If A is regular then $A = L(R)$ for some regular expr R

Proof: Give conversion DFA $M \rightarrow R$

WAIT! Need new concept first.

Generalized NFA

Defn: A Generalized Nondeterministic Finite Automaton (GNFA) is similar to an NFA, but allows regular expressions as transition labels



For convenience we will assume:

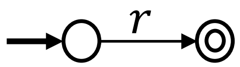
- One accept state, separate from the start state
- One arrow from each state to each state, except
  a) only exiting the start state
  b) only entering the accept state

We can easily modify a GNFA to have this special form.

GNFA → Regular Expressions

Lemma: Every GNFA G has an equivalent regular expression R
Proof: By induction on the number of states k of G

Basis $(k = 2)$:

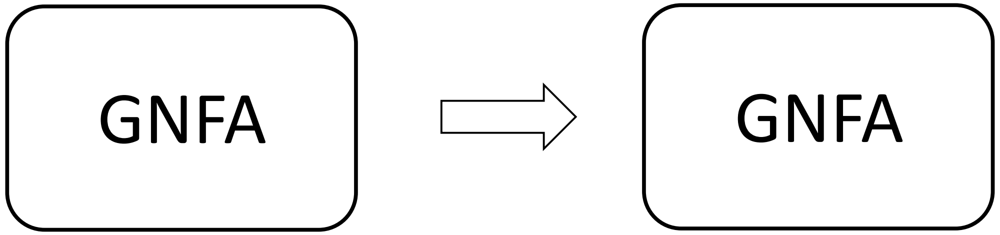
$G =$ 

Remember: G is in special form

Let $R = r$

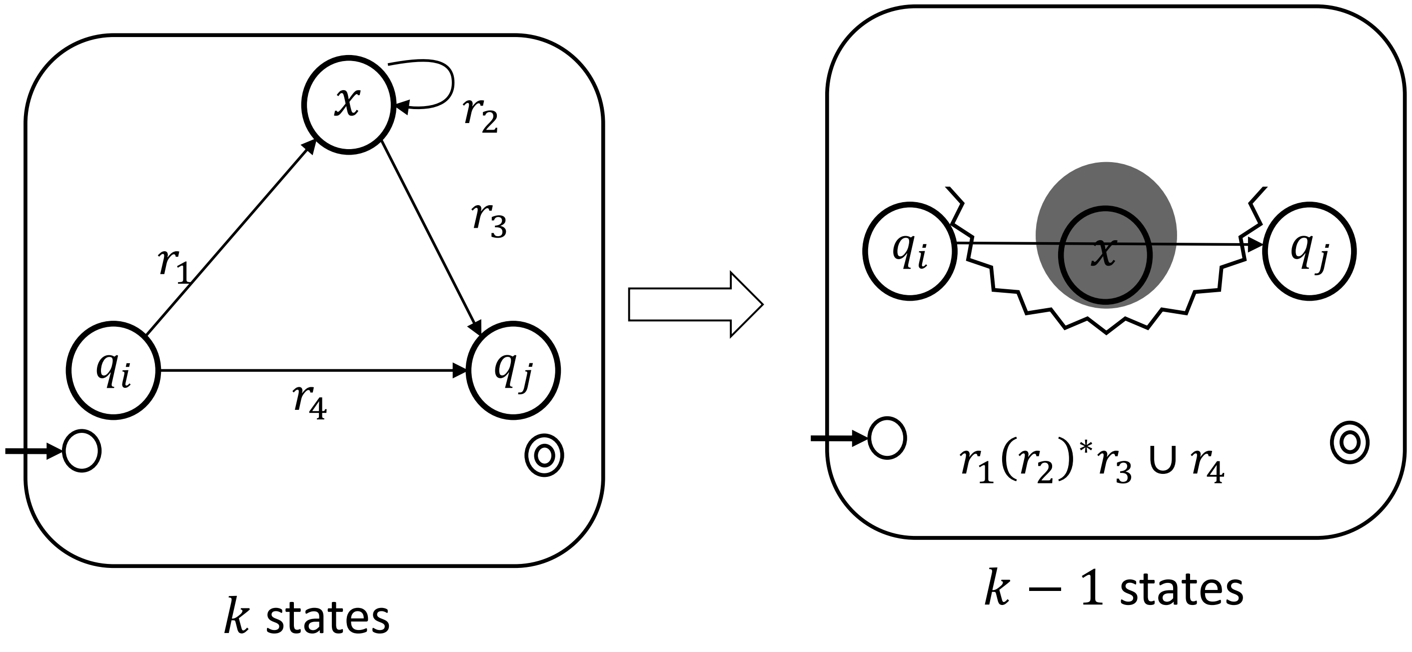
Induction step $(k > 2)$: Assume Lemma true for $k - 1$ states and prove for k states

IDEA: Convert k-state GNFA to equivalent $(k - 1)$ -state GNFA



k states $\qquad$ $k - 1$ states

k-state GNFA $\rightarrow$ $(k-1)$-state GNFA



1. Pick any state x except the start and accept states.
2. Remove x.
3. Repair the damage by recovering all paths that went through x.
4. Make the indicated change for each pair of states q_i, q_j .

Thus DFAs and regular expressions are equivalent.

Non-Regular Languages

How do we show a language is not regular?

- Remember, to show a language *is* regular, we give a DFA.
- <u>To show a language is *not* regular, we must give a proof.</u>
- It is not enough to say that you couldn't find a DFA for it, therefore the language isn't regular.

Two examples: Here $\Sigma = \{0,1\}$.

1. Let $B = \{w | \ w$ has equal numbers of 0s and 1s$\}$

Intuition: B is not regular because DFAs cannot count unboundedly.

Moral: You need to give a proof.

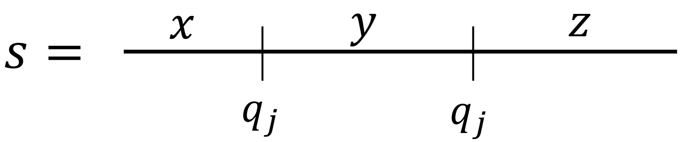
Method for Proving Non-regularity

Pumping Lemma: For every regular language A, there is a number p (the "pumping length") such that if $s \in A$ and $|s| \geq p$ then $s = xyz$ where

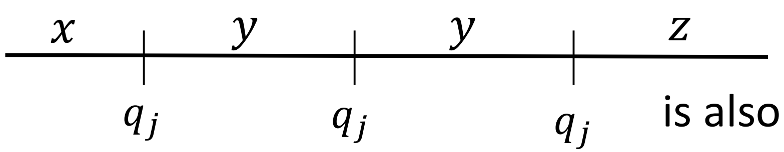
1) $xy^iz \in A$ for all $i \geq 0$ $\qquad y^i = \underbrace{yy\cdots y}_{i}$
2) $y \neq \varepsilon$
3) $|xy| \leq p$

Informally: A is regular → every long string in A can be pumped and the result stays in A.

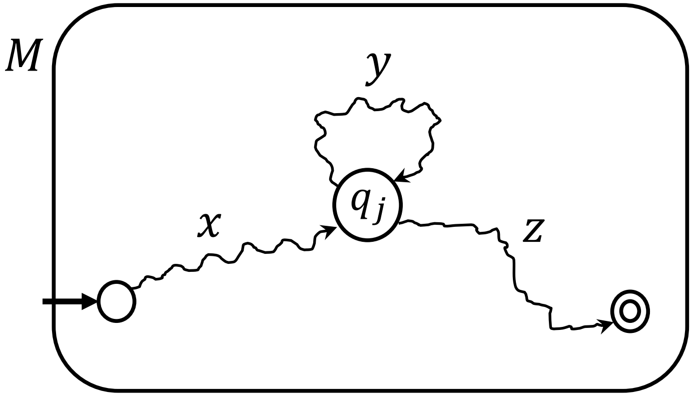
Proof: Let DFA M recognize A. Let p be the number of states in M. Pick $s \in A$ where $|s| \geq p$.



M will repeat a state q_j when reading s because s is so long.



is also accepted



The path that M follows when reading s.

Example 1 of Proving Non-regularity

Pumping Lemma: For every regular language A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = xyz$ where

1) $xy^iz \in A$ for all $i \geq 0$ $\qquad y^i = yy \cdots y$
2) $y \neq \varepsilon$
3) $|xy| \leq p$

Let $D = \{0^k1^k \mid k \geq 0\}$

Show: D is not regular

Proof by Contradiction:

Assume (to get a contradiction) that D is regular.
The pumping lemma gives p as above. Let $s = 0^p1^p \in D$.
Pumping lemma says that can divide $s = xyz$ satisfying the 3 conditions.

$$s = \underline{000 \cdots 000111 \cdots 111}$$
$$x \mid y \mid \qquad z$$
$$\leftarrow \leq p \rightarrow$$

But $xyyz$ has excess 0s and thus $xyyz \notin D$ contradicting the pumping lemma.
Therefore our assumption (D is regular) is false. We conclude that D is not regular.

Example 2 of Proving Non-regularity

Pumping Lemma: For every regular language A, there is a p such that if $s \in A$ and $|s| \geq p$ then $s = xyz$ where

1) $xy^i z \in A$ for all $i \geq 0$ $\qquad y^i = yy \cdots y$
2) $y \neq \varepsilon$
3) $|xy| \leq p$

Let $F = \{ww|\ w \in \Sigma^*\}$. Say $\Sigma^* = \{0,1\}$.

Show: F is not regular

Proof by Contradiction:

Assume (for contradiction) that F is regular.

The pumping lemma gives p as above. Need to choose $s \in F$. Which s?

Try $s = 0^p 0^p \in F$.

Try $s = 0^p 1 0^p 1 \in F$. Show cannot be pumped $s = xyz$ satisfying the 3 conditions.

$xyyz \notin F$ Contradiction! Therefore F is not regular.

$$s = \underbrace{000 \cdots 000000 \cdots 000}_{x \,|\, y \,|\, z}$$

$\leftarrow \leq p \rightarrow$

$y = 00$

$$s = \underbrace{000 \cdots 001000 \cdots 001}_{x \,|\, y \,|\, z}$$

$\leftarrow \leq p \rightarrow$

Example 3 of Proving Non-regularity

Variant: Combine closure properties with the Pumping Lemma.

Let $B = \{w|\ w \text{ has equal numbers of 0s and 1s}\}$

Show: B is not regular

Proof by Contradiction:

Assume (for contradiction) that B is regular.

We know that 0^*1^* is regular so $B \cap 0^*1^*$ is regular (closure under intersection).

But $D = B \cap 0^*1^*$ and we already showed D is not regular. Contradiction!

Therefore our assumption is false, so B is not regular.

Context Free Grammars

G_1

$$\left.\begin{aligned} &S \to 0S1 \\ &S \to R \\ &R \to \varepsilon \end{aligned}\right\}$$

In G_1:

Rule: Variable → string of variables and terminals
Variables: Symbols appearing on left-hand side of rule
Terminals: Symbols appearing only on right-hand side
Start Variable: Top left symbol

Example of G_1 generating a string

Tree of substitutions

Resulting string

Grammars generate strings

1. Write down start variable
2. Replace any variable according to a rule
   Repeat until only terminals remain
3. Result is the generated string
4. $L(G)$ is the language of all generated strings.

$\in L(G_1)$

$$L(G_1) = \{0^k 1^k \mid k \geq 0\}$$

Quick review of today

Summary: DFAs, NFAs, regular expressions are all equivalent

2. Proving languages not regular by using the pumping lemma and closure properties
3. Context Free Grammars