

18.404/6.840 Intro to the Theory of Computation

Instructor: Mike Sipser

Office Hours 4:00 – 5:30 Tuesdays

TAs: Office Hours TBD

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

Recitations start Friday

- Optional unless you need them!
- Hourly 10-2pm, online. On Sept 11, noon and 2pm → in-person

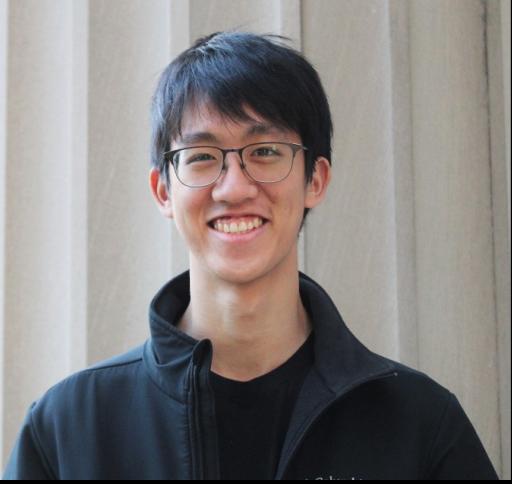
Homework, Exams, Quizzes

- See Course Information on homepage math.mit.edu/18.404
- First Pset due Sept 10. Posted on homepage

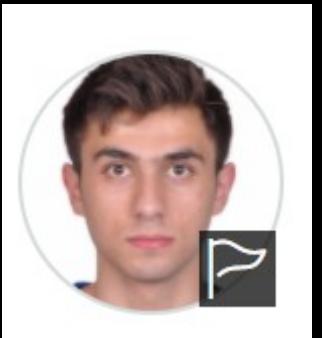
Our TAs



Alex



Thomas



Fadi



Abbas



Damian



Emily

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 in practice?
- 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 starting this Friday

- Not recorded; notes will be posted
- Two convert to in-person on Sept 11
- Review concepts and more examples
- Optional unless you are having difficulty
Participation can raise low grades
- Attend any recitation

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.
Creativity will be needed for psets and exams.

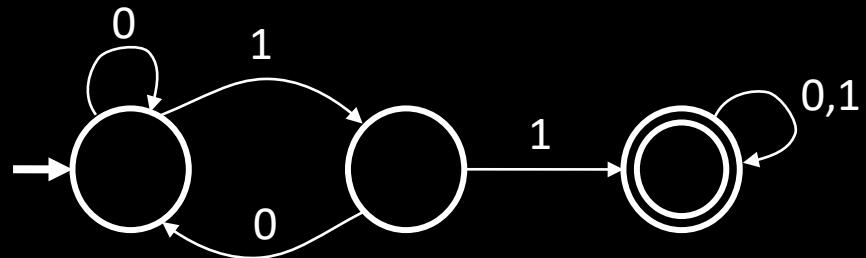
Collaboration policy on homework

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

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:

Transitions: $\xrightarrow{1}$

Start state: $\xrightarrow{\quad}$

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 \rightarrow Accept

00101 \rightarrow Reject

accepts exactly those strings in where contains substring

Say that **is the language of** and that **recognizes** and that .

Finite Automata – Formal Definition

Defn: A finite automaton is a 5-tuple

finite set of states

finite set of alphabet symbols

transition function

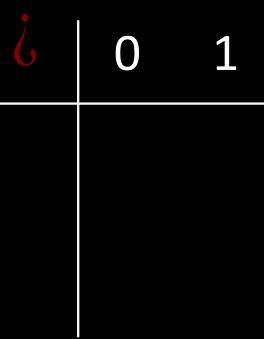
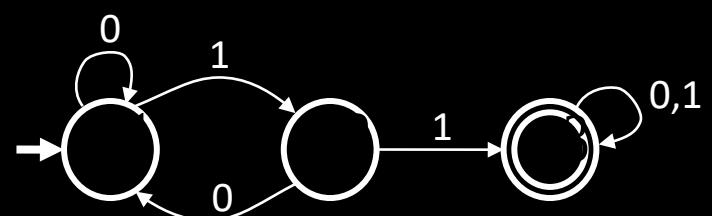
start state

set of accept states

means



Example:



Finite Automata – Computation

Strings and languages

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

Defn: accepts string each
if there is a sequence of states
where:

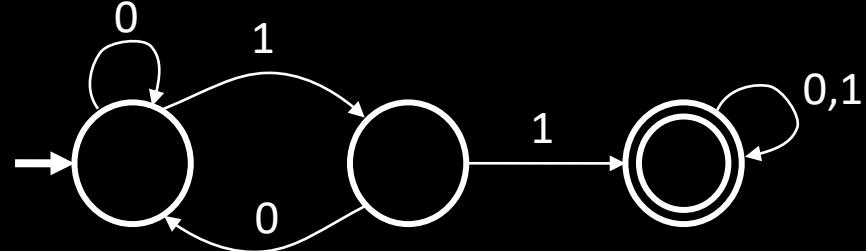
-
- for
-

Recognizing languages

-
- is the language of
- recognizes

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

Regular Languages – Examples



Therefore is regular

More examples:

Let L has an even number of 1s
is regular (make automaton for practice).

Let L has equal numbers of 0s and 1s
is not regular (we will prove).

Goal: Understand the regular languages

Regular Expressions

Regular operations. Let L be languages:

- Union: or
- Concatenation: and
- Star: each for
 - Note: always

Example. Let good, bad and boy, girl.

- $\{good, bad, boy, girl\}$
- $\{goodboy, goodgirl, badboy, badgirl\}$
- $\{\epsilon, good, bad, goodgood, goodbad, badgood, badbad, goodgoodgood, goodgoodbad, \dots\}$

Regular expressions

- Built from ϵ , members [Atomic]
- By using [Composite]

Examples:

- gives all strings over
- gives all strings that end with 1
- all strings that contain 11

Goal: Show finite automata equivalent to regular expressions

Closure Properties for Regular Languages

Theorem: If L_1 and L_2 are regular languages, so is $L_1 \cup L_2$ (closure under union)

Proof: Let M_1 recognize L_1 and M_2 recognize L_2 .

Construct a new DFA M recognizing $L_1 \cup L_2$.
The DFA M should accept input w if either M_1 or M_2 accept w .

Mini-quiz 3

In the proof, if M_1 and M_2 are finite automata where M_1 has n states and M_2 has m states. Then how many states does M have?

- (a) $n + m$
- (b) $n \times m$
- (c) n^m

Components of :

and

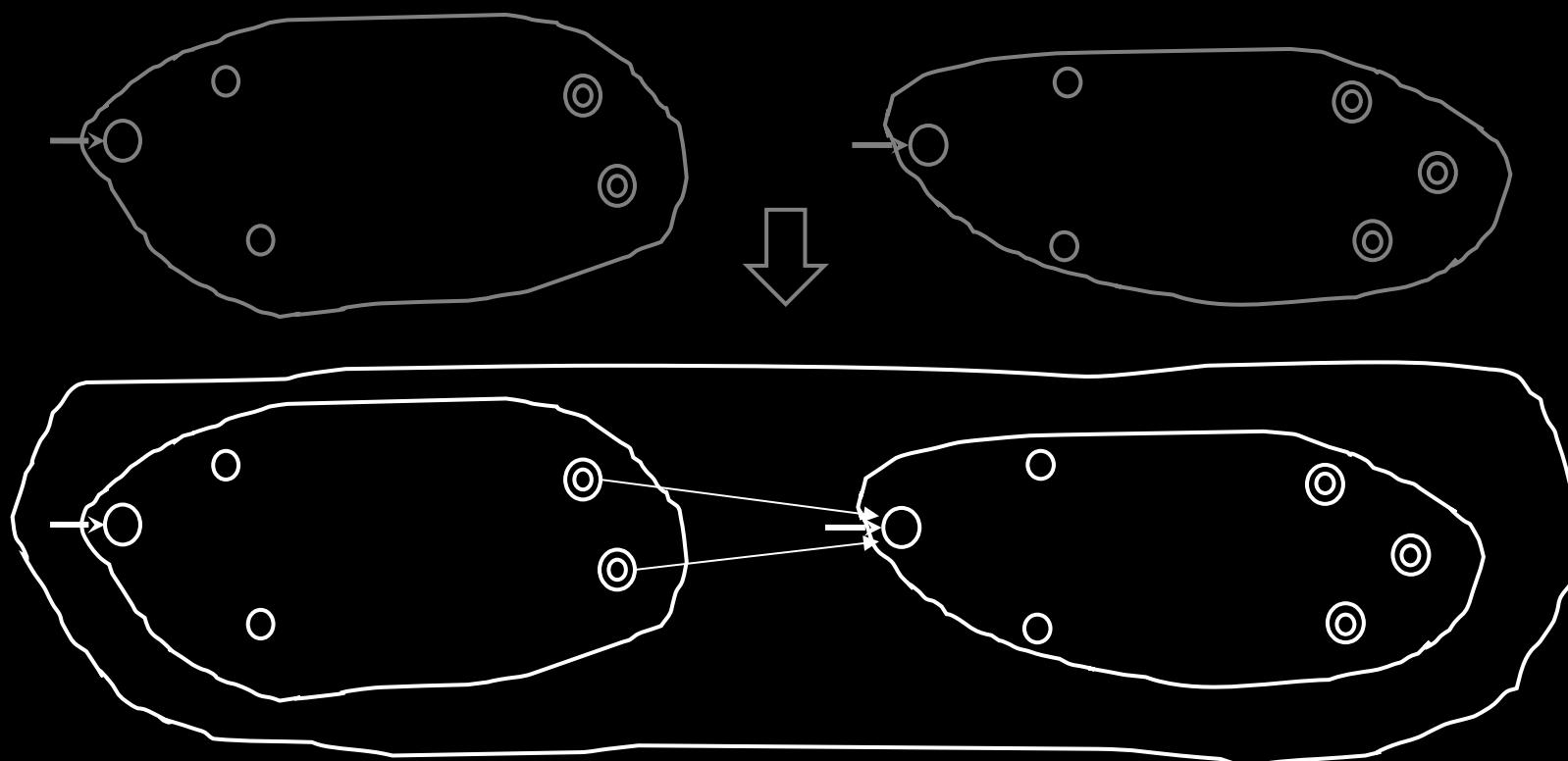
— NO! [gives intersection]

Closure Properties continued

Theorem: If are regular languages, so is (closure under)

Proof: Let recognize
recognize

Construct recognizing



should accept input
if where
accepts and accepts .



Doesn't work: Where to split ?



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
5. Started: Closure under , to be continued...