18.404/6.840 Intro to the Theory of Computation

Instructor: Mike Sipser

2021.10.9 Jiawei Wang

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. Creativity will be needed for psets and exams.

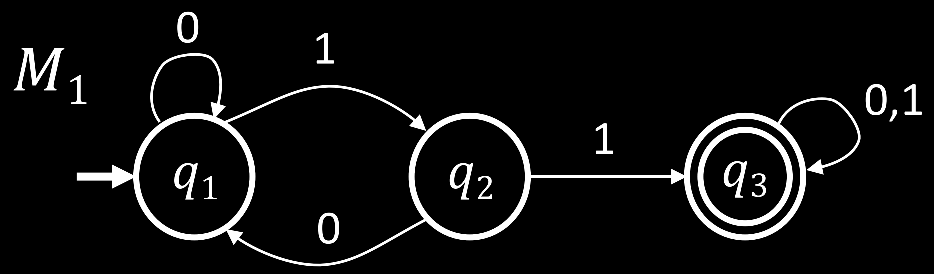
Collaboration policy on homework

- 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?**

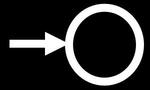
Let's begin: Finite Automata



You can think of it as representing a computer that has a very limited and small amount of memory

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

M1->Collection of these strings

M_1 accepts exactly those strings in A where

$A = \{w|\ w \text{ contains substring } 11\}$.

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

3 –same-way of saying the same thing

Finite Automata – Formal Definition

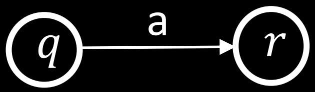
The reason for having a formal definition is, for one thing, it allows us to be very precise, then we'll know exactly what we mean by a finite automaton and then we'll know exactly what we mean by a finite automaton and should answer any question about what counts and what doesn't count. Another reason is that we can use its formal notation rather than as a kind of a picture when we want to represents them in formal articles.

Defn: A finite automaton 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 $\delta\colon Q \times \Sigma \rightarrow Q$

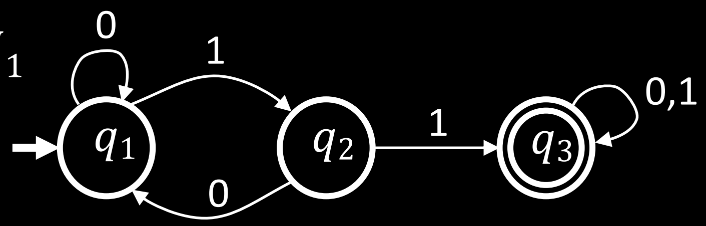
$\delta\,(q,\ a) = r$ means 

q_0 start state (only one)

F set of accept states

Example:

M_1



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

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

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

$F = \{q_3\}$

$\delta =$

	0	1
q_1	q_1	q_2
q_2	q_1	q_3
q_3	q_3	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 $1 \leq i \leq n$
- $r_n \in F$

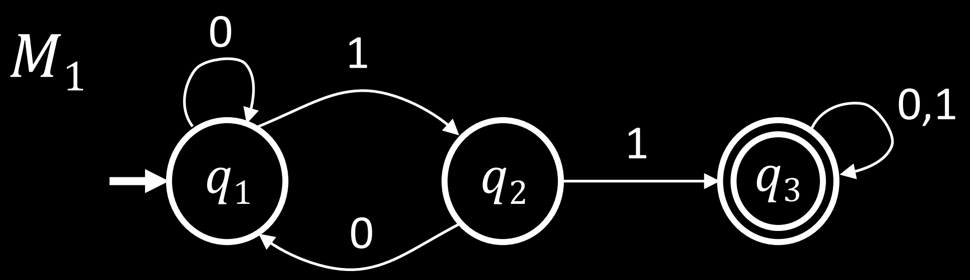
The I-th member of the sequence is obtained by looking at the previous one

Recognizing languages

- $L(M) = \{w | \ M \text{ accepts } w\}$
- $L(M)$ 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
$M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognize A_2

Construct $M = (Q, \Sigma, \delta, q_0, F)$ recognizing $A_1 \cup A_2$

M should accept input w if either M_1 or M_2 accept w.

Check-in 1.1

In the proof, if M_1 and M_2 are finite automata where M_1 has k_1 states and M_2 has k_2 states Then how many states does M have?

(a) $k_1 + k_2$
(b) $(k_1)^2 + (k_2)^2$
(c) $k_1 \times k_2$ ✓

Components of $\boldsymbol{M}$:

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

$q_0 = (q_1, q_2)$

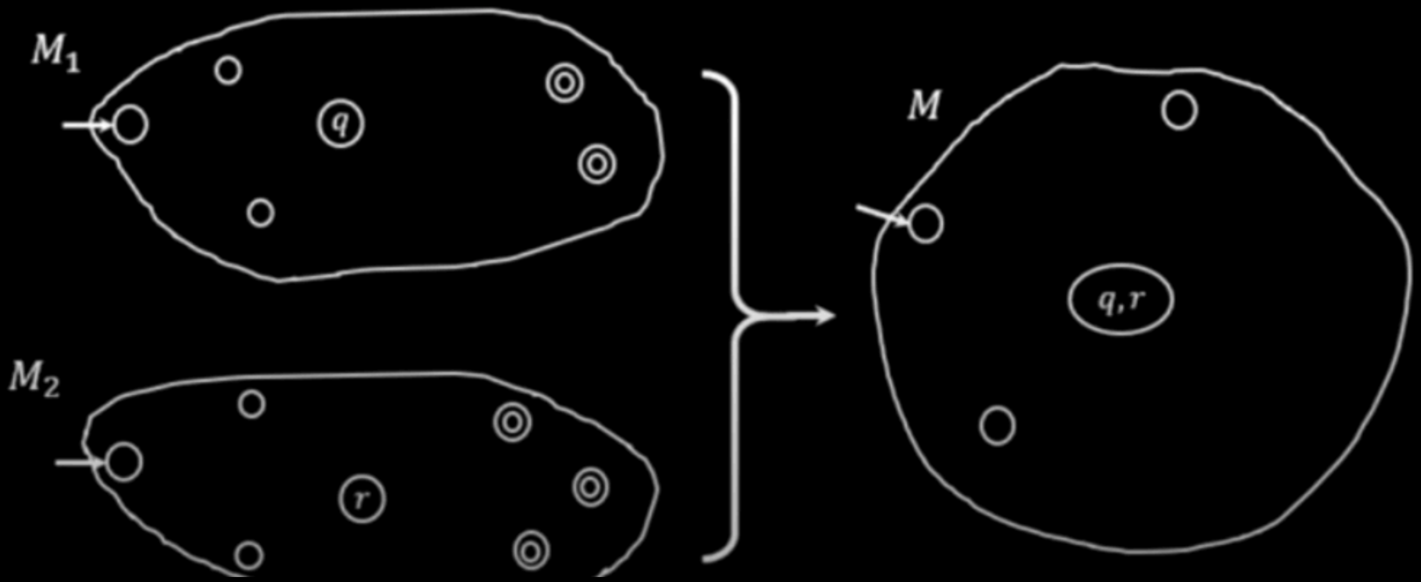
$\delta((q, r), a) = (\delta_1(q, a), \delta_2(r, a))$

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

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

F ={(r1,r2)|r1 ∈ F1 or r2 ∈ F2}.

Check-in 1.1



Components of $\boldsymbol{M}$:

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

$$q_0 = (q_1, q_2)$$

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

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

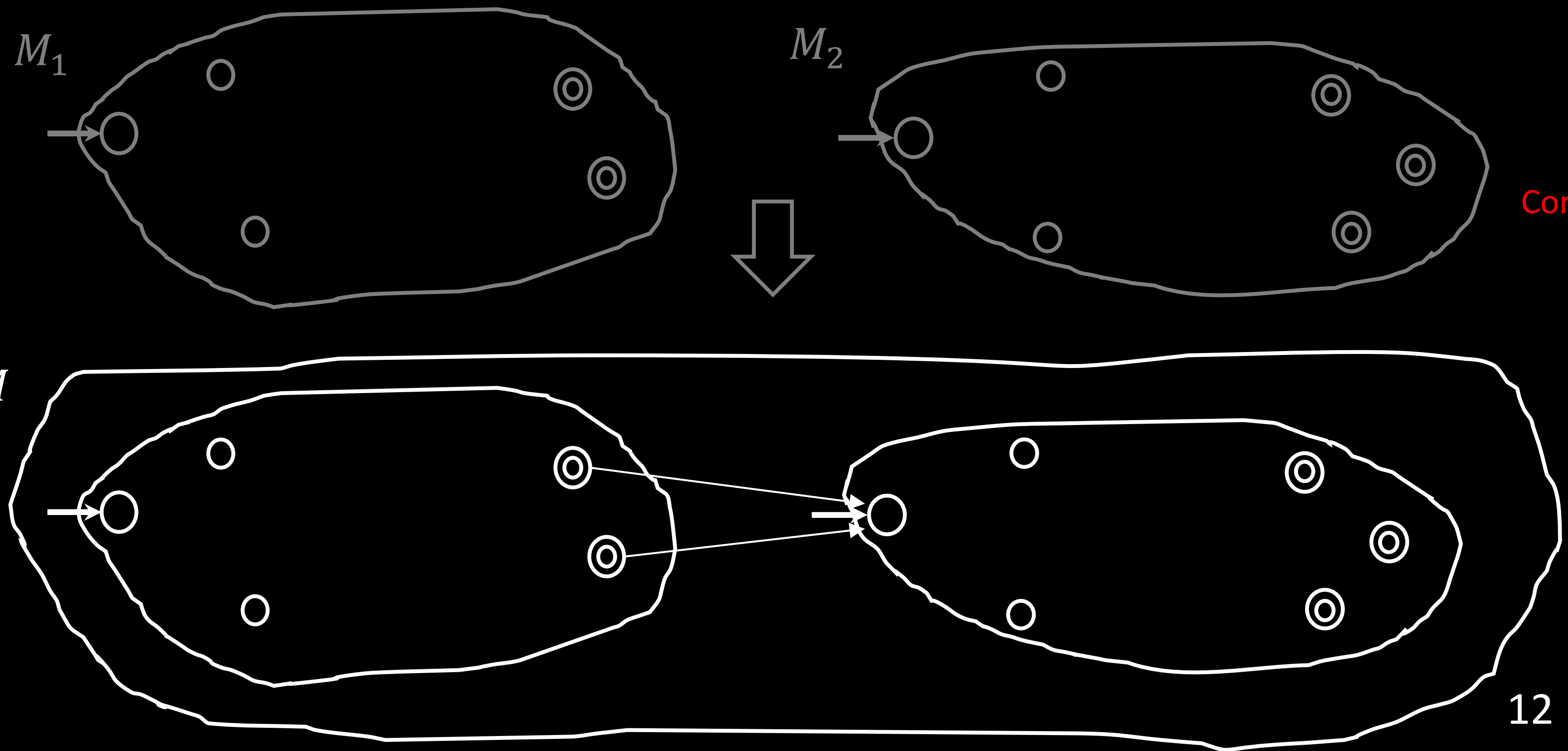
$$F = (F_1 \times Q_2) \cup (Q_1 \times F_2)$$

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
$M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognize A_2

Construct $M = (Q, \Sigma, \delta, q_0, F)$ recognizing A_1A_2



Connect M1 and M2

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…