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 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**
- 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

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

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

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

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

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

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

Finite Automata – Formal Definition

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

- $Q$ finite set of states
- $\Sigma$ finite set of alphabet symbols
- $\delta$ transition function $\delta: Q \times \Sigma \to Q$
- $q_0$ start state
- $F$ set of accept states

**Example:**

$M_1 = (Q, \Sigma, \delta, q_0, F)$

$\delta = \begin{array}{c|cc}
0 & 1 \\
\hline
q_1 & q_1 & q_2 \\
q_2 & q_1 & q_3 \\
q_3 & q_3 & q_3 \\
\end{array}$

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

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

$F = \{q_3\}$
Strings and languages
- A **string** is a finite sequence of symbols in $\Sigma$
- A **language** is a set of strings (finite or infinite)
- The **empty string** $\varepsilon$ is the string of length 0
- The **empty language** $\emptyset$ is the set with no strings

**Defn:** $M$ accepts string $w = w_1w_2 \ldots w_n$ 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 $1 \leq i \leq n$
- $r_n \in F$

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

Let $M_1$ be a finite automaton with states $q_1$, $q_2$, and $q_3$. The transitions are:
- From $q_1$ on 0 to $q_1$.
- From $q_1$ on 1 to $q_2$.
- From $q_2$ on 1 to $q_3$.
- From $q_2$ on 0 to $q_2$.
- From $q_3$ on 0,1 to $q_3$.

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

Therefore $A$ is regular.

More examples:

Let $B = \{ w \mid w \text{ has an even number of } 1s \}$

$B$ is regular (make automaton for practice).

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

$C$ is not regular (we will prove).

Goal: Understand the regular languages
Regular Expressions

Regular operations. Let $A, B$ be languages:

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

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

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

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

Examples:
- $(0 \cup 1)^* = \Sigma^*$ gives all strings over $\Sigma$
- $\Sigma^*1$ gives all strings that end with 1
- $\Sigma^*11\Sigma^* = \text{all strings that contain } 11 = L(M_1)$

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 \( M \):**

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

\( q_0 = (q_1, q_2) \)

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

\( F = F_1 \times F_2 \) **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 \( A_1 \)
\( M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2) \) recognize \( A_2 \)

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

\( M \) 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...