CSCI 361 Lecture 13:

Decidability

Shikha Singh

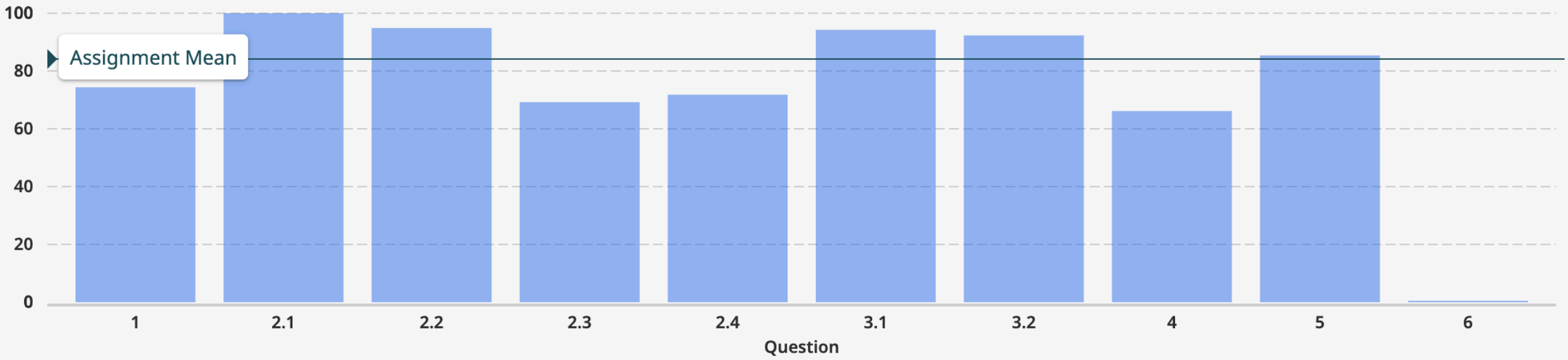
Announcements & Logistics

- Pick up **reading assignment** for Tues Oct 29
- **HW 5** released, due next Wed Oct 30
  - Questions about TM and decidability
- **Graded midterm feedback returned:**
  - Mean: **83.9%**, Median: **86.5%**
- Reminders: CS events
  - Gourd Party today in CS common room 3.30 pm
  - ***Pre-registration Info Session*** (WS & Spring 2025)

Midterm Discussion

- Curved: Across the board +2/52 applied (3.84% points)
- Midterm is just one part of overall assessment:
  - Accounts for **only** 25% of final grade
- Breakdown by question:
  - Q4 was the most challenging
  - 2c and 2d next
  - Did well on others

- Attendance, Readings & Class Participation (10%)
- Assignments (30%)
- Survey Paper and Presentation (10%)
- Midterm Exam (25%)
- Final Exam (25%)



So Far

- Defined Turing machines, created example TMs
- Discussed examples and variants of TM
    - Equivalence with multi-tape TM and non-deterministic TM
- Church-Turing thesis:

Intuitive notion of algorithms	equals	*Turing machine algorithms*

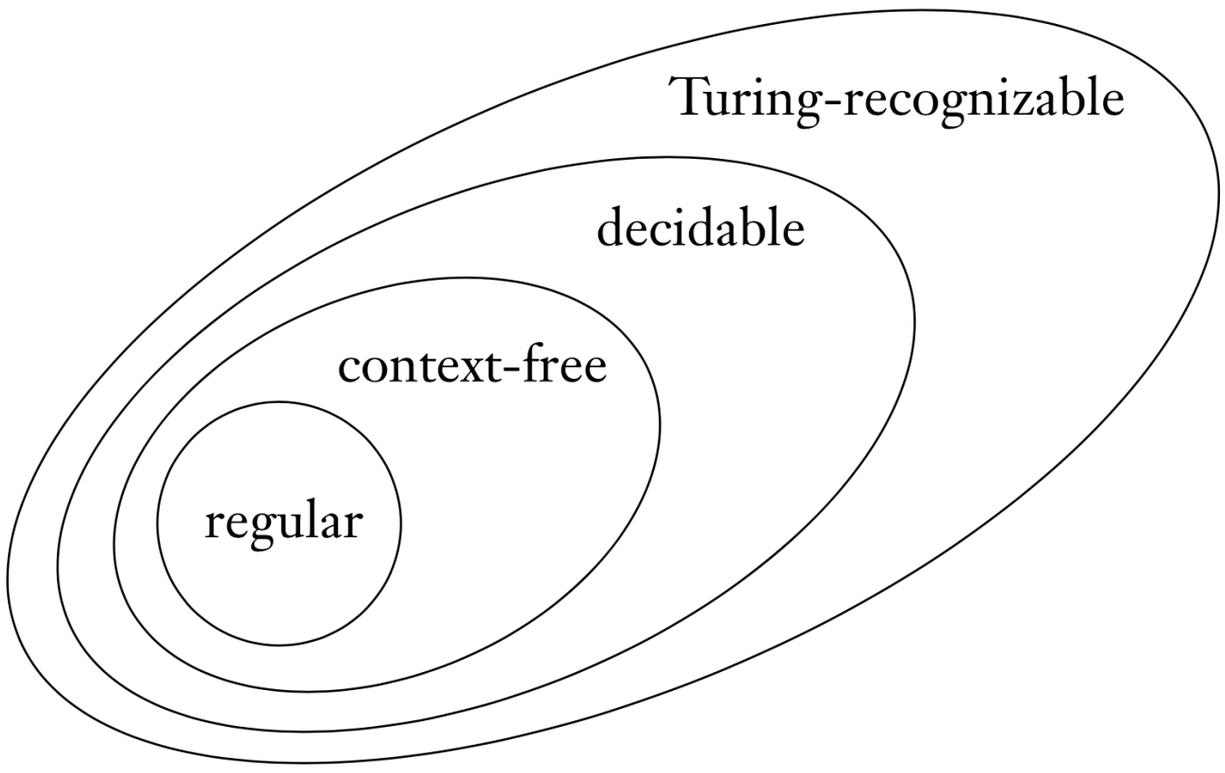
Today and Next Week

- Show TMs are more powerful than CFLs
- Moving towards problems that are unsolvable
  - In our terminology, undecidable languages
- Building towards undecidability:
  - Deciding semantic properties of DFAs/CFGs
  - Introduce idea of reductions

Models of Computation vs Power

- Finite automaton
- Push-down automaton
- Turing machine

- Sequencing, repetition
- Function calls/ simple recursion
- Simple loops as well as multiple recursive calls



Towards Unsolvability

- Moving towards finding out what problems are unsolvable
  - That is, not decidable by Turing machines
- These problems are about the behavior of other machines
- Programming language analog:
  - Once you have a PL, we write an interpreter for the language
  - Python interpreter for Python programs
    - Input is any Python program
- Code is data: can be input to programs/TM

PL View: Syntactic vs Semantic Properties

- Semantic property: properties that do no depending on the "code/ description" of the program but rather depend on the function it computes (about its behavior on inputs)
  - Program accepts some string containing an even # of 1s
- Syntactic property: depend on the particular syntax of the program (the description of the machine itself)
  - Program contains an if-else block
- Software verification is all about making sure programs meet certain specifications/requirements
- PL analog for what's coming: How hard is software verification really?

Code is Data

- Will soon design TMs that take other TMs as input and try answer semantic properties of the input TM
- To get ready, we will first spend some time showing that when the model of computation is restricted (automaton/CFGs), we can actually answer semantic questions about them
- All these questions will be framed as decision problems

Example Problems

- In general, is it useful to be able to check a program's output on a given input: constantly used in verification
- Acceptance problems: is this string accepted by this machine?
  - $A_{\text{DFA}} = \{\langle M, w\rangle \mid M \text{ is a DFA and } w \in L(M)\}$
    - Similar variant for NFA and regular expression
  - $A_{\text{CFG}} = \{\langle G, w\rangle \mid G \text{ is a CFG and } w \in L(G)\}$
    - Similar variant for PDA
  - **(Next class).** $A_{\text{TM}} = \{\langle M, w\rangle \mid T \text{ is a TM and } w \in L(M)\}$

Empty-ness Checks

- Does this program to solve a decision problem ever output 1?
- Emptiness problems:
  - $E_{\text{DFA}} = \{\langle M \rangle \mid M \text{ is a DFA and } L(M) = \emptyset\}$
    - Similar variant for NFA and regular expression
  - $E_{\text{CFG}} = \{\langle G \mid G \text{ is a CFG and } L(G) = \emptyset\}$
    - Similar variant for PDA
  - **(Next class).** $E_{\text{TM}} = \{\langle M, w \rangle \mid T \text{ is a TM and } L(M) = \emptyset\}$

Equivalence Checks

- Given two programs, do their output match on all inputs?
- Equivalence problems:
  - $EQ_{\text{DFA}} = \{\langle M, N\rangle \mid M, N \text{ are DFAs and } L(M) = L(N)\}$
    - Similar variant for NFA/regular expression combinations
  - **(Next class).** $EQ_{\text{CFG}} =$
    $\{\langle G_1, G_2\rangle \mid G_1, G_2 \text{ are CFGs and } L(G_1) = L(G_2)\}$
    - Similar variant for PDA
  - **(Next class).** $EQ_{\text{TM}} =$
    $\{\langle M, N\rangle \mid M, N \text{ are TMs and } L(M) = L(N)\}$

Encodings: Code is Data

- Notice the use of angular brackets to represent the intput
  - E.g. $\langle G \rangle$ is the encoding of G if G is a graph
- All these languages have inputs that are machines
- Need a way to encode general objects such as DFAs, CFGs, TMs into strings over a finite alphabet
- The exact encoding scheme is usually unimportant and rarely explicitly discussed
- In the first example we do, we will discuss the encoding of a DFA
- In later examples, we omit these details and assume the TM is given a reasonable encoding

Switch to Algorithms

- As Sipser puts it, we are at a "turning point" in the theory of computation
- Using the Church-Turing thesis, we will assume that TM can implement the algorithms we have discussed in class
  - E.g., a TM can covert a regular expression to an NFA, a PDA to a CFG, etc.
  - The details of how this conversion is handled is not important

Acceptance Problem for DFAs

Theorem. The language

$A_{\text{DFA}} = \{\langle M, w\rangle \mid M \text{ is a DFA and } w \in L(M)\}$ is decidable.

- Assume the DFA $M = (Q, \Sigma, \delta, q_0, F)$ is encoded as a string consisting of the states, followed by alphabet, followed by a list of triples $(q, x, \delta(q, x))$, start state and final states (each separated by #)
- Simulate the DFA M on w (how would this work?)
- Accept if M accepts, otherwise reject

Using Algorithms and Subroutines

Theorem. The following languages are decidable.

$$A_{\text{NFA}} = \{\langle N, w\rangle \mid M \text{ is a NFA and } w \in L(N)\}$$

$$A_{\text{REX}} = \{\langle R, w\rangle \mid R \text{is a regular expression and } w \in L(R)\}$$

What about CFGs?

Theorem. $A_{\text{CFG}} = \{\langle G, w\rangle \mid G \text{ is a CFG and } w \in L(G)\}$ is decidable.

Question. How do we check if a grammar accepts a given string?

What about CFGs?

Theorem. $A_{\text{CFG}} = \{\langle G, w\rangle \mid G \text{ is a CFG and } w \in L(G)\}$ is decidable.

Proof. NTM that decides this language:

On input $\langle G, w\rangle$ where G is a CFG:

- Starting from the start state, guess a transition from G
- Keep guessing until a string of non-terminals is reached
- If w is the same as string produced, accept; else reject

What about TMs?

Question. Is $A_{\mathrm{TM}} = \{\langle M, w\rangle \mid T \text{ is a TM and } w \in L(M)\}$ decidable?

Emptiness Testing

Theorem. $E_{\text{DFA}} = \{\langle M \rangle \mid M \text{ is a DFA and } L(M) = \varnothing\}$ is decidable.

Question. How do we check if a DFA never accepts a string?

Emptiness Testing for DFAs

Theorem. $E_{\text{DFA}} = \{\langle M \rangle \mid M \text{ is a DFA and } L(M) = \varnothing\}$ is decidable.

Proof. TM that decides this language:

On input $\langle M \rangle$ where M is a DFA:

- Mark the start state of M
- Repeat until no new states are marked:
  - Mark any state that has a transition coming from a marked state
- If no accept state is marked, accept; otherwise reject

Emptiness Testing for CFGs

Theorem. $E_{\text{CFG}} = \{\langle G \mid G \text{ is a CFG and } L(G) = \varnothing\}$ is decidable.

Question. How do we check if a grammar never generates a string?

Emptiness Testing for CFGs

Theorem. $E_{\text{CFG}} = \{\langle G \mid G \text{ is a CFG and } L(G) = \varnothing\}$ is decidable.

Proof. TM that decides this language:

On input $\langle G \rangle$ where G is a CFG:

- Mark all the terminals in G
- Repeat until no new variables get marked:
  - Mark any variable $A \to u_1 u_2 \cdots u_k$ such that each u_i on the RHS is already marked
- If start variable S is marked, accept; otherwise reject

Class Exercise

Problem. Show that

$ALL_{\text{DFA}} = \{\langle M \rangle \mid M \text{ is a DFA and } L(M) = \Sigma^*\}$ is decidable.

$EQ_{\text{DFA}} = \{\langle M, N \rangle \mid M, N \text{ are DFAs and } L(M) = L(N)\}$ is decidable.

Hint. Try to reduce it to one of the problems we have already discussed!

What about CFG Equivalence?

Question. Is $EQ_{\text{CFG}} =$

$\{\langle G_1, G_2\rangle \mid G_1, G_2$ are CFGs and $L(G_1) = L(G_2)\}$ decidable?