Lecture 13:

• More decidable languages
• Universal Turing Machine
• Countability

Reading:
Sipser Ch 4.1, 4.2

Mark Bun
October 25, 2022
Last Time

Church-Turing Thesis

v1: The basic TM (and all equivalent models) capture our intuitive notion of algorithms

v2: Any physically realizable model of computation can be simulated by the basic TM

Decidable languages (from language theory)

\[ A_{DFA} = \{ \langle D, w \rangle \mid \text{DFA } D \text{ accepts input } w \} \], etc.

Today: More decidable languages

What languages are undecidable? How can we prove so?
A “universal” algorithm for recognizing regular languages

\[ A_{\text{DFA}} = \{ \langle D, w \rangle \mid \text{DFA } D \text{ accepts } w \} \]

**Theorem:** \( A_{\text{DFA}} \) is decidable

**Proof:** Define a (high-level) 3-tape TM \( M \) on input \( \langle D, w \rangle \):

1. Check if \( \langle D, w \rangle \) is a valid encoding (reject if not)
2. Simulate \( D \) on \( w \), i.e.,
   - Tape 2: Maintain \( w \) and head location of \( D \)
   - Tape 3: Maintain state of \( D \), update according to \( \delta \)
3. **Accept** if \( D \) ends in an accept state, **reject** otherwise
Other decidable languages

\[ A_{DFA} = \{ \langle D, w \rangle \mid \text{DFA } D \text{ accepts } w \} \]

\[ A_{NFA} = \{ \langle N, w \rangle \mid \text{NFA } N \text{ accepts } w \} \]

\[ A_{REX} = \{ \langle R, w \rangle \mid \text{regular expression } R \text{ generates } w \} \]
NFA Acceptance

Which of the following describes a decider for $A_{NFA} = \{\langle N, w \rangle \mid \text{NFA } N \text{ accepts } w \}$?

a) Using a deterministic TM, simulate $N$ on $w$, always making the first nondeterministic choice at each step. Accept if it accepts, and reject otherwise.

b) Using a deterministic TM, simulate all possible choices of $N$ on $w$ for 1 step of computation, 2 steps of computation, etc. Accept whenever some simulation accepts.

c) Use the subset construction to convert $N$ to an equivalent DFA $M$. Simulate $M$ on $w$, accept if it accepts, and reject otherwise.
Regular Languages are Decidable

Theorem: Every regular language $L$ is decidable

Proof 1: If $L$ is regular, it is recognized by a DFA $D$. Convert this DFA to a TM $M$. Then $M$ decides $L$.

Proof 2: If $L$ is regular, it is recognized by a DFA $D$. The following TM $M_D$ decides $L$.

On input $w$: [Is $w \in L$ or not?]

1. Run the decider for $A_{DFA}$ on input $\langle D, w \rangle$
2. Accept if the decider accepts; reject otherwise

Correctness:

If $w \in L$ then $\langle D, w \rangle \in A_{DFA} \Rightarrow$ decider accepts
If $w \notin L$ then $\langle D, w \rangle \notin A_{DFA} \Rightarrow$ decider rejects.
Classes of Languages

- Regular
- Decidable
- Recognizable
More Decidable Languages: Emptiness Testing

Theorem: \( E_{DFA} = \{ \langle D \rangle \mid D \text{ is a DFA such that } L(D) = \emptyset \} \) is decidable

Proof: The following TM decides \( E_{DFA} \)

On input \( \langle D \rangle \), where \( D \) is a DFA with \( k \) states:

1. Perform \( k \) steps of breadth-first search on state diagram of \( D \) to determine if an accept state is reachable from the start state

2. **Reject** if a DFA accept state is reachable; accept otherwise

Correctness analysis:

If \( \langle D \rangle \in E_{DFA} \), then \( \forall w, \emptyset \) does not reach an accept state by reading input \( w \) in \( k \) steps

If \( \langle D \rangle \notin E_{DFA} \), then \( \exists \) a path from start to accept of length \( \leq k \) in TM

So BFS will find it \( \Rightarrow \) rejects
$E_{DFA}$ Example

$$D =$$

- **Example:**
  - Example with a diagram of a DFA.
  - Example text explaining the DFA's behavior.
  - Example text highlighting states and transitions.

- **Explanation:**
  - Diagram showing states and transitions.
  - States and transitions labeled with symbols.
  - Example text describing the DFA's states and transitions.

- **Conclusion:**
  - No accepting states.
  - DFA not reachable.
  - Conclusion text summarizing the DFA's properties.

- **Diagram Elements:**
  - States: $q_0, q_1, q_2, q_3, q_4, q_5$
  - Transitions:
    - $q_0 \rightarrow 0, 0, 1$
    - $q_1 \rightarrow 1, 0, 1$
    - $q_2 \rightarrow 0$
    - $q_3 \rightarrow 1$
    - $q_4 \rightarrow 0, 0, 1$
    - $q_5 \rightarrow 0$

- **Example Text:**
  - Example text explaining the DFA's behavior.
  - Example text highlighting states and transitions.

- **DFA States:**
  - Accepting states:
    - No accepting states.

- **DFA Transitions:**
  - Transitions:
    - $q_0 \rightarrow 0, 0, 1$
    - $q_1 \rightarrow 1, 0, 1$
    - $q_2 \rightarrow 0$
    - $q_3 \rightarrow 1$
    - $q_4 \rightarrow 0, 0, 1$
    - $q_5 \rightarrow 0$

- **DFA Conclusion:**
  - Conclusion text summarizing the DFA's properties.

- **DFA Implementation:**
  - Code or design details for implementing the DFA.

- **DFA Use Cases:**
  - Use cases for the DFA in real-world applications.

- **DFA Alternatives:**
  - Alternatives to the DFA for solving similar problems.

- **DFA Benefits:**
  - Benefits of using the DFA in various scenarios.

- **DFA Limitations:**
  - Limitations of the DFA in certain situations.

- **DFA Advantages:**
  - Advantages of the DFA over other models.

- **DFA Challenges:**
  - Challenges in implementing or using the DFA.

- **DFA Improvements:**
  - Improvements to the DFA for enhanced performance.

- **DFA Applications:**
  - Applications of the DFA in various fields.

- **DFA Analysis:**
  - Analysis of the DFA's behavior and performance.

- **DFA Evaluation:**
  - Evaluation of the DFA against other models.

- **DFA Comparison:**
  - Comparison of the DFA with other models in terms of efficiency, performance, and usability.

- **DFA Comparison:**
  - Comparison of the DFA with other models in terms of efficiency, performance, and usability.

- **DFA Comparison:**
  - Comparison of the DFA with other models in terms of efficiency, performance, and usability.
New Deciders from Old: Equality Testing

\( EQ_{DFA} = \{ \langle D_1, D_2 \rangle \mid D_1, D_2 \text{ are DFAs and } L(D_1) = L(D_2) \} \)

**Theorem:** \( EQ_{DFA} \) is decidable

**Proof:** The following TM decides \( EQ_{DFA} \)

On input \( \langle D_1, D_2 \rangle \), where \( \langle D_1, D_2 \rangle \) are DFAs:

1. Construct DFA \( D \) recognizing the **symmetric difference**
   \( L(D_1) \triangle L(D_2) = \{ w \mid w \text{ is in exactly one of } L(D_1) \text{ or } L(D_2) \} \)

2. Run the decider for \( E_{DFA} \) on \( \langle D \rangle \) and return its output

Analysis:

1) \( \langle D_1, D_2 \rangle \in EQ_{DFA} \Rightarrow L(D_1) = L(D_2) \Rightarrow L(D_1) \triangle L(D_2) = \emptyset \)
   \( \Rightarrow L(D) = \emptyset \Rightarrow \text{decider accept} \)

2) \( \langle D_1, D_2 \rangle \notin EQ_{DFA} \Rightarrow L(D_1) \neq L(D_2) \Rightarrow L(D_1) \triangle L(D_2) \neq \emptyset \)
   \( \Rightarrow L(D) \neq \emptyset \Rightarrow \text{decider reject} \)
Symmetric Difference

\[ A \triangle B = \{ w \mid w \in A \text{ or } w \in B \text{ but not both} \} \]

\[ A \triangle B = (A \setminus B) \cup (B \setminus A) = (A \cap \overline{B}) \cup (\overline{A} \cap B) \]

We showed: Given \( D_1, D_2 \), can construct NFAs recognizing 
\[ L(D_1), \ L(D_1) \cap L(D_2), \ L(D_2) \cup L(D_1) \]

Moreover, can be done on a TM!
Universal Turing Machine
Meta-Computational Languages

\[ A_{\text{DFA}} = \{ \langle D, w \rangle | \text{DFA } D \text{ accepts } w \} \]
\[ A_{\text{TM}} = \{ \langle M, w \rangle | \text{TM } M \text{ accepts } w \} \]

\[ E_{\text{DFA}} = \{ \langle D \rangle | \text{DFA } D \text{ recognizes the empty language } \emptyset \} \]
\[ E_{\text{TM}} = \{ \langle M \rangle | \text{TM } M \text{ recognizes the empty language } \emptyset \} \]

\[ E_{Q_{\text{DFA}}} = \{ \langle D_1, D_2 \rangle | D_1 \text{ and } D_2 \text{ are DFAs, } L(D_1) = L(D_2) \} \]
\[ E_{Q_{\text{TM}}} = \{ \langle M_1, M_2 \rangle | M_1 \text{ and } M_2 \text{ are TMs, } L(M_1) = L(M_2) \} \]
The Universal Turing Machine

\[ A_{TM} = \{ \langle M, w \rangle \mid M \text{ is a TM that accepts input } w \} \]

**Theorem:** \( A_{TM} \) is Turing-recognizable

The following “Universal TM” \( U \) recognizes \( A_{TM} \)

On input \( \langle M, w \rangle \):
1. Simulate running \( M \) on input \( w \)
2. If \( M \) accepts, accept. If \( M \) rejects, reject.

Correctness:
- If \( \langle M, w \rangle \in A_{TM} \): In simulation, \( M \) accepts \( w \), so \( U \) accepts.
- If \( \langle M, w \rangle \notin A_{TM} \) then \( M \) does not accept \( w \).
  - Case 1: \( M \) rejects \( w \) \( \Rightarrow \) simulation rejects \( \Rightarrow U \) rejects.
  - Case 2: \( M \) loops on \( w \) \( \Rightarrow \) simulation loops \( \Rightarrow U \) loops.
Universal TM and $A_{TM}$

Why is the Universal TM not a decider for $A_{TM}$?

The following “Universal TM” $U$ recognizes $A_{TM}$

On input $\langle M, w \rangle$:
1. Simulate running $M$ on input $w$
2. If $M$ accepts, accept. If $M$ rejects, reject.

a) It may reject inputs $\langle M, w \rangle$ where $M$ accepts $w$
b) It may accept inputs $\langle M, w \rangle$ where $M$ rejects $w$
\[\text{c)}\] It may loop on inputs $\langle M, w \rangle$ where $M$ loops on $w$
d) It may loop on inputs $\langle M, w \rangle$ where $M$ accepts $w$
More on the Universal TM

"It is possible to invent a single machine which can be used to compute any computable sequence. If this machine $U$ is supplied with a tape on the beginning of which is written the S.D ["standard description"] of some computing machine $M$, then $U$ will compute the same sequence as $M$.”

- Turing, “On Computable Numbers...” 1936

• Foreshadowed general-purpose programmable computers
• No need for specialized hardware: Virtual machines as software

Harvard architecture: Separate instruction and data pathways

von Neumann architecture: Programs can be treated as data
Undecidability

$A_{TM}$ is Turing-recognizable via the Universal TM

...but it turns out $A_{TM}$ (and $E_{TM}, EQ_{TM}$) is **undecidable**

i.e., computers cannot solve these problems no matter how much time they are given

How can we prove this?
First, a mathematical interlude...
Countability and Diagonalization
What’s your intuition?

Which of the following sets is the “biggest”?

a) The natural numbers: \( \mathbb{N} = \{1, 2, 3, \ldots \} \)

b) The even numbers: \( E = \{2, 4, 6, \ldots \} \)

c) The positive powers of 2: \( POW2 = \{2, 4, 8, 16, \ldots \} \)

\(\text{d)}\) They all have the same size
Set Theory Review

A function $f: A \rightarrow B$ is

- **1-to-1 (injective)** if $f(a) \neq f(a')$ for all $a \neq a'$

- **onto (surjective)** if for all $b \in B$, there exists $a \in A$ such that $f(a) = b$

- **a correspondence (bijective)** if it is 1-to-1 and onto, i.e., every $b \in B$ has a unique $a \in A$ with $f(a) = b$
How can we compare sizes of infinite sets?

Definition: Two sets have the same size if there is a bijection between them.

A set is countable if

- it is a finite set, or
- it has the same size as \( \mathbb{N} \), the set of natural numbers

i.e. \( \exists \text{ a bijection } f : \mathbb{N} \rightarrow A \)
Examples of countable sets

- $\emptyset$
- $\{0,1\}$
- $\{0, 1, 2, \ldots, 8675309\}$

- $E = \{2, 4, 6, 8, \ldots\}$ $f : \mathbb{N} \to E$ $f(i) = 2i$
- $SQUARES = \{1, 4, 9, 16, 25, \ldots\}$ $f(i) = i^2$
- $POW2 = \{2, 4, 8, 16, 32, \ldots\}$ $f(i) = 2^i$

$|E| = |SQUARES| = |POW2| = |\mathbb{N}|$. 