LECTURE 14

Last time
• Turing Machine Variants
• Church-Turing Thesis

Today
• Universal TM
• Decidable languages
• Designing deciders

Homework 5 due
Homework 6 out
The Church-Turing Thesis (1936)

L is recognized by a program for some computer*

\[ \uparrow \]

L is recognized by a TM

History

- 23 Hilbert’s problems (1900)
  - stated at International Congress of Mathematicians
  - 10th problem: Give a procedure for determining if a polynomial in \( k \) variables has an integral root.

* The computer must be “reasonable”
I-clicker problem (frequency: AC)

The language corresponding to Hilbert’s 10th problem is

A. not Turing-recongnizable
B. Turing-recongnizable, but not decidable
C. decidable
D. regular
E. More than one choice above works.
• Since TMs and programming languages are equivalent, we can think of TMs as programs.
• Since programs are strings, we can consider languages whose elements are programs.
Can we encode a Turing Machine as a string of 0s and 1s?

- $\langle O \rangle$ denotes an encoding of object $O$ as a string

states $Q = \{0, 1, \ldots, n-1\}$

tape symbols $\Gamma = \{0, 1, \ldots, m-1\}$
(first $k$ are input symbols)

$0^n10^m10^k10^s10^t10^r10^u1\ldots \langle \delta \rangle$

$\delta : \langle (p, a), (q, b, L) \rangle = 0^p10^a10^q10^b10$
Since TMs and programming languages are equivalent, we can think of TMs as programs.

Since programs are strings, we can consider languages whose elements are programs.

\( \langle M \rangle \) denotes an encoding of a TM M as a string

**Theorem.** We can make a Universal TM, a TM that takes any TM description \( \langle M \rangle \) and any string \( w \) as input and simulates the computation of \( M \) on \( w \).

\[
\langle w \rangle = \langle w_1, \ldots, w_n \rangle = \langle w_1 \rangle 10 \langle w_2 \rangle 1 \ldots 0 \langle w_n \rangle
\]
Similarly, we can encode DFAs, NFAs, regular expressions, PDAs, CFGs, etc into strings of 0s and 1s.

We can define the following languages:

$$A_{DFA} = \{ \langle D,w \rangle \mid D \text{ is a DFA that accepts string } w \}$$

$$A_{NFA} = \{ \langle N,w \rangle \mid N \text{ is an NFA that accepts string } w \}$$

$$A_{CFG} = \{ \langle G,w \rangle \mid G \text{ is a CFG that generates string } w \}$$
**Theorem.** $A_{DFA}$ is decidable.

**Proof:** The following TM M decides $A_{DFA}$.

$M = \text{`` On input } \langle D, w \rangle, \text{ where } D \text{ is a DFA and } w \text{ is a string:} \quad$

1. Check if input (to M) is legal, reject if not.  
   (This step is assumed to be the first step of every algorithm.)

2. Simulate $D$ on $w$.

3. **Accept** if $D$ ends in an accept state. O.w. reject.”

**Corollary.** $A_{NFA}$ is decidable.

(1. Convert input NFA $N$ to an equivalent DFA $D$.)

Sofya Raskhodnikova; based on slides by Nick Hopper
Theorem. $A_{\text{CFG}}$ is decidable.

$A_{\text{CFG}} = \{ \langle G, w \rangle \mid G \text{ is a CFG that generates string } w \}$
Chomsky Normal Form for CFGs

- Can have a rule $S \rightarrow \varepsilon$.
- All remaining rules are of the form $A \rightarrow BC \quad A, B, C \in V$
  $A \rightarrow a \quad a \in \Sigma$
- Cannot have $S$ on the RHS of any rule.

**Lemma.** Any CFG can be converted into an equivalent CFG in Chomsky normal form. *(Proof in Sipser.)*

**Lemma.** If $G$ is in Chomsky normal form, any derivation of string $w$ of length $n$ in $G$ has $2n - 1$ steps.
Lemma. If $G$ is in Chomsky normal form, any derivation of string $w$ of length $n$ in $G$ has $2n - 1$ steps.

Proof idea:

• Only rules of the form $A \rightarrow BC$ increase the number of symbols: need to apply rules of this form $n - 1$ times.

• Only rules of the form $A \rightarrow a$ replace variables with terminals: need to apply rules of this form $n$ times.
Theorem. $A_{CFG}$ is decidable.

$A_{CFG} = \{ \langle G, w \rangle \mid G \text{ is a CFG that generates string } w \}$

Proof: The following TM $M$ decides $A_{CFG}$.

$M =$ "On input $\langle G, w \rangle$, where $G$ is a CFG and $w$ is a string:

1. Convert $G$ to Chomsky normal form.
2. Let $n = |w|$.
3. Test all derivations with $2n - 1$ steps.
4. Accept if any derived $w$. O.w. reject.”"
Examples of decidable languages

$$A_{\text{DFA}} = \{ \langle D, w \rangle \mid D \text{ is a DFA that accepts string } w \}$$

$$A_{\text{NFA}} = \{ \langle N, w \rangle \mid N \text{ is an NFA that accepts string } w \}$$

$$A_{\text{CFG}} = \{ \langle G, w \rangle \mid G \text{ is a CFG that generates string } w \}$$
More decidable languages

\[ E_{\text{DFA}} = \{ \langle D \rangle \mid D \text{ is a DFA that recognizes the empty language} \} \]

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

\[ E_{\text{CFG}} = \{ \langle G \rangle \mid G \text{ is a CFG that generates the empty language} \} \]
Theorem. \( E_{\text{DFA}} \) is decidable.

\[ E_{\text{DFA}} = \{ \langle D \rangle \mid D \text{ is a DFA that recognizes } \emptyset \}. \]

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

\( M = \)`` On input \( \langle D \rangle \), where \( D \) is a DFA:

1. Use BFS to determine if an accepting state of \( D \) is reachable from from its start state.
2. Accept if not. O.w. reject.””
**Theorem.** $\text{EQ}_{\text{DFA}}$ is decidable.

$\text{EQ}_{\text{DFA}} = \{ \langle D_1, D_2 \rangle \mid D_1, D_2 \text{ are DFAs } \& \ L(D_1) = L(D_2) \}$

**Proof:** The following TM $M$ decides $\text{EQ}_{\text{DFA}}$.

$M = \text{``On input } \langle D_1, D_2 \rangle, \text{ where } D_1, D_2 \text{ are DFAs:}$$

1. Construct a DFA $D$ that recognizes the set difference of $L(D_1)$ and $L(D_2)$.
2. Run the decider for $\text{E}_{\text{DFA}}$ on $<D>$.
3. If it accepts, accept. O.w. reject.”
Theorem. $E_{CFG}$ is decidable.

$E_{CFG} = \{ \langle G \rangle \mid G \text{ is a CFG that recognizes } \emptyset. \}$

Proof: The following TM $M$ decides $E_{CFG}$.

$M = \text{``On input } \langle G \rangle, \text{ where } G \text{ is a CFG:}$$
1. Mark all terminals in $G$.
2. Repeat until no new variable is marked:
3. Mark any variable $A$ where $G$ has a rule $A \rightarrow \cdots$ and each variable/terminal on the RHS is already marked.
4. Accept if the start variable is unmarked. O.w. reject."
Exercises

• Prove that the following language is decidable:

\[ R_{DFA} = \{ \langle D, w \rangle \mid D \text{ is a DFA that rejects string } w \} \]

• Formulate the following problem as a language and prove that it is decidable:

Given a PDA and a string, determine if the PDA accepts the string.

\[ A_{PDA} = \{ \langle P, w \rangle \mid P \text{ is a PDA that accepts string } w \} \]

Can a TM just simulate \( P \) on \( w \), accept if it accepts and reject o.w.?
A decider for $A_{PDA}$ can, on input $<P, w>$

A. simulate $P$ on $w$, accept if it accepts and reject o.w.

B. convert $P$ to an equivalent CFG $G$ and then run a decider for $A_{CFG}$, accept if it accepts and reject o.w.

C. convert $P$ to an equivalent CFG $G$ and then run a decider for $A_{CFG}$, accept if it rejects and accept o.w.

D. None of the above.

E. More than one choice above works.