CS170 Computation Theory

Lecture 8

September 28, 2023

Megumi Ando

Tufts University | School of Engineering Computer Science

Review of Last Lecture

- Notation for Encodings and TMs
- Decision procedures for DFAs

Today's Topics

- ~~Notation for Encodings and TMs~~
- ~~Decision procedures for DFAs~~

- Decision procedures for CFGs
- The reducibility method

Recall: T-recognizability and Decidability

Definition (p. 170): Let M be a TM. Let $A = \{w \mid M \text{ accepts } w\}$. Then, A is the language recognized by M, i.e., $A = L(M)$.

Definition (p. 170): A language is T-recognizable if there is a TM that recognizes it.

Definition (p. 170): A TM M is a decider if it halts on all inputs.

Definition (p. 170): A language A is T-decidable (or just decidable) if $A = L(M)$ for some TM decider M.

Acceptance Problem for CFGs

Theorem (p.198): The language $A_{\text{CFG}} = \{\langle G, w \rangle \mid G \text{ is a CFG and } w \in L(G)\}$ is decidable.

Proof Attempt 1: Let $D'_{A_{\text{CFG}}}$ be the following TM.

$D'_{A_{\text{CFG}}}$ = "On input $\langle B, w \rangle$ where B is a CFG and w is a string,

1. Convert B to an equivalent PDA B' using the procedure from Lecture 4.
2. Simulate the computation of B' on w.
3. If B' ends in an accept state, *accept*. Otherwise, *reject.*"

$D'_{A_{\text{CFG}}}$ decides A_{CFG}.

Acceptance Problem for CFGs

Theorem (p.198): The language $A_{\text{CFG}} = \{\langle G, w\rangle \mid G \text{ is a CFG and } w \in L(G)\}$ is decidable.

Proof Attempt 1: Let $D'_{A_{\text{CFG}}}$ be the following TM.

$D'_{A_{\text{CFG}}}$ = "On input $\langle B, w\rangle$ where B is a CFG and w is a string,

1. Convert B to an equivalent PDA B' using the procedure from Lecture 4.
2. Simulate the computation of B' on w.
3. If B' ends in an accept state, *accept*. Otherwise, *reject.*"

$D'_{A_{\text{CFG}}}$ decides A_{CFG}.

(ϵ-transitions make argument tricky.)

Chomsky Normal Form

Definition (p.109): A CFG is in Chomsky normal form if every rule is:

$$T \to UV$$

$$T \to a$$

where U and V are not the start variable S. (With the one exception: $S \to \epsilon$.)

Lemma 1: Every CFL can be generated by a CFG in Chomsky normal form.

Proof in Sipser Book (p. 109).

Lemma 2

Lemma 2

Lemma 2: If G is a CFG in Chomsky normal form and $w \in L(G)$, then every derivation of w has $2|w| - 1$ steps.

Lemma 2

Lemma 2: If G is a CFG in Chomsky normal form and $w \in L(G)$, then every derivation of w has $2|w| - 1$ steps.

Proof:

(1) Let $w = w_1 w_2 \ldots w_n$ be any string in $L(G)$.

Lemma 2

Lemma 2: If G is a CFG in Chomsky normal form and $w \in L(G)$, then every derivation of w has $2|w| - 1$ steps.

Proof:

(1) Let $w = w_1 w_2 \ldots w_n$ be any string in $L(G)$.

(2) It takes $n - 1$ steps to "expand" the start variable into n variables $V_1 V_2 \ldots V_n$.

Lemma 2

Lemma 2: If G is a CFG in Chomsky normal form and $w \in L(G)$, then every derivation of w has $2|w| - 1$ steps.

Proof:

(1) Let $w = w_1 w_2 \ldots w_n$ be any string in $L(G)$.

(2) It takes $n - 1$ steps to “expand” the start variable into n variables $V_1 V_2 \ldots V_n$.

(3) For each variable V_i, it takes one step to substitute it for a terminal symbol.

Q.E.D.

Acceptance Problem for CFGs (con't)

Theorem (p.194): The language $A_{\text{CFG}} = \{\langle G, w\rangle \mid G$ is a CFG and $w \in L(G)\}$ is decidable.

Acceptance Problem for CFGs (con't)

Theorem (p.194): The language $A_{\text{CFG}} = \{\langle G, w\rangle \mid G$ is a CFG and $w \in L(G)\}$ is decidable.

Proof: Let $D_{A_{\text{CFG}}}$ be the following TM.

Acceptance Problem for CFGs (con't)

Theorem (p.194): The language $A_{\text{CFG}} = \{\langle G, w\rangle \mid G$ is a CFG and $w \in L(G)\}$ is decidable.

Proof: Let $D_{A_{\text{CFG}}}$ be the following TM.

$D_{A_{\text{CFG}}}$= "On input $\langle B, w\rangle$ where B is a CFG and w is a string,

Acceptance Problem for CFGs (con't)

Theorem (p.194): The language $A_{\text{CFG}} = \{\langle G, w\rangle \mid G$ is a CFG and $w \in L(G)\}$ is decidable.

Proof: Let $D_{A_{\text{CFG}}}$ be the following TM.

$D_{A_{\text{CFG}}}$= "On input $\langle B, w\rangle$ where B is a CFG and w is a string,

1. Convert B to an equivalent CFG B' in Chomsky normal form.

Acceptance Problem for CFGs (con't)

Theorem (p.194): The language $A_{\text{CFG}} = \{\langle G, w\rangle \,|\, G$ is a CFG and $w \in L(G)\}$ is decidable.

Proof: Let $D_{A_{\text{CFG}}}$ be the following TM.

$D_{A_{\text{CFG}}}$= "On input $\langle B, w\rangle$ where B is a CFG and w is a string,

1. Convert B to an equivalent CFG B' in Chomsky normal form.
2. Try all derivations of length $2\,|\,w\,|\,-1$.

Acceptance Problem for CFGs (con't)

Theorem (p.194): The language $A_{\text{CFG}} = \{\langle G, w\rangle \mid G \text{ is a CFG and } w \in L(G)\}$ is decidable.

Proof: Let $D_{A_{\text{CFG}}}$ be the following TM.

$D_{A_{\text{CFG}}}$= "On input $\langle B, w\rangle$ where B is a CFG and w is a string,

1. Convert B to an equivalent CFG B' in Chomsky normal form.
2. Try all derivations of length $2|w| - 1$.
3. If any derivation generates w, *accept*. Otherwise, *reject.*"

Acceptance Problem for CFGs (con't)

Theorem (p.194): The language $A_{\text{CFG}} = \{\langle G, w \rangle \mid G \text{ is a CFG and } w \in L(G)\}$ is decidable.

Proof: Let $D_{A_{\text{CFG}}}$ be the following TM.

$D_{A_{\text{CFG}}}$= "On input $\langle B, w \rangle$ where B is a CFG and w is a string,

1. Convert B to an equivalent CFG B' in Chomsky normal form.
2. Try all derivations of length $2|w| - 1$.
3. If any derivation generates w, *accept*. Otherwise, *reject."*

$D_{A_{\text{CFG}}}$ decides A_{CFG}.

Acceptance Problem for CFGs (con't)

Theorem (p.194): The language $A_{\text{CFG}} = \{\langle G, w\rangle \mid G \text{ is a CFG and } w \in L(G)\}$ is decidable.

Proof: Let $D_{A_{\text{CFG}}}$ be the following TM.

$D_{A_{\text{CFG}}}$ = "On input $\langle B, w\rangle$ where B is a CFG and w is a string,

1. Convert B to an equivalent CFG B' in Chomsky normal form.
2. Try all derivations of length $2|w| - 1$.
3. If any derivation generates w, *accept*. Otherwise, *reject.*"

$D_{A_{\text{CFG}}}$ decides A_{CFG}.

Each derivation takes $O(|w|)$ sub-steps.

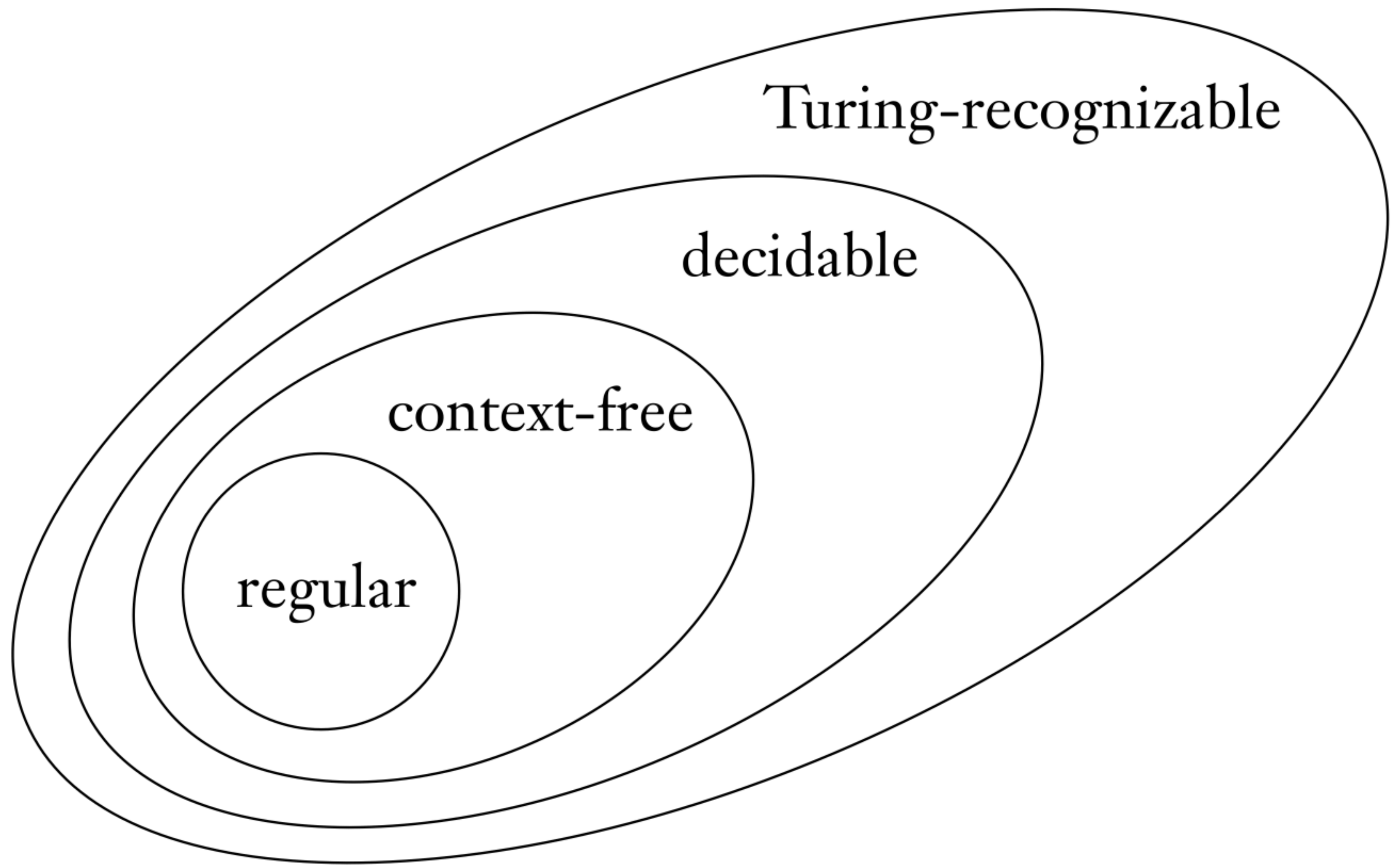
Q.E.D.

Check-In 1 (Break)

Prove that every context-free language is decidable.

(*Hint*: Consider what we just saw in the last slide: A_{CFG}.)

Check-In 1 (Break)



Emptiness Problem for CFGs

Theorem (p.199): The language $E_{\text{CFG}} = \{\langle B \rangle \mid B \text{ is a CFG and } L(B) = \emptyset\}$ is decidable.

Emptiness Problem for CFGs

Theorem (p.199): The language $E_{\text{CFG}} = \{\langle B \rangle \mid B \text{ is a CFG and } L(B) = \emptyset\}$ is decidable.

Proof: Let $D_{E_{\text{CFG}}}$ be the following TM.

Emptiness Problem for CFGs

Theorem (p.199): The language $E_{\text{CFG}} = \{\langle B \rangle \mid B \text{ is a CFG and } L(B) = \emptyset\}$ is decidable.

Proof: Let $D_{E_{\text{CFG}}}$ be the following TM.

$D_{E_{\text{CFG}}}$= "On input $\langle B \rangle$ where B is a CFG,

Emptiness Problem for CFGs

Theorem (p.199): The language $E_{\text{CFG}} = \{\langle B \rangle \mid B \text{ is a CFG and } L(B) = \emptyset\}$ is decidable.

Proof: Let $D_{E_{\text{CFG}}}$ be the following TM.

$D_{E_{\text{CFG}}}$= "On input $\langle B \rangle$ where B is a CFG,

1. Mark all terminals.

Emptiness Problem for CFGs

Theorem (p.199): The language $E_{\text{CFG}} = \{\langle B \rangle \mid B \text{ is a CFG and } L(B) = \varnothing\}$ is decidable.

Proof: Let $D_{E_{\text{CFG}}}$ be the following TM.

$D_{E_{\text{CFG}}}$= "On input $\langle B \rangle$ where B is a CFG,

1. Mark all terminals.
2. Repeat until no new variables are marked: mark all occurrences of variable V if $V \to u_1 u_2 \ldots u_k$ is a rule, and all u_i's are marked.

Emptiness Problem for CFGs

Theorem (p.199): The language $E_{\text{CFG}} = \{\langle B \rangle \mid B \text{ is a CFG and } L(B) = \emptyset\}$ is decidable.

Proof: Let $D_{E_{\text{CFG}}}$ be the following TM.

$D_{E_{\text{CFG}}}$= "On input $\langle B \rangle$ where B is a CFG,

1. Mark all terminals.
2. Repeat until no new variables are marked: mark all occurrences of variable V if $V \to u_1 u_2 \ldots u_k$ is a rule, and all u_i's are marked.
3. If start variable is not marked, accept. Else, reject."

Emptiness Problem for CFGs

Theorem (p.199): The language $E_{\text{CFG}} = \{\langle B \rangle \mid B \text{ is a CFG and } L(B) = \emptyset\}$ is decidable.

Proof: Let $D_{E_{\text{CFG}}}$ be the following TM.

$D_{E_{\text{CFG}}}$= "On input $\langle B \rangle$ where B is a CFG,

1. Mark all terminals.
2. Repeat until no new variables are marked: mark all occurrences of variable V if $V \to u_1 u_2 \ldots u_k$ is a rule, and all u_i's are marked.
3. If start variable is not marked, accept. Else, reject."

$D_{E_{\text{CFG}}}$ decides E_{CFG}.

Emptiness Problem for CFGs

Theorem (p.199): The language $E_{\text{CFG}} = \{\langle B \rangle \mid B \text{ is a CFG and } L(B) = \varnothing\}$ is decidable.

Proof: Let $D_{E_{\text{CFG}}}$ be the following TM.

$D_{E_{\text{CFG}}}$= “On input $\langle B \rangle$ where B is a CFG,

1. Mark all terminals.
2. Repeat until no new variables are marked: mark all occurrences of variable V if $V \to u_1 u_2 \ldots u_k$ is a rule, and all u_i’s are marked.
3. If start variable is not marked, accept. Else, reject.”

$D_{E_{\text{CFG}}}$ decides E_{CFG}.

$D_{E_{\text{CFG}}}$ always halts within $O(|V|)$ sub-steps.

Q.E.D.

Check In 2 (Break)

Fake Theorem: The language
$ALL_{\text{CFG}} = \{\langle G \rangle \,|\, G$ is a CFG such that $L(G) = \Sigma^*\}$ is decidable.

Fake Proof: Let $D_{ALL_{\text{CGF}}}$ be the following TM.

$D_{ALL_{\text{CGF}}}$= "On input $\langle G \rangle$ where G is a CFG,

1. Determine the CFG $\overline{G}$ that generates the language $\overline{L(G)}$.
2. Run $D_{E_{\text{CFG}}}$ (from previous slide) on $\langle \overline{G} \rangle$.
3. If $D_{E_{\text{CFG}}}$ accepts, accept. Else, reject."

$D_{ALL_{\text{CGF}}}$ decides ALL_{CFG}.

What's wrong with this proof?

Summary Table of Decidability

	Acceptance Problem	Emptiness Problem	Equivalence Problem
DFA			
CFG			
TM			

Summary Table of Decidability

	Acceptance Problem	**Emptiness Problem**	**Equivalence Problem**
DFA	Decidable (Lecture 7)		
CFG			
TM			

Summary Table of Decidability

	Acceptance Problem	Emptiness Problem	Equivalence Problem
DFA	Decidable (Lecture 7)	Decidable (Lecture 7)	
CFG			
TM			

Summary Table of Decidability

	Acceptance Problem	**Emptiness Problem**	**Equivalence Problem**
DFA	Decidable (Lecture 7)	Decidable (Lecture 7)	Decidable (Lecture 7)
CFG			
TM			

Summary Table of Decidability

	Acceptance Problem	**Emptiness Problem**	**Equivalence Problem**
DFA	Decidable (Lecture 7)	Decidable (Lecture 7)	Decidable (Lecture 7)
CFG	Decidable		
TM			

Summary Table of Decidability

	Acceptance Problem	Emptiness Problem	Equivalence Problem
DFA	Decidable (Lecture 7)	Decidable (Lecture 7)	Decidable (Lecture 7)
CFG	Decidable	Decidable	
TM			

Summary Table of Decidability

	Acceptance Problem	Emptiness Problem	Equivalence Problem
DFA	Decidable (Lecture 7)	Decidable (Lecture 7)	Decidable (Lecture 7)
CFG	Decidable	Decidable	Not Decidable
TM			

Summary Table of Decidability

	Acceptance Problem	Emptiness Problem	Equivalence Problem
DFA	Decidable (Lecture 7)	Decidable (Lecture 7)	Decidable (Lecture 7)
CFG	Decidable	Decidable	Not Decidable
TM	Not Decidable	Not Decidable	Not Decidable

Roadmap for Proving Undecidable Problems

To show that a language B is undecidable by a reduction:

1. Show that there is an undecidable problem: A (next lecture).
2. Show that A reduces to B: we can decide A by using a decider for B as a subroutine.
   - **Implication:** B decidable $\Longrightarrow$ A decidable

Anatomy of a Proof by Reduction

Anatomy of a Proof by Reduction

- A proof by reduction is essentially a proof by contradiction.

Anatomy of a Proof by Reduction

- A proof by reduction is essentially a proof by contradiction.
- Suppose A is a known undecidable problem.

Anatomy of a Proof by Reduction

- A proof by reduction is essentially a proof by contradiction.
- Suppose A is a known undecidable problem.
- We want to prove … **Thm:** B is undecidable.

Anatomy of a Proof by Reduction

- A proof by reduction is essentially a proof by contradiction.
- Suppose A is a known undecidable problem.
- We want to prove … **Thm:** B is undecidable.
- **Proof by Reduction:**

Anatomy of a Proof by Reduction

- A proof by reduction is essentially a proof by contradiction.
- Suppose A is a known undecidable problem.
- We want to prove … $\boxed{\textbf{Thm: } B \text{ is undecidable.}}$
- **Proof by Reduction:**

    (1) For the sake of reaching a contradiction, assume B is decidable.

Anatomy of a Proof by Reduction

- A proof by reduction is essentially a proof by contradiction.
- Suppose A is a known undecidable problem.
- We want to prove … $\boxed{\textbf{Thm: } B \text{ is undecidable.}}$
- **Proof by Reduction:**

  (1) For the sake of reaching a contradiction, assume B is decidable.

  (2) So, there exists a decider D_B for B.

Anatomy of a Proof by Reduction

- A proof by reduction is essentially a proof by contradiction.
- Suppose A is a known undecidable problem.
- We want to prove … **Thm:** B is undecidable.
- **Proof by Reduction:**

  (1) For the sake of reaching a contradiction, assume B is decidable.

  (2) So, there exists a decider D_B for B.

  (3) Construct a procedure D_A that uses D_B to decide A.

Anatomy of a Proof by Reduction

- A proof by reduction is essentially a proof by contradiction.
- Suppose A is a known undecidable problem.
- We want to prove … **Thm:** B is undecidable.
- **Proof by Reduction:**

  (1) For the sake of reaching a contradiction, assume B is decidable.

  (2) So, there exists a decider D_B for B.

  (3) Construct a procedure D_A that uses D_B to decide A.

  (4) Conclude that A is decidable.

Anatomy of a Proof by Reduction

- A proof by reduction is essentially a proof by contradiction.
- Suppose A is a known undecidable problem.
- We want to prove … $\boxed{\textbf{Thm: } B \text{ is undecidable.}}$
- **Proof by Reduction:**

  (1) For the sake of reaching a contradiction, assume B is decidable.

  (2) So, there exists a decider D_B for B.

  (3) Construct a procedure D_A that uses D_B to decide A.

  (4) Conclude that A is decidable.

  (5) Line (4) contradicts assumption that A is undecidable. **Q.E.D.**

Acceptance Problem for TMs

Acceptance Problem for TMs

Let $A_{\mathsf{TM}} = \{\langle M, w\rangle \mid M$ is a TM and $w \in L(M)\}$.

Acceptance Problem for TMs

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

Theorem: A_{TM} is recognizable.

Acceptance Problem for TMs

Let $A_{\mathsf{TM}} = \{\langle M, w\rangle \mid M$ is a TM and $w \in L(M)\}$.

Theorem: A_{TM} is recognizable.

Consider the following TM $R_{A_{\mathsf{TM}}}$ recognizes A_{TM}.

Acceptance Problem for TMs

Let $A_{\mathsf{TM}} = \{\langle M, w\rangle \mid M$ is a TM and $w \in L(M)\}$.

Theorem: A_{TM} is recognizable.

Consider the following TM $R_{A_{\mathsf{TM}}}$ recognizes A_{TM}.

$R_{A_{\mathsf{TM}}}$= “On input $\langle M, w\rangle$ where M is a TM and w is a string:

Acceptance Problem for TMs

Let $A_{\text{TM}} = \{\langle M, w \rangle \,|\, M$ is a TM and $w \in L(M)\}$.

Theorem: A_{TM} is recognizable.

Consider the following TM $R_{A_{\text{TM}}}$ recognizes A_{TM}.

$R_{A_{\text{TM}}}$= "On input $\langle M, w \rangle$ where M is a TM and w is a string:

1. Simulate M on w.

Acceptance Problem for TMs

Let $A_{\mathsf{TM}} = \{\langle M, w \rangle \,|\, M$ is a TM and $w \in L(M)\}$.

Theorem: A_{TM} is recognizable.

Consider the following TM $R_{A_{\mathsf{TM}}}$ recognizes A_{TM}.

$R_{A_{\mathsf{TM}}}$= "On input $\langle M, w \rangle$ where M is a TM and w is a string:

1. Simulate M on w.
2. Accept if M halts and accepts.

Acceptance Problem for TMs

Let $A_{\mathsf{TM}} = \{\langle M, w\rangle \mid M$ is a TM and $w \in L(M)\}$.

Theorem: A_{TM} is recognizable.

Consider the following TM $R_{A_{\mathsf{TM}}}$ recognizes A_{TM}.

$R_{A_{\mathsf{TM}}}$= “On input $\langle M, w\rangle$ where M is a TM and w is a string:

1. Simulate M on w.
2. Accept if M halts and accepts.
3. Reject if M halts and rejects.”

Acceptance Problem for TMs

Let $A_{\text{TM}} = \{\langle M, w\rangle \,|\, M$ is a TM and $w \in L(M)\}$.

Theorem: A_{TM} is recognizable.

Consider the following TM $R_{A_{\text{TM}}}$ recognizes A_{TM}.

$R'_{A_{\text{TM}}}$= “On input $\langle M, w\rangle$ where M is a TM and w is a string:

1. Simulate M on w.
2. Accept if M halts and accepts.
3. Reject if M halts and rejects.
4. Reject if M never halts.”

Acceptance Problem for TMs

Let $A_{\mathsf{TM}} = \{\langle M, w \rangle \mid M$ is a TM and $w \in L(M)\}$.

Theorem: A_{TM} is recognizable.

Consider the following TM $R_{A_{\mathsf{TM}}}$ recognizes A_{TM}.

$R'_{A_{\mathsf{TM}}}$= "On input $\langle M, w \rangle$ where M is a TM and w is a string:

1. Simulate M on w.
2. Accept if M halts and accepts.
3. Reject if M halts and rejects.
4. ~~Reject if M never halts."~~

Not a legal TM action.

Acceptance Problem for TMs

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

Theorem: A_{TM} is recognizable.

Consider the following TM $R_{A_{\text{TM}}}$ recognizes A_{TM}.

$R'_{A_{\text{TM}}}$= "On input $\langle M, w\rangle$ where M is a TM and w is a string:

1. Simulate M on w.
2. Accept if M halts and accepts.
3. Reject if M halts and rejects.
4. ~~Reject if M never halts."~~

Not a legal TM action.

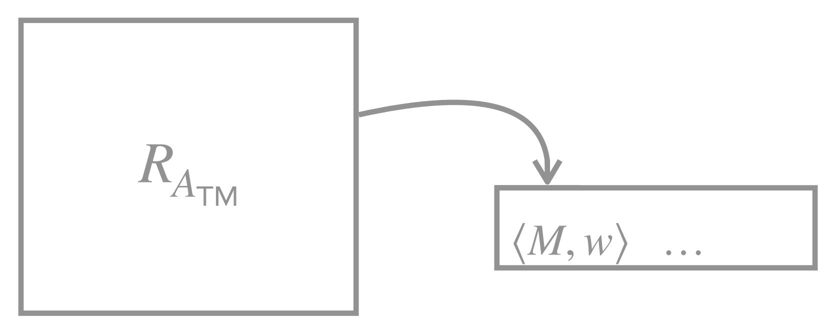


Fig. Turing's original universal TM, capable of simulating any other TM.

Example: Proving Undecidability via a Reduction

Example: Proving Undecidability via a Reduction

Theorem (p.207): The language $A_{\mathsf{TM}} = \{\langle M, w \rangle \mid M$ is a TM and $w \in L(M)\}$ is undecidable.

Example: Proving Undecidability via a Reduction

Theorem (p.207): The language $A_{\mathsf{TM}} = \{\langle M, w\rangle \mid M$ is a TM and $w \in L(M)\}$ is undecidable.

Proof: Next lecture.

Example: Proving Undecidability via a Reduction

Theorem (p.207): The language $A_{\mathrm{TM}} = \{\langle M, w \rangle \mid M \text{ is a TM and } w \in L(M)\}$ is undecidable.

Proof: Next lecture.

Theorem (p.216): The language $HALT_{\mathrm{TM}} = \{\langle M, w \rangle \mid M \text{ is a TM and halts on input } w\}$ is undecidable.

Example: Proving Undecidability via a Reduction

Theorem (p.207): The language $A_{\mathrm{TM}} = \{\langle M, w \rangle \mid M$ is a TM and $w \in L(M)\}$ is undecidable.

Proof: Next lecture.

Theorem (p.216): The language $HALT_{\mathrm{TM}} = \{\langle M, w \rangle \mid M$ is a TM and halts on input $w\}$ is undecidable.

Proof by reduction.

Example: Proving Undecidability via a Reduction

A:

Theorem (p.207): The language $A_{\text{TM}} = \{\langle M, w \rangle \mid M \text{ is a TM and } w \in L(M)\}$ is undecidable.

Proof: Next lecture.

Theorem (p.216): The language B: $HALT_{\text{TM}} = \{\langle M, w \rangle \mid M \text{ is a TM and halts on input } w\}$ is undecidable.

Proof by reduction.

Proof that the Halting Problem is Undecidable

Proof that the Halting Problem is Undecidable

Proof by Reduction:

(1) For the sake of reaching a contradiction, assume $HALT_{\text{TM}}$ is decidable.

Proof that the Halting Problem is Undecidable

Proof by Reduction:

(1) For the sake of reaching a contradiction, assume $HALT_{\text{TM}}$ is decidable.

(2) So, there exists a decider $D_{HALT_{\text{TM}}}$ for $HALT_{\text{TM}}$.

Proof that the Halting Problem is Undecidable

Proof by Reduction:

(1) For the sake of reaching a contradiction, assume $HALT_{\text{TM}}$ is decidable.

(2) So, there exists a decider $D_{HALT_{\text{TM}}}$ for $HALT_{\text{TM}}$.

(3) Let $D_{A_{\text{TM}}}$ be the following TM that uses $D_{HALT_{\text{TM}}}$ to decide A_{TM}:

Proof that the Halting Problem is Undecidable

Proof by Reduction:

(1) For the sake of reaching a contradiction, assume $HALT_{\text{TM}}$ is decidable.

(2) So, there exists a decider $D_{HALT_{\text{TM}}}$ for $HALT_{\text{TM}}$.

(3) Let $D_{A_{\text{TM}}}$ be the following TM that uses $D_{HALT_{\text{TM}}}$ to decide A_{TM}:

"On input $\langle M, w \rangle$ where M is a TM and w is a string,

Proof that the Halting Problem is Undecidable

Proof by Reduction:

(1) For the sake of reaching a contradiction, assume $HALT_{\mathsf{TM}}$ is decidable.

(2) So, there exists a decider $D_{HALT_{\mathsf{TM}}}$ for $HALT_{\mathsf{TM}}$.

(3) Let $D_{A_{\mathsf{TM}}}$ be the following TM that uses $D_{HALT_{\mathsf{TM}}}$ to decide A_{TM}:

"On input $\langle M, w \rangle$ where M is a TM and w is a string,

1. Run $D_{HALT_{\mathsf{TM}}}$ on $\langle M, w \rangle$.

Proof that the Halting Problem is Undecidable

Proof by Reduction:

(1) For the sake of reaching a contradiction, assume $HALT_{\text{TM}}$ is decidable.

(2) So, there exists a decider $D_{HALT_{\text{TM}}}$ for $HALT_{\text{TM}}$.

(3) Let $D_{A_{\text{TM}}}$ be the following TM that uses $D_{HALT_{\text{TM}}}$ to decide A_{TM}:

"On input $\langle M, w \rangle$ where M is a TM and w is a string,

1. Run $D_{HALT_{\text{TM}}}$ on $\langle M, w \rangle$.
2. If $D_{HALT_{\text{TM}}}$ rejects (i.e., M doesn't halt on w), reject.

Proof that the Halting Problem is Undecidable

Proof by Reduction:

(1) For the sake of reaching a contradiction, assume $HALT_{\text{TM}}$ is decidable.

(2) So, there exists a decider $D_{HALT_{\text{TM}}}$ for $HALT_{\text{TM}}$.

(3) Let $D_{A_{\text{TM}}}$ be the following TM that uses $D_{HALT_{\text{TM}}}$ to decide A_{TM}:

"On input $\langle M, w \rangle$ where M is a TM and w is a string,

1. Run $D_{HALT_{\text{TM}}}$ on $\langle M, w \rangle$.
2. If $D_{HALT_{\text{TM}}}$ rejects (i.e., M doesn't halt on w), reject.
3. If $D_{HALT_{\text{TM}}}$ accepts, simulate M on w until it halts.

Proof that the Halting Problem is Undecidable

Proof by Reduction:

(1) For the sake of reaching a contradiction, assume $HALT_{\text{TM}}$ is decidable.

(2) So, there exists a decider $D_{HALT_{\text{TM}}}$ for $HALT_{\text{TM}}$.

(3) Let $D_{A_{\text{TM}}}$ be the following TM that uses $D_{HALT_{\text{TM}}}$ to decide A_{TM}:

"On input $\langle M, w \rangle$ where M is a TM and w is a string,

1. Run $D_{HALT_{\text{TM}}}$ on $\langle M, w \rangle$.
2. If $D_{HALT_{\text{TM}}}$ rejects (i.e., M doesn't halt on w), reject.
3. If $D_{HALT_{\text{TM}}}$ accepts, simulate M on w until it halts.
4. If M accepts, accept. Otherwise, reject."

Proof that the Halting Problem is Undecidable

Proof by Reduction:

(1) For the sake of reaching a contradiction, assume $HALT_{\text{TM}}$ is decidable.

(2) So, there exists a decider $D_{HALT_{\text{TM}}}$ for $HALT_{\text{TM}}$.

(3) Let $D_{A_{\text{TM}}}$ be the following TM that uses $D_{HALT_{\text{TM}}}$ to decide A_{TM}:

"On input $\langle M, w \rangle$ where M is a TM and w is a string,

1. Run $D_{HALT_{\text{TM}}}$ on $\langle M, w \rangle$.
2. If $D_{HALT_{\text{TM}}}$ rejects (i.e., M doesn't halt on w), reject.
3. If $D_{HALT_{\text{TM}}}$ accepts, simulate M on w until it halts.
4. If M accepts, accept. Otherwise, reject."

(4) So, we've reached a contradiction, namely that A is decidable. **Q.E.D.**

Summary of Today's Lecture

- Decision procedures for CFGs
- The reducibility method

Acknowledgements

- These slides are based on lecture notes on Theory of Computation from other universities, namely Michael Sipser (MIT), Lorenzo De Stefani (Brown).

- **Errata:** If you let us know of any errors in the slides, we'll fix them and acknowledge you here!