Lec 17. Reduction and undecidable languages I

Eunjung Kim

Characterizing decidability

A language $A \subseteq \Sigma^*$ is said to be co-Turing-recognizable if its complement (i.e. $\Sigma^* \setminus A$) is Turing-recognizable.

Turing-recognizable and co-Turing-recognizable

A language is decidable if and only if it is Turing-recognizable and co-Turing-recognizable.

Characterizing decidability

A language $A \subseteq \Sigma^*$ is said to be co-Turing-recognizable if its complement (i.e. $\Sigma^* \setminus A$) is Turing-recognizable.

Turing-recognizable and co-Turing-recognizable

A language is decidable if and only if it is Turing-recognizable and co-Turing-recognizable.

- The direction ($\Rightarrow$) is straightforward. (How so?)
- For the direction ($\Leftarrow$), let M_1 and M_2 be two TMs recognizing A and $\bar{A}$. Build a new TM M which runs both M_1 and M_2 simultaneously on $w \in \Sigma^*$ and outputs

$$M(w) = \begin{cases} \text{ACCEPT} & \text{if } M_1(w) = \text{ACCEPT} \\ \text{REJECT} & \text{if } M_2(w) = \text{ACCEPT} \end{cases}$$

  Clearly M decides A.

Halting problem is undecidable

Halting problem: $HALT_{TM} = \{\langle M, w\rangle \mid M \text{ is TM and } M \text{ halts on } w\}$.

From undecidability of A_{TM}, we derive

$HALT_{TM}$ is undecidable.

Halting problem is undecidable

Halting problem: $HALT_{TM} = \{\langle M, w\rangle \mid M \text{ is TM and } M \text{ halts on } w\}$.

From undecidability of A_{TM}, we derive

$HALT_{TM}$ is undecidable.

Proof: we crucially use that $A_{TM} = \{\langle M, w\rangle \mid M \text{ is TM and } M \text{ accepts } w\}$ is undecidable.

HALTING PROBLEM IS UNDECIDABLE

Suppose the contrary; let D be a decider TM for $HALT_{TM}$.

Then, we can build a decider A for A_{TM} that works as follows:

A upon input $\langle M, w \rangle$

1. Run D on the string $\langle M, w \rangle$.
2. if D accepts $\langle M, w \rangle$, then A simulates M on w and outputs

$$A(\langle M, w \rangle) = \begin{cases} \text{YES} & \text{if } M(w) = \text{YES} \\ \text{NO} & \text{if } M(w) = \text{NO} \end{cases}$$

3. if D rejects $\langle M, w \rangle$ (i.e. M loops on w), then A reject $\langle M, w \rangle$.

So far

We have seen two techniques to prove undecidability.

1. Diagonal argument: to settle the undecidability of A_{TM}.
2. Reduction, a.k.a. Turing-reduction: used for showing that Halting problem is undecidable.

A Turing-reducible to B

We say that A is Turing-reducible to B if there is a TM deciding A which uses a (hypothetical) TM deciding B.
Observe: Undecidability of A implies the undecidability of B. ("B is as hard as A")

So far

We have seen two techniques to prove undecidability.

1. Diagonal argument: to settle the undecidability of A_{TM}.
2. Reduction, a.k.a. Turing-reduction: used for showing that Halting problem is undecidable.

A Turing-reducible to B

We say that A is Turing-reducible to B if there is a TM deciding A which uses a (hypothetical) TM deciding B.
Observe: Undecidability of A implies the undecidability of B. ("B is as hard as A")

The hypothetical TM deciding B is also called an oracle for B.

3. We' shall see a third technique called mapping-reduction a.k.a many-one reduction.

Emptiness problem is undecidable

Emptiness problem: $E_{TM} = \{M : M \text{ is TM and } L(M) = \emptyset\}$.

Undecidability of E_{TM}

E_{TM} is undecidable.

Emptiness problem is undecidable

Emptiness problem: $E_{TM} = \{M : M \text{ is TM and } L(M) = \emptyset\}$.

Undecidability of E_{TM}

E_{TM} is undecidable.

- Prove that if E_{TM} has a decider E, then this can be used (as a subroutine) to build a decider D for $A_{TM} = \{(M, w) : M \text{ is TM and } M \text{ accepts } w\}$.
- This contradicts the undecidability of A_{TM}.

Emptiness problem is undecidable

Emptiness problem: $E_{TM} = \{M : M \text{ is TM and } L(M) = \emptyset\}$.

Undecidability of E_{TM}

E_{TM} is undecidable.

- D upon an input $\langle M, w \rangle$ does the following:
  1. Compute an encoding of a TM M_o^w such that

$$L(M_o^w) = \begin{cases} \{w\} & \text{if } M \text{ accepts } w \\ \emptyset & \text{if } M \text{ does not accept } w \end{cases}$$

  2. Run R on $\langle M_o^w \rangle$.
  3. D outputs

$$\begin{cases} ?????????? & \text{if } E(\langle M_o^w \rangle) = \text{YES} \\ ?????????? & \text{if } E(\langle M_o^w \rangle) = \text{NO} \end{cases}$$

- How to design (compute) such TM M_o^w?

Emptiness problem is undecidable

How does a TM M_o^w work internally?

$$L(M_o^w) = \begin{cases} \{w\} & \text{if } M \text{ accepts } w \\ \emptyset & \text{if } M \text{ does not accept } w \end{cases}$$

TM M_o^w does the following on input string x:

$$M_o^w(x) = \begin{cases} \textsc{No} & \text{if } x \neq w \\ \textsc{Yes} & \text{if } x = w \text{ and } M \text{ accepts } w \end{cases}$$

M_o^w may loop on w. But it's fine to obtain a decider D as D does not simulate M_o^w but only computes its encoding.

Non-emptiness is undecidable

$SOME_{TM} = \{M : M \text{ is TM and } L(M) \neq \emptyset\}$.

Undecidability of $SOME_{TM}$ from that of E_{TM}

$SOME_{TM}$ is undecidable.

- Key idea: Reduce from E_{TM} to $SOME_{TM}$ by designing a decider D of E_{TM} using the hypothetical TM S deciding $SOME_{TM}$.
- D upon an input $\langle M \rangle$ does the following:
  1. Run S on $\langle M \rangle$.
  2. D outputs

$$D(\langle M \rangle) = \begin{cases} ??????? & \text{if } S(\langle M \rangle) = \text{Yes} \\ ??????? & \text{if } S(\langle M \rangle) = \text{No} \end{cases}$$

REG_{TM} IS UNDECIDABLE

$REG_{TM} = \{M : M \text{ is TM and } L(M) \text{ is a regular language}\}$.

UNDECIDABILITY OF REG_{TM}

REG_{TM} is undecidable.

- Key idea: reduction from A_{TM} to REG_{TM}.
- D, upon an input $\langle M, w\rangle$, does the following:
  1. Compute & write an encoding $\langle M_o^{M,w}\rangle$ of TM $M_o^{M,w}$ such that

$$L(M_o^{M,w}) = \begin{cases} \{0^n 1^n \mid n \geq 0\} & \text{if } M \text{ does not accept } w \\ \Sigma^* & \text{if } M \text{ accepts } w \end{cases}$$

  2. Run R on $\langle M_o^{M,w}\rangle$.
  3. D outputs

$$\begin{cases} \text{YES} & \text{if } R \text{ outputs YES} \\ \text{NO} & \text{if } R \text{ outputs NO} \end{cases}$$

REG_{TM} IS UNDECIDABLE

How does a TM $M_o^{M,w}$ work internally?

$$L(M_o^{M,w}) = \begin{cases} \{0^n1^n \mid n \geq 0\} & \text{if } M \text{ does not accept } w \\ \Sigma^* & \text{if } M \text{ accepts } w \end{cases}$$

TM $M_o^{M,w}$ does the following on input string x:

$$M_o^{M,w}(x) = \begin{cases} \text{YES} & \text{if } x \text{ is of the form } 0^n1^n \text{ for some } n \geq 0 \\ \text{YES} & \text{if } M \text{ accepts } w \end{cases}$$

$M_o^{M,w}$ may loop on w. But it's fine to obtain a decider D as D does not simulate $M_o^{M,w}$ but only computes its encoding.

Linear Bounded Automaton

Linear bounded Automaton

A linear bounded automaton is a Turing machine with the following restriction: its header is not allowed to move off the portion of the (single) tape containing the input. When the TM instructs the header to move to the right of the right-end of the input, then it stays where it is.

Key observation: For an input of length n, a linear bounded automaton on w can go through at most

$$|Q| \cdot n \cdot |\Gamma|^n$$

distinct configurations. This means

Halting problem for LBA

A linear bounded automaton M halts on an input w of length n if it halts in the first $|Q| \cdot n \cdot |\Gamma|^n$ steps. Does the converse hold?

Linear Bounded Automaton

$HALT_{LBA} = \{\langle M, w\rangle : M \text{ is LBA and } M \text{ halts on } w\}$.
$A_{LBA} = \{\langle M, w\rangle : M \text{ is LBA and } M \text{ accepts } w\}$.

Decidability of some problems related to LBA

$HALT_{LBA}$ and A_{LBA} are decidable.

We can construct a TM D which does the following on $\langle M, w\rangle$:

1. D simulates M on w for $|Q| \cdot n \cdot |\Gamma|^n$ steps.
2. If D halts, output YES.
3. If D did not halt, output NO.

E_{LBA} IS UNDECIDABLE

$E_{LBA} = \{\langle M \rangle : M \text{ is LBA and } L(M) = \emptyset\}$.

UNDECIDABILITY OF E_{LBA}

E_{LBA} is undecidable.

- Key idea: Reduce from A_{TM} to E_{LBA}.
- D upon an input $\langle M, w \rangle$ does the following:
  1. Compute & write an encoding $\langle B^{M,w} \rangle$ of LBA $B^{M,w}$ s.t.

$$L(B^{M,w}) = \begin{cases} \{\text{all accepting computation histories of } M \text{ on } w\} \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{if } M \text{ accepts } w \\ \emptyset \quad \text{if } M \text{ does not accept } w \end{cases}$$

  2. Run E on $\langle B^{M,w} \rangle$.
  3. D outputs

$$\begin{cases} \text{NO} & \text{if } E \text{ outputs YES} \\ \text{YES} & \text{if } E \text{ outputs NO} \end{cases}$$