CS244: Theory of Computation

Fu Song
ShanghaiTech University

Fall 2022

- Turing machine is a model of a general purpose computer
- Several examples of problems that are solvable on a Turing machine
- One example of a problem, A_{TM}, that is computationally unsolvable
- Examine more unsolvable problems
- Introduce the primary method, reducibility, for proving that problems are computationally unsolvable

Outline

Reducibility

An informal definition of reducibility

- Given two problems A and B, a reduction is a way converting the problem A to the problem B
- If we have a solution for B, by reduction, we get a solution for A
- If A is undecidable, by reduction, we prove that B is undecidable

Outline

Undecidable Problems from Language Theory

The halting problem

$$HALT_{\mathsf{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ halts on input } w\}.$$

Theorem

$HALT_{\mathsf{TM}}$ is undecidable.

Proof

Reduction from the membership problem of TM i.e., A_{TM}, which was proved undecidable, to $HALT_{\mathsf{TM}}$

Assume R decides $HALT_{\mathsf{TM}}$, construct a TM S which decides A_{TM}.

S on input $\langle M, w \rangle$:

1. Run R on $\langle M, w \rangle$ (by assumption that R decides $HALT_{\mathsf{TM}}$, R must halt)
2. If R reject (i.e., M never halts on w), then reject.
3. Otherwise R must accept, simulate M on w until it halts.
4. If M has accepts, then accept; if M has rejected, reject.

If R decides $HALT_{\mathsf{TM}}$, then S decides A_{TM}

Testing emptiness

$$E_{\mathsf{TM}} = \left\{ \langle M \rangle \;\middle|\; M \text{ is a TM and } L(M) = \emptyset \right\}.$$

Theorem

E_{TM} is undecidable.

Proof (1)

Reduction from the membership problem of TM i.e., A_{TM}, which was proved undecidable, to E_{TM}

Assume R decides E_{TM}. How to construct the TM S that decides A_{TM}?

S on input $\langle M, w \rangle$:

1. Run R on $\langle M \rangle$ (by assumption that R decides E_{TM}, R must halt)
2. If R accept, (i.e., $E_{\mathsf{TM}} = \emptyset$), then reject.
3. Otherwise R must reject, then $E_{\mathsf{TM}} \neq \emptyset$.

We do not know whether $w \in L(M)$ or not.
Then, S cannot decide A_{TM}.

Instead, we run R on a TM M_1 *obtained by restricting M to a specific string w such that M accepts w iff $L(M_1) \neq \emptyset$.*

Proof (1)

Reduction from the membership problem of TM i.e., A_{TM},
which was proved undecidable, to E_{TM}

For every TM M and string w, we construct an M_1:

M_1 on input x:

1. If $x \neq w$, then reject.
2. If $x = w$, run M on w and accept if M does.

Then

$$M \text{ accepts } w \quad \Longleftrightarrow \quad L(M_1) \neq \emptyset.$$

Proof (2)

$$M \text{ accepts } w \iff L(M_1) \neq \emptyset.$$

Assume R decides E_{TM}. Then the following TM S decides A_{TM}.

S on input $\langle M, w \rangle$:

1. Use the description of M and w to construct the TM M_1.
2. Run R on input $\langle M_1 \rangle$.
3. If R accepts (i.e., $L(M_1) = \emptyset$), then reject;
4. if R rejects (i.e., $L(M_1) \neq \emptyset$), then accept.

If R decides E_{TM}, then S decides A_{TM}.

Testing regularity

$$REGULAR_{\mathsf{TM}} = \{\langle M \rangle \mid M \text{ is a TM and } L(M) \text{ is a regular language}\}.$$

Theorem

$REGULAR_{\mathsf{TM}}$ is undecidable.

Proof (1)

Reduction from the membership problem of TM i.e., A_{TM}, which was proved undecidable, to $REGULAR_{\mathsf{TM}}$

Assume R decides $REGULAR_{\mathsf{TM}}$. How to construct the TM S that decides A_{TM}?

S takes $\langle M, w \rangle$ as input such that a TM M_2 accepts a regular language iff M accepts w.

R accepts M_2 iff M accepts w.

Proof (1)

For every TM M and string w we construct an M_2:

M_2 on input x:

1. If x has the form $0^n 1^n$, then accept.
2. If x does not have the form $0^n 1^n$, then run M on w and accept if M does.

Then

$$M \text{ accepts } w \quad \Longleftrightarrow \quad L(M_2) \text{ is regular.}$$

Indeed

- If M accepts w, then $L(M_2) = \Sigma^*$ (regular language)
- If M does not accept w, then $L(M_2) = \{0^n 1^n \mid n \geq 0\}$ (context-free language but not regular)

Proof (2)

$$M \text{ accepts } w \iff L(M_2) \text{ is regular.}$$

Assume R decides $REGULAR_{\mathrm{TM}}$. Then the following S decides A_{TM}.
S on input $\langle M, w \rangle$:

1. Use the description of M and w to construct the TM M_2.
2. Run R on input $\langle M_2 \rangle$.
3. If R accepts, then accept; if R rejects, then reject.

Generalization, check whether a Turing machine is CFL, a decidable language, or even a finite language can be shown to be undecidable with similar proofs

Quiz

$$CFG_{\mathrm{TM}} = \{\langle M\rangle \mid M \text{ is a TM and } L(M) \text{ is a context-free language}\}.$$

Theorem

CFG_{TM} is undecidable.

Rice's Theorem

- Let P be any property of the language of a TM, defined as a language of TM encodings $\langle M \rangle$
  - P is nontrivial, if P is not empty and not universal
  - P is fair, if $L(M_1) = L(M_2)$, then $\langle M_1 \rangle \in P \iff \langle M_2 \rangle \in P$

Theorem

Nontrivial and fair P is undecidable, i.e., checking whether a given TM M has property P is undecidable.

Proof (1)

- Assume that P is fair and nontrivial.
- Let R be the decider of P.
- Let $T_\emptyset$ be a TM such that $L(T_\emptyset) = \emptyset$. We assume that $\langle T_\emptyset \rangle \notin P$, otherwise we can consider the property $\overline{P}$ (decidable language P is closed under complementation)
- Since P is nontrivial, there exists a TM T such that $\langle T \rangle \in P$
- Design a TM S to decide A_{TM} using P's decider R to distinguish T and $T_\emptyset$

Proof (2)

M_w on input x:

1. Simulate M on w. If it halts and rejects, then reject. If it accepts, then togo step 2.
2. Simulate T on x. If it accepts, then accept.

- If M accepts w, then M_w simulates T
- Otherwise, M_w simulates $T_\emptyset$

Now, we can use R to determine whether $\langle M_w \rangle \in P$,

$$\langle M_w \rangle \in P \iff w \in L(M)$$

Proof (3)

S on input $\langle M, w \rangle$:

1. Construct M_w from M and w
2. Simulate R on M_w. If it accepts, then accept; If it rejects, then reject;

$$\langle M_w \rangle \in P \iff w \in L(M)$$

S becomes a decider of A_{TM}.

Reducibility

- So far, we prove undecidability by reducing from A_{TM}
- Indeed, we can prove undecidability by reducing from any known undecidable languages such as $REGULAR_{\mathsf{TM}}$ and E_{TM}.
- We demonstrate it by proving that equality testing of TM is undecidable via reducing from E_{TM}.

Testing equality

$$EQ_{\mathsf{TM}} = \big\{\langle M_1, M_2\rangle \;\big|\; M_1 \text{ and } M_2 \text{ are TMs and } L(M_1) = L(M_2)\big\}.$$

Theorem

EQ_{TM} is undecidable.

Reduction from the emptiness problem of TM i.e., E_{TM}, which was proved undecidable, to EQ_{TM}

Proof

Reduction from the emptiness problem of TM i.e., E_{TM},
which was proved undecidable, to EQ_{TM}

Idea: for every $M_\emptyset \in E_{\mathsf{TM}}$: $\langle M, M_\emptyset \rangle \in EQ_{\mathsf{TM}} \iff M \in E_{\mathsf{TM}}$,

Assume R decides EQ_{TM}. Then we can decide E_{TM} as follows.

S on input $\langle M \rangle$:

1. Run R on input $\langle M, M_\emptyset \rangle$, where $M_\emptyset$ is a TM that rejects all inputs.
2. If R accepts, then accept; if R rejects, then reject.

Reductions via computation histories

Computation histories

Definition

Let M be a TM and w an input string. An accepting computation history for M on w is a sequence of configurations.

$$C_1, \ldots, C_\ell,$$

where C_1 is the start configuration of M on w, C_ℓ is an accepting configuration of M, and each C_i legally follows from C_{i-1} according to the rules of M.

A rejecting computation history for M on w is defined similarly, except that C_ℓ is a rejecting configuration.

Recall Linear Bounded Automata

Definition

A linear bounded automaton (LBA) is a TM wherein the tape head isn't permitted to move off the portion of the tape containing the input.

If the machine tries to move its head off either end of the input, the head stays where it is.

$$A_{\text{LBA}} = \{\langle M, w\rangle \mid M \text{ is an LBA that accepts } w\}.$$

Theorem

A_{LBA} is decidable.

Lemma

Let M be an LBA with q states and g symbols in the tape alphabet. There are exactly qng^n distinct configurations of M for a tape length n.

Proof

Theorem

A_{LBA} is decidable.

L on input $\langle M, w \rangle$:

1. Simulate M on w for qng^n steps or until it halts.
2. If M has halted, accept if it has accepted and reject if it has rejected. If it has not halted, reject.

If M on w has not halted within qng^n steps, it must be repeating a configuration and therefore looping.

Testing emptiness

$$E_{\mathsf{LBA}} = \{\langle M\rangle \mid B \text{ is an LBA and } L(B) = \emptyset\}.$$

Theorem

E_{LBA} is undecidable.

Reduction from A_{TM} to E_{LBA}

Idea: construct a LBA B that accepts all the accepting sequences of computations (i.e., accepting computation histories $\sharp C_1 \sharp \ldots \sharp C_\ell \sharp$) of M on w.

$$L(B) \neq \emptyset \iff w \in L(M)$$

An LBA recognizing computation histories

Let M be a TM and w an input string.

On input x, the LBA B works as follows:

1. breaks up x according to the delimiters $\sharp$ into strings $C_1, \ldots, C_\ell$;
2. determines whether C_i's satisfy
   - 2.1 C_1 is the start configuration for M on w, i.e., $q_0 w$
   - 2.2 each C_{i+1} legally follows from C_i, i.e., $w_1 x_1 q x_2 w_2 \to w_1 abc w_2$
   - 2.3 C_ℓ is an accepting configuration, i.e., $w_1 q_f w_2$

Then

$$M \text{ accepts } w \quad \Longleftrightarrow \quad L(B) \neq \emptyset.$$

Proof

$$M \text{ accepts } w \iff L(B) \neq \emptyset.$$

Assume R decides E_{LBA}. Then the following S decides A_{TM}.

S on input $\langle M, w \rangle$:

1. Construct LBA B from M and w.
2. Run R on input $\langle B \rangle$.
3. If R rejects (i.e., $L(B) \neq \emptyset$), then accept; if R accepts (i.e., $L(B) \neq \emptyset$), then reject.

Recall: Membership and emptiness of CFG are decidable. But the universal of CFG is undecidable.

$$ALL_{\mathsf{CFG}} = \{\langle G\rangle \mid G \text{ is a CFG and } L(G) = \Sigma^*\}.$$

Theorem
ALL_{CFG} is undecidable.

Proof (1)

Let M be a TM and w a string. We will construct a CFG G such that

$$\begin{aligned} M \text{ accepts } w \quad &\iff \quad L(G) \neq \Sigma^* \\ &\iff \quad G \text{ does not generate} \\ & \quad \text{the accepting computation history for } M \text{ on } w. \end{aligned}$$

Proof (2)

An accepting computation history for M on w appears as

$$\#C_1\#C_2\#\cdots\#C_\ell\#,$$

where C_i is the configuration of M on the ith step of the computation on w.

Then, G generates all strings

1. that do not start with C_1,
2. that do not end with an accepting configuration, or
3. in which C_i does not properly yield C_{i+1} under the rule of M.

Proof (3)

We construct a PDA D and then convert it to G.

1. D starts by nondeterministically branching to guess which of the three conditions to check.
2. The first and the second are straightforward.
3. The third branch accepts if some C_i does not properly yield C_{i+1}.
   3.1 It scans the input and nondeterministically decides that it has come to C_i.
   3.2 It pushes C_i onto the stack until it reads $\#$.
   3.3 Then D pops the stack to compare with C_{i+1}: they are almost the same except around the head position, where the difference is dictated by the transition function of M.
   3.4 D accepts if there is a mismatch or an improper update.

Proof (4)

A minor problem: when D pops C_i off the stack, it is in reverse order.

We write the accepting computation history as

$$\# \underbrace{\longrightarrow}_{C_1} \# \underbrace{\longrightarrow}_{C_2^{\mathcal{R}}} \# \underbrace{\longrightarrow}_{C_3} \# \underbrace{\longrightarrow}_{C_4^{\mathcal{R}}} \# \cdots \# \underbrace{\longrightarrow}_{C_\ell} \#$$

Outline

Post Correspondence Problem (PCP)

- Undecidability is not confined to problems concerning automata
- There are many undecidability problems
- An undecidable problem concerning simple manipulations of strings, called the Post Correspondence Problem (PCP)

Post Correspondence Problem (PCP)

Definition

Given a finite alphabet Σ with at least two elements and a set of pairs

$$P = \{\frac{t_1}{b_1}, \frac{t_2}{b_2}, \cdots, \frac{t_m}{b_m}\}$$

where $t_i, b_i \in \Sigma^+$ for all $1 \le i \le m$,

the Post Correspondence Problem is to determine whether P has a match, namely, there exists a sequence of indices $1 \le i_1, i_2, \cdots, i_k \le m$ such that

$$t_{i_1} t_{i_2} \cdots t_{i_k} = b_{i_1} b_{i_2} \cdots b_{i_k}$$

Consider $P = \{\frac{b}{ca}, \frac{a}{ab}, \frac{ca}{a}, \frac{abc}{c}\}$ and indices $2, 1, 3, 2, 4$:

$$\frac{a}{ab} \frac{b}{ca} \frac{ca}{a} \frac{a}{ab} \frac{abc}{c}$$

Post Correspondence Problem (PCP)

Theorem

The PCP problem is undecidable.

Idea:

- Reducing from A_{TM} to P such that the TM M accepts w iff P has a match
- We encode an accepting computation history of M on w as a match of P, i.e., for the match

$$t_{i_1} t_{i_2} \cdots t_{i_k} = b_{i_1} b_{i_2} \cdots b_{i_k} = \sharp C_1 \sharp C_2 \sharp \cdots \sharp C_\ell \sharp$$

Proof (1)

$\text{MPCP} = \{\langle P \rangle \mid \text{P is a PCP with match that starts with index } 1\}$

Theorem

The MPCP problem is undecidable.

Proof (2)

Given a TM M and input w, construct MPCP P':

1. The first domino: $\frac{\sharp}{\sharp q_0 w \sharp} \in P'$
2. Simulate moving right: for every $a, b \in \Gamma, p, q \in Q$ such that $q \neq q_{reject}$ and $\delta(q, a) = (p, b, R)$: $\frac{qa}{bp} \in P'$
3. Simulate moving left: for every $a, b, c \in \Gamma, p, q \in Q$ such that $q \neq q_{reject}$ and $\delta(q, a) = (p, b, L)$: $\frac{cqa}{pcb} \in P'$
4. Preserve the other tape content: for every $a \in \Gamma$: $\frac{a}{a} \in P'$
5. Add blank: $\frac{\sharp}{\llcorner\!\lrcorner \sharp} \in P'$
6. eliminate adjacent symbols at accept state: for every $a \in \Gamma$, $\frac{a q_{accept}}{q_{accept}}, \frac{q_{accept} a}{q_{accept}} \in P'$
7. Happy ending: $\frac{q_{accept} \sharp\sharp}{\sharp} \in P'$

$q_0 101 \sharp 1 q_1 01 \sharp 11 q_2 1 \sharp 1 q_3 10 \iff$

$$\frac{\sharp}{\sharp q_0 101 \sharp} \frac{q_0 1}{1 q_1} \frac{0}{0} \frac{1}{1} \frac{\sharp}{\sharp} \frac{1}{1} \frac{q_1 0}{1 q_2} \frac{1}{1} \frac{\sharp}{\sharp} \frac{1}{1} \frac{1 q_2 1}{q_3 10} \frac{\sharp}{\sharp} \frac{1 q_3}{q_3} \frac{1}{1} \frac{0}{0} \frac{\sharp}{\sharp} \frac{q_3 1}{q_3} \frac{0}{0} \frac{\sharp}{\sharp} \frac{q_3 0}{q_3} \frac{\sharp}{\sharp} \frac{q_3 \sharp\sharp}{\sharp}$$

Proof (3)

From MPCP P' to PCP P:

- The first domino: $\frac{\sharp}{\sharp q_0 w_1 \cdots w_n \sharp} \in P' \iff \frac{\star\sharp}{\star\sharp\star q_0 \star w_1 \star \cdots \star w_n \star \sharp \star} \in P$
- For other domino: $\frac{x_1 x_2 \cdots x_m}{y_1 y_2 \cdots y_n} \in P' \iff \frac{\star x_1 \star x_2 \cdots \star x_m}{y_1 \star y_2 \star \cdots y_n \star} \in P$
- Match the additional star: $\frac{\star\$}{\$} \in P$

$$\frac{\sharp}{\sharp q_0 101 \sharp}\,\frac{q_0 1}{1 q_1}\,\frac{0}{0}\,\frac{1}{1}\,\frac{\sharp}{\sharp}\,\frac{1}{1}\,\frac{q_1 0}{1 q_2}\,\frac{1}{1}\,\frac{\sharp}{\sharp}\,\frac{1}{1}\,\frac{1 q_2 1}{q_3 10}\,\frac{\sharp}{\sharp}\,\frac{1 q_3}{q_3}\,\frac{1}{1}\,\frac{0}{0}\,\frac{\sharp}{\sharp}\,\frac{q_3 1}{q_3}\,\frac{0}{0}\,\frac{\sharp}{\sharp}\,\frac{q_3 0}{q_3}\,\frac{\sharp}{\sharp}\,\frac{q_3 \sharp\sharp}{\sharp}$$

$$\iff \frac{\star\sharp}{\star\sharp\star q_0 \star 1 \star 0 \star 1 \star \sharp \star}\,\frac{\star q_0 \star 1}{1 \star q_1 \star}\,\frac{\star 0}{0 \star}\,\frac{\star 1}{1 \star}\,\frac{\star\sharp}{\sharp\star}\,\frac{\star 1}{1 \star}\,\frac{\star q_1 \star 0}{1 \star q_2 \star}\,\frac{\star 1}{1 \star}\,\frac{\star\sharp}{\sharp\star}\,\frac{\star 1}{1 \star}\,\frac{\star 1 \star q_2 \star 1}{q_3 \star 1 \star 0 \star}\,\frac{\star\sharp}{\sharp\star}$$

$$\frac{\star 1 \star q_3}{q_3 \star}\,\frac{\star 1}{1 \star}\,\frac{\star 0}{0 \star}\,\frac{\star\sharp}{\sharp\star}\,\frac{\star q_3 \star 1}{q_3 \star}\,\frac{\star 0}{0 \star}\,\frac{\star\sharp}{\sharp\star}\,\frac{\star q_3 \star 0}{q_3 \star}\,\frac{\star\sharp}{\sharp\star}\,\frac{\star q_3 \star \sharp \star \sharp}{\sharp\star}\,\frac{\star\$}{\$}$$

Proof (3)

S on input $\langle M, w \rangle$:

1. Construct the PCP problem P from M and w
2. If P has a match, then accept; otherwise reject

Outline

Mapping Reducibility
Many-to-one Reducibility

Yet another proof technique

Computable functions

Definition

A function $f : \Sigma^* \to \Sigma^*$ is computable function if some Turing machine M, on every input w, halts with $f(w)$ on its tape.

Function $+ := \lambda\langle n, m\rangle.\langle n + m\rangle$ is computable for $n, m \in \mathbb{N}$

Formal definition of mapping reducibility

Definition

Language A is mapping reducible to language B, written $A \leq_{\mathrm{m}} B$, if there is a computable function $f : \Sigma^* \to \Sigma^*$, where for every $w \in \Sigma^*$

$$w \in A \iff f(w) \in B.$$

The function f is called the reduction from A to B.

Theorem

$$A \leq_{\mathrm{m}} B \iff \overline{A} \leq_{\mathrm{m}} \overline{B}.$$

$$\begin{aligned}
(A \leq_{\mathrm{m}} B) &\iff (\exists f . w \in A \iff f(w) \in B) \\
&\iff (\exists f . w \notin A \iff f(w) \notin B) \\
&\iff (\exists f . w \in \overline{A} \iff f(w) \in \overline{B}) \\
&\iff \overline{A} \leq_{\mathrm{m}} \overline{B}
\end{aligned}$$

$$w \in A \iff f(w) \in B.$$

Theorem

If $A \leq_{\mathrm{m}} B$ and B is decidable, then A is decidable.

Corollary

If $A \leq_{\mathrm{m}} B$ and A is undecidable, then B is undecidable.

$A_{\mathsf{TM}} \leq_{\mathrm{m}} HALT_{\mathsf{TM}}$

S on input $\langle M, w\rangle$: (previous reduction)

1. Run R on $\langle M, w\rangle$
2. If R reject (i.e., M never halts on w), then reject.
3. Otherwise R must accept, simulate M on w until it halts.
4. If M has accepts, then accept; if M has rejected, reject.

R decides $HALT_{\mathsf{TM}}$ iff S decides A_{TM}

F on input $\langle M, w\rangle$: (computable function $f : A_{\mathsf{TM}} \to HALT_{\mathsf{TM}}$)

1. Construct the following machine $M'(x)$.
   1.1 Run M on x.
   1.2 If M accepts, then accept.
   1.3 If M rejects, then enter a loop.
2. Output $\langle M', w\rangle$.

$$\langle M, w\rangle \in A_{\mathsf{TM}} \iff f(\langle M, w\rangle) = \langle M', w\rangle \in HALT_{\mathsf{TM}}$$

$E_{\text{TM}} \leq_{\mathrm{m}} EQ_{\text{TM}}$

S on input $\langle M \rangle$: (previous reduction)

1. Run R on input $\langle M, M_\emptyset \rangle$, where $M_\emptyset$ is a TM that rejects all inputs.
2. If R accepts, then accept; if R rejects, then reject.

S on input $\langle M \rangle$: (computable function $f : E_{\text{TM}} \to EQ_{\text{TM}}$)

1. Construct a TM encoding $\langle M_\emptyset \rangle$ such that $L(M_\emptyset) = \emptyset$.
2. Output: $\langle M, M_\emptyset \rangle$.

$$\langle M \rangle \in E_{\text{TM}} \iff f(\langle M \rangle) = \langle M, M_\emptyset \rangle \in EQ_{\text{TM}}$$

Not all undecidable reduction can be characterized by mapping reducibility

Theorem

E_{TM} is undecidable.

Reduction from A_{TM} to E_{TM}

S on input $\langle M, w \rangle$:

1. Construct the TM M_1 such that M accepts $w \iff L(M_1) \neq \emptyset$.
2. Run R on input $\langle M_1 \rangle$.
3. If R accepts (i.e., $L(M_1) = \emptyset$), then reject;
4. if R rejects (i.e., $L(M_1) \neq \emptyset$), then accept.

R decides E_{TM} iff S decides A_{TM}.

However, A_{TM} is not mapping reducible to E_{TM}

$$\langle M, w \rangle \in A_{\mathsf{TM}} \iff f(\langle M, w \rangle) = M' \notin E_{\mathsf{TM}}$$

Theorem

If $A \leq_{\mathrm{m}} B$ and B is Turing-recognizable, then A is Turing-recognizable.

Corollary

If $A \leq_{\mathrm{m}} B$ and A is not Turing-recognizable, then B is not Turing-recognizable.

Theorem

EQ_{TM} is neither Turing-recognizable nor co-Turing-recognizable.

Proof (1)

To show EQ_{TM} is not Turing-recognizable, we prove $A_{\mathsf{TM}} \leq_{\mathrm{m}} \overline{EQ_{\mathsf{TM}}}$:

F on input $\langle M, w \rangle$:

1. Construct the following two machines M_1 and M_2.
   1.1 M_1 reject any input.
   1.2 M_2 accepts any input if M accepts w.
2. Output $\langle M_1, M_2 \rangle$.

$$(M_1 \neq M_2) \iff w \in L(M)$$

Then

- $(A_{\mathsf{TM}} \leq_{\mathrm{m}} \overline{EQ_{\mathsf{TM}}}) \iff (\overline{A_{\mathsf{TM}}} \leq_{\mathrm{m}} EQ_{\mathsf{TM}})$
- Since $\overline{A_{\mathsf{TM}}}$ is known not-Turing-recognizable, then EQ_{TM} is not-Turing-recognizable

Proof (2)

To show $\overline{EQ_{\mathsf{TM}}}$ is not Turing-recognizable, we prove $A_{\mathsf{TM}} \leq_{\mathrm{m}} EQ_{\mathsf{TM}}$:

G on input $\langle M, w \rangle$:

1. Construct the following two machines M_1 and M_2.
   1.1 M_1 accepts any input.
   1.2 M_2 accepts any input if M accepts w.
2. Output $\langle M_1, M_2 \rangle$.

$$(M_1 = M_2) \iff w \in L(M)$$

Then

- $(A_{\mathsf{TM}} \leq_{\mathrm{m}} EQ_{\mathsf{TM}}) \iff (\overline{A_{\mathsf{TM}}} \leq_{\mathrm{m}} \overline{EQ_{\mathsf{TM}}})$
- $\overline{A_{\mathsf{TM}}}$ is known not-Turing-recognizable, then $\overline{EQ_{\mathsf{TM}}}$ is not-Turing-recognizable

Not all undecidable reduction can be characterized by mapping reducibility

Theorem

E_{TM} is undecidable.

Reduction from A_{TM} to E_{TM}

S on input $\langle M, w \rangle$:

1. Construct the TM M_1 such that M accepts $w \iff L(M_1) \neq \emptyset$.
2. Run R on input $\langle M_1 \rangle$.
3. If R accepts (i.e., $L(M_1) = \emptyset$), then reject;
4. if R rejects (i.e., $L(M_1) \neq \emptyset$), then accept.

R decides E_{TM} iff S decides A_{TM}.

However, A_{TM} is not mapping reducible to E_{TM}

$$\langle M, w \rangle \in A_{\mathsf{TM}} \iff f(\langle M, w \rangle) = M' \notin E_{\mathsf{TM}}$$

Do there exist any other proper mapping reduction?

Theorem

A_{TM} is not mapping reducible to E_{TM}

1. Assume A_{TM} is mapping reducible to E_{TM}, i.e., $A_{\mathsf{TM}} \leq_{\mathrm{m}} E_{\mathsf{TM}}$ via the computable function f
2. Then $\overline{A_{\mathsf{TM}}} \leq_{\mathrm{m}} \overline{E_{\mathsf{TM}}}$ via the computable function f
3. Since $\overline{A_{\mathsf{TM}}}$ is not-Turing-recognizable, then $\overline{E_{\mathsf{TM}}}$ is not-Turing-recognizable.
4. Consider the following TM S that recognizes $\overline{E_{\mathsf{TM}}}$.

S on input $\langle M \rangle$, where M is a TM

1. Repeat the following for $i = 1, 2, 3, \ldots$
2. Run M for i steps on each input, s_1, s_2, $\ldots$, s_j.
3. If M accepts some input s_j, then accept. Otherwise, continue forever.