Computability Theory

Tian-Ming Bu

East China Normal University

Outline

Outline

Preliminaries

Chapter 3: The Church-Turing Thesis

Chapter 4: Undecidability

Chapter 5: Reducibility

Strings and languages

- Alphabet: any nonempty finite set of symbols
- String over an alphabet: a finite sequence of symbols from that alphabet
- Length of a string w: $|w|$
- Empty string: ε
- Lexicographic ordering
- Language: set of strings

Finite and Infinite Sets

- Two sets A and B are equinumerous if there is a bijection $f : A \mapsto B$.
- A set is finite if it is equinumerous with $\{1, 2, \ldots, n\}$ for some natural number n.
- A set is infinite if it is not finite.
- A set is countably infinite if it is equinumerous with $\mathbb{N}$.
- A set is countable if it is finite or countably infinite.
- A set is uncountable if it is not countable.

Finite and Infinite Sets

- Two sets A and B are equinumerous if there is a bijection $f : A \mapsto B$.
- A set is finite if it is equinumerous with $\{1, 2, \ldots, n\}$ for some natural number n.
- A set is infinite if it is not finite.
- A set is countably infinite if it is equinumerous with $\mathbb{N}$.
- A set is countable if it is finite or countably infinite.
- A set is uncountable if it is not countable.

Theorem

For any alphabet Σ, the language Σ^* is countable.

Finite Automata

A ***finite automaton*** is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

1. Q is a finite set called the ***states***,
2. Σ is a finite set called the ***alphabet***,
3. $\delta \colon Q \times \Sigma \longrightarrow Q$ is the ***transition function***,[1]
4. $q_0 \in Q$ is the ***start state***, and
5. $F \subseteq Q$ is the ***set of accept states***.[2]

Finite Automata

- A configuration of a DFA is any element of $Q \times \Sigma^*$.
- We say (q, w) yields (q', w') in one step (written $(q, w) \vdash_M (q', w')$) if $\exists a \in \Sigma$ such that $w = aw'$ and $\delta(q, a) = q'$.
- M accepts w if there is a state $q \in F$ such that $(q_0, w) \vdash_M^* (q, \varepsilon)$.
- If A is the set of all strings that machine M accepts, we say A is the language of machine M and write $L(M) = A$.
- We say M recognizes A.
- DFA=NFA.
- A language is called a regular language if some finite automaton recognizes it.
- The language $B = \{0^n 1^n | n \geq 0\}$ is not a regular language.

Outline

Turing Machines

A ***Turing machine*** is a 7-tuple, $(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$, where Q, Σ, Γ are all finite sets and

1. Q is the set of states,
2. Σ is the input alphabet not containing the ***blank symbol*** $\sqcup$,
3. Γ is the tape alphabet, where $\sqcup \in \Gamma$ and $\Sigma \subseteq \Gamma$,
4. $\delta\colon Q \times \Gamma \longrightarrow Q \times \Gamma \times \{\mathrm{L}, \mathrm{R}\}$ is the transition function,
5. $q_0 \in Q$ is the start state,
6. $q_{\text{accept}} \in Q$ is the accept state, and
7. $q_{\text{reject}} \in Q$ is the reject state, where $q_{\text{reject}} \neq q_{\text{accept}}$.

Configuration

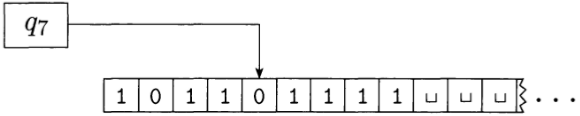


FIGURE 3.4
A Turing machine with configuration $1011q_701111$

- We say configuration C_1 yields configuration C_2 if the Turing machine can legally go from C_1 to C_2 in a single step.
- Start configuration, accepting configuration, rejecting configuration, halting configuration.
- A Turing machine M accepts input w if a sequence of configurations $C_1, \ldots, C_k$ exists, where
  1. C_1 is the start configuration of M on input w
  2. each C_i yields C_{i+1}
  3. C_k is an accepting configuration.

Languages

Definition

A language A is Turing-recognizable or recursively enumerable iff there exists some Turing machine M such that $\forall w, w \in A \Leftrightarrow M$ accepts w. We say A is recognized by M.

Definition

A language A is Turing-decidable or recursive iff there exists some Turing machine M such that $\forall w, w \in A \Rightarrow M$ accepts w, and $w \notin A \Rightarrow M$ rejects w. We say the language A is decided by the decider M.

Languages

Definition

A language A is Turing-recognizable or recursively enumerable iff there exists some Turing machine M such that $\forall w, w \in A \Leftrightarrow M$ accepts w. We say A is recognized by M.

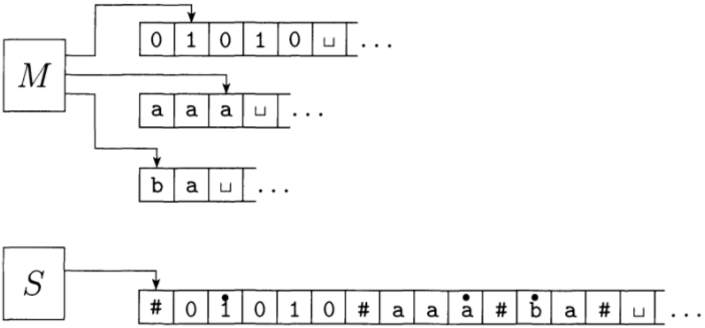
Definition

A language A is Turing-decidable or recursive iff there exists some Turing machine M such that $\forall w, w \in A \Rightarrow M$ accepts w, and $w \notin A \Rightarrow M$ rejects w. We say the language A is decided by the decider M.

Theorem

If a language A is Turing-decidable, it is Turing-recognizable.

Multitape Turing Machines



Nondeterministic Turing Machines

- $\delta : Q \times \Gamma \longrightarrow \mathcal{P}\{Q \times \Gamma \times \{L, R\}\}$
- A nondeterministic Turing machine M accepts input w if there exists a computation path from the start configuration to the accept configuration.
- A nondeterministic Turing machine M rejects input w if any computation path from the start configuration will lead to a reject configuration in finite steps.
- A language A is recognized by a nondeterministic Turing Machine M iff $\forall w, w \in A \Leftrightarrow M \text{ accepts } w$.
- A language A is decided by a nondeterministic Turing Machine M iff $\forall w, w \in A \Rightarrow M \text{ accepts } w$, and $w \notin A \Rightarrow M \text{ rejects } w$.
- A nondeterministic Turing Machine can be simulated by a deterministic Turing Machine in exponential time.

Hilbert's Tenth Problem

Let $D =$
$\{p | p$ is a integral coefficient polynomial with an integral root.$\}$.
Hilbert's tenth problem asks whether the set D is decidable.

Hilbert's Tenth Problem

Let $D =$
$\{p | p \text{ is a integral coefficient polynomial with an integral root.}\}$.
Hilbert's tenth problem asks whether the set D is decidable.

- D is Turing-recognizable.

Hilbert's Tenth Problem

Let $D =$
$\{p | p \text{ is a integral coefficient polynomial with an integral root.}\}$.
Hilbert's tenth problem asks whether the set D is decidable.

- D is Turing-recognizable.
- If p has only one variable, then D is Turing-decidable.

Hilbert's Tenth Problem

Let $D =$
$\{p | p \text{ is a integral coefficient polynomial with an integral root.}\}$.
Hilbert's tenth problem asks whether the set D is decidable.

- D is Turing-recognizable.
- If p has only one variable, then D is Turing-decidable.
- Yuri Matijasevič proved in 1970 that generally D is undecidable.

Outline

The Acceptance Problem

Theorem

Some languages are not Turing-recognizable.

The Acceptance Problem

Theorem

Some languages are not Turing-recognizable.

Definition (The Acceptance Problem)

$A_{\mathsf{TM}} = \{\langle M, w\rangle | M \text{ is a TM and } M \text{ accepts } w\}$.

The Acceptance Problem

Theorem
Some languages are not Turing-recognizable.

Definition (The Acceptance Problem)
$A_{\mathsf{TM}} = \{\langle M, w\rangle | M \text{ is a TM and } M \text{ accepts } w\}$.

- A_{TM} is Turing-recognizable.

The Acceptance Problem

Theorem
Some languages are not Turing-recognizable.

Definition (The Acceptance Problem)
$A_{\mathsf{TM}} = \{\langle M, w\rangle | M \text{ is a TM and } M \text{ accepts } w\}$.

- A_{TM} is Turing-recognizable.
- If A_{TM} is Turing decidable, then every Turing-recognizable language is Turing-decidable.

The Acceptance Problem

Theorem
Some languages are not Turing-recognizable.

Definition (The Acceptance Problem)
$A_{\text{TM}} = \{\langle M, w\rangle | M \text{ is a TM and } M \text{ accepts } w\}$.

- A_{TM} is Turing-recognizable.
- If A_{TM} is Turing decidable, then every Turing-recognizable language is Turing-decidable.
- A_{TM} is undecidable.

A_{TM} is undecidable

1.

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\cdots$
M_1	*accept*		*accept*		
M_2	*accept*	*accept*	*accept*	*accept*	
M_3					$\cdots$
M_4	*accept*	*accept*			
$\vdots$			$\vdots$		

A_{TM} is undecidable

1.

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\cdots$
M_1	*accept*		*accept*		
M_2	*accept*	*accept*	*accept*	*accept*	
M_3					$\cdots$
M_4	*accept*	*accept*			
$\vdots$		$\vdots$			

2. Define the language $A = \Sigma^* - \{\langle M_i \rangle | M_i \text{ accepts } \langle M_i \rangle\}$

A_{TM} is undecidable

1.

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\cdots$
M_1	*accept*		*accept*		
M_2	*accept*	*accept*	*accept*	*accept*	
M_3					$\cdots$
M_4	*accept*	*accept*			
$\vdots$		$\vdots$			

2. Define the language $A = \Sigma^* - \{\langle M_i \rangle | M_i \text{ accepts } \langle M_i \rangle\}$
3. A is unrecognizable. Thus, A is undecidable.

A_{TM} is undecidable

1.

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\cdots$
M_1	*accept*		*accept*		
M_2	*accept*	*accept*	*accept*	*accept*	
M_3					$\cdots$
M_4	*accept*	*accept*			
$\vdots$			$\vdots$		

2. Define the language $A = \Sigma^* - \{\langle M_i \rangle | M_i \text{ accepts } \langle M_i \rangle\}$
3. A is unrecognizable. Thus, A is undecidable.
4. $\overline{A}$ is undecidable.

A_{TM} is undecidable

1.

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\cdots$
M_1	*accept*		*accept*		
M_2	*accept*	*accept*	*accept*	*accept*	
M_3					$\cdots$
M_4	*accept*	*accept*			
$\vdots$		$\vdots$			

2. Define the language $A = \Sigma^* - \{\langle M_i \rangle | M_i \text{ accepts } \langle M_i \rangle\}$
3. A is unrecognizable. Thus, A is undecidable.
4. $\overline{A}$ is undecidable.
5. If A_{TM} is decidable, so is $\overline{A}$.

A_{TM} is undecidable

1.

	$\langle M_1 \rangle$	$\langle M_2 \rangle$	$\langle M_3 \rangle$	$\langle M_4 \rangle$	$\cdots$
M_1	*accept*		*accept*		
M_2	*accept*	*accept*	*accept*	*accept*	
M_3					$\cdots$
M_4	*accept*	*accept*			
$\vdots$			$\vdots$		

2. Define the language $A = \Sigma^* - \{\langle M_i \rangle | M_i \text{ accepts } \langle M_i \rangle\}$
3. A is unrecognizable. Thus, A is undecidable.
4. $\overline{A}$ is undecidable.
5. If A_{TM} is decidable, so is $\overline{A}$.
6. So both $\overline{A}$ and A_{TM} are undecidable.

Comments

1. $\overline{A_{\mathsf{TM}}}$ is unrecognizable.

Comments

1. $\overline{A_{\mathsf{TM}}}$ is unrecognizable.

2. $\overline{A}$ is recognizable.

Comments

1. $\overline{A_{\mathsf{TM}}}$ is unrecognizable.

2. $\overline{A}$ is recognizable.

3. A is unrecognizable.

Comments

1. $\overline{A_{\mathsf{TM}}}$ is unrecognizable.
2. $\overline{A}$ is recognizable.
3. A is unrecognizable.
4. The class of Turing-recognizable languages is not closed under complement.

Outline

Mapping Reducibility

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

Mapping Reducibility

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

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

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

The function f is called the ***reduction*** of A to B.

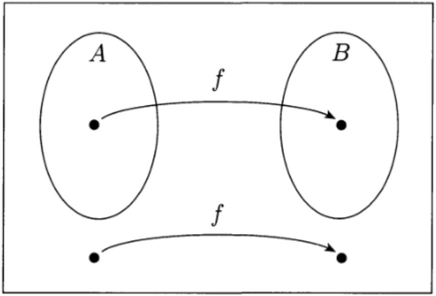
Mapping Reducibility

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

Language A is ***mapping reducible*** to language B, written $A \leq_m B$, if there is a computable function $f\colon \Sigma^* \longrightarrow \Sigma^*$, where for every w,

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

The function f is called the ***reduction*** of A to B.



Theorem

If $A \leq_m B$ *and* B *is decidable/recognizable, then* A *is decidable/recognizable.*

Theorem

If $A \leq_m B$ and B is decidable/recognizable, then A is decidable/recognizable.

Corollary

If $A \leq_m B$ and A is undecidable/unrecognizable, then B is undecidable/unrecognizable.

Theorem

If $A \leq_m B$ and B is decidable/recognizable, then A is decidable/recognizable.

Corollary

If $A \leq_m B$ and A is undecidable/unrecognizable, then B is undecidable/unrecognizable.

Lemma

If $A \leq_m B$, then $\overline{A} \leq_m \overline{B}$.

Turing Reducibility

An ***oracle*** for a language B is an external device that is capable of reporting whether any string w is a member of B. An ***oracle Turing machine*** is a modified Turing machine that has the additional capability of querying an oracle. We write M^B to describe an oracle Turing machine that has an oracle for language B.

Turing Reducibility

An ***oracle*** for a language B is an external device that is capable of reporting whether any string w is a member of B. An ***oracle Turing machine*** is a modified Turing machine that has the additional capability of querying an oracle. We write M^B to describe an oracle Turing machine that has an oracle for language B.

Language A is ***Turing reducible*** to language B, written $A \leq_{\mathrm{T}} B$, if A is decidable relative to B.

Turing Reducibility

An ***oracle*** for a language B is an external device that is capable of reporting whether any string w is a member of B. An ***oracle Turing machine*** is a modified Turing machine that has the additional capability of querying an oracle. We write M^B to describe an oracle Turing machine that has an oracle for language B.

Language A is ***Turing reducible*** to language B, written $A \leq_{\mathrm{T}} B$, if A is decidable relative to B.

Theorem

If $A \leq_T B$ and B is decidable, then A is decidable.

$HALT_{\mathsf{TM}}$

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

$HALT_{\mathsf{TM}}$

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

Theorem
$HALT_{TM}$ *is undecidable.*

$HALT_{\mathsf{TM}}$

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

Theorem

$HALT_{TM}$ is undecidable.

Proof.

$A_{\mathsf{TM}} \leq_T HALT_{\mathsf{TM}}$.

$HALT_{\mathsf{TM}}$

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

Theorem

$HALT_{TM}$ is undecidable.

Proof.

S: $A_{\mathsf{TM}} \leq_T \overset{R}{HALT_{\mathsf{TM}}}$.

$S =$ "On input $\langle M, w\rangle$, an encoding of a TM M and a string w:

1. Run TM R on input $\langle M, w\rangle$.
2. If R rejects, *reject*.
3. If R accepts, simulate M on w until it halts.
4. If M has accepted, *accept*; if M has rejected, *reject*."

$\square$

E_{TM}

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

E_{TM}

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

Theorem

E_{TM} is undecidable.

E_{TM}

$E_{\mathrm{TM}} = \{\langle M\rangle | M \text{ is a TM and } L(M) = \emptyset\}$.

Theorem

E_{TM} is undecidable.

Proof.

$A_{\mathrm{TM}} \leq_T E_{\mathrm{TM}}$.

E_{TM}

$E_{\mathrm{TM}} = \{\langle M \rangle | M \text{ is a TM and } L(M) = \emptyset\}$.

Theorem

E_{TM} *is undecidable.*

Proof.

$A_{\mathrm{TM}} \leq_T E_{\mathrm{TM}}$.

M_1 = "On input x:

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

E_{TM}

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

Theorem

E_{TM} is undecidable.

Proof.

$A_{\mathsf{TM}} \leq_T E_{\mathsf{TM}}$.

M_1 = "On input x:

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

S = "On input $\langle M, w\rangle$, an encoding of a TM M and a string w:

1. Use the description of M and w to construct the TM M_1 just described.
2. Run R on input $\langle M_1\rangle$.
3. If R accepts, *reject*; if R rejects, *accept*."

$\square$

$REGULAR_{\mathsf{TM}}$

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

$REGULAR_{\mathsf{TM}}$

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

Theorem
$REGULAR_{TM}$ is undecidable.

$REGULAR_{\mathsf{TM}}$

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

Theorem
$REGULAR_{TM}$ is undecidable.

Proof.
$A_{\mathsf{TM}} \leq_m REGULAR_{\mathsf{TM}}$.

$REGULAR_{\mathsf{TM}}$

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

Theorem
$REGULAR_{TM}$ is undecidable.

Proof.
$A_{\mathsf{TM}} \leq_m REGULAR_{\mathsf{TM}}$.
$S =$ "On input $\langle M, w \rangle$, where M is a TM and w is a string:

1. Construct the following TM M_2.
   $M_2 =$ "On input x:
   1. If x has the form $0^n 1^n$, *accept*.
   2. If x does not have this form, run M on input w and *accept* if M accepts w."
2. Run R on input $\langle M_2 \rangle$.
3. If R accepts, *accept*; if R rejects, *reject*."

□

EQ_{TM}

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

EQ_{TM}

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

Theorem

EQ_{TM} is undecidable.

EQ_{TM}

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

Theorem

EQ_{TM} is undecidable.

Proof.

$E_{\mathsf{TM}} \leq_m EQ_{\mathsf{TM}}$.

EQ_{TM}

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

Theorem

EQ_{TM} is undecidable.

Proof.

$E_{\mathrm{TM}} \leq_m EQ_{\mathrm{TM}}$.

$S =$ "On input $\langle M\rangle$, where M is a TM:

1. Run R on input $\langle M, M_1\rangle$, where M_1 is a TM that rejects all inputs.
2. If R accepts, *accept*; if R rejects, *reject*."

□

Theorem

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

Theorem

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

Proof.

We prove $\overline{A_{\mathrm{TM}}} \leq_{\mathrm{m}} EQ_{\mathrm{TM}}$ through $A_{\mathrm{TM}} \leq_{\mathrm{m}} \overline{EQ_{\mathrm{TM}}}$.

Theorem

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

Proof.

We prove $\overline{A_{\mathsf{TM}}} \leq_{\mathrm{m}} EQ_{\mathsf{TM}}$ through $A_{\mathsf{TM}} \leq_{\mathrm{m}} \overline{EQ_{\mathsf{TM}}}$.

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

1. Construct the following two machines M_1 and M_2.

$M_1 =$ "On any input:

1. *Reject*."

$M_2 =$ "On any input:

1. Run M on w. If it accepts, *accept*."

2. Output $\langle M_1, M_2\rangle$."

Theorem

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

Proof.

We prove $\overline{A_{\mathsf{TM}}} \leq_{\mathsf{m}} EQ_{\mathsf{TM}}$ through $A_{\mathsf{TM}} \leq_{\mathsf{m}} \overline{EQ_{\mathsf{TM}}}$.

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

1. Construct the following two machines M_1 and M_2.

   $M_1 =$ "On any input:

   1. *Reject*."

   $M_2 =$ "On any input:

   1. Run M on w. If it accepts, *accept*."

2. Output $\langle M_1, M_2\rangle$."

We prove $\overline{A_{\mathsf{TM}}} \leq_{\mathsf{m}} \overline{EQ_{\mathsf{TM}}}$ through $A_{\mathsf{TM}} \leq_{\mathsf{m}} EQ_{\mathsf{TM}}$.

Theorem

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

Proof.

We prove $\overline{A_{\mathsf{TM}}} \leq_{\mathsf{m}} EQ_{\mathsf{TM}}$ through $A_{\mathsf{TM}} \leq_{\mathsf{m}} \overline{EQ_{\mathsf{TM}}}$.

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

1. Construct the following two machines M_1 and M_2.

   $M_1 =$ "On any input:

   1. *Reject*."

   $M_2 =$ "On any input:

   1. Run M on w. If it accepts, *accept*."
2. Output $\langle M_1, M_2 \rangle$."

We prove $\overline{A_{\mathsf{TM}}} \leq_{\mathsf{m}} \overline{EQ_{\mathsf{TM}}}$ through $A_{\mathsf{TM}} \leq_{\mathsf{m}} EQ_{\mathsf{TM}}$.

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

1. Construct the following two machines M_1 and M_2.

   $M_1 =$ "On any input:

   1. *Accept*."

   $M_2 =$ "On any input:

   1. Run M on w.
   2. If it accepts, *accept*."
2. Output $\langle M_1, M_2 \rangle$."

$\square$