# Computability and Complexity

Lecture 7

Computable functions
Mapping reducibility
Results and examples

given by Jiri Srba

# Proof Template for Showing Undecidability by Reduction

We know that language $A$ is undecidable. By reducing $A$ to $B$ we want to show that the language $B$ too.

# Proof Template for Showing Undecidability by Reduction

We know that language $A$ is undecidable. By reducing $A$ to $B$ we want to show that the language $B$ too.

1. By contradiction assume that we have a decider $M_B$ for $B$.
2. Using $M_B$ we construct a decider $M_A$ for the language $A$:

$M_A =$ "On input $\langle I_A \rangle$:

1. Algorithmically construct an input $\langle I_B \rangle$ for $M_B$ s.t.:

a) if $\langle I_A \rangle \in A$ then $\langle I_B \rangle \in B$
if $\langle I_A \rangle \notin A$ then $\langle I_B \rangle \notin B$

2. Run the decider $M_B$ on $\langle I_B \rangle$:
Case a): $M_A$ accepts iff $M_B$ accepted.
"

3. We know that $M_A$ cannot exist, so $M_B$ cannot exist either.
4. Conclusion: the language $B$ is undecidable.

# Proof Template for Showing Undecidability by Reduction

We know that language $A$ is undecidable. By reducing $A$ to $B$ we want to show that the language $B$ too.

1. By contradiction assume that we have a decider $M_B$ for $B$.
2. Using $M_B$ we construct a decider $M_A$ for the language $A$:

$M_A =$ "On input $\langle I_A \rangle$:

1. Algorithmically construct an input $\langle I_B \rangle$ for $M_B$ s.t.:

a) if $\langle I_A \rangle \in A$ then $\langle I_B \rangle \in B$
if $\langle I_A \rangle \notin A$ then $\langle I_B \rangle \notin B$

b) if $\langle I_A \rangle \in A$ then $\langle I_B \rangle \notin B$
if $\langle I_A \rangle \notin A$ then $\langle I_B \rangle \in B$

2. Run the decider $M_B$ on $\langle I_B \rangle$:
Case a): $M_A$ accepts iff $M_B$ accepted.
Case b): $M_A$ accepts iff $M_B$ rejected."

3. We know that $M_A$ cannot exist, so $M_B$ cannot exist either.
4. Conclusion: the language $B$ is undecidable.

# Computable Function

Idea: Turing machines can also compute functions $f : \Sigma^* \to \Sigma^*$.

# Computable Function

Idea: Turing machines can also compute functions $f : \Sigma^* \to \Sigma^*$.

**Definition (Computable Function)**

A function $f : \Sigma^* \to \Sigma^*$ is a computable function iff there exists a TM $M_f$ which on any given input $w \in \Sigma^*$

- always halts, and
- leaves just $f(w)$ on its tape.

# Computable Function

Idea: Turing machines can also compute functions $f : \Sigma^* \to \Sigma^*$.

**Definition (Computable Function)**

A function $f : \Sigma^* \to \Sigma^*$ is a computable function iff there exists a TM $M_f$ which on any given input $w \in \Sigma^*$

- always halts, and
- leaves just $f(w)$ on its tape.

Examples:

- Let $f(w) \stackrel{\text{def}}{=} ww$ be a function. Then $f$ is computable.
- Let $f(\langle n_1, n_2 \rangle) \stackrel{\text{def}}{=} \langle n \rangle$ where $n_1$ and $n_2$ are integers and $n = n_1 * n_2$. Then $f$ is computable.

# Example of a Computable Function

$$f(w) \stackrel{\text{def}}{=} \begin{cases} \langle M' \rangle & \text{if } w = \langle M \rangle \text{ and } M \text{ is a TM, and } M' \text{ is} \\ & \text{a never rejecting TM s.t. } L(M) = L(M') \\ & \\ \epsilon & \text{if } w \text{ is not an encoding of a TM} \end{cases}$$

# Example of a Computable Function

$$f(w) \stackrel{\text{def}}{=} \begin{cases} \langle M' \rangle & \text{if } w = \langle M \rangle \text{ and } M \text{ is a TM, and } M' \text{ is} \\ & \text{a never rejecting TM s.t. } L(M) = L(M') \\ \epsilon & \text{if } w \text{ is not an encoding of a TM} \end{cases}$$

**Function $f$ is computable by the following TM $M_f$:**

"On input $w$:

1. If $w$ is not an encoding of a TM, erase the tape and accept.
2. If $w = \langle M \rangle$ for some TM $M$ then replace every occurrence of $q_{reject}$ in $\langle M \rangle$ (which is on the tape) with a newly added state $q_{loop}$, and add the transitions $\delta(q_{loop}, a) = (q_{loop}, a, R)$ for all tape symbols $a$.
3. Accept."

# Mapping Reducibility

**Definition (Mapping Reducibility, $A \leq_m B$)**

Let $A, B \subseteq \Sigma^*$. We say that language $A$ is mapping reducible to language $B$, written $A \leq_m B$, iff

1. there is a computable function $f : \Sigma^* \to \Sigma^*$ such that
2. for every $w \in \Sigma^*$:

$$w \in A \quad \text{if and only if} \quad f(w) \in B$$

or equivalently:

- if $w \in A$ then $f(w) \in B$, and
- if $w \notin A$ then $f(w) \notin B$.

Remark:
This kind of reducibility is also called many-one reducibility.

**Theorem**

$A_{TM} \leq_m HALT_{TM}$

Proof: Construct a computable function $f$ which on input $\langle M, w \rangle$ returns $\langle M', w' \rangle$ such that

$$\langle M, w \rangle \in A_{TM} \text{ if and only if } \langle M', w' \rangle \in HALT_{TM}.$$

# Example: $A_{TM} \leq_m HALT_{TM}$

**Theorem**

$A_{TM} \leq_m HALT_{TM}$

Proof: Construct a computable function $f$ which on input $\langle M, w \rangle$ returns $\langle M', w' \rangle$ such that

$$\langle M, w \rangle \in A_{TM} \text{ if and only if } \langle M', w' \rangle \in HALT_{TM}.$$

$M_f =$ "On input $\langle M, w \rangle$:

1. Construct the following machine $M'$:

   $M' =$ "On input $x$:

   1. Run $M$ on $x$.
   2. If $M$ accepted, $M'$ accepts.
   3. If $M$ rejected, $M'$ loops."

2. Output $\langle M', w \rangle$."

# Other Examples

In previous lecture we showed that:

- $A_{TM} \leq_m MPCP$
- $MPCP \leq_m PCP$

# Results Concerning Mapping Reducibility

## Main Theorem

Assume that $A \leq_m B$. Then:

1. If $B$ is decidable then $A$ is decidable.

## Main Theorem

Assume that $A \leq_m B$. Then:

1. If $B$ is decidable then $A$ is decidable.
2. If $A$ is undecidable then $B$ is undecidable.

# Results Concerning Mapping Reducibility

## Main Theorem

Assume that $A \leq_m B$. Then:

1. If $B$ is decidable then $A$ is decidable.
2. If $A$ is undecidable then $B$ is undecidable.
3. If $B$ is recognizable then $A$ is recognizable.

# Results Concerning Mapping Reducibility

## Main Theorem

Assume that $A \leq_m B$. Then:

1. If $B$ is decidable then $A$ is decidable.
2. If $A$ is undecidable then $B$ is undecidable.
3. If $B$ is recognizable then $A$ is recognizable.
4. If $A$ is not recognizable then $B$ is not recognizable.

# Results Concerning Mapping Reducibility

## Main Theorem

Assume that $A \leq_m B$. Then:

1. If $B$ is decidable then $A$ is decidable.
2. If $A$ is undecidable then $B$ is undecidable.
3. If $B$ is recognizable then $A$ is recognizable.
4. If $A$ is not recognizable then $B$ is not recognizable.
5. $\overline{A} \leq_m \overline{B}$

# Results Concerning Mapping Reducibility

**Main Theorem**

Assume that $A \leq_m B$. Then:

1. If $B$ is decidable then $A$ is decidable.
2. If $A$ is undecidable then $B$ is undecidable.
3. If $B$ is recognizable then $A$ is recognizable.
4. If $A$ is not recognizable then $B$ is not recognizable.
5. $\overline{A} \leq_m \overline{B}$

Useful observation:

- Let us take some non-recognizable language, like e.g. $\overline{A_{TM}}$.
- If for some language $B$ we show that $\overline{A_{TM}} \leq_m B$ and $\overline{A_{TM}} \leq_m \overline{B}$ then $B$ is not recognizable, neither co-recognizable.

# Example of Use

$$EQ_{TM} \stackrel{\text{def}}{=} \{\langle M_1, M_2\rangle \mid \ M_1 \text{ and } M_2 \text{ are TMs s.t. } L(M_1) = L(M_2)\ \}$$

**Theorem**

The language $EQ_{TM}$ is neither recognizable nor co-recognizable.

Proof:

We show that

- $\overline{A_{TM}} \leq_m EQ_{TM}$, and
- $\overline{A_{TM}} \leq_m \overline{EQ_{TM}}$.

# Proof: $\overline{A_{TM}} \leq_m EQ_{TM}$

We prove that $\overline{A_{TM}} \leq_m EQ_{TM}$ by showing that

$$A_{TM} \leq_m \overline{EQ_{TM}}.$$

# Proof: $\overline{A_{TM}} \leq_m EQ_{TM}$

We prove that $\overline{A_{TM}} \leq_m EQ_{TM}$ by showing that

$$A_{TM} \leq_m \overline{EQ_{TM}}.$$

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

1. Construct the following machines $M_1$ and $M_2$:

   $M_1 =$ "On input $x$: reject"

   $M_2 =$ "On input $x$: erase $x$; Run $M$ on $w$ and $M_2$ accepts iff $M$ accepted."

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

# Proof: $\overline{A_{TM}} \leq_m EQ_{TM}$

We prove that $\overline{A_{TM}} \leq_m EQ_{TM}$ by showing that

$$A_{TM} \leq_m \overline{EQ_{TM}}.$$

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

1. Construct the following machines $M_1$ and $M_2$:
   $M_1$ = "On input $x$: reject"
   $M_2$ = "On input $x$: erase $x$; Run $M$ on $w$ and
   $M_2$ accepts iff $M$ accepted."
2. Output $\langle M_1, M_2 \rangle$."

Note: $M_f$ halts for any given input.

**Observe:**

- If $M$ accepts $w$ then $L(M_1) \neq L(M_2)$.
- If $M$ does not accept $w$ then $L(M_1) = L(M_2)$.

Hence $f$ is a mapping reduction from $A_{TM}$ to $\overline{EQ_{TM}}$.

# Proof: $\overline{A_{TM}} \leq_m \overline{EQ_{TM}}$

We prove that $\overline{A_{TM}} \leq_m \overline{EQ_{TM}}$ by showing that

$$A_{TM} \leq_m EQ_{TM}.$$

# Proof: $\overline{A_{TM}} \leq_m \overline{EQ_{TM}}$

We prove that $\overline{A_{TM}} \leq_m \overline{EQ_{TM}}$ by showing that

$$A_{TM} \leq_m EQ_{TM}.$$

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

1. Construct the following machines $M_1$ and $M_2$:

   $M_1 =$ "On input $x$: accept"

   $M_2 =$ "On input $x$: erase $x$; Run $M$ on $w$ and $M_2$ accepts iff $M$ accepted."

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

# Proof: $\overline{A_{TM}} \leq_m \overline{EQ_{TM}}$

We prove that $\overline{A_{TM}} \leq_m \overline{EQ_{TM}}$ by showing that

$$A_{TM} \leq_m EQ_{TM}.$$

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

1. Construct the following machines $M_1$ and $M_2$:

   $M_1 =$ "On input $x$: accept"

   $M_2 =$ "On input $x$: erase $x$; Run $M$ on $w$ and $M_2$ accepts iff $M$ accepted."

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

Note: $M_f$ halts for any given input.

**Observe:**

- If $M$ accepts $w$ then $L(M_1) = L(M_2)$.
- If $M$ does not accept $w$ then $L(M_1) \neq L(M_2)$.

Hence $f$ is a mapping reduction from $A_{TM}$ to $EQ_{TM}$.

# Exam Questions

- Definitions of computable function and mapping reducibility.
- Examples of mapping reducibility.
- Main theorem (5 parts).
- Technique to show that a language is not recognizable nor co-recognizable.