CS 483 - Introduction to the Theory of Computation

Lecture 15

Mapping Reduction

Reducing problem A to problem B means that we can use the solution of problem B to answer problem A.

Mapping Reduction

Reducing problem A to problem B means that we can use the solution of problem B to answer problem A.

A mapping reduction (or many-one reduction) from language $A \in \Sigma_1^*$ to language $B \in \Sigma_2^*$ is a **computable** function: $f: \Sigma_1^* \to \Sigma_2^*$, such that

$$x \in A \Leftrightarrow f(x) \in B.$$

In other words, $f(A) \subseteq B$ and $f(\overline{A}) \subseteq \overline{B}$.

We write $A \leq_{\mathrm{m}} B$ if there is a mapping reduction from A to B.

Mapping Reduction

Theorem. If $A \leq_{\mathrm{m}} B$ and B is decidable (resp. recognizable), then A is also decidable (resp. recognizable).

Mapping Reduction

Theorem. If $A \leq_{\mathrm{m}} B$ and B is decidable (resp. recognizable), then A is also decidable (resp. recognizable).

(Equivalently we say that **R** and **RE** are both closed under $\leq_{\mathrm{m}}$.)

Mapping Reduction

Theorem. If $A \leq_{\mathrm{m}} B$ and B is decidable (resp. recognizable), then A is also decidable (resp. recognizable).

Proof. Assume B is decided (recognized) by M, and f is the reduction from A to B. Construct $M'(x)$ that decides (recognizes) A:

- Compute $f(x)$;
- Run $M(f(x))$, and accept or reject accordingly.

Mapping Reduction

Theorem. If $A \leq_{\mathrm{m}} B$ and B is decidable (resp. recognizable), then A is also decidable (resp. recognizable).

Proof. Assume B is decided (recognized) by M, and f is the reduction from A to B. Construct $M'(x)$ that decides (recognizes) A:

- Compute $f(x)$;
- Run $M(f(x))$, and accept or reject accordingly.

Corollary. If $A \leq_{\mathrm{m}} B$ and A is undecidable (resp. unrecognizable), then B is also undecidable (resp. unrecognizable).

Mapping Reduction

Example 1:

$$L' = \{ M \mid M \text{ is a Turing machine}, M(M) = 1 \} \leq_{\mathrm{m}} A_{\mathrm{TM}}$$

Mapping Reduction

Example 1:

$$L' = \{ M \mid M \text{ is a Turing machine}, M(M) = 1 \} \leq_{\mathrm{m}} A_{\mathrm{TM}}$$

Let $f(M) = (M, M)$. Then f is computable, and

$$M \in L \Leftrightarrow (M, M) \in A_{\mathrm{TM}}$$

Mapping Reduction

Example 2:

$$\text{HALT} \leq_{\text{m}} A_{\text{TM}}$$

Mapping Reduction

Example 2:

$$\text{HALT} \leq_{\text{m}} A_{\text{TM}}$$

Let $f(M, w) = (M', w)$, where $M'(x)$ runs $M(x)$ and accepts if it halted.

Then f is computable and

$$(M, w) \in \text{HALT} \Leftrightarrow (M', w) \in A_{\text{TM}}$$

Mapping Reduction

Example 3:

$$A_{\mathrm{TM}} \leq_{\mathrm{m}} \mathrm{HALT}$$

Mapping Reduction

Example 3:

$$A_{\mathrm{TM}} \leq_{\mathrm{m}} \textsc{Halt}$$

"If $(M, w) \notin \textsc{Halt}$, reject; otherwise run $U(M, w)$" is NOT mapping reduction!

Mapping Reduction

Example 3:

$$A_{\mathrm{TM}} \leq_{\mathrm{m}} \textsc{Halt}$$

"If $(M, w) \notin \textsc{Halt}$, reject; otherwise run $U(M, w)$" is NOT mapping reduction!

Let $f(M, w) = (M', w)$ where $M'(w)$:

- Run $M(w)$;
  - If it accepted, accept;
  - If it rejected, transition into an infinite loop.

Then f is computable and $(M, w) \in A_{\mathrm{TM}} \Leftrightarrow (M', w) \in \textsc{Halt}$

RE-Completeness

A language L is **RE**-hard, if $L' \leq_{\mathbf{m}} L$ for every $L' \in$ **RE**.

(In other words, L is at least as hard as every problem in **RE**.)

RE-Completeness

A language L is **RE**-hard, if $L' \leq_{\mathrm{m}} L$ for every $L' \in$ **RE**.

(In other words, L is at least as hard as every problem in **RE**.)

A language L is **RE**-complete, if L is **RE**-hard and $L \in$ **RE**.

(In other words, L completely characterizes **RE**: $\{ L' \mid L' \leq_{\mathrm{m}} L \}$)

RE-Completeness

A language L is **RE**-hard, if $L' \leq_{\mathrm{m}} L$ for every $L' \in$ **RE**.

(In other words, L is at least as hard as every problem in **RE**.)

A language L is **RE**-complete, if L is **RE**-hard and $L \in$ **RE**.

(In other words, L completely characterizes **RE**: $\{ L' \mid L' \leq_{\mathrm{m}} L \}$)

The same definition of hardness and completeness goes for every class of languages $\mathcal{C}$.

(with a specified type of reduction that $\mathcal{C}$ is closed under)

RE-Completeness

Theorem. A_{TM} is **RE**-complete.

We know that $A_{\mathrm{TM}} \in$ **RE**; only need to show $L \leq_{\mathrm{m}} A_{\mathrm{TM}}$ for every $L \in$ **RE**.

RE-Completeness

Theorem. A_{TM} is **RE**-complete.

We know that $A_{\mathrm{TM}} \in$ **RE**; only need to show $L \leq_{\mathrm{m}} A_{\mathrm{TM}}$ for every $L \in$ **RE**.

Suppose L is recognized by M. Let $f(w) = (M, w)$.

Then f is computable (as $\langle M \rangle$ is a fixed string) and

$$w \in L \Leftrightarrow M(w) = 1 \Leftrightarrow (M, w) \in A_{\mathrm{TM}}$$

RE-Completeness

Corollary. HALT is **RE**-complete.

RE-Completeness

Corollary. HALT is **RE**-complete.

This is because the mapping reduction is transitive:

If $A \leq_{\mathrm{m}} B$ and $B \leq_{\mathrm{m}} C$ then $A \leq_{\mathrm{m}} C$.

RE-Completeness

Corollary. HALT is **RE**-complete.

This is because the mapping reduction is transitive:

$$\text{If } A \leq_{\mathrm{m}} B \text{ and } B \leq_{\mathrm{m}} C \text{ then } A \leq_{\mathrm{m}} C.$$

Let f be the reduction from A to B, and g be the reduction from B to C.
Then $g \circ f$ is a reduction from A to C:

$$x \in A \Leftrightarrow f(x) \in B \Leftrightarrow g(f(x)) \in C.$$

RE-Hardness

Theorem. The following language is **RE**-hard:

$$\textsc{Halt} \times \overline{\textsc{Halt}} = \{ (M_1, w_1, M_2, w_2) \mid M_1(w_1) \neq \bot, M_2(w_2) = \bot \}$$

RE-Hardness

Theorem. The following language is **RE**-hard:

$$\textsc{Halt} \times \overline{\textsc{Halt}} = \{ (M_1, w_1, M_2, w_2) \mid M_1(w_1) \neq \bot, M_2(w_2) = \bot \}$$

Fix some $(M_2, w_2) \in \overline{\textsc{Halt}}$. Then $f(M, w) = (M, w, M_2, w_2)$ is a mapping reduction from $\textsc{Halt}$ to $\textsc{Halt} \times \overline{\textsc{Halt}}$.

RE-Hardness

Theorem. The following language is **RE**-hard:

$$\textsc{Halt} \times \overline{\textsc{Halt}} = \{ (M_1, w_1, M_2, w_2) \mid M_1(w_1) \neq \bot, M_2(w_2) = \bot \}$$

Fix some $(M_2, w_2) \in \overline{\textsc{Halt}}$. Then $f(M, w) = (M, w, M_2, w_2)$ is a mapping reduction from $\textsc{Halt}$ to $\textsc{Halt} \times \overline{\textsc{Halt}}$.

For the same reason, there is also a mapping reduction from $\overline{\textsc{Halt}}$ to $\textsc{Halt} \times \overline{\textsc{Halt}}$, so $\textsc{Halt} \times \overline{\textsc{Halt}} \notin$ **RE**.

R-Hardness?

Hardness and completeness for **R** under mapping reductions is trivial:

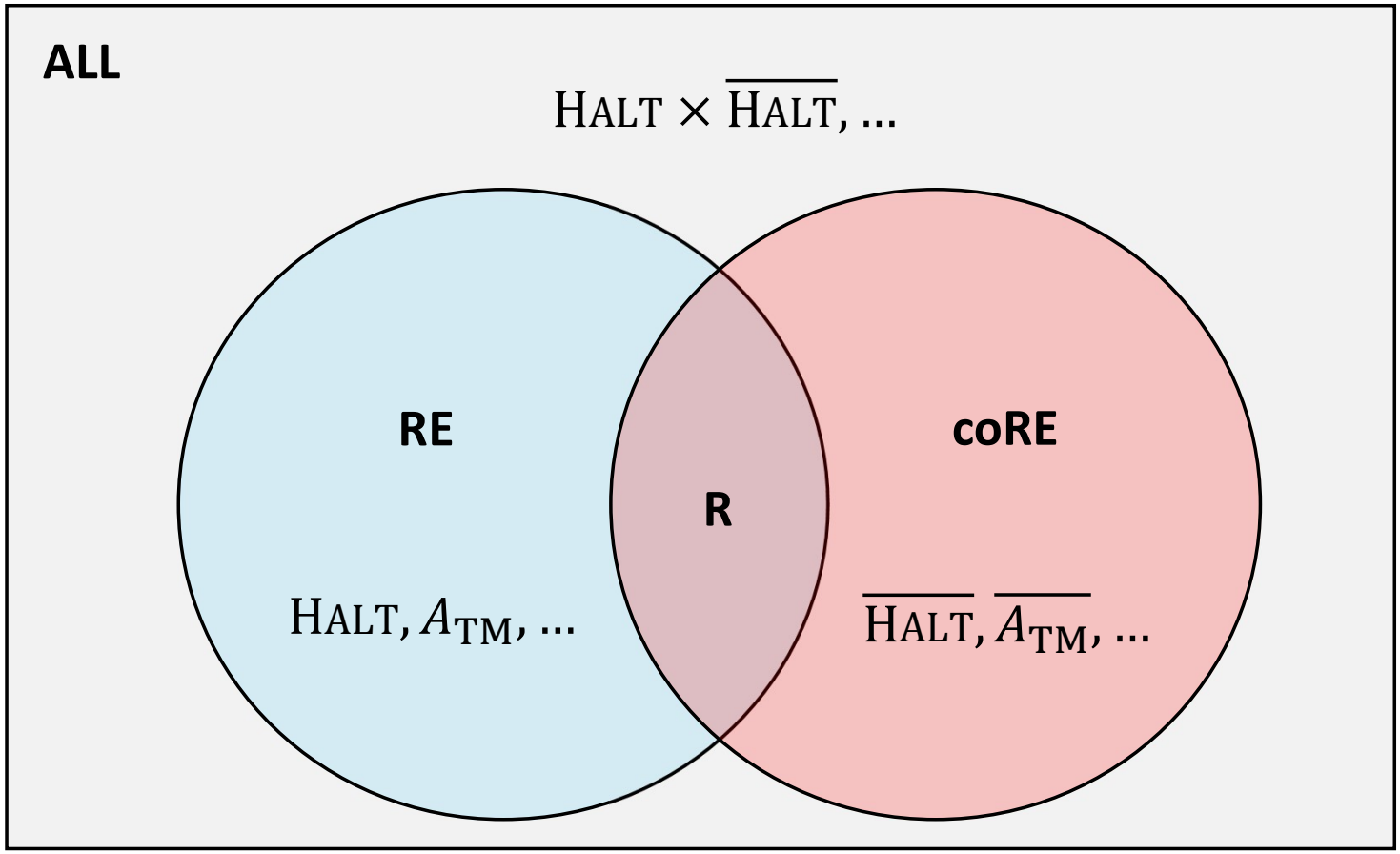
Every language $L \notin \{\emptyset, \Sigma^*\}$ is **R**-hard.

Take $L' \in \mathbf{R}$, and fix $x \in L$, $y \in \overline{L}$. Then

$$f(w) = \begin{cases} x & w \in L' \\ y & w \notin L' \end{cases}$$

is a mapping reduction from L' to L.

The Venn Diagram



Rice's Theorem

Theorem. Let $L \subseteq \mathrm{TM} = \{ M \mid M \text{ is a Turing machine} \}$ be a non-trivial property of Turing machines, i.e.

- $L \notin \{\emptyset, \mathrm{TM}\}$;
- Whenever $M_1(x) = M_2(x)$ for every x, we have $M_1 \in L \Leftrightarrow M_2 \in L$.

Then either L or $\overline{L}$ is **RE**-hard (and thus L is undecidable).

Rice's Theorem

Proof. Let $M_{\mathbf{no_halt}}$ be a Turing machine that never halts on any input.

Assume $M_{\mathbf{no_halt}} \notin L$ and take arbitrary $M_{\mathbf{a}} \in L$. We will show $\textsc{Halt} \leq_{\mathbf{m}} L$.

(If $M_{\mathbf{no_halt}} \in L$, we can instead show $\textsc{Halt} \leq_{\mathbf{m}} \overline{L}$)

Rice's Theorem

Proof. Let $M_{\mathbf{no_halt}}$ be a Turing machine that never halts on any input.

Assume $M_{\mathbf{no_halt}} \notin L$ and take arbitrary $M_{\mathbf{a}} \in L$. We will show $\textsc{Halt} \leq_{\mathbf{m}} L$.

Let $f(M, w) = M'_{M,w}$, where $M'_{M,w}$ is the Turing machine that on input y:

- Run $M(w)$ (with hardwired M and w);
  - If it halted then run $M_{\mathbf{a}}(y)$ and output accordingly.

Rice's Theorem

Proof. Let $M_{\mathbf{no_halt}}$ be a Turing machine that never halts on any input.

Assume $M_{\mathbf{no_halt}} \notin L$ and take arbitrary $M_{\mathbf{a}} \in L$. We will show $\textsc{Halt} \leq_{\mathbf{m}} L$.

Let $f(M, w) = M'_{M,w}$, where $M'_{M,w}$ is the Turing machine that on input y:

- Run $M(w)$ (with hardwired M and w);
  - If it halted then run $M_{\mathbf{a}}(y)$ and output accordingly.

If $(M, w) \in \textsc{Halt}$, then $M'_{M,w}(y) = M_{\mathbf{a}}(y)$ for every y, so $M'_{M,w} \in L$.

If $(M, w) \notin \textsc{Halt}$, then $M'_{M,w}(y) = M_{\mathbf{no_halt}}(y)$ for every y, so $M'_{M,w} \notin L$.

Rice's Theorem

Examples:

- $\{ M \mid \text{TM } M \text{ accepts } \varepsilon \}$

Rice's Theorem

Examples:

- $\{ M \mid \text{TM } M \text{ accepts } \varepsilon \}$ is **RE**-complete.
- $\{ M \mid \text{TM } M \text{ halts on some } w \text{ with even length} \}$

Rice's Theorem

Examples:

- $\{ M \mid \text{TM } M \text{ accepts } \varepsilon \}$ is **RE**-complete.
- $\{ M \mid \text{TM } M \text{ halts on some } w \text{ with even length} \}$ is **RE**-complete.
- $\{ M \mid \text{TM } M \text{ does not halt on any input} \}$

Rice's Theorem

Examples:

- $\{ M \mid \text{TM } M \text{ accepts } \varepsilon \}$ is **RE**-complete.
- $\{ M \mid \text{TM } M \text{ halts on some } w \text{ with even length} \}$ is **RE**-complete.
- $\{ M \mid \text{TM } M \text{ does not halt on any input} \}$ is **coRE**-complete.
- $\text{ALL}_{\text{TM}} = \{ M \mid M \text{ is a TM}, L(M) = \Sigma^* \}$

Rice's Theorem

Examples:

- $\{ M \mid \text{TM } M \text{ accepts } \varepsilon \}$ is **RE**-complete.
- $\{ M \mid \text{TM } M \text{ halts on some } w \text{ with even length} \}$ is **RE**-complete.
- $\{ M \mid \text{TM } M \text{ does not halt on any input} \}$ is **coRE**-complete.
- $\text{ALL}_{\text{TM}} = \{ M \mid M \text{ is a TM}, L(M) = \Sigma^* \}$ is **RE**-hard and **coRE**-hard

We can show $\overline{\text{HALT}} \leq_{\text{m}} \text{ALL}_{\text{TM}}$. Let $f(M, w)$ be the Turing machine $M'(t)$:

- Run $M(w)$ for t steps. Accept if it didn't halt in t steps.

Rice's Theorem

Can we use Rice's theorem on

$$L' = \{ M \mid M \text{ is a Turing machine}, M(M) = 1 \}?$$

Rice's Theorem

Can we use Rice's theorem on

$$L' = \{ M \mid M \text{ is a Turing machine}, M(M) = 1 \}?$$

No, because L' is NOT a property of Turing machines.

For instance, $M_1(M)$ decides whether M contains at most 100 states.

We can implement M_1 efficiently so that $M_1 \in L'$, but there is another equivalent M_2 such that $M_2(M_2) = M_1(M_2) = 0$ so $M_2 \notin L'$.