Mapping Reducibility
Formalism for Reducibility

Clarifies the previously-seen Reducibility approaches. (Is a.k.a. many-one reducibility)

Mapping Reducibility is the use of a computable function to convert instances of problem A to instances of problem B.

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

**Example:** arithmetic operator +
- **Input:** $\langle m, n \rangle$
- **Output:** sum of $m$ and $n$

**Example:** TM that never moves left off tape.
- **Input:** $\langle M \rangle$
- **Output:** $\langle M' \rangle$ where $L(M) = L(M')$
Language $A$ is mapping reducible to language $B$, written $A \leq_m B$, if there is a computable function $f : \Sigma^* \rightarrow \Sigma^*$, where for every $w$, $w \in A \iff f(w) \in B$.

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

A mapping reduction of $A$ to $B$ provides a way to convert questions about membership testing in $A$ to membership testing in $B$.

To test whether $w \in A$, we use the reduction $f$ to map $w$ to $f(w)$ and test whether $f(w) \in B$.

**Question:** Why is this called a reduction?

**Note:** Mapping Reducibility may seem like a repeat of previous lectures (and, granted, it is very similar), but there are a few important subtleties which we will address throughout the lecture.
Theorem: Decidability and Undecidability

“If \( A \leq_m B \) and \( B \) is decidable, then \( A \) is decidable.”

**PROOF:**
We let \( M \) be the decider for \( B \) and \( f \) be the reduction from \( A \) to \( B \). We describe a decider \( N \) for \( A \) as follows.

\[ N = “\text{On input } w:\text{ 1. Compute } f(w).\text{ 2. Run } M \text{ on input } f(w) \text{ and output whatever } M \text{ outputs.”} \]

Clearly, if \( w \in A \), then \( f(w) \in B \) because \( f \) is a reduction from \( A \) to \( B \). Thus, \( M \) accepts \( f(w) \) whenever \( w \in A \). Therefore, \( N \) works as desired.

Corollary:
“If \( A \leq_m B \) and \( A \) is undecidable, then \( B \) is undecidable.”

Also:
“If \( A \leq_m B \) and \( B \) is recognizable, then \( A \) is recognizable.”

Also:
“If \( A \leq_m B \) and \( A \) is unrecognizable, then \( B \) is unrecognizable.”

**BUT!**
What if \( B \) is undecidable? What does that prove about \( A \)?
What if \( A \) is decidable?
Theorem: Decidability and Undecidability

“If $A \leq_m B$ and $B$ is decidable, then $A$ is decidable.”

**PROOF:**
We let $M$ be the decider for $B$ and $f$ be the reduction from $A$ to $B$. We describe a decider $N$ for $A$ as follows.

1. Compute $f(w)$.
2. Run $M$ on input $f(w)$ and output whatever $M$ outputs.

Clearly, if $w \in A$, then $f(w) \in B$ because $f$ is a reduction from $A$ to $B$ whenever $w \in A$, desired.

**Corollary:**
“If $A \leq_m B$ and $A$ is undecidable, then $B$ is undecidable.”

Also:
“If $A \leq_m B$ and $B$ is recognizable, then $A$ is recognizable.”

Also:
“If $A \leq_m B$ and $A$ is unrecognizable, then $B$ is unrecognizable.”

**BUT!**
What if $B$ is undecidable? What does that prove about $A$?
What if $A$ is decidable?
Theorem - \( \text{HALT}_{TM} \) is undecidable

Original Method:

\( \text{HALT}_{TM} = \{ \langle M, w \rangle \mid M \text{ is a TM and } M \text{ halts on input } w \} \)

Let's assume that TM \( R \) decides \( \text{HALT}_{TM} \). Construct TM \( S \) to decide \( A_{TM} \) as follows.

\( S = \text{“On input } \langle M, w \rangle \text{, an encoding of a TM } M \text{ and a string } w:} \)

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."

Clearly, if \( R \) decides \( \text{HALT}_{TM} \) then \( S \) decides \( A_{TM} \). This is the contradiction. Because \( A_{TM} \) is undecidable, \( \text{HALT}_{TM} \) also must be undecidable.

Mapping Reduction:

\( \langle M, w \rangle \in A_{TM} \text{ if and only if } \langle M', w' \rangle \in \text{HALT}_{TM}. \)

The following machine \( F \) computes a reduction \( f \).

\( F = \text{“On input } \langle M, w \rangle :} \)

1. Construct the following machine \( M' \).
   \( M' = \text{“On input } x:} \)
   a. Run \( M \) on \( x. \)
   b. If \( M \) accepts, accept.
   c. If \( M \) rejects, enter a loop.”
2. Output \( \langle M', w \rangle. \)"

How are these different?
PCP has two reductions:
\[ A_{TM} \leq^m_{m} \text{MPCP} \]
\[ \text{MPCP} \leq^m_{m} \text{PCP} \]

Is Mapping Reduction transitive?

**PROOF:**

Suppose \( A \leq^m_{m} B \) and \( B \leq^m_{m} C \) Then there are computable functions \( f \) and \( g \) such that \( x \in A \iff f(x) \in B \) and \( y \in B \iff g(y) \in C \).

Consider the composition function \( h(x) = g(f(x)) \).

We can build a TM that computes \( h \) as follows:

First, simulate a TM for \( f \) (such a TM exists because we assumed that \( f \) is computable) on input \( x \) and call the output \( y \).

Then simulate a TM for \( g \) on \( y \). The output is \( h(x) = g(f(x)) \).

Therefore, \( h \) is a computable function. Moreover, \( x \in A \iff h(x) \in C \).

Hence \( A \leq^m_{m} C \) via the reduction function \( h \).
More Theorems Re-examined: $E_{TM}$

Original Method:

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

$M_1 = \text{“On input } x:\text{ 1. If } x \neq w, \text{ reject. 2. If } x = w, \text{ run } M \text{ on input } w \text{ and accept if } M \text{ does.”} \}$

Assume that TM $R$ decides $E_{TM}$ and construct TM $S$ that decides $A_{TM}$ as follows.

$S = \text{“On input } \langle M, w \rangle, \text{ an encoding of a TM } M \text{ and a string } w:\text{ 1. Use the description of } M \text{ and } w \text{ to construct the TM } M_1 \text{ just described. 2. Run } R \text{ on input } \langle M_1 \rangle. 3. If } R \text{ accepts, reject; if } R \text{ rejects, accept.”} \}$

Problem: The mapping in the proof is actually $A_{TM}$ to $\neg E_{TM}$ (pay attention to the negation).

Notice: Decidability is not affected by complementation. But can we create a pure mapping reduction?

Proof that a Mapping Reduction is impossible:
Suppose for a contradiction that $A_{TM} \leq_m E_{TM}$ via reduction $f$. It follows from the definition of mapping reducibility that $\neg A_{TM} \leq_m \neg E_{TM}$ via the same reduction function $f$. However, $\neg E_{TM}$ (Exercise 4.5) is Turing-recognizable and $\neg A_{TM}$ is not Turing-recognizable.
Theorem: $\text{EQ}_\text{TM}$ is neither TR nor co-TR

$A_{\text{TM}} \leq_m \overline{\text{EQ}_\text{TM}}$

$F = \text{"On input } \langle M, w \rangle \text{, where } M \text{ is a TM and } w \text{ is a string:} $
1. Construct the following two machines, $M_1$ and $M_2$.
   $M_1 = \text{"On any input:} $
   1. \text{Reject.}"$
   $M_2 = \text{"On any input:} $
   1. \text{Run } M \text{ on } w. \text{If it accepts, Accept."}$
2. Output $\langle M_1, M_2 \rangle$."

$A_{\text{TM}} \leq_m \text{EQ}_\text{TM}$

$F = \text{"On input } \langle M, w \rangle \text{, where } M \text{ is a TM and } w \text{ is a string:} $
1. Construct the following two machines, $M_1$ and $M_2$.
   $M_1 = \text{"On any input:} $
   1. \text{Accept.}"$
   $M_2 = \text{"On any input:} $
   1. \text{Run } M \text{ on } w. \text{If it accepts, Accept."}$
2. Output $\langle M_1, M_2 \rangle$."