Mapping Reducibility

there and back again...

A CS Student’s Tale.
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 \( L(M) = L(M') \)
Mapping Reducibility Formalism

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.

$N = “$On input $w:$
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 $B$. 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 - $\text{HALT}_{\text{TM}}$ is undecidable

**Original Method:**

$\text{HALT}_{\text{TM}} = \{\langle M, w \rangle | M \text{ is a TM and } M \text{ halts on input } w\}$

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

$S = “\text{On input } \langle M, w \rangle, \text{ an encoding of a TM } M \text{ and a string } w:\n1. \text{ Run TM } R \text{ on input } \langle M, w \rangle.$
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 $\text{HALT}_{\text{TM}}$, then $S$ decides $A_{\text{TM}}$. This is the contradiction. Because $A_{\text{TM}}$ is undecidable, $\text{HALT}_{\text{TM}}$ also must be undecidable.

**Mapping Reduction:**

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

The following machine $F$ computes a reduction $f$.

$F = “\text{On input } \langle M, w \rangle:\n1. \text{ Construct the following machine } M'.
   \quad M' = “\text{On input } x:\n      a. \text{ Run } M \text{ on } x.
      b. \text{ If } M \text{ accepts, accept.}
      c. \text{ If } M \text{ rejects, enter a loop}.”$
2. Output $\langle M', w \rangle.”$

How are these different?
More Theorems Re-examined: PCP

PCP has two reductions:

\[ A_{\text{TM}} \leq_m \text{MPCP} \]

\[ \text{MPCP} \leq_m \text{PCP} \]

Is Mapping Reduction transitive?

PROOF:
Suppose \( A \leq_m B \) and \( B \leq_m C \). 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 \( A \leq_m C \) via the reduction function \( h. \)
More Theorems Re-examined: \( E_{TM} \)

**Original Method:**

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

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

Assume that TM \( R \) decides \( E_{TM} \) and construct TM \( S \) that decides \( A_{TM} \) as follows.

\( S = \) “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.”

**Mapping Reduction**

**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

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

$F =$ “On input $\langle M, w \rangle$, where $M$ is a TM and $w$ is 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 $\langle M_1, M_2 \rangle$.”

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

$F =$ “On input $\langle M, w \rangle$, where $M$ is a TM and $w$ is 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 $w$. If it accepts, accept.”
   2. Output $\langle M_1, M_2 \rangle$.”