So far we've reduced problems to each other via an oracle that gives us (often undecidable) answers. For example, given an oracle (a decider) for $E_{TM}$, we created a decider for $A_{TM}$. Since $A_{TM}$ is $DREC$, it follows that $E_{TM}$ is $DREC$. This is a Turing reduction, denoted as $A_{TM} \leq_T E_{TM}$.

In other words, if $A \leq_T B$, then $A$ is decidable relative to $B$. From this, we get the following theorem.

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

It is very tempting to think $A \leq_T B \implies B \in RE \implies A \in RE$.

After all, if you can map (accepting) instances of $A$ to (accepting) instances of $B$, then if $B$ is recognizable, surely $A$ must be too.

There is, however, hidden additional structure there. If you map $x \in A$ to $f(x) \in B$, then you also map $x \notin A$ to $f(x) \notin B$.

This is key! A general Turing reduction does not have this property. For example, we've shown that $A_{TM} \leq_T E_{TM}$, but $A_{TM} \notin RE$ and $E_{TM} \in RE$. The reduction we performed is more honestly written as $A_{TM} \leq_T \overline{E_{TM}}$. Note $\overline{E_{TM}} \in RE$.

Let's formalize this notion.

A function $f: \Sigma^* \to \Sigma^*$ is computable if there exists a $TM$ $M$ such that for all inputs $w \in \Sigma^*$, $M(w)$ halts with exactly $f(w)$ on its tape.
A language \( A \) is mapping reducible to a language \( B \), written \( A \leq m B \), if there is a computable function \( f: \Sigma^* \rightarrow \Sigma^* \) such that for every input \( w \in \Sigma^* \), \( w \in A \iff f(w) \in B \). The function \( f \) is called the reduction from \( A \) to \( B \).

On an intuitive level, when we write a reduction \( A \leq m B \), it means that \( A \) is no harder than \( B \) or, vice versa, \( B \) is at least as hard as \( A \).

Mapping reductions give us a bit more information than Turing reductions.

**Thm.** If \( A \leq m B \) and \( B \in \text{DEC} \), then \( A \in \text{DEC} \).

** Pf.** Let \( D \) be a decider for \( B \). Then \( D(f(w)) \) decides \( A \).

**Cor.** If \( A \leq m B \) and \( A \in \text{DEC} \), then \( B \in \text{DEC} \).

We can recover old theorems via mapping reductions.

**Thm.** \( \text{HALT} \leq m \text{DEC} \)

** Pf.** Define the TM \( M' \) to be

\[ N_M = \text{"On input } w\text{,}
\begin{align*}
1) & \text{Run } M(w) \\
2) & \text{Accept if } M(w) \text{ accepts.} \\
3) & \text{Loop forever."}
\end{align*} \]

Clearly \( N_M(w) \) halts iff \( M(w) \) accepts. So if we define the obviously computable function \( f(<M,w>) = <M',w> \) (improperly formatted strings are left unmodified), then \( <M,w> \in A_M \iff <M',w> \in \text{HALT}_M \).
So $A_{TM} \leq_m \text{HALT}_{TM}$, but $A_{TM} \notin \text{DEC}$, hence $\text{HALT}_{TM} \notin \text{DEC}$. 

But wait! There's more we can learn from $A_{TM} \leq_m \text{HALT}_{TM}$.

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

**Pf.** The same reduction yields the result.

**Thm.** If $A \leq_m B$ and $B \in \text{RE}$, then $A \in \text{RE}$.

**Pf.** Identical to the DEC case except we have recognizers instead of deciders (this is what we would have hoped $\leq_T$ would do).

**Cor.** If $A \leq_m B$ and $A \notin \text{RE}$, then $B \notin \text{RE}$.

**Cor.** If $A \leq_m B$ and $B \in \text{co-RE}$, then $A \in \text{co-RE}$.

**Pf.** $A \leq_m B \Rightarrow \overline{A} \leq_m \overline{B}$. Since $B \in \text{co-RE}$, $\overline{B} \in \text{RE}$, then $A \in \text{RE} \Rightarrow A \in \text{co-RE}$.

**Cor.** If $A \leq_m B$ and $A \notin \text{co-RE}$, then $B \notin \text{co-RE}$.

These theorems will show up again later in time/space resource restricted reductions, but first an example.

**Thm.** $E_{TM} \in \text{RE}$ and $E_{TM} \notin \text{co-RE}$.

**Pf.** We give two reductions: $A_{TM} \leq_m E_{TM}$ and $A_{TM} \leq_m \overline{E_{TM}}$.

Consider the TMs $R$ and $N_{w,v}$, where

$R =$ "On input $w$,
1) Reject"

$N_{w,v} =$ "On input $v$,
1) Run $M(w)$
2) Accept if $M(w)$ accepts and reject otherwise."

Then $L(R) = \emptyset$ and $L(N_{w,v}) = \{ \Sigma^* \mid M(w) \text{ does not accept} \}$.
So if we define the computable function

\[ f(v) = \begin{cases} 
\langle R, R \rangle & v \neq \langle M, w \rangle \\
\langle R, N_{M, w} \rangle & v = \langle M, w \rangle, 
\end{cases} \]

then clearly \( v \in A_{im} \) iff \( f(v) \in EQ_{im} \).

So \( A_{im} \notin EQ_{im} \), but \( A_{im} \notin \text{co-RE} \), so \( EQ_{im} \notin \text{co-RE} \), hence \( EQ_{im} \notin \text{RE} \).

Now let the TM T be

\[ T = \text{"On input } w, \]
\[ \text{1) Accept."} \]

So \( L(T) = \Sigma^* \), and we define the computable function

\[ f(v) = \begin{cases} 
v & v \neq \langle M, w \rangle \\
\langle T, N_{M, w} \rangle & v = \langle M, w \rangle. 
\end{cases} \]

Then we have \( v \in A_{im} \) iff \( f(v) \in EQ_{im} \).

So \( A_{im} \notin EQ_{im} \), but \( A_{im} \notin \text{co-RE} \), so \( EQ_{im} \notin \text{co-RE} \).

A few final notes. For a class of languages \( C^* \), we say a language \( A \) is \( C^* \)-HARD if \( \forall B \in C^* \), \( B \leq_m A \). If \( A \) is also a member of \( C^* \), then \( A \) is \( C^* \)-COMPLETE. (\( A \in C^* \) and \( A \in C^* \)-HARD \( \Rightarrow A \in C^* \)-COMPLETE).

Any language \( A \notin \text{RE} \) is necessarily \( \text{RE-HARD} \). Similarly, \( A \notin \text{co-RE} \) implies \( A \in \text{co-RE-HARD} \). To see why this is the case has to do with the arithmetic hierarchy.