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 $A_{TM} \leq_{T} DEC$, it follows that $E_{TM} \leq_{T} DEC$. 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:

(Thm) If $A \leq_{T} B$ and $B$ is decidable, then $A$ is decidable.

It is very tempting to think $A \leq_{T} B \Rightarrow B \in RE \Rightarrow A \in RE$.

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 here. If you map $x \in A$ to $f(x) \in B$, then you also map $x \in \overline{A}$ to $f(x) \in \overline{B}$. 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 $A_{TM} \in RE$ and $E_{TM} \notin RE$. The reduction we performed is more honestly written as $A_{TM} \leq_{T} E_{TM}$. Note $E_{TM} \notin RE$.

Let’s formalize this notion.

A function $f: \Sigma^{*} \rightarrow \Sigma^{*}$ 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 $A \leq B$, 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.** $\text{HALT}_{TM} \in \text{DEC}$

**Pf.** Define the TM $M'$ to be

$N_M =$ "On input $w$
1) Run $M(w)$
2) Accept if $M(w)$ accepts.
3) Loop forever"

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 $<M, w> \in A_{TM}$ iff $<M', w> \in \text{HALT}_{TM}$.
So $A_T \leq_m \text{HALT}_m$, but $A_T \not\leq \text{DEC}$, hence $\text{HALT}_m \not\leq \text{DEC}$.

But wait! There's more we can learn from $A_T \leq_m \text{HALT}_m$.

**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 $A \not\in \text{RE}$, then $B \not\in \text{RE}$.

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

**Pf** $A \leq_m B \implies \overline{A} \leq_m \overline{B}$. Since $B \in \text{coRE}$, $\overline{\text{RE}} \implies \overline{\text{RE}} \implies \overline{\text{RE}} \implies A \in \text{coRE}$.

**Cor** If $A \leq_m B$ and $A \not\in \text{coRE}$, then $B \not\in \text{coRE}$.

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

**Thm** $\text{EQ}_T \leq_m \text{RE}$ and $\text{EQ}_T \not\leq \text{coRE}$.

**Pf** We give two reductions: $A_T \leq_m \text{EQ}_T$ and $A_T \leq_m \text{EQ}_T$.

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

$$R = \text{"On input } w, \text{ 1) Reject"}$$

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

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

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

then clearly \( u \in A_{TM} \) iff \( f(u) \in \overline{EQ_{TM}} \).

So \( A_{TM} \subseteq EQ_{TM} \), but \( A_{TM} \not\in \text{co-RE} \), so \( EQ_{TM} \not\in \text{co-RE} \), hence \( EQ_{TM} \not\in \text{RE} \).

Now let the TM T be

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

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

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

Then we have \( u \in A_{TM} \) iff \( f(u) \in \overline{EQ_{TM}} \).

So \( A_{TM} \subseteq EQ_{TM} \), but \( A_{TM} \not\in \text{co-RE} \), so \( EQ_{TM} \not\in \text{co-RE} \).

\[ \square \]

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

Any language \( A \not\in \text{RE} \) is necessarily \( \text{RE-HARD} \). Similarly, \( A \not\in \text{co-RE} \) implies \( A \in \text{co-RE-HARD} \). To see why this is the case has to do with the arithmetic hierarchy.
Exercise: Here's an easy language we reduce to a hard language:

\[ \text{STEP}_{TM} = \{ \langle M, w, i \rangle \mid M(w) \text{ halts within } i \text{ steps} \} \]

This looks like \( \text{HALT}_{TM} \), so let's do that. We want a TM that halts iff it halts within \( i \) steps. We give that with the TM \( \xi_i \) below.

\[ \xi_i: \text{ On input } w, \]

1) Run \( M(w) \) for \( i \) steps
2) If \( M \) halted, accept
3) Otherwise, loop forever

Clearly, \( \xi \) halts iff \( M(w) \) halts within \( i \) steps. If we make the reduction \( f(\langle M, w, i \rangle) = \langle \xi_i, w \rangle \), then that gives us \( \text{STEP}_{TM} \leq_m \text{HALT}_{TM} \).

Of course, \( \text{STEP}_{TM} \in \text{DEC} \), and that's easy to show. It also means \( \text{STEP}_{TM} \leq_m \text{HALT}_{TM} \). We show this by giving a TM \( L_i \) that halts iff \( M(w) \) does not halt within \( i \) steps.

\[ L_i: \text{ On input } \langle M, w \rangle, \]

1) Run \( M(w) \) for \( i \) steps
2) If \( M \) halted, loop forever
3) Accept

So if we define \( f(\langle M, w, i \rangle) = \langle L_i, w \rangle \), then \( f \) is a mapping reduction from \( \text{STEP}_{TM} \) to \( \text{HALT}_{TM} \).

This was all really easy. Let's try something more challenging.
Ex

\( UHALT_{TM} = \{ \langle M \rangle \mid M \text{ halts on all inputs} \} \)

In other words, \( UHALT_{TM} \) is the class of all deciders. Rice's Theorem doesn't give us that \( UHALT_{TM} \) is \( \# \text{DEC} \) for free, but this isn't very interesting. Let's consider if it's at all computable.

First, we'll show \( HALT_{TM} \leq_m UHALT_{TM} \). To do this, consider the TM \( H_{M,w} \).

\[ H_{M,w} = \begin{align*}
\text{On input } \langle \Gamma \rangle, \\
1) \text{ Run } M(\langle \Gamma \rangle). \\
2) \text{ Reject}
\end{align*} \]

So \( H_{M,w} \) halts on all inputs iff \( M(\langle \Gamma \rangle) \) halts. Note that we don't care if it accepts or rejects, as its language is unimportant to us.

If we define \( f(\langle M, \langle \Gamma \rangle \rangle) = \langle H_{M,w} \rangle \), then \( f \) is a mapping reduction from \( HALT_{TM} \) to \( UHALT_{TM} \). From this, we conclude that \( UHALT_{TM} \uparrow \text{co-RE} \).

\( UHALT_{TM} \) seems harder than \( HALT_{TM} \) since it's asking about all inputs, so we should also be able to show \( \overline{UHALT_{TM}} \uparrow \text{co-RE} \).

We can get it from \( HALT_{TM} \leq_m UHALT_{TM} \), since \( UHALT_{TM} \uparrow \text{co-RE} \) yields the result.

Consider the TM \( L_{M,w} \) below.

\[ L_{M,w} = \begin{align*}
\text{On input } \langle \Gamma \rangle, \\
1) \text{ Run } M(\langle \Gamma \rangle) \text{ for } 100 \text{ steps.} \\
2) \text{ If } M(\langle \Gamma \rangle) \text{ halted, loop forever.} \\
3) \text{ Accept.}
\end{align*} \]

So \( L_{M,w} \) halts on all inputs iff \( M(\langle \Gamma \rangle) \) does not halt. In other words, \( L_{M,w} \) does not halt on at least one input if \( M(\langle \Gamma \rangle) \) halts. So then \( f(\langle M, \langle \Gamma \rangle \rangle) = \langle L_{M,w} \rangle \) is a mapping reduction from \( HALT_{TM} \) to \( UHALT_{TM} \).