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 $A_{TM} \notin DEC$, it follows that $E_{TM} \notin DEC$. This is a Turing reduction, denoted as $A_{TM} \leq_T E_{TM}$.

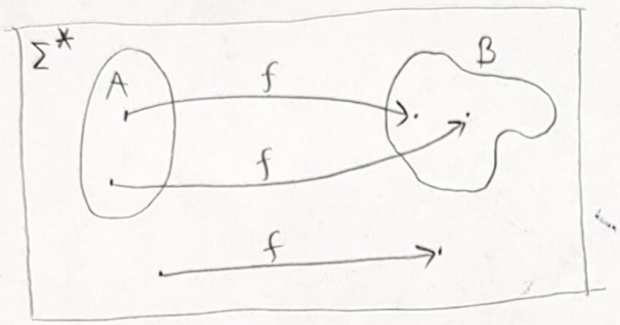
In other words if $A \leq_T B$, then A is decidable relative to B. From this, we get the following theorem.

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 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 $A_{TM} \in RE$ and $E_{TM} \notin RE$. 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 <u>mapping reducable</u> to a language B, written $A \leq_m B$, if there is a computable function $f: \Sigma^* \to \Sigma^*$ such that for every input $w \in \Sigma^*$, $w \in A \iff f(w) \in B$.
The function f is called the <u>reduction</u> 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.

<u>Thm</u> If $A \leq_m B$ and $B \in DEC$, then $A \in DEC$.

<u>Pf</u> Let D be a decider for B. Then $D(f(w))$ decides A.

<u>Cor</u> If $A \leq_m B$ and $A \notin DEC$, then $B \notin DEC$.

We can recover old theorems via mapping reductions.

<u>Thm</u> $HALT_{TM} \notin DEC$

<u>Pf</u> 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 $f(\langle M, w\rangle) = \langle M', w\rangle$ (improperly formatted strings are left unmodified), then $\langle M, w\rangle \in A_{TM}$ iff $\langle M', w\rangle \in HALT_{TM}$.

So $A_{TM} \leq_m HALT_{TM}$, but $A_{TM} \notin DEC$, hence $HALT_{TM} \notin DEC$. □

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

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 $B \in RE$, then $A \in RE$.

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 $A \notin RE$, then $B \notin RE$.

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

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

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

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

Thm) $EQ_{TM} \notin RE$ and $EQ_{TM} \notin co\text{-}RE$.

Pf) We give two reductions: $A_{TM} \leq_m \overline{EQ_{TM}}$ and $A_{TM} \leq_m EQ_{TM}$.
Consider the TMs R and $N_{M,w}$, where

R = "On input w,
1) Reject"

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

Then $L(R) = \emptyset$ and $L(N_{M,w}) = \begin{cases} \emptyset & M(w) \text{ does not accept} \\ \Sigma^* & M(w) \text{ accepts.} \end{cases}$

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 $v \in A_{TM}$ iff $f(v) \in \overline{EQ_{TM}}$.

So $A_{TM} \leq \overline{EQ_{TM}}$, but $A_{TM} \notin$ co-RE, so $\overline{EQ_{TM}} \notin$ co-RE, hence $EQ_{TM} \notin$ RE.

Now let the TM T be

T = "On input w,
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 $v \in A_{TM}$ iff $f(v) \in EQ_{TM}$.

So $A_{TM} \leq_m EQ_{TM}$, but $A_{TM} \notin$ co-RE, so $EQ_{TM} \notin$ co-RE.

□

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

Any language $A \notin$ RE is necessarily RE-HARD. Similarly, $A \notin$ co-RE implies $A \in$ co-RE-HARD. To see why this is the case has to do with the arithmetic heirarchy.

Ex) Here's an easy language we reduce to a hard language.

$$STEP_{TM} = \{\langle M, w, i\rangle \mid M(w) \text{ halts within } i \text{ steps}\}$$

This looks like $HALT_{TM}$, so let's do that. We want a TM that halts iff it halts within i steps. We give that with the TM $S_{M,i}$ below.

$S_{M,i}$ = On input w,
1) Run $M(w)$ for i steps
2) If M halted, accept
3) Otherwise, loop forever

Clearly, $S_{M,i}$ halts iff $M(w)$ halts within i steps. If we make the reduction $f(\langle M, w, i\rangle) = \langle S_{M,i}, w\rangle$, then that gives us $STEP_{TM} \leq_m HALT_{TM}$.

Of course, $STEP_{TM} \in DEC$, and that's easy to show. It also means $\overline{STEP_{TM}} \leq_m HALT_{TM}$. We show this by giving a TM $L_{M,i}$ that halts iff $M(w)$ does not halt within i steps.

$L_{M,i}$ = On input w
1) Run $M(w)$ for i steps
2) If M halted, loop forever
3) Accept

So if we define $f(\langle M, w, i\rangle) = \langle L_{M,i}, w\rangle$, then f is a mapping reduction from $\overline{STEP_{TM}}$ to $HALT_{TM}$.

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 does not give us that $UHALT_{TM} \notin DEC$ 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 $H_{M,w}$.

$H_{M,w}$ = On input v,
1) Run $M(w)$.
2) Reject

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

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

$UHALT_{TM}$ seems harder than $HALT_{TM}$ since it's asking about all inputs, so we should also be able to show $UHALT_{TM} \notin RE$.

We can get it from $HALT_{TM} \leq_m \overline{UHALT_{TM}}$, since $\overline{UHALT_{TM}} \notin$ co-RE yields the result.

Consider the TM $L_{M,w}$ below.

$L_{M,w}$ = On input v,
1) Run $M(w)$ for $|v|$ steps
2) If $M(w)$ halted, loop forever
3) Accept

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