### Other Undecidable Problems

**Strategy for more problems:**

1. Assume for purpose of contradiction that problem $P$ decid.
2. Show that if $P$ were decidable, another problem $Q$ (that we have shown undecidable) would be decidable. CONTR.

**Th.** $E_{TM} = \{<M> | M \text{ is a TM and } L(M) \text{ is empty}\}$ is undecidable.

**Proof idea:** We show if $E_{TM}$ were decidable, then $A_{TM}$ would be.

So suppose $E_{TM}$ decided by $R$. We want decider $S$ for $A_{TM}$.

Could we use $R$ directly for $S$? $R$ accepts $<M>$ iff $L(M) = \emptyset$.

So if $L(M) = \emptyset$, then $S$ should reject $w$.

But if $L(M) \neq \emptyset$, then we’re stuck!!???

---

### Other Undecidable Problems

**Trick:** Instead of running $R$ on $<M>$, we run it on a different machine $<S^{M,w}>$ (defined for specific $M$ and $w$). $S^{M,w}$ takes input $x$, and:

1. if $x \neq w$, it rejects.
2. if $x = w$, run $M$ on input $w$, and accept if $M$ does.

$S^{M,w}$ accepts at most one input, $w$; and it accepts $w$ iff $M$ accepts $w$.

so $L(S^{M,w}) = \emptyset$ iff $M$ rejects $w$

so $L(S^{M,w}) \neq \emptyset$ iff $M$ accepts $w$

This allows us to use problem $E_{TM}$!
**E<sub>TM</sub> is Undecidable**

Proof: Suppose E<sub>TM</sub> were decidable by R. We show that A<sub>TM</sub> would be decidable, a contradiction.

We first define S<sup>M,w</sup> for given M and w as:

- On input x
  1. if x ≠ w, reject.
  2. if x = w, run M on w and accept if M does.

We now define decider S for A<sub>TM</sub> as follows:

- On input <M,w>
  1. use M to construct S<sup>M,w</sup>
  2. Run R on <S<sup>M,w</sup>>
  3. If R accepts, S rejects; if R rejects, S accepts.

**Other Undecidable Problems**

We can use other problems than A<sub>TM</sub> to get a contradiction!

Th. EQ<sub>TM</sub> = {<M1,M2> | M1 is a TM , L(M1) = L(M2)}

is undecidable.

Proof: We show if EQ<sub>TM</sub> is decidable, E<sub>TM</sub> is, Contr.

Proof idea: EQ<sub>TM</sub> tests if 2 languages =, E<sub>TM</sub> if language = ∅

So E<sub>TM</sub> is a special case of EQ<sub>TM</sub>.

Proof: Suppose R decides EQ<sub>TM</sub>. Construct S to decide E<sub>TM</sub> as follows.

S: on input <M>
  1. Run R on <M, Rej>, where Rej is TM with L(Rej) = ∅
  2. If R accepts, S accepts; if R rejects, S rejects.

CONTR.!
Mapping Reducibility

Intuition: to solve a problem, reduce it to another

Ex. to solve problem of getting A in course, reduce to
getting A on quizzes, on homework, and on final.

We’ve already used this idea to prove more
problems undecidable

Ex: If \( \text{HALT}_{TM} \) is decidable, then \( A_{TM} \) is decidable.

But \( A_{TM} \) already shown undecidable, CONTR.

Mapping reducibility makes reduction of problems precise

\( A \leq_m B \): intuitively, if B has a solution, can
use to solve for A.

We will use computable mapping \( m \) of \( A \) to B

---

Computable Functions

TM as computer of output, not recognizer

Output is what is on tape when halt

Def.: \( f : \Sigma^* \rightarrow \Sigma^* \) is a computable function if

there is some TM M which computes \( f \):

on input \( w \), M halts with just \( f(w) \) on its tape.

Note: \( M \) must always halt!

Ex. computable function: TM M that, with i and j on tape,
computes \([i\] (erases i and j, when done)
computes \([* \) function

Other computable functions: \( \text{factorial(<n>)} \)

\( \text{gcd(<m,n>)} \) \( \text{prime(<n>) = <nth prime no}> \)

Transformers of TM:

\( f(<M>) = <M’> \) where \( L(M) = L(M’), \) and \( M’ \) never
moves its tape off the left end
Mapping Reducibility

Def. 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 \( w \) in \( \Sigma^* \),

\[ w \text{ in } A \quad \Rightarrow \quad f(w) \text{ in } B \]

\( f \) is called a reduction.

*Can test membership in A by membership in B*

Ex. \( E_{TM} \leq_{m} EQ_{TM} \)

\[ f: \Sigma^* \rightarrow \Sigma^* \]

\[ \langle M \rangle \quad \rightarrow \quad f(\langle M \rangle) = \langle M, \text{Rej} \rangle \quad \text{ (where } L(\text{Rej}) = \emptyset \text{ )} \]

\( f \) appends to \( \langle M \rangle \) the repr. \( \langle \text{Rej} \rangle \); \( f(w) = w \) ow.

\[ \langle M \rangle \text{ in } E_{TM} \quad \Rightarrow \quad f(\langle M \rangle) \text{ in } EQ_{TM} \]

\( f \) is computable:

Uses of Mapping Reducibility

Th: If \( A \leq_{m} B \) and B is decidable, then A is decid.

Proof: Let \( D \) be the decider for B, and \( f \) the reduc.

\( A \rightarrow_{m} B \). We define decider \( D' \) for A as follows:

\( D' \): on input \( w \)

1. compute \( f(w) \)

2. Run \( D \) on input \( f(w) \); if \( D \) accepts, \( D' \) accepts.
   
   if \( D \) rejects, \( D' \) rejects.

   Since \( w \) in \( A \rightarrow f(w) \) in B by def. map. reduc.,
   
   and \( D' \) accepts \( w \rightarrow D \) accepts \( f(w) \),

   we conclude \( D' \) accepts A.

Since \( D \) is a decider, \( D' \) is a decider.

Therefore, \( A \) is decidable.
**Uses of Mapping Reducibility**

Cor: If $A \leq_m B$ and $A$ is undecidable, then $B$ is undecidable.

Proof: Suppose $B$ were decidable. By the prev. Th., since $A \leq_m B$, then $A$ would be decidable. Contradiction.

Ex. If $A_{TM} \leq_m HALT_{TM}$ and $A_{TM}$ undecidable

$\Rightarrow$ $HALT_{TM}$ undecidable

To show $A_{TM} \leq_m HALT_{TM}$, need computable $f$:

\[
\begin{align*}
\text{Ex. if } A_{TM} & \leq_m HALT_{TM} \text{ and } A_{TM} \text{ undecidable} \\
\text{To show } A_{TM} & \leq_m HALT_{TM}, \text{ need computable } f : <M,w> & \mapsto <M',w'> \\
\text{Then } <M,w> & \text{ in } A_{TM} \iff <M',w'> \text{ in } HALT_{TM}
\end{align*}
\]

---

**Uses of Mapping Reducibility**

TM $F$ computes $f$ as follows:

$F$: on input $<M,w>$

1. Construct $M'$:
   
   $M'$: on input $x$
   
   1. Run $M$ on $x$.
   
   2. If $M$ accepts, $M'$ accepts.
      
      If $M$ rejects, loop infinitely.

2. Output $<M',w>$

Then $<M,w> \in A_{TM} \iff M$ accepts $w \iff M'$ halts on $w \iff <M',w> \in HALT_{TM}$
Uses of Mapping Reducibility

Th. $E_{TM}$ is undecidable \textit{(revisited)}

A reduced to $E_{TM}$

Qu: is it a mapping reducibility?

$<M,w> \rightarrow <M'>$

M accepts w $\iff L(M') \neq \phi$

$<M,w>$ in A $\iff <M'>$ in $E_{TM}$

so we showed $A_{TM} \leq_{m} E_{TM}$

not $A_{TM} \leq_{m} E_{TM}$ \textit{(can't be done)}

We could still prove the th., because

$E_{TM}$ decidable $\iff E_{TM}$ decidable

Turing-Recognizability & Mapping Reducibility

Th. If $A \leq_{m} B$ and B Turing-recognizable, then $A$ is Turing-recognizable.

Proof: Suppose B is recognized by TM R, and f: $A \rightarrow B$ is a reduction. We define recognizer TM M for A as follows:

M: on input w
1. compute f(w)
2. run R on input f(w); \text{ if R accepts, M acc.}
   \text{if R rejects, M rejects.}
Since w in $A \iff f(w)$ in B, by def. map. reduc., and M accepts w $\iff$ R accepts f(w)
we conclude M accepts A. M is certainly a recognizer. Therefore, A is Turing-recognizable.
**Turing-Recognizability & Mapping Reducibility**

<table>
<thead>
<tr>
<th>Cor. If $A \leq^m_B$ and $A$ not Turing-recognizable, then $B$ is not Turing-recognizable.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Proof:</strong> Suppose $B$ were T. R. By the previous th., since $A \leq^m_B$, then $A$ would be T.R. Contradiction.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Th. $EQ_{TM}$ is not Turing-recognizable.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Proof:</strong> We show that $A_{TM} \leq^m_{TM} EQ_{TM}$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>F: on input $&lt;M,w&gt;$ construct Rej and M2 so that $L(Rej) = \emptyset$</th>
</tr>
</thead>
<tbody>
<tr>
<td>M2: on input $x$, run $M$ on $w$. If $M$ accepts, accept. $L(M2)$ is either $\Sigma^*$ (if $M$ accepts $w$) or $\emptyset$ (ow)</td>
</tr>
</tbody>
</table>

| $M$ accepts $w \iff \emptyset = L(Rej) \equiv L(M2)$ |
Turing-Recognizability & Mapping Reducibility

Th. $\overline{EQ_{TM}}$ is not Turing-recognizable.

Proof: We show that $A_{TM} \leq_m EQ_{TM}$

Want $<M,w> \rightarrow <M_1,M_2>

M accepts w $\iff$ $L(M_1) = L(M_2)$

G: on input $<M,w>$ construct Acc and M2 so

that $L(Acc) = \Sigma^*$

M2: on input x, run M on w. If M accepts, accept.

$L(M_2)$ is either $\Sigma^*$ (if M accepts w) or $\phi$ (ow)

M accepts w $\iff$ $\Sigma^* = L(Acc) = L(M_2)$

$EQ_{TM}$ and $\overline{EQ_{TM}}$ are both Turing-unrecognizable