### Other Undecidable Problems

*Strategy for more problems:
1. Assume for purpose of contradiction that problem \( \text{P decid} \).
2. Show that if \( \text{P} \) were decidable, another problem \( \text{Q} \) (that we have shown undecidable) would be decidable. \( \text{CONTR}. \)

**Th.** \( E_{TM} = \{ \langle M \rangle \mid M \text{ is a TM and } L(M) \text{ is empty} \} \) 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 \( \langle M \rangle \) iff \( L(M) = \phi \).

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

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

---

### Other Undecidable Problems

*Trick:* Instead of running \( R \) on \( \langle M \rangle \), we run it on a different machine \( \langle S^{M,w} \rangle \) (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 \( L( S^{M,w} ) = \phi \) iff \( M \) rejects \( w \)

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

This allows us to use problem \( E_{TM} \)!
**E\textsubscript{TM} is Undecidable**

Proof: Suppose \( E_{\text{TM}} \) were decidable by \( R \). We show that \( A_{\text{TM}} \) would be decidable, a contradiction.

We first define \( S_{M,w} \) for given \( M \) and \( w \) as:

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

We now define decider \( S \) for \( A_{\text{TM}} \) as follows:

On input \( <M,w> \)
1. use \( M \) to construct \( S_{M,w} \)
2. Run \( R \) on \( <S_{M,w}> \)
3. If \( R \) accepts, \( S \) rejects; if \( R \) rejects, \( S \) accepts.

**Other Undecidable Problems**

We can use other problems than \( A_{\text{TM}} \) to get a contradiction!

**Th. \( \text{EQ}_{\text{TM}} = \{<M1,M2> | M \text{ is a TM, } L(M1) = L(M2)\} \) is undecidable.**

Proof: We show if \( \text{EQ}_{\text{TM}} \) is decidable, \( E_{\text{TM}} \) is, Contr.

Proof idea: \( \text{EQ}_{\text{TM}} \) tests if 2 languages \( = \), \( E_{\text{TM}} \) if language \( = \) \( \phi \)

So \( E_{\text{TM}} \) is a special case of \( \text{EQ}_{\text{TM}} \).

Proof: Suppose \( R \) decides \( \text{EQ}_{\text{TM}} \). Construct \( S \) to decide \( E_{\text{TM}} \) as follows.

\( S \): on input \( <M> \)
1. Run \( R \) on \( <M, \text{Rej}> \), where \( \text{Rej} \) is \( \text{TM} \) with \( L(\text{Rej}) = \phi \)
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 $\text{HALT}_\text{TM}$ is decidable, then $A_{\text{TM}}$ is decidable.

But $A_{\text{TM}}$ 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. HW 3 problem 6: TM that, with i and j on tape, computes $i^j$ (if on just 1 tape, and erased i and j, then this TM would compute * function)

Other computable functions: factorial(<n>)

$\text{gcd}(<m,n>)$ 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 $\Sigma^*$,

$$w \in A \iff f(w) \in B$$

$f$ is called a reduction.

Can test membership in $A$ by membership in $B$.

**Ex.** $E_{TM} \leq_m EQ_{TM}$

$f: \Sigma^* \rightarrow \Sigma^*$

$$<M> \mapsto f(<M>) = <M,\text{Rej}> \quad (\text{where } L(\text{Rej}) = \phi)$$

$f$ appends to $<M>$ the repr. $<\text{Rej}>$; $f(w) = w$ ow.

$$<M> \in E_{TM} \iff f(<M>) \in EQ_{TM}$$

$f$ 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. $A \leq_m B$. 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 $w \in A \iff f(w) \in B$ by def. map. reduc., and $D'$ accepts $w \iff D$ 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 $A \leq^m_m B$ and $A$ is undecidable, then $B$ is undecidable.

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

Ex. If $A_{TM} \leq^m_m HALT_{TM}$ and $A_{TM}$ undecidable

$\longrightarrow$ HALT_{TM} undecidable

To show $A_{TM} \leq^m_m HALT_{TM}$, need computable $f : <M,w> \longrightarrow <M',w'>$

$<M,w>$ in $A_{TM}$ iff $<M',w'>$ in $HALT_{TM}$

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 $A$ $\iff$ $M$ accepts $w$ $\iff$

$TM$

$M'$ halts on $w$ $\iff$

$<M',w>$ in $HALT_{TM}$
Uses of Mapping Reducibility

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

A \( \leq_m \) reduced to \( E_{TM} \)

Qu: is it a mapping reducibility?

\[ <M,w> \longrightarrow <M'> \]

\( M \) accepts \( w \) \( \iff \) \( L(M') = \phi \)

\[ <M,w> \text{ in } A \longrightarrow <M'> \text{ in } E_{TM} \]

so we showed \( A_{TM} \leq E_{TM} \)

not \( A_{TM} \leq m E_{TM} \) (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:

\[ M: \text{ on input } w \]

1. compute \( f(w) \)
2. run \( R \) on input \( f(w) \); if \( R \) accepts, \( M \) acc. if \( R \) rejects, \( M \) rejects.

Since \( w \) in \( A \leftrightarrow f(w) \) in \( B \), by def. map. reduc., and \( M \) accepts \( w \leftrightarrow R \) accepts \( f(w) \)

we conclude \( M \) accepts \( A \). \( M \) is certainly a recognizer. Therefore, \( A \) is Turing-recognizable.