Next: reducibility

- We still need to *prove* that the Halting problem is undecidable.
- Do more examples of undecidable problems.
- Try to get a general technique for proving undecidability.
Halting problem

- Assume that it is decidable. So there is a TM $R$ that decides
  $\text{HALT} = \{<M,w>| M \text{ is a TM and } M \text{ halts on } w\}$
- Use $R$ as a subroutine to get a TM $S$ to decide
  $A_{TM} = \{<M,w> | M \text{ is a TM that accepts } w \}$
- Therefore $A_{TM}$ is decidable. CONTRADICTION!
- Details follow ….

Halting problem - 2

$S =$ “On input $<M,w>$
- Run TM $R$ on input $<M,w>$.
- If $R$ rejects, REJECT.
- If $R$ accepts, simulate $M$ on $w$ until it halts.
- If $M$ has accepted, ACCEPT, else REJECT”
More undecidability

\[ E_{\text{TM}} = \{ <M> | \text{M is a TM and } L(M) = \emptyset \} \]

We mentioned that \( E_{\text{TM}} \) is co-TM recognizable.
We will prove next that \( E_{\text{TM}} \) is undecidable.

Intuition: You cannot solve this problem UNLESS you solve the halting problem!!

But this is hard to formalize, so we use \( A_{\text{TM}} \).
Instead.

\[ 13-07-29 \quad \text{CSE 2001, Summer 2013} \]

\[ 5 \]

\[ E_{\text{TM}} \text{ is undecidable} \]

Assume \( R \) decides \( E_{\text{TM}} \). Use \( R \) to design TM \( S \) to decide \( A_{\text{TM}} \).

- Given a TM \( M \) and input \( w \), define a new TM \( M' \):
  - If \( x \neq w \), reject
  - If \( x = w \), accept iff \( M \) accepts \( w \)

\[ S = \text{“On input } <M,w> \]

- **Construct \( M' \) as above.**
- Run TM \( R \) on input \( <M'> \).
- If \( R \) accepts, REJECT; If \( R \) rejects, ACCEPT.”
**EQ_{TM} is undecidable**

If this is decidable, then we can solve E_{TM}!! (You need to check equality with TM M_{1} that rejects all inputs)  
Assume R decides EQ_{TM}. Use R to design TM S to decide E_{TM}.

S = “On input <M>
• Run TM R on input <M, M_{1}>.
• If R accepts, ACCEPT; If R rejects, REJECT.”

---

**The running idea**

All our proofs had a common structure  
• The first undecidable proof was hard – used diagonalization/self-reference.  
• For the rest, we assumed decidable and used it as a subroutine to design TM’s that decide known undecidable problems.  
• Can we make this technique more structured?
Mapping Reducibility

Thus far, we used reductions informally:

If “knowing how to solve A” implied “knowing how to solve B”, then we had a reduction from B to A.

Sometimes we had to negate the answer to the “∈A?” question, sometimes not. In general, it was unspecified which transformations were allowed around the “∈A?”-part of the reduction.

Let us make this formal…

Computable Functions

A function \( f: \Sigma^* \rightarrow \Sigma^* \) is a TM-computable function if there is a Turing machine that on every input \( w \in \Sigma^* \) halts with just \( f(w) \) on the tape.

All the usual computations (addition, multiplication, sorting, minimization, etc.) are all TM-computable.

Note: alterations to TMs, like “given a TM M, we can make an M’ such that…” can also be described by computable functions that satisfy \( f(<M>) = <M’> \).
A language $A$ is mapping reducible to a another language $B$ if there is a TM-computable function $f: \Sigma^* \rightarrow \Sigma^*$ such that:

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

for every $w \in \Sigma^*$.

Terminology/notation:
- $A \leq_m B$
- function $f$ is the reduction of $A$ to $B$
- also called: “many-one reducible”

The language $B$ can be more difficult than $A$. Typically, the image $f(A)$ is only a subset of $B$, and $f(\Sigma^*\backslash A)$ a subset of $\Sigma^*\backslash B$.

“Image $f(A)$ can be the easy part of $B$.”
Previous mappings used

\[ A_{TM} \leq_m HALT_{TM} \]

\[ F = \text{“On input } <M,w> \text{”} \]
- Construct TM \( M' \) = “on input x:
  - Run M on x
  - If M accepts, ACCEPT
  - If M rejects, enter infinite loop.”
- Output \( <M',w> \)”

Check: f maps \( <M,w> \) to \( <M', w'> \).
\( <M,w> \in A_{TM} \iff <M',w> \in HALT_{TM} \)

Previous mappings used - 2

Recall: \( M_1 \) rejects all inputs. Assume \( R \) decides \( EQ_{TM} \). Use \( R \) to design TM \( S \) to decide \( E_{TM} \).

\[ S = \text{“On input } <M> \text{”} \]
- Run TM \( R \) on input \( <M, M_1> \).
- If \( R \) accepts, ACCEPT; If \( R \) rejects, REJECT.”

Check: f maps \( <M> \) to \( <M, M_1> \).
\( <M> \in E_{TM} \iff <M, M_1> \in EQ_{TM} \)
Decidability obeys \( \leq_m \) Ordering

**Theorem 5.22**: If \( A \leq_m B \) and \( B \) is TM-decidable, then \( A \) is TM-decidable.

**Proof**: Let \( M \) be the TM that decides \( B \) and \( f \) the reducing function from \( A \) to \( B \). Consider the TM:

On input \( w \):
1) Compute \( f(w) \)
2) Run \( M \) on \( f(w) \) and give the same output.

By definition of \( f \): if \( w \in A \) then \( f(w) \in B \).

\( M \) “accepts” \( f(w) \) if \( w \in A \), and
\( M \) “rejects” \( f(w) \) if \( w \notin A \).

Undecidability obeys \( \leq_m \) Order

**Corollary 5.23**: If \( A \leq_m B \) and \( A \) is undecidable, then \( B \) is undecidable as well.

**Proof**: Language \( A \) undecidable and \( B \) decidable contradicts the previous theorem.

Extra: If \( A \leq_m B \), then also for the complements \( (\Sigma^* \setminus A) \leq_m (\Sigma^* \setminus B) \).

**Proof**: Let \( f \) be the reducing function of \( A \) to \( B \) with \( w \in A \iff f(w) \in B \). This same computable function also obeys “\( v \in (\Sigma^* \setminus A) \iff f(v) \in (\Sigma^* \setminus B) \)” for all \( v \in \Sigma^* \).
Recognizability and $\leq_m$

**Theorem 5.28:** If $A \leq_m B$ and $B$ is TM-recognizable, then $A$ is TM-recognizable.

**Proof:** Let $M$ be the TM that recognizes $B$ and $f$ the reducing function from $A$ to $B$. Again the TM:

1) Compute $f(w)$
2) Simulate $M$ on $f(w)$ and give the same result.

By definition of $f$: $w \in A$ equivalent with $f(w) \in B$.

$M$ “accepts” $f(w)$ if $w \in A$, and

$M$ “rejects” $f(w)$/does not halt on $f(w)$ if $w \notin A$.

Unrecognizability and $\leq_m$

**Corollary 5.29:** If $A \leq_m B$ and $A$ is not Turing-recognizable, then $B$ is not recognizable as well.

**Proof:** Language $A$ not TM-recognizable and $B$ recognizable contradicts the previous theorem.

**Extra:** If $A \leq_m B$ and $A$ is not co-TM recognizable, then $B$ is not co-Turing-recognizable as well.

**Proof:** If $A$ is not co-TM-recognizable, then the complement ($\Sigma^*\overline{A}$) is not TM recognizable.

By $A \leq_m B$ we also know that ($\Sigma^*\overline{A}$) $\leq_m$ ($\Sigma^*\overline{B}$).

Previous corollary: ($\Sigma^*\overline{B}$) not TM recognizable, hence $B$ not co-Turing-recognizable.
Decidable $A \leq_m B$

If $A$ is a decidable language, then $A \leq_m B$ for every nontrivial $B$. (Let $1 \in B$ and $0 \notin B$.)

Because $A$ is decidable, there exists a TM $M$ such that $M$ outputs “accept” on every $x \in A$, and “reject” on $x \notin A$. We can use this $M$ for a TM-computable function $f$ with

$$f(x) = \begin{cases} 1 & x \in A \\ 0 & x \notin A \end{cases}$$

"The function $f$ does all the decision-work”

---

$E_{TM}$ Revisited

Recall: The emptiness language was defined as

$$E_{TM} = \{ <M> \mid M \text{ is a TM with } L(M) = \emptyset \}$$

$E_{TM}$ is not Turing recognizable.

Simple proof via ($\overline{A}_{TM} \leq_m E_{TM}$):

Let $f$ on input $<M,w>$ give $<M'>$ as output with:

$M'$: Ignore input
- Run $M$ on $w$
  - If $M$ accepted $w$ then “accept”
  - otherwise “reject”

Now:

$$<M,w> \in \overline{A}_{TM} \iff f(<M,w>) = <M'> \in E_{TM}$$
Something still unproven...

**EQ**\textsubscript{TM} is not TM Recognizable

Proof (by showing \( \overline{A_{TM}} \leq_m EQ_{TM} \)):

Let \( f \) on input \( <M,w> \) give \( <M_1,M_2> \) as output with:

- \( M_1 \): “reject” on all inputs
- \( M_2 \): Ignore input
  - Run \( M \) on \( w \)
  - “accept” if \( M \) accepted \( w \)

We see that with this TM-computable \( f \):

\[ <M,w> \in \overline{A_{TM}} \iff f(<M,w>) = <M_1,M_2> \in EQ_{TM} \]

Because \( \overline{A_{TM}} \) is not recognizable, so is \( EQ_{TM} \).
**EQ\textsuperscript{TM} not co-TM Recognizable**

**Proof** (by showing \(A\textsuperscript{TM} \leq \text{m} EQ\textsuperscript{TM}\)):

Let \(f\) on input \(<M,w>\) give \(<M_1,M_2>\) as output with:

- \(M_1\): “accept” on all inputs
- \(M_2\): Ignore input
  - Run \(M\) on \(w\)
  - “accept” if \(M\) accepted \(w\)

We see that with this TM-computable \(f\):

\(<M,w> \in A\textsuperscript{TM} \iff f(<M,w>) = <M_1,M_2> \in EQ\textsuperscript{TM}\)

Because \(A\textsuperscript{TM}\) is not co-recognizable, so is \(EQ\textsuperscript{TM}\).

---

**Partial \(\leq \text{m}\) Ordering**

\[\text{TRIVIAL} \leq \text{m} \text{ DECIDABLE} \leq \text{m} EQ\textsuperscript{TM}\]

\[\text{TRIVIAL} \leq \text{m} A\textsuperscript{TM} \leq \text{m} EQ\textsuperscript{TM}\]

\[\text{TRIVIAL} \leq \text{m} \bar{A}\textsuperscript{TM} \leq \text{m} EQ\textsuperscript{TM}\]