Outline

• reductions
• many-one reductions
• undecidable problems
  – computation histories
  – surprising contrasts between decidable/undecidable
• Rice’s Theorem
Example reduction

• Preceding reduction proved:

**Theorem**: $A_{TM}$ is undecidable.

Proof (recap):

– suppose $A_{TM}$ is decidable
– we showed how to use $A_{TM}$ to decide HALT
– conclude HALT is decidable. Contradiction.
Another example

• Try to prove undecidable:

\[ E_{TM} = \{<M> : L(M) = \emptyset\} \]

• which problem should we **reduce from**?
  – \( HALT = \{<M, w> : M \text{ halts on input } w\} \)
  – \( A_{TM} = \{<M, w> : M \text{ accepts input } w\} \)

• Some things we can do:
  – check if \( <M> \in E_{TM} \)
  – construct another TM \( M' \) and check if \( <M'> \in E_{TM} \)
Another example

- We are given input $<M, w>$
- We want to use a procedure that decides $E_{TM}$ to decide if $<M, w> \in A_{TM}$

**Idea:**
- check if $<M> \in E_{TM}$
- if not?
  - helpful if could make $M$ reject everything except possibly $w$. 
Another example

• Construct TM $M'$:
  – on input $x$, if $x \neq w$, then reject
  – else simulate $M$ on $x$, and accept if $M$ does.

• on input $\langle M, w \rangle$
  – construct $M'$ from description of $M$
  – check if $M' \in E_{\text{TM}}$
    • if no, $M$ must accept $w$; ACCEPT
    • if yes, $M$ cannot accept $w$; REJECT
Another example

• Preceding reduction proved:

**Theorem**: $E_{TM}$ is undecidable.

Proof (recap):

– suppose $E_{TM}$ is decidable
– we showed how to use $E_{TM}$ to decide $A_{TM}$
– conclude $A_{TM}$ is decidable. Contradiction.
Example reduction

• We proved
  \[ A_{TM} = \{<M, w> : M \text{ accepts input } w\} \]
  undecidable, by reduction from
  \[ \text{HALT} = \{<M, w> : M \text{ halts on input } w\} \]

• We proved
  \[ E_{TM} = \{<M> : L(M) = \emptyset\} \]
  undecidable by reduction from \( A_{TM} \)
Definition of reduction

• Can you reduce co-HALT to HALT?

• We know that HALT is RE
• Does this show that co-HALT is RE?
  – recall, we showed co-HALT is not RE

• our current notion of reduction cannot distinguish complements
Definition of reduction

• More refined notion of reduction:
  – “many-one” reduction (commonly)
  – “mapping” reduction (book)

A \[ \overset{f}{\longrightarrow} \]\ B

\begin{tabular}{c|c}
yes & yes \\
no & no \\
\end{tabular}

reduction from language A to language B
Definition of reduction

- function $f$ should be computable

**Definition:** $f : \Sigma^* \rightarrow \Sigma^*$ is computable if there exists a TM $M_f$ such that on every $w \in \Sigma^*$, $M_f$ halts on $w$ with $f(w)$ written on its tape.
Definition of reduction

• Notation: “A many-one reduces to B” is written

\[ A \leq_m B \]

– “yes maps to yes and no maps to no” means:

\[ w \in A \text{ maps to } f(w) \in B \text{ & } w \notin A \text{ maps to } f(w) \notin B \]

• B is at least as “hard” as A

  – more accurate: B at least as “expressive” as A
Using reductions

**Definition:** $A \leq_m B$ if there is a computable function $f$ such that for all $w$

\[ w \in A \iff f(w) \in B \]

**Theorem:** If $A \leq_m B$ and $B$ is decidable then $A$ is decidable.

**Proof:**

- decider for $A$: on input $w$, compute $f(w)$, run decider for $B$, do whatever it does.
Using reductions

• Main use: given language NEW, prove it is undecidable by showing OLD $\leq_m$ NEW, where OLD known to be undecidable
  – proof by contradiction
  – if NEW decidable, then OLD decidable
  – OLD undecidable. Contradiction.

• common to reduce in wrong direction.

• review this argument to check yourself.
Using reductions

**Theorem**: if $A \leq_m B$ and $B$ is RE then $A$ is RE

**Proof**: 
- TM for recognizing $A$: on input $w$, compute $f(w)$, run TM that recognizes $B$, do whatever it does.

• Main use: given language NEW, prove it is not RE by showing OLD $\leq_m$ NEW, where OLD known to be not RE.
Many-one reduction example

• Showed $E_{TM}$ undecidable. Consider:
  \[ \text{co-}E_{TM} = \{<M> : L(M) \neq \emptyset\} \]

\[ A_{TM} \quad f \quad \text{yes} \quad f \quad \text{no} \]
\[ \text{co-}E_{TM} \quad \text{yes} \quad \text{no} \]

• $f(<M, w>) = <M'>$ where $M'$ is TM that
  • on input $x$, if $x \neq w$, then reject
  • else simulate $M$ on $x$, and accept if $M$ does

• $f$ clearly computable
Many-one reduction example

- yes maps to yes?
  - if \(<M, w> \in A_{TM}\) then \(f(M, w) \in \text{co-}E_{TM}\)

- no maps to no?
  - if \(<M, w> \notin A_{TM}\) then \(f(M, w) \notin \text{co-}E_{TM}\)

- \(f(<M, w>) = <M'>\)
  where \(M'\) is TM that
  - on input \(x\), if \(x \neq w\), then reject
  - else simulate \(M\) on \(x\), and accept if \(M\) does

- \(f\) clearly computable
Undecidable problems

**Theorem:** The language

\[ \text{REGULAR} = \{ \langle M \rangle : M \text{ is a TM and } L(M) \text{ is regular} \} \]

is undecidable.

**Proof:**

– reduce from \( A_{TM} \) (i.e. show \( A_{TM} \leq_m \text{REGULAR} \))
– what should \( f(<M, w>) \) produce?
Undecidable problems

Proof:

\[ f(<M, w>) = <M'> \] described below

on input \( x \):

• if \( x \) has form \( 0^n1^n \), accept

• else simulate \( M \) on \( w \) and accept \( x \) if \( M \) accepts

• is \( f \) computable?

• YES maps to YES?

\[ <M, w> \in A_{TM} \Rightarrow f(M, w) \in \text{REGULAR} \]

• NO maps to NO?

\[ <M, w> \notin A_{TM} \Rightarrow f(M, w) \notin \text{REGULAR} \]