Reduction?

A problem $P_1$ reduces to a problem $P_2$ means:

“If we have a solution for $P_2$, then we have a solution for $P_1$.”
Reduction?

A problem $P_1$ reduces to a problem $P_2$ means:

“If we have a solution for $P_2$, then we have a solution for $P_1$.”

Which is not true?

A. $A_{TM}$ reduces to $HALT_{TM}$
B. $HALT_{TM}$ reduces to $A_{TM}$
C. $EQ_{DFA}$ reduces to $E_{DFA}$
D. $\Sigma^*$ reduces to $EQ_{CFG}$
E. $EQ_{CFG}$ reduces to $\Sigma^*$
Reduction?

A problem $P_1$ reduces to a problem $P_2$ means:

“If we have a solution for $P_2$, then we have a solution for $P_1$.”

If $P_1$ is decidable, must $P_2$ also be decidable?

A. Yes
B. No
C. I don’t know.

Solving a problem means deciding membership in a language.
Reduction?

A problem $P_1$ reduces to a problem $P_2$ means:

“If we have a solution for $P_2$, then we have a solution for $P_1$.”

If $P_2$ is decidable, must $P_1$ also be decidable?

A. Yes
B. No
C. I don’t know.

Solving a problem means deciding membership in a language
Reduction?

A problem $P_1$ reduces to a problem $P_2$ means:

“If we have a solution for $P_2$, then we have a solution for $P_1$.”

If $P_2$ is decidable, must $P_1$ also be decidable?

A. Yes
B. No
C. I don’t know.

Solving a problem means deciding membership in a language

Contrapositive?
Reduction?

If \( P_1 \) reduces to \( P_2 \) and \( P_1 \) is undecidable, then \( P_2 \) is undecidable.

**Strategy**: to prove that a problem \( P \) is undecidable, prove that a problem we know to be undecidable reduces to it.  

[Let \( P \) play the role of \( P_2 \).]
Claim: $E_{TM}$ is undecidable. (Theorem 5.2)

\[ E_{TM} = \{ \langle M \rangle | M \text{ is a TM and } L(M) \text{ is empty} \} \]

i.e. want to recognize codes of TMs that always reject /loop

• Proof by reduction?

To use proof by reduction to prove that $E_{TM}$ is undecidable, we must reduce an undecidable set to $E_{TM}$
Claim: $E_{TM}$ is undecidable.

- **Proof by reduction**
  
  - **Goal:** show that $A_{TM}$ reduces to $E_{TM}$.
  - i.e. Build an algorithm that uses a decider for $E_{TM}$ as a subroutine and that decides $A_{TM}$
  
  - **Assume:** have a TM, $R$, that decides $E_{TM}$
  
  - **Build:** new TM, $M_{ATM}$, that decides $A_{TM}$
    
    - Always halts
    - Accepts iff input $<M,w>$ and $w$ is in $L(M)$. 
Claim: $E_{TM}$ is undecidable.

- **Proof by reduction…**
  - Assume: have a TM, $R$, that decides $E_{TM}$
  - Build: new TM, $M_{ATM}$, that decides $A_{TM}$
    - Always halts
    - Accepts iff input $<M,w>$ and $w$ is in $L(M)$.

  "On input $<M,w>$:
  - First, build the TM $X$ which, on input $x$, ignores $x$ and simulates $M$ on $w$.\"
Claim: $E_{TM}$ is undecidable.

Proof by reduction…

- Assume: have a TM, $R$, that decides $E_{TM}$
- Build: new TM, $M_{ATM}$, that decides $A_{TM}$
  - Always halts
  - Accepts iff input $<M,w>$ and $w$ is in $L(M)$.

"On input $<M,w>$:
  - First, build the TM $X$ which, on input $x$, ignores $x$ and simulates $M$ on $w$.

Is $X$ a decider?
A. Yes
B. No
C. I don't know.
Claim: $E_{TM}$ is undecidable.

- Proof by reduction…
  - Assume: have a TM, $R$, that decides $E_{TM}$
  - Build: new TM, $M_{ATM}$, that decides $A_{TM}$
    - Always halts
    - Accepts iff input $<M, w>$ and $w$ is in $L(M)$.
  - "On input $<M, w>$:
    - First, build the TM $X$ which, on input $x$, ignores $x$ and simulates $M$ on $w$.

What does it mean if $L(X)$ is empty?
A. $x$ is the empty string
B. $x = w$
C. $X = M$
D. $M$ accepts $w$
E. $M$ rejects $w$
Claim: $E_{TM}$ is undecidable.

- **Proof by reduction…**
  - Assume: have a TM, $R$, that decides $E_{TM}$
  - Build: new TM, $M_{ATM}$, that decides $A_{TM}$
    - Always halts
    - Accepts iff input $<M,w>$ and $w$ is in $L(M)$.
  - "On input $<M,w>$:
    - First, build the TM $X$ which, on input $x$, ignores $x$ and simulates $M$ on $w$.
    - Run $R$ on $<X>$.
      - If $R$ accepts, *reject*; if $R$ rejects, *accept."
Claim: $E_{TM}$ is undecidable.

- **Proof by reduction…**
  - Assume: have a TM, $R$, that decides $E_{TM}$
  - Build: new TM, $M_{ATM}$, that decides $A_{TM}$
    - Always halts
    - Accepts iff input $<M,w>$ and $w$ is in $L(M)$.

- "On input $<M,w>$:
  - First, build the TM $X$ which, on input $x$, ignores $x$ and simulates $M$ on $w$.
  - Run $R$ on $<X>$.
    - If $R$ accepts, reject; if $R$ rejects, accept."

- Correctness: ...

Is this machine $M_{ATM}$ a decider?

A. Yes
B. No
C. I don’t know.
So far

Decidable

$A_{DFA}$
$E_{DFA}$
$EQ_{DFA}$

Undecidable

$A_{TM}$
$HALT_{TM}$
$E_{TM}$

To think about:
Are these undecidable languages recognizable?

Give algorithm!

Diagonalization OR reduction
General approach

To prove that \{ <M> | M is a TM and L(M) has property P } is undecidable

• Assume **towards a contradiction** that R is a decider for \{ <M> | M is a TM and L(M) has P }.
• Build decider for A<sup>TM</sup> by: "On input <M,w> 
  1. Construct a new TM X such that L(X) has P iff w in L(M) 
  2. Run R on <X>: if accepts, accept; if rejects, reject."

Note: sometimes easier to build X so that L(X) has P iff w **not** in L(M)
Reducing other problems?

\[ \text{INF}_{\text{TM}} = \{ <M> \mid M \text{ is TM and } L(M) \text{ is infinite} \} \]

To prove \( \text{INF}_{\text{TM}} \) is undecidable, we will reduce problems known to be undecidable to it. E.g.

- \( A_{\text{TM}} \): Input \( <M,w> \); Need to decide if \( w \) is in \( L(M) \).
- \( \text{HALT}_{\text{TM}} \): Input \( <M,w> \); Need to decide if \( M \) halts on \( w \).
- \( E_{\text{TM}} \): Input \( <M> \); need to decide if \( L(M) \) is empty.
Reducing other problems?

Let $R$ decide $\text{INF}_{\text{TM}} = \{ <M> | M \text{ is TM and } L(M) \text{ is infinite} \}$

$A_{\text{TM}}$: Input $<M,w>$; Need to decide if $w$ is in $L(M)$.

$M_{\text{ATM}}$: "On input $<M,w>$

1. Build $X$ such that $L(X)$ is infinite iff $w$ is in $L(M)$.
2. Run $R$ on $X$. If accepts, accept; if rejects, reject."

$X$: "On input $x$

1. Run $M$ on $w$. If accepts, accept; if rejects, reject."
Reducing other problems?

Let $R$ decide $\text{INF}_\text{TM} = \{ <M> \mid M \text{ is TM and } L(M) \text{ is infinite} \}$

$\text{HALT}_\text{TM}$: Input $<M, w>$; Need to decide if $M$ halts on $w$.

$M_{\text{HALT}}$: "On input $<M, w>$
1. Build $X$ such that $L(X)$ is infinite iff ________________.
2. Run $R$ on $X$. If accepts, accept; if rejects, reject."

$X$: "On input $x$
1. Run $M$ on $w$. If ________________."
Reducing other problems?

Let $R$ decide $\text{INF}_{\text{TM}} = \{ \langle M \rangle \mid M \text{ is TM and } L(M) \text{ is infinite} \}$.

$\text{HALT}_{\text{TM}}$: Input $\langle M, w \rangle$;

$M_{\text{HALT}}$: "On input $\langle M, w \rangle$

1. Build $X$ such that $L(X)$ is infinite iff ________________.
2. Run $R$ on $X$. If accepts, accept; if rejects, reject."

$X$: "On input $x$
1. Run $M$ on $w$. If ________________."
Reducing other problems?

Let $R$ decide $\textbf{INF}_{\text{TM}} = \{<M> | M \text{ is TM and } L(M) \text{ is infinite}\}$

$E_{\text{TM}}$: Input $<M>$; Need to decide if $L(M)$ is empty.

$M_{\text{ETM}}$: "On input $<M>$
1. Build $X$ such that $L(X)$ is infinite iff $L(M)$ is not empty.
2. Run $R$ on $X$. If accepts, reject; if rejects, accept."

$X$: "On input $x$
1. Run $M$ on ________________________."
Last example

\[ \text{EQ}_{\text{TM}} = \{ <M_1, M_2> \mid M_1 \text{ and } M_2 \text{ are TMs and } L(M_1) = L(M_2) \} \]

Claim: \( \text{EQ}_{\text{TM}} \) is undecidable.

How do we pick which problem to reduce?

Option 1: "Stick with what we know" … \( A_{\text{TM}} \)
Option 2: "The road less travelled" … \( E_{\text{TM}} \)
Last example

Option 1: "Stick with what we know" … \( A_{TM} \)

Given \( M_{EQ} \) deciding \( EQ_{TM} \), build \( M_{ATM} \): "On input \( <M,w> \),
1. Build machine \( M_{acc} \) that accepts all inputs.
2. Build machine \( X \) that ignores its input and runs \( M \) on \( w \).
3. Run \( M_{EQ} \) on \( <M_{acc}, X> \). If accepts, accept; if rejects, reject."
Last example

Option 2: "The road less travelled" … $E_{TM}$

Given $M_{EQ}$ deciding $EQ_{TM}$, build $M_{ETM}$: "On input $<M>$,
1. Build machine $M_{rej}$ that rejects all inputs.
2. Run $M_{EQ}$ on $<M_{rej}, M>$. If accepts, accept; if rejects, reject."
Exercise

Claim: Exactly one of $E_{TM}$ and its complement is recognizable.

Proof:

Why not both?

Which is?