Undecidability
CSCI 338

EQ_{TM}

Claim: $EQ_{TM} = \{\langle M, N\rangle : M, N$ are TMs and $L(M) = L(N)\}$ is undecidable.

Proof: Suppose EQ_{TM} is decidable and let TM H be its decider.

Build a TM S that decides A_{TM}:

S = on input $\langle N, \omega\rangle$

1. Construct TM M_1 on input $\langle x\rangle$:
   1. accept.
2. Construct TM M_2 on input $\langle y\rangle$:
   1. Run N on ω and accept if N does.
3. Run H on $\langle M_1, M_2\rangle$.
4. If H accepts, accept. If H rejects, reject.

$\boldsymbol{N}$ **accepts** $\boldsymbol{\omega}$

$\Updownarrow$

$\boldsymbol{L(M_2) = \Sigma^*}$

If N accepts ω, then M_1 and M_2 have the same language (Σ^*). If N does not accept ω, then they have different languages. Thus S decides A_{TM}. (bad!)

EQ_{TM}

Claim: $EQ_{TM} = \{\langle M, N\rangle : M, N$ are TMs and $L(M) = L(N)\}$ is undecidable.

Proof: Suppose EQ_{TM} is decidable and let TM H be its decider.

Build a TM S that decides E_{TM}:

S = on input $\langle P\rangle$

1.

$$\boxed{\text{To show } EQ_{TM} \text{ is undecidable, use it to decide } E_{TM}.}$$

EQ_{TM}

Claim: $EQ_{TM} = \{\langle M, N\rangle : M, N$ are TMs and $L(M) = L(N)\}$ is undecidable.

Proof: Suppose EQ_{TM} is decidable and let TM H be its decider.

Build a TM S that decides E_{TM}:

S = on input $\langle P\rangle$

1.

We have a way (H) to test if two TMs have the same language.

How could we use that to test if a TM's language is empty?

Plan: ?

EQ_{TM}

Claim: $EQ_{TM} = \{\langle M, N\rangle : M, N$ are TMs and $L(M) = L(N)\}$ is undecidable.

Proof: Suppose EQ_{TM} is decidable and let TM H be its decider.

Build a TM S that decides E_{TM}:

S = on input $\langle P\rangle$

1.

We have a way (H) to test if two TMs have the same language.

How could we use that to test if a TM's language is empty?

Plan: Make a TM with an empty language and use H to compare it to input to E_{TM}.

EQ_{TM}

Claim: $EQ_{TM} = \{\langle M, N\rangle : M, N$ are TMs and $L(M) = L(N)\}$ is undecidable.

Proof: Suppose EQ_{TM} is decidable and let TM H be its decider.

Build a TM S that decides E_{TM}:

S = on input $\langle P\rangle$

1. Construct TM M_2 on input $\langle x\rangle$:
    1. reject.

EQ_{TM}

Claim: $EQ_{TM} = \{\langle M, N\rangle : M, N$ are TMs and $L(M) = L(N)\}$ is undecidable.

Proof: Suppose EQ_{TM} is decidable and let TM H be its decider.

Build a TM S that decides E_{TM}:

S = on input $\langle P\rangle$

1. Construct TM M_2 on input $\langle x\rangle$:
    1. reject.

$\boldsymbol{L(M_2) = ?}$

EQ_{TM}

Claim: $EQ_{TM} = \{\langle M, N\rangle : M, N$ are TMs and $L(M) = L(N)\}$ is undecidable.

Proof: Suppose EQ_{TM} is decidable and let TM H be its decider.

Build a TM S that decides E_{TM}:

S = on input $\langle P\rangle$

1. Construct TM M_2 on input $\langle x\rangle$:
   1. <u>reject</u>.

$\boldsymbol{L(M_2) = \emptyset}$

EQ_{TM}

Claim: $EQ_{TM} = \{\langle M, N\rangle : M, N$ are TMs and $L(M) = L(N)\}$ is undecidable.

Proof: Suppose EQ_{TM} is decidable and let TM H be its decider.

Build a TM S that decides E_{TM}:

S = on input $\langle P\rangle$

1. Construct TM M_2 on input $\langle x\rangle$:
   1. reject.
2. ?

EQ_{TM}

Claim: $EQ_{TM} = \{\langle M, N\rangle : M, N$ are TMs and $L(M) = L(N)\}$ is undecidable.

Proof: Suppose EQ_{TM} is decidable and let TM H be its decider.

Build a TM S that decides E_{TM}:

S = on input $\langle P\rangle$

1. Construct TM M_2 on input $\langle x\rangle$:
   1. reject.
2. Run H on $\langle P, M_2\rangle$.

EQ_{TM}

Claim: $EQ_{TM} = \{\langle M, N\rangle : M, N$ are TMs and $L(M) = L(N)\}$ is undecidable.

Proof: Suppose EQ_{TM} is decidable and let TM H be its decider.

Build a TM S that decides E_{TM}:

S = on input $\langle P\rangle$

1. Construct TM M_2 on input $\langle x\rangle$:
   1. reject.
2. Run H on $\langle P, M_2\rangle$.
3. If H accepts, ___?___. If H rejects, ___?___.

EQ_{TM}

Claim: $EQ_{TM} = \{\langle M, N\rangle : M, N \text{ are TMs and } L(M) = L(N)\}$ is undecidable.

Proof: Suppose EQ_{TM} is decidable and let TM H be its decider.

Build a TM S that decides E_{TM}:

S = on input $\langle P\rangle$

1. Construct TM M_2 on input $\langle x\rangle$:
    1. <u>reject</u>.
2. Run H on $\langle P, M_2\rangle$.
3. If H accepts, <u>accept</u>. If H rejects, <u>reject</u>.

EQ_{TM}

Claim: $EQ_{TM} = \{\langle M, N\rangle : M, N$ are TMs and $L(M) = L(N)\}$ is undecidable.

Proof: Suppose EQ_{TM} is decidable and let TM H be its decider.

Build a TM S that decides E_{TM}:

S = on input $\langle P\rangle$

1. Construct TM M_2 on input $\langle x\rangle$:
    1. reject.
2. Run H on $\langle P, M_2\rangle$.
3. If H accepts, accept. If H rejects, reject.

If…?

EQ_{TM}

Claim: $EQ_{TM} = \{\langle M, N\rangle : M, N$ are TMs and $L(M) = L(N)\}$ is undecidable.

Proof: Suppose EQ_{TM} is decidable and let TM H be its decider.

Build a TM S that decides E_{TM}:

S = on input $\langle P\rangle$

1. Construct TM M_2 on input $\langle x\rangle$:
   1. <u>reject</u>.
2. Run H on $\langle P, M_2\rangle$.
3. If H accepts, <u>accept</u>. If H rejects, <u>reject</u>.

If $L(P) = \emptyset$, …?

EQ_{TM}

Claim: $EQ_{TM} = \{\langle M, N\rangle : M, N$ are TMs and $L(M) = L(N)\}$ is undecidable.

Proof: Suppose EQ_{TM} is decidable and let TM H be its decider.

Build a TM S that decides E_{TM}:

S = on input $\langle P\rangle$

1. Construct TM M_2 on input $\langle x\rangle$:
   1. reject.
2. Run H on $\langle P, M_2\rangle$.
3. If H accepts, accept. If H rejects, reject.

If $L(P) = \emptyset$, M_2 and P will have the same language (since $L(M_2) = \emptyset$) and…?

EQ_{TM}

Claim: $EQ_{TM} = \{\langle M, N\rangle : M, N$ are TMs and $L(M) = L(N)\}$ is undecidable.

Proof: Suppose EQ_{TM} is decidable and let TM H be its decider.

Build a TM S that decides E_{TM}:

S = on input $\langle P\rangle$

1. Construct TM M_2 on input $\langle x\rangle$:
    1. <u>reject</u>.
2. Run H on $\langle P, M_2\rangle$.
3. If H accepts, <u>accept</u>. If H rejects, <u>reject</u>.

If $L(P) = \emptyset$, M_2 and P will have the same language (since $L(M_2) = \emptyset$) and S will accept.

EQ_{TM}

Claim: $EQ_{TM} = \{\langle M, N\rangle : M, N$ are TMs and $L(M) = L(N)\}$ is undecidable.

Proof: Suppose EQ_{TM} is decidable and let TM H be its decider.

Build a TM S that decides E_{TM}:

S = on input $\langle P\rangle$

1. Construct TM M_2 on input $\langle x\rangle$:
    1. reject.
2. Run H on $\langle P, M_2\rangle$.
3. If H accepts, accept. If H rejects, reject.

If $L(P) = \emptyset$, M_2 and P will have the same language (since $L(M_2) = \emptyset$) and S will accept. If $L(P) \neq \emptyset$, …?

EQ_{TM}

Claim: $EQ_{TM} = \{\langle M, N\rangle : M, N$ are TMs and $L(M) = L(N)\}$ is undecidable.

Proof: Suppose EQ_{TM} is decidable and let TM H be its decider.

Build a TM S that decides E_{TM}:

S = on input $\langle P\rangle$

1. Construct TM M_2 on input $\langle x\rangle$:
   1. reject.
2. Run H on $\langle P, M_2\rangle$.
3. If H accepts, accept. If H rejects, reject.

If $L(P) = \emptyset$, M_2 and P will have the same language (since $L(M_2) = \emptyset$) and S will accept. If $L(P) \neq \emptyset$, M_2 and P will not have the same language and…?

EQ_{TM}

Claim: $EQ_{TM} = \{\langle M, N\rangle : M, N$ are TMs and $L(M) = L(N)\}$ is undecidable.

Proof: Suppose EQ_{TM} is decidable and let TM H be its decider.

Build a TM S that decides E_{TM}:

S = on input $\langle P\rangle$

1. Construct TM M_2 on input $\langle x\rangle$:
   1. <u>reject</u>.
2. Run H on $\langle P, M_2\rangle$.
3. If H accepts, <u>accept</u>. If H rejects, <u>reject</u>.

If $L(P) = \emptyset$, M_2 and P will have the same language (since $L(M_2) = \emptyset$) and S will accept. If $L(P) \neq \emptyset$, M_2 and P will not have the same language and S will reject.

EQ_{TM}

Claim: $EQ_{TM} = \{\langle M, N\rangle : M, N$ are TMs and $L(M) = L(N)\}$ is undecidable.

Proof: Suppose EQ_{TM} is decidable and let TM H be its decider.

Build a TM S that decides E_{TM}:

S = on input $\langle P\rangle$

1. Construct TM M_2 on input $\langle x\rangle$:
   1. reject.
2. Run H on $\langle P, M_2\rangle$.
3. If H accepts, accept. If H rejects, reject.

If $L(P) = \emptyset$, M_2 and P will have the same language (since $L(M_2) = \emptyset$) and S will accept. If $L(P) \neq \emptyset$, M_2 and P will not have the same language and S will reject. Therefore, S is a decider for E_{TM}, which is a contradiction, so EQ_{TM} is undecidable.

$REGULAR_{TM}$

Claim: $REGULAR_{TM} = \{\langle M \rangle : M$ is a TM and $L(M)$ is regular$\}$ is undecidable.

Proof:

?

$REGULAR_{TM}$

Claim: $REGULAR_{TM} = \{\langle M \rangle : M$ is a TM and $L(M)$ is regular$\}$ is undecidable.

Proof: Suppose $REGULAR_{TM}$ is decidable and let TM H be its decider.

$REGULAR_{TM}$

Claim: $REGULAR_{TM} = \{\langle M \rangle : M$ is a TM and $L(M)$ is regular$\}$ is undecidable.

Proof: Suppose $REGULAR_{TM}$ is decidable and let TM H be its decider.

Build a TM S that decides A_{TM}:

S = on input $\langle N, \omega \rangle$

1.

$\boxed{\text{To show } REGULAR_{TM} \text{ is undecidable, use it to decide } A_{TM}.}$

$REGULAR_{TM}$

Claim: $REGULAR_{TM} = \{\langle M \rangle : M$ is a TM and $L(M)$ is regular$\}$ is undecidable.

Proof: Suppose $REGULAR_{TM}$ is decidable and let TM H be its decider.

Build a TM S that decides A_{TM}:

S = on input $\langle N, \omega \rangle$

1.

Plan: Build a TM whose language is regular if N accepts ω and not regular if N does not accept ω.

$REGULAR_{TM}$

Claim: $REGULAR_{TM} = \{\langle M\rangle : M$ is a TM and $L(M)$ is regular$\}$ is undecidable.

Proof: Suppose $REGULAR_{TM}$ is decidable and let TM H be its decider.

Build a TM S that decides A_{TM}:

S = on input $\langle N, \omega\rangle$

1. Construct TM M_2 on input $\langle x\rangle$:

?

$L(M_2)$ is regular
$\Updownarrow$
N accepts ω

Plan: Build a TM whose language is regular if N accepts ω and not regular if N does not accept ω.

$REGULAR_{TM}$

Claim: $REGULAR_{TM} = \{\langle M \rangle : M$ is a TM and $L(M)$ is regular$\}$ is undecidable.

Proof: Suppose $REGULAR_{TM}$ is decidable and let TM H be its decider.

Build a TM S that decides A_{TM}:

S = on input $\langle N, \omega \rangle$

1. Construct TM M_2 on input $\langle x \rangle$:
    1. If $x \in \{ \quad ??? \quad \}$, <u>accept</u>.
    2. If $x \notin \{ \quad ??? \quad \}$, run N on ω and <u>accept</u> if N does.

$$\boxed{\begin{array}{c} L(M_2) \text{ is regular} \\ \Updownarrow \\ N \text{ accepts } \omega \end{array}}$$

Plan: Build a TM whose language is regular if N accepts ω and not regular if N does not accept ω.

$REGULAR_{TM}$

Claim: $REGULAR_{TM} = \{\langle M \rangle : M$ is a TM and $L(M)$ is regular$\}$ is undecidable.

Proof: Suppose $REGULAR_{TM}$ is decidable and let TM H be its decider.

Build a TM S that decides A_{TM}:

S = on input $\langle N, \omega \rangle$

1. Construct TM M_2 on input $\langle x \rangle$:
    1. If $x \in \{0^n 1^n : n \geq 0\}$, <u>accept</u>.
    2. If $x \notin \{0^n 1^n : n \geq 0\}$, run N on ω and <u>accept</u> if N does.

$L(M_2)$ is regular

$\Updownarrow$

N accepts ω

Plan: Build a TM whose language is regular if N accepts ω and not regular if N does not accept ω.

$REGULAR_{TM}$

Claim: $REGULAR_{TM} = \{\langle M \rangle : M$ is a TM and $L(M)$ is regular$\}$ is undecidable.

Proof: Suppose $REGULAR_{TM}$ is decidable and let TM H be its decider.

Build a TM S that decides A_{TM}:

S = on input $\langle N, \omega \rangle$

1. Construct TM M_2 on input $\langle x \rangle$:
   1. If $x \in \{0^n 1^n : n \geq 0\}$, <u>accept</u>.
   2. If $x \notin \{0^n 1^n : n \geq 0\}$, run N on ω and <u>accept</u> if N does.

$\boldsymbol{L(M_2) = ?}$

Plan: Build a TM whose language is regular if N accepts ω and not regular if N does not accept ω.

$REGULAR_{TM}$

Claim: $REGULAR_{TM} = \{\langle M\rangle : M$ is a TM and $L(M)$ is regular$\}$ is undecidable.

Proof: Suppose $REGULAR_{TM}$ is decidable and let TM H be its decider.

Build a TM S that decides A_{TM}:

S = on input $\langle N, \omega\rangle$

1. Construct TM M_2 on input $\langle x\rangle$:
    1. If $x \in \{0^n1^n : n \geq 0\}$, accept.
    2. If $x \notin \{0^n1^n : n \geq 0\}$, run N on ω and accept if N does.

$\boldsymbol{L(M_2) = 0^n1^n}$ **or** $\boldsymbol{\Sigma^*}$

Plan: Build a TM whose language is regular if N accepts ω and not regular if N does not accept ω.

$REGULAR_{TM}$

Claim: $REGULAR_{TM} = \{\langle M \rangle : M$ is a TM and $L(M)$ is regular$\}$ is undecidable.

Proof: Suppose $REGULAR_{TM}$ is decidable and let TM H be its decider.

Build a TM S that decides A_{TM}:

S = on input $\langle N, \omega \rangle$

1. Construct TM M_2 on input $\langle x \rangle$:
   1. If $x \in \{0^n 1^n : n \geq 0\}$, accept.
   2. If $x \notin \{0^n 1^n : n \geq 0\}$, run N on ω and accept if N does.
2. ?

$REGULAR_{TM}$

Claim: $REGULAR_{TM} = \{\langle M\rangle : M$ is a TM and $L(M)$ is regular$\}$ is undecidable.

Proof: Suppose $REGULAR_{TM}$ is decidable and let TM H be its decider.

Build a TM S that decides A_{TM}:

S = on input $\langle N, \omega\rangle$

1. Construct TM M_2 on input $\langle x\rangle$:
   1. If $x \in \{0^n 1^n : n \geq 0\}$, <u>accept</u>.
   2. If $x \notin \{0^n 1^n : n \geq 0\}$, run N on ω and <u>accept</u> if N does.
2. Run H on $\langle M_2\rangle$.

$REGULAR_{TM}$

Claim: $REGULAR_{TM} = \{\langle M \rangle : M$ is a TM and $L(M)$ is regular$\}$ is undecidable.

Proof: Suppose $REGULAR_{TM}$ is decidable and let TM H be its decider.

Build a TM S that decides A_{TM}:

S = on input $\langle N, \omega \rangle$

1. Construct TM M_2 on input $\langle x \rangle$:
   1. If $x \in \{0^n 1^n : n \geq 0\}$, <u>accept</u>.
   2. If $x \notin \{0^n 1^n : n \geq 0\}$, run N on ω and <u>accept</u> if N does.
2. Run H on $\langle M_2 \rangle$.
3. If H accepts, <u>accept</u>. If H rejects, <u>reject</u>.

$REGULAR_{TM}$

Claim: $REGULAR_{TM} = \{\langle M \rangle : M$ is a TM and $L(M)$ is regular$\}$ is undecidable.

Proof: Suppose $REGULAR_{TM}$ is decidable and let TM H be its decider.

Build a TM S that decides A_{TM}:

S = on input $\langle N, \omega \rangle$

1. Construct TM M_2 on input $\langle x \rangle$:
    1. If $x \in \{0^n 1^n : n \geq 0\}$, <u>accept</u>.
    2. If $x \notin \{0^n 1^n : n \geq 0\}$, run N on ω and <u>accept</u> if N does.
2. Run H on $\langle M_2 \rangle$.
3. If H accepts, <u>accept</u>. If H rejects, <u>reject</u>.

If N accepts ω, $L(M_2) = \Sigma^*$ (regular).

$REGULAR_{TM}$

Claim: $REGULAR_{TM} = \{\langle M \rangle : M$ is a TM and $L(M)$ is regular$\}$ is undecidable.

Proof: Suppose $REGULAR_{TM}$ is decidable and let TM H be its decider.

Build a TM S that decides A_{TM}:

S = on input $\langle N, \omega \rangle$

1. Construct TM M_2 on input $\langle x \rangle$:
    1. If $x \in \{0^n 1^n : n \geq 0\}$, <u>accept</u>.
    2. If $x \notin \{0^n 1^n : n \geq 0\}$, run N on ω and <u>accept</u> if N does.
2. Run H on $\langle M_2 \rangle$.
3. If H accepts, <u>accept</u>. If H rejects, <u>reject</u>.

If N accepts ω, $L(M_2) = \Sigma^*$ (regular). If N does not accept ω, $L(M_2) = \{0^n 1^n : n \geq 0\}$ (not regular).

$REGULAR_{TM}$

Claim: $REGULAR_{TM} = \{\langle M \rangle : M$ is a TM and $L(M)$ is regular$\}$ is undecidable.

Proof: Suppose $REGULAR_{TM}$ is decidable and let TM H be its decider.

Build a TM S that decides A_{TM}:

S = on input $\langle N, \omega \rangle$

1. Construct TM M_2 on input $\langle x \rangle$:
    1. If $x \in \{0^n 1^n : n \geq 0\}$, accept.
    2. If $x \notin \{0^n 1^n : n \geq 0\}$, run N on ω and accept if N does.
2. Run H on $\langle M_2 \rangle$.
3. If H accepts, accept. If H rejects, reject.

If N accepts ω, $L(M_2) = \Sigma^*$ (regular). If N does not accept ω, $L(M_2) = \{0^n 1^n : n \geq 0\}$ (not regular). So, deciding if $L(M_2)$ is regular will determine if N accepts ω.

$REGULAR_{TM}$

Claim: $REGULAR_{TM} = \{\langle M \rangle : M$ is a TM and $L(M)$ is regular$\}$ is undecidable.

Proof: Suppose $REGULAR_{TM}$ is decidable and let TM H be its decider.

Build a TM S that decides A_{TM}:

S = on input $\langle N, \omega \rangle$

1. Construct TM M_2 on input $\langle x \rangle$:
   1. If $x \in \{0^n 1^n : n \geq 0\}$, <u>accept</u>.
   2. If $x \notin \{0^n 1^n : n \geq 0\}$, run N on ω and <u>accept</u> if N does.
2. Run H on $\langle M_2 \rangle$.
3. If H accepts, <u>accept</u>. If H rejects, <u>reject</u>.

If N accepts ω, $L(M_2) = \Sigma^*$ (regular). If N does not accept ω, $L(M_2) = \{0^n 1^n : n \geq 0\}$ (not regular). So, deciding if $L(M_2)$ is regular will determine if N accepts ω. Therefore, S is a decider for A_{TM}, so $REGULAR_{TM}$ is undecidable.

When in doubt use A_{TM}!!!

Unrecognizable Language

Claim: A language is decidable $\Longleftrightarrow$ it and its complement are Turing-recognizable.

Proof:

Unrecognizable Language

Claim: A language is decidable $\Leftrightarrow$ it and its complement are Turing-recognizable.

Proof: $\Longrightarrow$ If a language is decidable, its complement is also decidable (just reverse accept/reject conditions) and decidable languages are recognizable.

Given decider T for A, make decider for $\bar{A}$:

M = on input ω

1. Run T on ω.
2. If T accepts, <u>reject</u>. If T rejects, <u>accept</u>.

Unrecognizable Language

Claim: A language is decidable $\Leftrightarrow$ it and its complement are Turing-recognizable.

Proof: $\Longrightarrow$ If a language is decidable, its complement is also decidable (just reverse accept/reject conditions) and decidable languages are recognizable.

$\Longleftarrow$ If A and $\bar{A}$ are both Turing-recognizable, let M_1 and M_2 be recognizers for A and $\bar{A}$.

Unrecognizable Language

Claim: A language is decidable $\Leftrightarrow$ it and its complement are Turing-recognizable.

Proof: $\Longrightarrow$ If a language is decidable, its complement is also decidable (just reverse accept/reject conditions) and decidable languages are recognizable.

$\Longleftarrow$ If A and $\bar{A}$ are both Turing-recognizable, let M_1 and M_2 be recognizers for A and $\bar{A}$. Consider the following TM:

M = on input ω

1. Run both M_1 and M_2 on ω in parallel (alternate instructions).
2. If M_1 accepts, <u>accept</u>. If M_2 accepts, <u>reject</u>.

Unrecognizable Language

Claim: A language is decidable $\Leftrightarrow$ it and its complement are Turing-recognizable.

Proof: $\Longrightarrow$ If a language is decidable, its complement is also decidable (just reverse accept/reject conditions) and decidable languages are recognizable.

$\Longleftarrow$ If A and $\bar{A}$ are both Turing-recognizable, let M_1 and M_2 be recognizers for A and $\bar{A}$. Consider the following TM:

M = on input ω

1. Run both M_1 and M_2 on ω in parallel (alternate instructions).
2. If M_1 accepts, <u>accept</u>. If M_2 accepts, <u>reject</u>.

Since $\omega \in A$ or $\bar{A}$, M_1 or M_2 must accept (halts on input). Thus, M is a decider for A.

Unrecognizable Language

Claim: $\overline{HALT_{TM}} = \{\langle M, \omega \rangle : M$ is a TM and M does not halt on $\omega\}$ is not Turing-recognizable.

Proof:

Unrecognizable Language

Claim: $\overline{HALT_{TM}} = \{\langle M, \omega\rangle\text{: } M$ is a TM and M does not halt on ω} is not Turing-recognizable.

Proof: Suppose $\overline{HALT_{TM}}$ was Turing-recognizable. Let T be its recognizer (i.e., ????).

Unrecognizable Language

Claim: $\overline{HALT_{TM}} = \{\langle M, \omega\rangle \colon M$ is a TM and M does not halt on $\omega\}$ is not Turing-recognizable.

Proof: Suppose $\overline{HALT_{TM}}$ was Turing-recognizable. Let T be its recognizer (i.e., T will accept if a TM does ***not*** halt on some input).

Unrecognizable Language

Claim: $\overline{HALT_{TM}} = \{\langle M, \omega \rangle: M$ is a TM and M does not halt on ω} is not Turing-recognizable.

Proof: Suppose $\overline{HALT_{TM}}$ was Turing-recognizable. Let T be its recognizer (i.e., T will accept if a TM does ***not*** halt on some input).

Consider S on $\langle M, \omega \rangle$:

Unrecognizable Language

Claim: $\overline{HALT_{TM}} = \{\langle M, \omega\rangle\colon M$ is a TM and M does not halt on ω} is not Turing-recognizable.

Proof: Suppose $\overline{HALT_{TM}}$ was Turing-recognizable. Let T be its recognizer (i.e., T will accept if a TM does ***not*** halt on some input).

Consider S on $\langle N, \omega\rangle$:

1. Run N on ω.
2. accept.

Unrecognizable Language

Claim: $\overline{HALT_{TM}} = \{\langle M, \omega\rangle \colon M$ is a TM and M does not halt on ω} is not Turing-recognizable.

Proof: Suppose $\overline{HALT_{TM}}$ was Turing-recognizable. Let T be its recognizer (i.e., T will accept if a TM does ***not*** halt on some input).

Consider S on $\langle N, \omega\rangle$:

1. Run N on ω.
2. accept.

???

Unrecognizable Language

Claim: $\overline{HALT_{TM}} = \{\langle M, \omega\rangle : M$ is a TM and M does not halt on ω} is not Turing-recognizable.

Proof: Suppose $\overline{HALT_{TM}}$ was Turing-recognizable. Let T be its recognizer (i.e., T will accept if a TM does ***not*** halt on some input).

Consider S on $\langle N, \omega\rangle$:

1. Run N on ω.
2. accept.

$\boldsymbol{HALT_{TM}}$ **recognizer!**

Unrecognizable Language

Claim: $\overline{HALT_{TM}} = \{\langle M, \omega\rangle : M$ is a TM and M does not halt on ω} is not Turing-recognizable.

Proof: Suppose $\overline{HALT_{TM}}$ was Turing-recognizable. Let T be its recognizer (i.e., T will accept if a TM does ***not*** halt on some input).

Consider S on $\langle N, \omega\rangle$:

1. Run N on ω.
2. accept.

$\boldsymbol{HALT_{TM}}$ **recognizer!**

Consider V on $\langle N, \omega\rangle$:

Unrecognizable Language

Claim: $\overline{HALT_{TM}} = \{\langle M, \omega\rangle : M$ is a TM and M does not halt on $\omega\}$ is not Turing-recognizable.

Proof: Suppose $\overline{HALT_{TM}}$ was Turing-recognizable. Let T be its recognizer (i.e., T will accept if a TM does ***not*** halt on some input).

Consider S on $\langle N, \omega\rangle$:

1. Run N on ω.
2. accept.

$\boldsymbol{HALT_{TM}}$ **recognizer!**

Consider V on $\langle N, \omega\rangle$:

1. Run T on $\langle N, \omega\rangle$ and run S on $\langle N, \omega\rangle$ in parallel.

Unrecognizable Language

Claim: $\overline{HALT_{TM}} = \{\langle M, \omega\rangle : M$ is a TM and M does not halt on $\omega\}$ is not Turing-recognizable.

Proof: Suppose $\overline{HALT_{TM}}$ was Turing-recognizable. Let T be its recognizer (i.e., T will accept if a TM does ***not*** halt on some input).

Consider S on $\langle N, \omega\rangle$:

1. Run N on ω.
2. accept.

$\boldsymbol{HALT_{TM}}$ **recognizer!**

Consider V on $\langle N, \omega\rangle$:

1. Run T on $\langle N, \omega\rangle$ and run S on $\langle N, \omega\rangle$ in parallel.
2. If T accepts, reject. If S accepts, accept.

Unrecognizable Language

Claim: $\overline{HALT_{TM}} = \{\langle M, \omega \rangle : M$ is a TM and M does not halt on $\omega\}$ is not Turing-recognizable.

Proof: Suppose $\overline{HALT_{TM}}$ was Turing-recognizable. Let T be its recognizer (i.e., T will accept if a TM does ***not*** halt on some input).

Consider S on $\langle N, \omega \rangle$:

1. Run N on ω.
2. accept.

$HALT_{TM}$ recognizer!

Consider V on $\langle N, \omega \rangle$:

1. Run T on $\langle N, \omega \rangle$ and run S on $\langle N, \omega \rangle$ in parallel.
2. If T accepts, reject. If S accepts, accept.

???

Unrecognizable Language

Claim: $\overline{HALT_{TM}} = \{\langle M, \omega\rangle : M$ is a TM and M does not halt on ω} is not Turing-recognizable.

Proof: Suppose $\overline{HALT_{TM}}$ was Turing-recognizable. Let T be its recognizer (i.e., T will accept if a TM does ***not*** halt on some input).

Consider S on $\langle N, \omega\rangle$:

1. Run N on ω.
2. accept.

$HALT_{TM}$ recognizer!

Consider V on $\langle N, \omega\rangle$:

1. Run T on $\langle N, \omega\rangle$ and run S on $\langle N, \omega\rangle$ in parallel.
2. If T accepts, reject. If S accepts, accept.

$HALT_{TM}$ decider!

Unrecognizable Language

Claim: $\overline{HALT_{TM}} = \{\langle M, \omega\rangle : M$ is a TM and M does not halt on $\omega\}$is not Turing-recognizable.

Proof:

A language is decidable $\Leftrightarrow$ it and its complement are Turing-recognizable.

Unrecognizable Language

Claim: $\overline{HALT_{TM}} = \{\langle M, \omega\rangle : M$ is a TM and M does not halt on ω}is not Turing-recognizable.

Proof: $HALT_{TM}$ is not decidable

A language is decidable ⟺ it and its complement are Turing-recognizable.

Unrecognizable Language

Claim: $\overline{HALT_{TM}} = \{\langle M, \omega \rangle: M$ is a TM and M does not halt on ω}is not Turing-recognizable.

Proof: $HALT_{TM}$ is not decidable $\Longrightarrow HALT_{TM}$ and $\overline{HALT_{TM}}$ cannot both be Turing-recognizable (otherwise $HALT_{TM}$ would be decidable).

A language is decidable ⟺ it and its complement are Turing-recognizable.

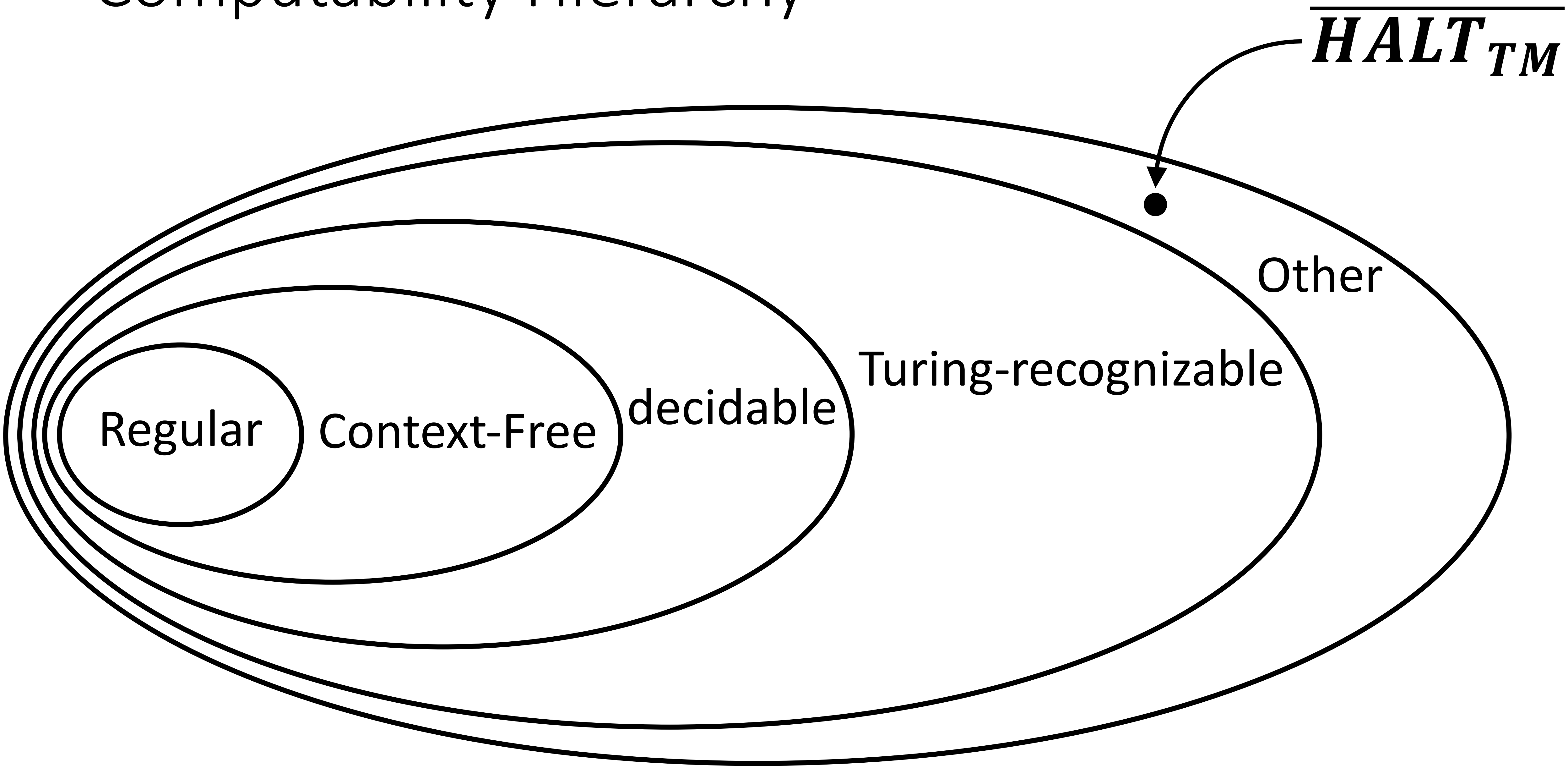
Unrecognizable Language

Claim: $\overline{HALT_{TM}} = \{\langle M, \omega \rangle : M$ is a TM and M does not halt on $\omega\}$is not Turing-recognizable.

Proof: $HALT_{TM}$ is not decidable $\Longrightarrow HALT_{TM}$ and $\overline{HALT_{TM}}$ cannot both be Turing-recognizable (otherwise $HALT_{TM}$ would be decidable). Since $HALT_{TM}$ is Turing-recognizable, $\overline{HALT_{TM}}$ cannot be Turing-recognizable.

A language is decidable $\Longleftrightarrow$ it and its complement are Turing-recognizable.

Computability Hierarchy



Beyond Decidability

What if $HALT_{TM}$ were “decidable”?

Beyond Decidability

What if $HALT_{TM}$ were “decidable”?

Goldbach’s Conjecture:

- 280-year-old open problem.
- Every integer ≥ 2 is sum of two primes.

Beyond Decidability

What if $HALT_{TM}$ were "decidable"?

Goldbach's Conjecture:

- 280-year-old open problem.
- Every integer ≥ 2 is sum of two primes.

Consider G on $\langle x \rangle$:

1. For $n = 2$, check each pair of prime number $< n$.
2. If no pair sums to n, <u>reject</u>.
3. Increment n and loop to step 1.

Beyond Decidability

```
public boolean G() {
   int i = 2;
   while (true) {
      boolean found = false;
      for (int n = 1; n < i; n++) {
         for (int m = 1; m < i; m++) {
            if (isPrime(n) && isPrime(m) && m + n = i) {
               found = true;
            }
         }
      }
      if (!found) {
         return false;
      }
      i++;
   }
}
```

Beyond Decidability

What if $HALT_{TM}$ were "decidable"?

Goldbach's Conjecture:

- 280-year-old open problem.
- Every integer ≥ 2 is sum of two primes.

Consider G on $\langle x \rangle$:

1. For $n = 2$, check each pair of prime number $< n$.
2. If no pair sums to n, reject.
3. Increment n and loop to step 1.

What does it mean if G halts?
What does it mean if G does not halt?

Beyond Decidability

What if $HALT_{TM}$ were "decidable"?

Goldbach's Conjecture:

- 280-year-old open problem.
- Every integer ≥ 2 is sum of two primes.

Consider G on $\langle x \rangle$:

1. For $n = 2$, check each pair of prime number $< n$.
2. If no pair sums to n, reject.
3. Increment n and loop to step 1.

What does it mean if G halts? **Goldbach's conjecture is false!**

What does it mean if G does not halt? **Goldbach's conjecture is true!**

Beyond Decidability

What if $HALT_{TM}$ were "decidable"?

Goldbach's Conjecture:

- 280-year-old open problem.
- Every integer ≥ 2 is sum of two primes.

Consider G on $\langle x \rangle$:

1. For $n = 2$, check each pair of prime number $< n$.
2. If no pair sums to n, reject.
3. Increment n and loop to step 1.

What does it mean if G halts? **Goldbach's conjecture is false!**

What does it mean if G does not halt? **Goldbach's conjecture is true!**

Turns out you can do this for lots of open problems over natural numbers (twin prime conjecture,…)