Review
CSCI 338
Test 2 Logistics

1. During class on Wednesday 3/27.
2. You can bring your book and any notes you would like, but no electronic devices.
3. You may assume anything proven in class or on homeworks.
4. Four questions:
   1) Show a language is decidable (5 points).
   2) Show a language is not decidable (10 points).
   3) Show a language is not decidable (5 points).
   4) Show a language is not decidable (1-2 points).
Claim: \( \text{FINITE}_{TM} = \{ \langle M \rangle : M \text{ is a TM and accepts a finite number of strings} \} \) is undecidable.

Proof: Suppose \( \text{FINITE}_{TM} \) is decidable and let TM \( H \) be its decider.

Build a TM \( S \) that decides \( A_{TM} \):

\[
S = \text{on input } \langle N, \omega \rangle
\]

1. Construct TM \( M_2 \) on input \( \langle x \rangle \):
   1. Run \( N \) on \( \omega \) and accept if \( N \) does.
   2. Run \( H \) on \( \langle M_2 \rangle \).
   3. If \( H \) accepts, reject. If \( H \) rejects, accept.

If \( N \) accepts \( \omega \), \( L(M_2) = \Sigma^* \) (infinite) and \( S \) accepts. If \( N \) does not accept \( \omega \), \( L(M_2) = \emptyset \) (finite) and \( S \) rejects. Therefore, \( S \) is a decider for \( A_{TM} \), which is a contradiction, so \( \text{FINITE}_{TM} \) is undecidable.
Claim: \(EVEN_{TM} = \{\langle M \rangle: M \text{ is a TM and } L(M) \text{ contains all even length strings}\}\) is undecidable.

Proof: Suppose \(EVEN_{TM}\) is decidable and let TM \(H\) be its decider.

Build a TM \(S\) that decides \(A_{TM}\):

1. Construct TM \(M_2\) on input \(\langle x \rangle\):
   1. If \(x\) has odd length, reject.
   2. If \(x\) has even length, run \(N\) on \(\omega\).
   3. If \(N\) accepts \(\omega\), accept. If \(N\) rejects \(\omega\), reject.
2. Run \(H\) on \(\langle M_2 \rangle\).
Claim: $EVEN_{TM} = \{\langle M \rangle: M \text{ is a TM and } L(M) \text{ contains all even length strings}\}$ is undecidable.

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

Build a TM $S$ that decides $A_{TM}$:

$S = \text{on input } \langle N, \omega \rangle$

1. Construct TM $M_2$ on input $\langle x \rangle$:
   1. If $x$ has odd length, reject. \[\text{ accept}\]
   2. If $x$ has even length, run $N$ on $\omega$.
   3. If $N$ accepts $\omega$, accept. If $N$ rejects $\omega$, reject.

2. Run $H$ on $\langle M_2 \rangle$.

$L(M_2)$ contains all even length strings
\[\updownarrow\]
$N$ accepts $\omega$
Claim: \( \text{EVEN}_{TM} = \{ \langle M \rangle : M \text{ is a TM and } L(M) \text{ contains all even length strings} \} \) is undecidable.

Proof: Suppose \( \text{EVEN}_{TM} \) is decidable and let TM \( H \) be its decider.

Build a TM \( S \) that decides \( A_{TM} \):

\[
S = \text{on input } \langle N, \omega \rangle
\]

1. Construct TM \( M_2 \) on input \( \langle x \rangle \):
   1. If \( x \) has odd length, reject.
   2. If \( x \) has even length, run \( N \) on \( \omega \).
   3. If \( N \) accepts \( \omega \), accept. If \( N \) rejects \( \omega \), reject.

2. Run \( H \) on \( \langle M_2 \rangle \).

Let \( L(M_2) \) contains all even length strings if \( N \) accepts \( \omega \) accept
Claim: \( \text{EVEN}_{TM} = \{ \langle M \rangle : M \text{ is a TM and } L(M) \text{ contains all even length strings} \} \) is undecidable.

Proof: Suppose \( \text{EVEN}_{TM} \) is decidable and let TM \( H \) be its decider.

Build a TM \( S \) that decides \( A_{TM} \):

\[
S = \text{on input } \langle N, \omega \rangle \\
1. \text{Construct TM } M_2 \text{ on input } \langle x \rangle : \\
   1. \text{If } x \text{ has odd length, reject.} \\
   2. \text{If } x \text{ has even length, run } N \text{ on } \omega. \\
   3. \text{If } N \text{ accepts } \omega, \text{ accept. If } N \text{ rejects } \omega, \text{ reject.} \\
2. \text{Run } H \text{ on } \langle M_2 \rangle.
\]

\( L(M_2) \) contains all even length strings

\( N \) accepts \( \omega \)
Claim: $EVEN_{TM} = \{\langle M \rangle : M$ is a TM and $L(M)$ contains all even length strings$\}$ is undecidable.

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

Build a TM $S$ that decides $A_{TM}$:

$S = \text{on input } \langle N, \omega \rangle$

1. Construct TM $M_2$ on input $\langle x \rangle$:
   1. Run $N$ on $\omega$ 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 $\omega$, $L(M_2) = \Sigma^*$ (contains all even length strings) and $S$ accepts.
If $N$ does not accept $\omega$, $L(M_2) = \emptyset$ (does not contain all even length strings) and $S$ rejects. Thus, $S$ is a decider for $A_{TM}$, which is a contradiction, so $EVEN_{TM}$ is undecidable.
Claim: $HALT - \varepsilon_{TM} = \{\langle M \rangle: M \text{ is a TM and halts on empty input (i.e. } \varepsilon)\}$ is undecidable.

Proof: Suppose $HALT - \varepsilon_{TM}$ is decidable and let TM $H$ be its decider.

Build a TM $S$ that decides $A_{TM}$:

$S = \text{ on input } \langle N, \omega \rangle$

1. Construct TM $M_2$ on input $\langle x \rangle$:
   1. Run $N$ on $\omega$ 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 $\omega$, $M_2$ halts on $\varepsilon$ and $S$ accepts. If $N$ does not accept $\omega$, $M_2$ does not halt on $\varepsilon$ and $S$ rejects. Therefore, $S$ is a decider for $A_{TM}$, which is a contradiction, so $HALT - \varepsilon_{TM}$ is undecidable.
Claim: $HALT - \varepsilon_{TM} = \{\langle M \rangle : M \text{ is a TM and halts on empty input (i.e. } \varepsilon)\}$ is undecidable.

Proof: Suppose $HALT - \varepsilon_{TM}$ is decidable and let TM $H$ be its decider. Build a TM $S$ that decides $A_{TM}$:

$S = \text{ on input } \langle N, \omega \rangle$

1. Construct TM $M_2$ on input $\langle x \rangle$:
   1. Run $N$ on $\omega$ and accept if $N$ accepts.
   2. Run $H$ on $\langle M_2 \rangle$.

\[
\begin{array}{c}
M_2 \text{ halts on } \varepsilon \\
\downarrow \\
N \text{ accepts } \omega
\end{array}
\]

\[\text{reject}\]
Claim: $HALT - \varepsilon_{TM} = \{\langle M \rangle: M$ is a TM and halts on empty input (i.e. $\varepsilon)\}$ is undecidable.

Proof: Suppose $HALT - \varepsilon_{TM}$ is decidable and let TM $H$ be its decider.

Build a TM $S$ that decides $A_{TM}$:

$S = \text{on input } \langle N, \omega \rangle$

1. Construct TM $M_2$ on input $\langle x \rangle$:
   1. If $x \neq \varepsilon$, reject.
   2. If $x = \varepsilon$, run $N$ on $\omega$ and accept if $N$ accepts.
2. Run $H$ on $\langle M_2 \rangle$. 

$M_2$ halts on $\varepsilon$ $\iff$ $N$ accepts $\omega$
Claim: \( \text{HALT} - \varepsilon_{TM} = \{ \langle M \rangle : M \text{ is a TM and halts on empty input (i.e. } \varepsilon) \} \) is undecidable.

Proof: Suppose \( \text{HALT} - \varepsilon_{TM} \) is decidable and let TM \( H \) be its decider.

Build a TM \( S \) that decides \( \mathcal{A}_{TM} \):

\[
S = \text{on input } \langle N, \omega \rangle
\]

1. Construct TM \( M_2 \) on input \( \langle x \rangle \):
   1. If \( x \neq \varepsilon \), reject. **accept**
   2. If \( x = \varepsilon \), run \( N \) on \( \omega \) and **accept** if \( N \) accepts.

2. Run \( H \) on \( \langle M_2 \rangle \).
Claim: $HALT - \varepsilon_{TM} = \{\langle M \rangle: M$ is a TM and halts on empty input (i.e. $\varepsilon)\}$ is undecidable.

Proof: Suppose $HALT - \varepsilon_{TM}$ 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 $x \neq \varepsilon$, reject.
   2. If $x = \varepsilon$, run $N$ on $\omega$ and accept if $N$ accepts.

2. Run $H$ on $\langle M_2 \rangle$. 

$M_2$ halts on $\varepsilon$

$\Downarrow$

$N$ accepts $\omega$
Claim: $HALT - \varepsilon_{TM} = \{\langle M \rangle: M \text{ is a TM and halts on empty input (i.e. } \varepsilon)\}$ is undecidable.

Proof: Suppose $HALT - \varepsilon_{TM}$ is decidable and let TM $H$ be its decider.

Build a TM $S$ that decides $A_{TM}$:

$S = \text{on input } \langle N, \omega \rangle$

1. Construct TM $M_2$ on input $\langle x \rangle$:
   1. If $x \neq \varepsilon$, reject.
   2. If $x = \varepsilon$, run $N$ on $\omega$ and accept if $N$ accepts.

2. Run $H$ on $\langle M_2 \rangle$.  

$M_2$ halts on $\varepsilon$  
$\uparrow$  
$N$ accepts $\omega$  

Rejects
Claim: \( \text{HALT} - \varepsilon_{TM} = \{ \langle M \rangle: M \text{ is a TM and halts on empty input (i.e. } \varepsilon) \} \) is undecidable.

Proof: Suppose \( \text{HALT} - \varepsilon_{TM} \) is decidable and let TM \( H \) be its decider.

Build a TM \( S \) that decides \( A_{TM} \):

\[
S = \text{on input } \langle N, \omega \rangle
\]

1. Construct TM \( M_2 \) on input \( \langle x \rangle \):
   1. If \( x \neq \varepsilon \), reject.
   2. If \( x = \varepsilon \), run \( N \) on \( \omega \) and accept if \( N \) accepts.
2. Run \( H \) on \( \langle M_2 \rangle \).

\( M_2 \) halts on \( \varepsilon \)
\( \uparrow \)
\( N \) accepts \( \omega \)

\( N \) not accepting is not the same as \( N \) rejecting!

Doesn't work! \( N \) not accepting is not the same as \( N \) rejecting!
Claim: $HALT - \varepsilon_{TM} = \{\langle M \rangle: M$ is a TM and halts on empty input (i.e. $\varepsilon)\}$ is undecidable.

Proof: Suppose $HALT - \varepsilon_{TM}$ is decidable and let TM $H$ be its decider.

Build a TM $S$ that decides $HALT_{TM}$:

$S = \text{on input } \langle N, \omega \rangle$

1. Construct TM $M_2$ on input $\langle x \rangle$:
   1. Run $N$ on $\omega$ and accept if $N$ does.
2. Run $H$ on $\langle M_2 \rangle$. 

\[
\begin{array}{c}
M_2 \text{ halts on } \varepsilon \\
\Downarrow
\\
N \text{ halts on } \omega
\end{array}
\]
Claim: \( \text{HALT} - \varepsilon_{TM} = \{ \langle M \rangle : M \text{ is a TM and halts on empty input (i.e. } \varepsilon) \} \) is undecidable.

Proof: Suppose \( \text{HALT} - \varepsilon_{TM} \) is decidable and let \( \text{TM } H \) be its decider.

Build a TM \( S \) that decides \( \text{HALT}_{TM} \):

\[
S = \text{on input } \langle N, \omega \rangle \\
1. \text{Construct } \text{TM } M_2 \text{ on input } \langle x \rangle : \\
   1. \text{Run } N \text{ on } \omega \text{ and accept if } N \text{ does.} \\
   2. \text{Run } H \text{ on } \langle M_2 \rangle.
\]

\[
\begin{array}{c}
M_2 \text{ halts on } \varepsilon \\
\Downarrow \\
N \text{ halts on } \omega
\end{array}
\]

Doesn’t work! \( M_2 \) does not (always) halt when \( N \) halts on \( \omega \)!
Claim: $HALT - \varepsilon_{TM} = \{\langle M \rangle: M \text{ is a TM and halts on empty input (i.e. } \varepsilon)\}$ is undecidable.

Proof: Suppose $HALT - \varepsilon_{TM}$ is decidable and let TM $H$ be its decider.

Build a TM $S$ that decides $HALT_{TM}$:

$S = \text{on input } \langle N, \omega \rangle$

1. Construct TM $M_2$ on input $\langle x \rangle$:
   1. Run $N$ on $\omega$.
   2. Accept if $N$ accepts and reject if $N$ rejects.
2. Run $H$ on $\langle M_2 \rangle$.
3. If $H$ accepts, accept. If $H$ rejects, reject.
Claim: $HALT - \varepsilon_{TM} = \{\langle M \rangle: M \text{ is a TM and halts on empty input (i.e. } \varepsilon)\}$ is undecidable.

Proof: Suppose $HALT - \varepsilon_{TM}$ is decidable and let TM $H$ be its decider.

Build a TM $S$ that decides $HALT_{TM}$:

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

1. Construct TM $M_2$ on input $\langle x \rangle$:
   1. Run $N$ on $\omega$.
   2. reject.

2. Run $H$ on $\langle M_2 \rangle$.

3. If $H$ accepts, accept. If $H$ rejects, reject.
Prove that $ALMOST\_ALL_{DFA} = \{\langle A, \omega \rangle : A \text{ is a DFA and } L(A) = \Sigma^* \setminus \{\omega\}\} \text{ is decidable. Remember to show that your algorithm meets the requirements of a decider.}$

You are allowed to use any deciders we learned in class or on homework!
Prove that $ANY_{TM} = \{\langle M \rangle: M \text{ is a TM and } L(M) \neq \emptyset \}$ is undecidable.
Prove that $ALMOST\_ALL_{DFA} = \{⟨A, ω⟩: A$ is a DFA and $L(A) = \Sigma^* \setminus \{ω\}\}$ is decidable. Remember to show that your algorithm meets the requirements of a decider.

Proof: Construct the following Turing machine to decide $ALMOST\_ALL_{DFA}$:

$M = \text{on input } ⟨A, ω⟩$

1. Run $A$ on $ω$ and reject if $A$ accepts.
2. Construct DFA $B$ that only accepts $ω$.
3. Construct DFA $C$ so that $L(C) = L(A) \cup L(B)$.
4. Run the decider for $ALL_{DFA}$ on $C$.
5. If the decider accepts, accept. If it rejects, reject.

Every DFA is guaranteed to halt on all input so step 1 will halt. Constructing $B$ and $C$ will occur in finite time, and running a decider is guaranteed to halt. Thus, $M$ is guaranteed to halt and is a decider.
Prove that \( \text{ANY}_{TM} = \{ \langle M \rangle : M \text{ is a TM and } L(M) \neq \emptyset \} \) is undecidable.

Proof: Suppose \( \text{ANY}_{TM} \) is decidable and let TM \( H \) be its decider.

Build a TM \( S \) that decides \( A_{TM} \):

\[
S = \text{on input } \langle N, \omega \rangle
\]

1. Construct TM \( M_2 \) on input \( \langle x \rangle \):
   1. Run \( N \) on \( \omega \) 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 \( \omega \), \( L(M_2) = \Sigma^* \) and \( S \) accepts. If \( N \) does not accept \( \omega \), \( L(M_2) = \emptyset \) and \( S \) rejects. Thus, \( S \) is a decider for \( A_{TM} \), which is a contradiction, so \( \text{ANY}_{TM} \) is undecidable.
Computability Hierarchy

- Regular
- Context-Free
- Decidable
- Turing-recognizable
- Other
Claim: \( \overline{\text{HALT}}_{TM} = \{ \langle M, \omega \rangle : M \text{ is a TM and } M \text{ does not halt on } \omega \} \) is not Turing-recognizable.
Unrecognizable Language

Claim: \( \overline{HALT_{TM}} = \{ \langle M, \omega \rangle: M \text{ is a TM and } M \text{ does not halt on } \omega \} \) 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 \textbf{not} halt on some input).
Claim: $\overline{HALT_{TM}} = \{\langle M, \omega \rangle: M \text{ is a TM and } M \text{ does not halt on } \omega \}$ 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 $\omega$.
2. accept.

$HALT_{TM}$ recognizer!
Unrecognizable Language

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

Proof: Suppose $\overline{\text{HALT}_{TM}}$ 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 $\omega$.
2. accept.

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.
Computability Hierarchy

- Regular
- Context-Free
- Decidable
- Turing-recognizable

$HALT_{TM}$
Unrecognizable Language

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

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

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

$M =$ on input $\omega$

1. Run both $M_1$ and $M_2$ on $\omega$ in parallel (alternate instructions).
2. If $M_1$ accepts, accept. If $M_2$ accepts, reject.

Since $\omega \in A$ or $\overline{A}$, $M_1$ or $M_2$ must accept (halts on input). Thus, $M$ is a decider for $A$. 
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 $\geq 2$ is sum of two primes.
Beyond Decidability

What if \( \text{HALT}_{TM} \) were “decidable”?

Goldbach’s Conjecture:
- 280-year-old open problem.
- Every integer \( \geq 2 \) 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.
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 $\geq 2$ 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 $\geq 2$ 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 $\geq 2$ 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,...)