Pre-lecture brain teaser

In the following languages, three are decidable and three are undecidable. Which are which?

- \( A_{\text{CFG}} = \{ \langle G, w \rangle \mid G \text{ is a CFG that accepts string } w \} \).
- \( E_{\text{CFG}} = \{ \langle G \rangle \mid G \text{ is a CFG and } L(G) = \emptyset \} \).
- \( \text{ALL}_{\text{CFG}} = \{ \langle G \rangle \mid G \text{ is a CFG and } L(G) = \Sigma^* \} \).
- \( A_{\text{LBA}} = \{ \langle M, w \rangle \mid M \text{ is a LBA that generates string } w \} \).
- \( E_{\text{LBA}} = \{ \langle M \rangle \mid M \text{ is a LBA where } L(M) = \emptyset \} \).
- \( \text{ALL}_{\text{LBA}} = \{ \langle M \rangle \mid M \text{ is a LBA where } L(M) = \Sigma^* \} \).
Pre-lecture brain teaser

In the following languages, three are decidable and three are undecidable. Which are which?

- \( A_{CFG} = \{ \langle G, w \rangle \mid G \text{ is a } CFG \text{ that accepts string } w \} \).
- \( E_{CFG} = \{ \langle G \rangle \mid G \text{ is a } CFG \text{ and } L(G) = \emptyset \} \).
- \( \text{ALL}_{CFG} = \{ \langle G \rangle \mid G \text{ is a } CFG \text{ and } L(G) = \Sigma^* \} \).
- \( A_{LBA} = \{ \langle M, w \rangle \mid M \text{ is a } LBA \text{ that accepts string } w \} \).
- \( E_{LBA} = \{ \langle M \rangle \mid M \text{ is a } LBA \text{ where } L(M) = \emptyset \} \).
- \( \text{ALL}_{LBA} = \{ \langle M \rangle \mid M \text{ is a } LBA \text{ where } L(M) = \Sigma^* \} \).
$A_{CFG}$ decidable?

$V = \{S, S_0\}$

$T = \{\epsilon, 0, 1\}$

$P = \{S \rightarrow \epsilon,
S \rightarrow 0S0,
S \rightarrow 1S1\}$
$A_{\mathit{CFG}}$ decidable?

YES!
$A_{CFG}$ decidable?

YES!

- $V = \{S\}$
- $T = \{0, 1\}$
- $P = \{S \to \epsilon \mid 0S0 \mid 1S1\}$
  (abbrev. for $S \to \epsilon, S \to 0S0, S \to 1S1$)

$|\omega| = n$
A \text{CFG} decidable?

YES!

\textbf{Lemma}

A CFG in Chomsky normal form can derive a string \( w \) in at most \( 2n - 1 \) steps! \textit{(Shown in Sipser textbook)}

Knowing this, we can just simulate all the possible rule combinations for \( 2^n \) steps and see if any of the resulting strings matches \( w \).
Is $E_{CFG}$ decidable?

1. Mark all terminal symbols in $G$
2. Repeat until no new variables get marked:
   2.1 Mark any variable $A$ where $G$ has the rule $A \rightarrow U_1 U_2 ... U_k$ where $U_i$ is a marked terminal/variable
3. If start variable is not marked, accept. Otherwise reject.

- $V = \{S\}
- T = \{0, 1\}
- P = \{S \rightarrow \varepsilon | 0 S 0, S \rightarrow 1 S 1\} (abbrev. for $S \rightarrow \varepsilon, S \rightarrow 0 S 0, S \rightarrow 1 S 1$)
$E_{CFG}$ decidable?

YES!
YES!

In this case, we just need to know if we can get from the start variable to a string with only terminal symbols.

1. Mark all terminal symbols in $G$
2. Repeat until no new variables get marked:
   2.1 Mark any variable $A$ where $G$ has the rule $A \rightarrow U_1U_2 \ldots U_k$ where $U_i$ is a marked terminal/variable
3. If start variable is not marked, accept. Otherwise reject.

- $V = \{S\}$
- $T = \{0, 1\}$
- $P = \{S \rightarrow \epsilon \mid 0S0 \mid 1S1\}$ (abbrev. for $S \rightarrow \epsilon, S \rightarrow 0S0, S \rightarrow 1S1$)
$ALL_{CFG}$ decidable?
ALL\textsubscript{CFG} decidable?

Nope
ALL\textsubscript{CFG} decidable?

Nope

Proof requires computation histories which are outside the scope of this course.
A_{LBA} decidable?

Remember an LBA has a finite tapes. Therefore we know:

1. A tape of length \( n \) where each cell can contain \( g \) symbols, you have \( g^n \) possible configurations.

2. The tape head can be in one of \( n \) positions and has \( q \) states yielding a tape that can be in \( q^n \) configurations.

3. Therefore the machine can be in \( q^n g^n \) configurations.

Lemma

If an LBA does not accept or reject in \( q^n g^n \) then it is stuck in a loop forever.
$A_{LBA}$ decidable?

YES!
A_LBA decidable?

YES!

Remember a LBA has a finite tapes. Therefore we know:

1. A tape of length $n$ where each cell can contain $g$ symbols, you have $g^n$ possible configurations.
2. The tape head can be in one of $n$ positions and has $q$ states yielding a tape that can be in $qn$ configurations.
3. Therefore the machine can be in $qng^n$ configurations.
A_LBA\ decidable?

YES!

Remember a LBA has a finite tapes. Therefore we know:

1. A tape of length $n$ where each cell can contain $g$ symbols, you have $g^n$ possible configurations.
2. The tape head can be in one of $n$ positions and has $q$ states yielding a tape that can be in $qn$ configurations.
3. Therefore the machine can be in $qng^n$ configurations.

**Lemma**

*If an LBA does not accept or reject in $qng^n$ then it is stuck in a loop forever.*
Decider for $A_{LBA}$ will:

1. Simulate $\langle M \rangle$ on $w$ for $qng^n$ steps.
   1.1 if accepts, then accept
   1.2 if rejects, then reject

2. If neither accepts or rejects, means it’s in a loop in which case, reject.
$E_{LBA}$ decidable?
$E_{LBA}$ decidable?

Nope
ELBA decidable?

Nope

Proof requires computational history trick, a story for another time......
ALL_{LBA} decidable?
ALL\textsubscript{LBA} decidable?

Nope
NOPE

No standard proof for this, but let’s look at a pattern:
Nope

No standard proof for this, but let’s look at a pattern:

So we sort’ve figure that $ALL_{LBA}$ isn’t decidable because we know (assuming you believe me) $ALL_{CFG}$ wasn’t (though intuition is never sufficient evidence).
Decidability across grammar complexities

<table>
<thead>
<tr>
<th></th>
<th>DFA</th>
<th>CFG</th>
<th>PDA</th>
<th>LBA</th>
<th>TM</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>U</td>
</tr>
<tr>
<td>E</td>
<td>D</td>
<td>D</td>
<td>D</td>
<td>U</td>
<td>U</td>
</tr>
<tr>
<td>ALL</td>
<td>D</td>
<td>U</td>
<td>U</td>
<td>U</td>
<td>U</td>
</tr>
</tbody>
</table>

Eventually problems get too tough....
The above proofs were somewhat repetitious...

...they imply a more general result.

**Theorem (Rice’s Theorem.)**

Suppose that \( L \) is a language of Turing machines; that is, each word in \( L \) encodes a \( TM \). Furthermore, assume that the following two properties hold.

(a) Membership in \( L \) depends only on the Turing machine’s language, i.e. if \( L(M) = L(N) \) then \( \langle M \rangle \in L \iff \langle N \rangle \in L \).

(b) The set \( L \) is “non-trivial,” i.e. \( L \neq \emptyset \) and \( L \) does not contain all Turing machines.

Then \( L \) is a undecidable. (Note: In an exam, you can’t just say undecidable because of Rice’s theorem.)
Un-/decidability practice problems
Available Undecidable languages

- \( L_{\text{Accept}} = \left\{ \langle M, w \rangle \mid M \text{ is a } TM \text{ and accepts } w \right\} \).
- \( L_{\text{HALT}} = \left\{ \langle M \rangle \mid M \text{ is a } TM \text{ and halts on } \varepsilon \right\} \).
Is the language:

\[ L_{\text{accept}} = \{ \langle m, w \rangle \mid M \text{ is a TM and accepts} \} \]

\[ L_{\text{HaltOnInput}} = \{ \langle M, w \rangle \mid M \text{ is a TM and halts on } w \} . \]
Practice 1: Halt on Input

Is the language:

$L_{\text{HaltOnInput}} = \left\{ \langle M, w \rangle \mid M \text{ is a TM and halts on } w \right\}.$
Practice 2: L has fooling set

Is the language:

\[ L_{\text{HasFooling}} = \{ \langle M \rangle \mid M \text{ is a TM and } L(M) \text{ has a fooling set} \} \]

\[ L_{\text{HasFooling}} \Rightarrow L_{\text{Halting}} \]

Diagram:

- ORAC (Oracle)
- Make \( M \)
- \( \langle M \rangle \)
- \( \langle w \rangle \)
- Dec Halton Input

Decision process:

- If \( x \) is \( \in \) and \( \neg |M| \)
  - Accept
- Reject
- Accept
Is the language: 

\[ L_{\text{HasFooling}} = \{ \langle M \rangle \mid M \text{ is a TM and } L(M) \text{ has a fooling set} \} \]
NP-Complete practice problems
A centipede is an undirected graph formed by a path of length $k$ with two edges (legs) attached to each node on the path as shown in the below figure. Hence, the centipede graph has $3k$ vertices. The **CENTIPEDE** problem is the following: given an undirected graph $G = (V, E)$ and an integer $k$, does $G$ contain a centipede of $3k$ vertices as a subgraph? Prove that **CENTIPEDE** is NP-Complete.
What do we need to do to prove Centipede is NP-Complete?

Prove: CENTIPEDE ≤ NP
CENTIPEDE ≤ NP-hard
Prove Centipede is in \textbf{NP}: \\

\textit{Certificate:} 3 vertex arrays \\
- backbone $[1 \ldots k]$ \\
- leg$_{1} [1 \ldots k]$ \\
- leg$_{2} [1 \ldots k]$ \\

\textit{Certifier:}
- Check for legs for $i = 1 \ldots k$ \\
- Make sure all vertices are distinct \\
- $(\text{leg}[i], \text{back}[i]) \in G$ \\
- $(\text{leg}[i], \text{back}[i+1]) \in G$
Prove Centipede is in NP-hard:

Reduction
Take \( \langle G \rangle = (V, E) \)
Copy \( \langle G \rangle = \langle G' \rangle \)
Add 2n vertices to \( G \)
for \( \forall v \in V \)
add edge between \( (v', v') \) and \( (v', v_2') \)
Prove Centipede is in \textbf{NP-hard}:

\textbf{Hamiltonian Path}: Given a graph $G$ (either directed or undirected), is there a path that visits every vertex exactly once

$HC \leq_p Centipede$
A quasi-satisfying assignment for a 3CNF boolean formula $\Phi$ is an assignment of truth values to the variables such that at most one clause in $\Phi$ does not contain a true literal. Prove that it is NP-complete to determine whether a given 3CNF boolean formula has a quasi-satisfying assignment.
A quasi-satisfying assignment for a 3CNF boolean formula $\Phi$ is an assignment of truth values to the variables such that at most one clause in $\Phi$ does not contain a true literal. Prove that it is NP-complete to determine whether a given 3CNF boolean formula has a quasi-satisfying assignment.

Prove quasiSAT is in NP

Certificate: truth assignment

Certificate:
A quasi-satisfying assignment for a 3CNF boolean formula $\Phi$ is an assignment of truth values to the variables such that at most one clause in $\Phi$ does not contain a true literal. Prove that it is NP-complete to determine whether a given 3CNF boolean formula has a quasi-satisfying assignment.

Prove quasiSAT is NP-hard

$$\phi' = \phi \lor (x \lor x \lor x) \lor (\bar{x} \lor \bar{x} \lor \bar{x})$$
Prove quasiSAT is NP-hard
Prove quasiSAT is NP-hard

**3SAT:** Given a boolean formula in conjunctive normal form, with exactly three distinct literals per clause, does the formula have a satisfying assignment.
Good luck on the exam