Today’s Topics

- Question: $A_{\text{TM}}$ and a possibly infinite alphabet?
- Review: Recursion Theorem
  - Proving $A_{\text{TM}}$ is undecidable with Recursion Theorem
- $A_{\text{LBA}}$ is decidable
- $E_{\text{LBA}}$ is undecidable
- $\text{ALL}_{\text{PDA}}$ is undecidable
- $E_{\text{2WAY-PDA}}$ is undecidable
- Review: Closure Properties
- Review: Computability Class Relationships
- Review: Common Languages
- Review: Pumping Lemmas
- Bonus: $B = \{ 0^i1^j2^k | i,j,k \geq 0 \text{ and } i \geq k \text{ or } j \geq k \}$
  - $B$ is a CFL
  - $B$ is not a regular language
Question: \( A_{TM} \) and possibly infinite alphabet

- TM alphabets are always finite
- So then how does \( A_{TM} \) recognize \(<M,w>\) when languages of M may differ from \( A_{TM} \)?

- Trick: Encode alphabet of M into some fixed alphabet, say \{0,1\}
- Express \( k^{th} \) symbol of M’s alphabet as \( 0^k1 \) for example
Review: Recursion Theorem

Goal is to show that a TM can retrieve its own description

- Define function $q: \Sigma^* \rightarrow \Sigma^*$ such that:
  - $q(w) = \langle P_w \rangle$
  - $q$ on $w$ returns description of TM that when run, prints $w$ onto tape and halts.
  - Straightforward to see that such function can exist
Review: Recursion Theorem (cont.)

TM SELF has two parts, A and B

1. Run A = \langle P_B \rangle
2. Run B = “1. Compute \( q(\text{tape contents}) \) to get \( \langle P_{\text{tape contents}} \rangle = \langle P_B \rangle = A \)
   2. Prepend A to tape (currently has B) to get AB
   3. Halt with AB = \langle SELF \rangle on tape”
Proving $A_{TM}$ is Undecidable with Recursion Theorem

Proof by Contradiction: Assume some TM H decides $A_{TM}$

Consider TM R: (diagonalization argument)

R = “On input $w$

1. Get own description $<R>$
2. Use H on $<R, w>$ to determine whether R accepts $w$
3. Do opposite of what H returns”

If H were to exist, then we could create such a TM R. But such TM R may never exist!
ALBA is Decidable

LBA has bounded tape, so bounded number of state configurations

\[ \text{\# state configurations} = |Q| \cdot w \cdot |\Gamma|^w \]

S = "on input <L, w>
1. Simulate L on w for |Q|\cdot w \cdot |\Gamma|^w iterations
2. Accept if simulation accepts
   Reject if simulation rejects or is not yet halted"
**E\textsubscript{LBA} is Undecidable**

Goal: To create a LBA that only accepts a valid comp. history of M on w

Reduce $A\textsubscript{TM}$ to $E\textsubscript{LBA}$ problem ($A\textsubscript{TM} \leq_m E\textsubscript{LBA}$)

Assume TM T decides $E\textsubscript{LBA}$. Use T to create TM S which decides $A\textsubscript{TM}$

$S =$ "on input $<M,w>$
1. Create a LBA L which accepts only on comp. history M on w
   - Check if legal starting config for input w
   - Check if legal accepting config at the end
   - Check that all transitions $C_i$ to $C_{i+1}$ are valid
2. Use TM T on L
3. If T accepts L, reject. Accept otherwise."
**ALL\textsubscript{PDA} is Undecidable**

Goal: Create a PDA which accepts all non-valid comp. histories and all valid comp. histories EXCEPT M on w

Similar reduction and proof as for $E_{\text{LBA}}$. Use computation history method. Assume TM T decides ALL\textsubscript{PDA}.

S = "on input <M,w>

1. Create a P PDA which does not accept valid comp history M on w
   Non-det. branch to the following conditions:
   - Accept if starting config is NOT valid
   - Accept if ending config is NOT valid
   - Go to some point on tape. Push Ci onto the stack and compare against Ci+1. If NOT valid transition accept.

2. Use TM T on P. If T accept, then reject. Otherwise, accept."
**ALL_{PDA} is Undecidable**

Similar reduction and proof as for $E_{LBA}$. Use computation history method.

\[18.404 \rightarrow 404.81\]
## Review: Closure Properties

<table>
<thead>
<tr>
<th></th>
<th>Regular Languages</th>
<th>CFLs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Closed</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td></td>
<td>Union</td>
<td>Union</td>
</tr>
<tr>
<td></td>
<td>Concatenation</td>
<td>Concatenation</td>
</tr>
<tr>
<td></td>
<td>Kleene Star</td>
<td>Kleene Star</td>
</tr>
<tr>
<td></td>
<td>Intersection</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Negation</td>
<td></td>
</tr>
<tr>
<td>Not-Closed</td>
<td></td>
<td>Not-Closed</td>
</tr>
<tr>
<td></td>
<td>Intersection</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Negation</td>
<td></td>
</tr>
</tbody>
</table>

**Note:**
\[
CFL \cap \text{Reg. lang.} = CFL
\]
Review: Computability Class Relationships
## Review: Common Languages

<table>
<thead>
<tr>
<th>T-Decidable</th>
<th>T-Recognizable (undecidable)</th>
<th>T-coRecognizable (undecidable)</th>
<th>T-Unrecognizable (neither T-recog nor T-coRecog)</th>
</tr>
</thead>
<tbody>
<tr>
<td>● $A_{DFA}$</td>
<td>● $A_{TM}$</td>
<td>● $E_{TM}$</td>
<td>● $EQ_{TM}$</td>
</tr>
<tr>
<td>● $A_{NFA}$</td>
<td>● $\text{negate}(\text{EQ}_{CFG})$</td>
<td>● $\text{negate}(A_{TM})$</td>
<td>● $\text{negate}(EQ_{TM})$</td>
</tr>
<tr>
<td>● $E_{DFA}$</td>
<td>● $\text{HALT}_{TM}$</td>
<td>● $E_{CFG}$</td>
<td></td>
</tr>
<tr>
<td>● $EQ_{DFA}$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>● $A_{CFG}$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>● $E_{CFG}$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Review: Pumping Lemmas (Regular Language)

For every regular language, there exists a pumping number $p \geq 1$ such that every string of length at least $p$ can be written as $w=xyz$ and satisfies:

- $|y| \geq 1$
- $|xy| \leq p$
- $(\forall n \geq 0) (xy^nz \in L)$
Review: Pumping Lemmas (CFLs)

For every CFL, there exists a pumping number $p \geq 1$ such that every string of length at least $p$ can be written as $s=uvxyz$ and satisfies:

- $(\forall n \geq 0) (uv^nxy^nz \in L)$
- $|vy| \geq 1$
- $|vxy| \leq p$

Note: Need to make sure that all slide-windows in the string that CANNOT be pumped!