Today’s Topics

- Re-explaining Non-CFL Language
  - \( \{ a^i b^j c^k \mid i > j > k \} \)
- Review: \( A_{TM} \) is Undecidable
- Proving Decidable
  - \( \{ <R, S> \mid R \text{ and } S \text{ are regular expressions and } L(R) \subseteq L(S) \} \)
  - \( \{ <R> \mid R \text{ is a regular expression and } L(R) \text{ is prefix-free} \} \)
  - \( \{ <D> \mid D \text{ is a DFA that accepts some palindrome} \} \)
  - \( \{ <D> \mid D \text{ is a DFA that accepts } w^R \text{ whenever it accepts } w \} \)
- Proving T-Recognizable
  - \( \{ <M> \mid M \text{ is a TM whose language is non-empty} \} \)
- Recap
- Bonus (time-permitting)
  - \( \{ <S> \mid S \text{ is a TM whose language is empty} \} \) is T-unrecognizable
  - 2TAPe = \( \{ \{M, w\} \mid M \text{ is a 2-tape TM that writes a non-blank symbol on 2nd tape on } w \} \)
    - Prove it is T-recognizable, but not T-decidable
Example: Proving Non-CFL Languages

Prove that \( \{ a^ib^jc^k \mid i > j > k \} \) is not a CFL

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

\[ s = a^{p+2}b^{p+1}c^p \]
Review: $A_{TM}$ is Undecidable

Proof by Contradiction

$A_{TM} = \{ <M, w> | M \text{ is a TM that accepts input } w \}$

- Assume TM $H$ decides $A_{TM}$
  - $H$ accepts $<M, w>$ iff $M$ accepts $w$
  - $H$ rejects $<M, w>$ iff $M$ rejects or loops on $w$
- Will prove that $H$ may never exist due to a contradiction
Recall assuming that \( H \) decides \( \text{ATM} = \{ <M, w> | M \text{ accepts } w \} \)

Use \( H \) to construct a TM \( D \)

\[
D = \text{“On input } <M> \\
1. \text{ Simulate } H \text{ on input } <M, <M>> \text{ ie: } (<M, w> \text{ where } w = <M>) \\
2. \text{ Reject if } H \text{ accepts. Accept if } H \text{ rejects.”}
\]

\( D \) accepts \( <M> \) iff \( M \) does not accept \( <M> \)

Contradiction: \( D \) accepts \( <D> \) iff \( <D> \) does not accept \( <D> \)
### Review: $A_{TM}$ is Undecidable

<table>
<thead>
<tr>
<th>All TM descriptions:</th>
<th>$\langle M_1 \rangle$</th>
<th>$\langle M_2 \rangle$</th>
<th>$\langle M_3 \rangle$</th>
<th>$\langle M_4 \rangle$</th>
<th>...</th>
<th>$\langle D \rangle$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_1$</td>
<td>acc</td>
<td>rej</td>
<td>acc</td>
<td>acc</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>$M_2$</td>
<td>rej</td>
<td>rej</td>
<td>rej</td>
<td>rej</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$M_3$</td>
<td>acc</td>
<td>acc</td>
<td>acc</td>
<td>acc</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>$M_4$</td>
<td>rej</td>
<td>rej</td>
<td>acc</td>
<td>acc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td>$\vdots$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$D$</td>
<td>rej</td>
<td>acc</td>
<td>rej</td>
<td>rej</td>
<td></td>
<td>??????</td>
</tr>
</tbody>
</table>
Proving Decidable

\{ <R, S> | R and S are regular expressions and L(R) \subseteq L(S) \}

D = "on input <R, S>
1. Convert R and S into DFA R' and S' respectively
2. Construct DFA T = R' intersect S'
3. Run EQ_DFA on <R', T> and return accordingly"

Proving Decidable

\{ <R> | R is a regular expression and L(R) is prefix-free \}

NOT prefix free = \{ "Star Wars", "b", "ac", "Star Wars is cool!" \}

D = "on input <R>
  1. Construct DFA R' from reg expr R
  2. Prune all out-going edges from accept states of R' to create DFA P
     (this filters all suffixed strings out of L(R'))
  3. Run EQ_DFA on R and P. Accept if EQ_DFA accepts. Reject otherwise."
Proving Decidable

\{ <D> | D is a DFA that accepts some palindrome \} palindrome = \{w+rev(w)\}

D has a palindrome -> intersection of L(D) and palindrome is non empty set
D has no palindrome -> intersection of L(D) and palindrome is empty set

Use construction from HW 2, problem 0.2: regular language \cap CFL = CFL

F = "on input <D>
1. Use construction from HW 2 to create PDA P that computes:
   CFL = reg lang intersect palindrome
2. Run E_PDA on P. Accept if E_PDA rejects. Reject otherwise."
Proving Decidable

\{ \langle D \rangle \mid \text{D is a DFA that accepts } w^R \text{ whenever it accepts } w \}
Proving T-Recognizable

\{ <M> \mid M \text{ is a TM whose language is non-empty} \}

R = "on input <M>
1. Simulate M on all inputs of \( \Sigma^* \) one by one
2. If M accepts any of the inputs, then accept"

If M really has empty language then will iterate forever over \( \Sigma^* \) and never terminate. But this is OK for T-Recog languages.
Recap