Decidable Problems about Context-Free Languages (CFLs)

Wed October 28, 2020
HW 5/6 questions?
HW6 out

• Covers material from Chapter 4

• “Show that <LANG> is decidable” ...
Last time: Decidable DFA Languages

- \( A_{\text{DFA}} = \{ \langle B, w \rangle | B \text{ is a DFA that accepts input string } w \} \)
- \( A_{\text{NFA}} = \{ \langle B, w \rangle | B \text{ is an NFA that accepts input string } w \} \)
- \( A_{\text{REX}} = \{ \langle R, w \rangle | R \text{ is a regular expression that generates string } w \} \)
- \( E_{\text{DFA}} = \{ \langle A \rangle | A \text{ is a DFA and } L(A) = \emptyset \} \)
- \( EQ_{\text{DFA}} = \{ \langle A, B \rangle | A \text{ and } B \text{ are DFAs and } L(A) = L(B) \} \)

Remember:

TMs = programs
This is your library
Thm: $A_{CFG}$ is a decidable language

$A_{CFG} = \{ \langle G, w \rangle \mid G \text{ is a CFG that generates string } w \}$

- Related to parsing!
  - E.g., is program $w$ a valid Python (with grammar $G$) program?
- Create a decider TM:
  - Try all possible derivations of $G$?
  - But this might never halt, e.g., if there is a rule like: $S \rightarrow 0S$ or $S \rightarrow S$
  - This TM would be a recognizer but not a decider
- **Idea:** Bound the number of derivation steps?
  - Stop after some length?
DEFINITION 2.8

A context-free grammar is in Chomsky normal form if every rule is of the form

\[ A \rightarrow BC \]
\[ A \rightarrow a \]

where \( a \) is any terminal and \( A, B, \) and \( C \) are any variables—except that \( B \) and \( C \) may not be the start variable. In addition, we permit the rule \( S \rightarrow \varepsilon \), where \( S \) is the start variable.
Chomsky Normal Form: Number of Steps

- To generate a string of length $n$:
  - $n$ steps: to generate all the terminals
  - $n - 1$ steps: to generate enough variables
  - **Total:** $2n - 1$ steps to generate length $n$ string
Thm: Every CFG has a Chomsky Normal Form

1. Add new start variable $S_0$ that does not appear on any RHS
   - i.e., add rule $S_0 \rightarrow S$, where $S$ is old start var

\[
\begin{align*}
S & \rightarrow ASA \mid aB \\
A & \rightarrow B \mid S \\
B & \rightarrow b \mid \varepsilon \\
S_0 & \rightarrow S \\
S & \rightarrow ASA \mid aB \\
A & \rightarrow B \mid S \\
B & \rightarrow b \mid \varepsilon
\end{align*}
\]
Thm: Every CFG has a Chomsky Normal Form

1. Add new start variable $S_0$ that does not appear on any RHS
   - I.e., add rule $S_0 \rightarrow S$, where $S$ is old start var

2. Remove all “empty” rules of the form $A \rightarrow \epsilon$
   - $A$ must not be the start variable
   - Then for every rule with $A$ on RHS, add new rule with $A$ deleted
     - E.g., if $R \rightarrow uAv$ is a rule, add $R \rightarrow uv$
     - Must cover all combinations if $A$ appears more than once in a RHS
       - E.g., if $R \rightarrow uAvAw$ is a rule, add 3 rules: $R \rightarrow uvAw$, $R \rightarrow uAvw$, $R \rightarrow uvw$
Thm: Every CFG has a Chomsky Normal Form

1. Add new start variable $S_0$ that does not appear on any RHS
   • I.e., add rule $S_0 \rightarrow S$, where $S$ is old start var

2. Remove all “empty” rules of the form $A \rightarrow \varepsilon$
   • $A$ must not be the start variable
   • Then for every rule with $A$ on RHS, add new rule with $A$ deleted
     • E.g., If $R \rightarrow uAv$ is a rule, add $R \rightarrow uv$
     • Must cover all combinations if $A$ appears more than once in a RHS
       • E.g., if $R \rightarrow uAvAw$ is a rule, add 3 rules: $R \rightarrow uvAw, R \rightarrow uAvw, R \rightarrow uvw$

3. Remove all “unit” rules of the form $A \rightarrow B$
   • Then, for every rule $B \rightarrow u$, add rule $A \rightarrow u$
Thm: Every CFG has a Chomsky Normal Form

1. Add new start variable $S_0$ that does not appear on any RHS
   • I.e., add rule $S_0 \rightarrow S$, where $S$ is old start var

2. Remove all “empty” rules of the form $A \rightarrow \varepsilon$
   • $A$ must not be the start variable
   • Then for every rule with $A$ on RHS, add new rule $S \rightarrow ASA$

3. Remove all unit rules of the form $A \rightarrow a$
   • Then, for every rule $B \rightarrow u$, add rule $A \rightarrow u$

4. Split up rules with RHS longer than length 2
   • E.g., $A \rightarrow wxyz$ becomes $A \rightarrow wB$, $B \rightarrow xc$, $C \rightarrow yz$

5. Replace all terminals on RHS with new rule
   • E.g., for above, add $W \rightarrow w$, $X \rightarrow x$, $Y \rightarrow y$, $Z \rightarrow z$
Thm: $A_{CFG}$ is a decidable language

$A_{CFG} = \{ \langle G, w \rangle \mid G \text{ is a CFG that generates string } w \}$

- Create decider:

$S = \text{“On input } \langle G, w \rangle \text{, where } G \text{ is a CFG and } w \text{ is a string:} \smallskip$
  1. Convert $G$ to an equivalent grammar in Chomsky normal form.
  2. List all derivations with $2n - 1$ steps, where $n$ is the length of $w$; except if $n = 0$, then instead list all derivations with one step.
  3. If any of these derivations generate $w$, accept; if not, reject.”
**Thm:** $E_{CFG}$ is a decidable language.

$$E_{CFG} = \{ \langle G \rangle \mid G \text{ is a CFG and } L(G) = \emptyset \}$$

- Recall:

$$E_{DFA} = \{ \langle A \rangle \mid A \text{ is a DFA and } L(A) = \emptyset \}$$

$$T = \text{“On input } \langle A \rangle, \text{ where } A \text{ is a DFA:}$$

1. Mark the start state of $A$.
2. Repeat until no new states get marked:
3. Mark any state that has a transition coming into it from any state that is already marked.
4. If no accept state is marked, accept; otherwise, reject.”

- “Reachability” (of accept state from start state)
Thm: $E_{CFG}$ is a decidable language.

$$E_{CFG} = \{ \langle G \rangle \mid G \text{ is a CFG and } L(G) = \emptyset \}$$

- Create decider that calculates reachability for grammar $G$
  - Except start from terminals, to avoid looping

$R =$ “On input $\langle G \rangle$, where $G$ is a CFG:

1. Mark all terminal symbols in $G$.
2. Repeat until no new variables get marked:
3. Mark any variable $A$ where $G$ has a rule $A \rightarrow U_1 U_2 \cdots U_k$ and each symbol $U_1, \ldots, U_k$ has already been marked.
4. If the start variable is not marked, accept; otherwise, reject.”
Thm: $\text{EQ}_{\text{CFG}}$ is a decidable language?

$$\text{EQ}_{\text{CFG}} = \{ \langle G, H \rangle \mid G \text{ and } H \text{ are CFGs and } L(G) = L(H) \}$$

- Recall: $\text{EQ}_{\text{DFA}} = \{ \langle A, B \rangle \mid A \text{ and } B \text{ are DFAs and } L(A) = L(B) \}$

- Use Symmetric Difference

  \[ L(C) = \emptyset \text{ iff } L(A) = L(B) \]

  - $C =$ complement, Union, intersection of machines A and B

- Can’t do this for CFLs!
  - Intersection and complement are not closed for CFLs!!!
Intersection of CFLs is **Not** Closed!

• If closed, then intersection of these CFLs should be a CFL:

\[ A = \{ a^m b^n c^n | m, n \geq 0 \} \]

\[ B = \{ a^n b^n c^m | m, n \geq 0 \} \]

• But \( A \cap B = \{ a^n b^n c^n | n \geq 0 \} \)

• Not a CFL!
Complement of a CFL is not Closed!

• If CFLs closed under complement:

\[
\text{if } G_1 \text{ and } G_2 \text{ context-free} \\
\overline{L(G_1)} \text{ and } \overline{L(G_2)} \text{ context-free} \\
\overline{L(G_1)} \cup \overline{L(G_1)} \text{ context-free} \\
\overline{L(G_1)} \cup \overline{L(G_1)} \text{ context-free} \\
\overline{L(G_1) \cap L(G_2)} \text{ context-free}
\]

DeMorgan’s Law!
Thm: $EQ_{CFG}$ is a decidable language?

$EQ_{CFG} = \{(G, H) | G \text{ and } H \text{ are CFGs and } L(G) = L(H)\}$

- No!
- Not recognizable either!
- You cannot decide whether two grammars are equal!
- (Can’t prove until Chapter 5)
Decidability of CFGs Recap

- $A_{CFG} = \{ \langle G, w \rangle \mid G \text{ is a CFG that generates string } w \}\$
  - Convert grammar to Chomsky Normal Form
  - Then check all possible derivations of length $2|w| - 1$ steps

- $E_{CFG} = \{ \langle G \rangle \mid G \text{ is a CFG and } L(G) = \emptyset \}$
  - Compute “reachability” of start variable from terminals

- $EQ_{CFG} = \{ \langle G, H \rangle \mid G \text{ and } H \text{ are CFGs and } L(G) = L(H) \}$
  - We couldn’t prove that this is decidable!
  - (Can’t use this when creating a decider)
Next time: Thms: $A_{TM}$ is Turing-recognizable

$A_{TM}$ is undecidable

$A_{TM} = \{ \langle M, w \rangle \mid M$ is a TM and $M$ accepts $w \}$

$U =$ “On input $\langle M, w \rangle$, where $M$ is a TM and $w$ is a string:

1. Simulate $M$ on input $w$.
2. If $M$ ever enters its accept state, accept; if $M$ ever enters its reject state, reject.”
Check-in Quiz 10/28
On gradescope
End of Class Survey 10/28
See course website