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

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

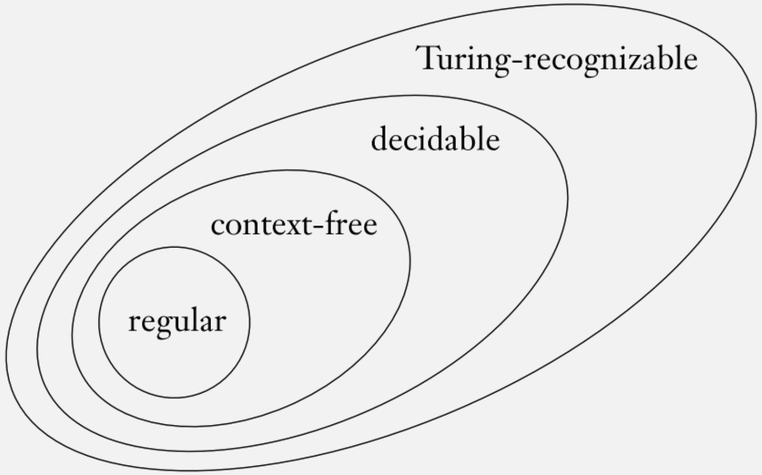
Remember:
TMs = programs
This is your library

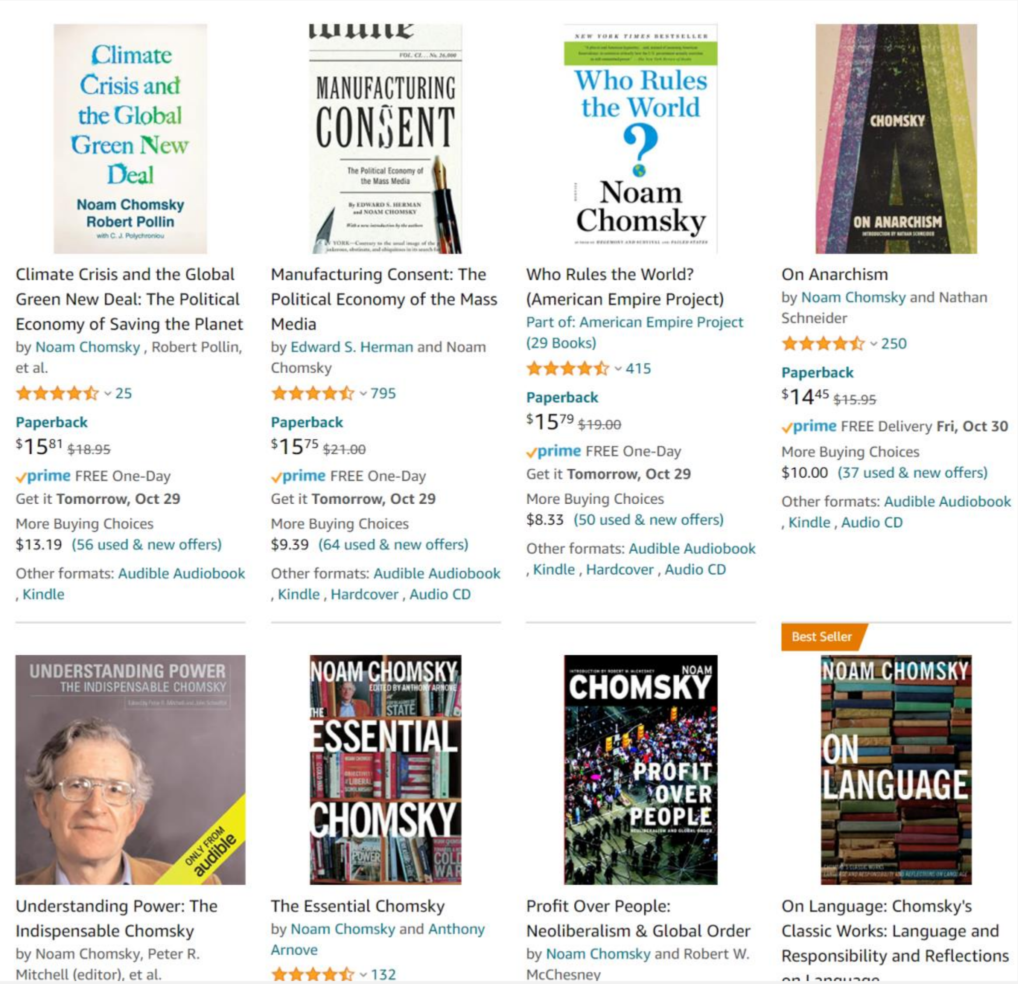
Thm: A_{CFG} is a decidable language

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

- 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 -> 0S or S -> S
  - This TM would be a recognizer but not a decider
- Idea: Bound the number of derivation steps?
  - Stop after some length?

Chomsky Normal Form





Chomsky Normal Form

DEFINITION 2.8

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

$$A \to BC$$
$$A \to a$$

2 kinds of rules

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 $S \to \varepsilon$, 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

Chomsky normal form

$$A \rightarrow BC$$
$$A \rightarrow a$$

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

Chomsky normal form

$$A \rightarrow BC$$
$$A \rightarrow a$$

$$\begin{aligned} S &\rightarrow ASA \mid \mathtt{a}B \\ A &\rightarrow B \mid S \\ B &\rightarrow \mathtt{b} \mid \varepsilon \end{aligned}$$

$$\begin{aligned} \boldsymbol{S_0} &\rightarrow \boldsymbol{S} \\ S &\rightarrow ASA \mid \mathtt{a}B \\ A &\rightarrow B \mid S \\ B &\rightarrow \mathtt{b} \mid \varepsilon \end{aligned}$$

Thm: Every CFG has a Chomsky Normal Form

Chomsky normal form

$A \to BC$

$A \to a$

1. Add new start variable S_0 that does not appear on any RHS
   - I.e., add rule $S_0 \to S$, where S is old start var
2. Remove all "empty" rules of the form $A \to \varepsilon$
   - A must not be the start variable
   - Then for every rule with A on RHS, add new rule with A deleted
     - E.g., If $R \to uAv$ is a rule, add $R \to uv$
   - Must cover all combinations if A appears more than once in a RHS
     - E.g., if $R \to uAvAw$ is a rule, add 3 rules: $R \to uvAw$, $R \to uAvw$, R -> uvw

$$
\begin{aligned}
S_0 &\to S \\
S &\to ASA \mid \mathtt{a}B \mid \mathbf{a} \\
A &\to B \mid S \mid \boldsymbol{\varepsilon} \\
B &\to \mathtt{b} \mid \varepsilon
\end{aligned}
\quad \Rightarrow \quad
\begin{aligned}
S_0 &\to S \\
S &\to ASA \mid \mathtt{a}B \mid \mathtt{a} \mid \boldsymbol{SA} \mid \boldsymbol{AS} \mid \boldsymbol{S} \\
A &\to B \mid S \mid \varepsilon \\
B &\to \mathtt{b}
\end{aligned}
$$

Thm: Every CFG has a Chomsky Normal Form

Chomsky normal form

$A \to BC$

$A \to a$

1. Add new start variable S_0 that does not appear on any RHS
   - I.e., add rule $S_0 \to S$, where S is old start var
2. Remove all "empty" rules of the form $A \to \varepsilon$
   - A must not be the start variable
   - Then for every rule with A on RHS, add new rule with A deleted
     - E.g., If $R \to uAv$ is a rule, add $R \to uv$
   - Must cover all combinations if A appears more than once in a RHS
     - E.g., if $R \to uAvAw$ is a rule, add 3 rules: $R \to uvAw$, $R \to uAvw$, R -> uvw
3. Remove all "unit" rules of the form $A \to B$
   - Then, for every rule $B \to u$, add rule $A \to u$

$$\begin{aligned} S_0 &\to S \\ S &\to ASA \mid \mathtt{a}B \mid \mathtt{a} \mid SA \mid AS \mid S \\ A &\to B \mid S \\ B &\to \mathtt{b} \end{aligned}$$

$$\begin{aligned} S_0 &\to S \mid \mathbf{ASA} \mid \mathbf{aB} \mid \mathbf{a} \mid \mathbf{SA} \mid \mathbf{AS} \\ S &\to ASA \mid \mathtt{a}B \mid \mathtt{a} \mid SA \mid AS \\ A &\to B \mid S \\ B &\to \mathtt{b} \end{aligned}$$

$$\begin{aligned} S_0 &\to ASA \mid \mathtt{a}B \mid \mathtt{a} \mid SA \mid AS \\ S &\to ASA \mid \mathtt{a}B \mid \mathtt{a} \mid SA \mid AS \\ A &\to S \mid \mathtt{b} \mid \mathbf{ASA} \mid \mathbf{aB} \mid \mathbf{a} \mid \mathbf{SA} \mid \mathbf{AS} \\ B &\to \mathtt{b} \end{aligned}$$

Thm: Every CFG has a Chomsky Normal Form

Chomsky normal form

$A \to BC$

$A \to a$

1. Add new start variable S_0 that does not appear on any RHS
   - I.e., add rule $S_0 \to S$, where S is old start var
2. Remove all "empty" rules of the form $A \to \varepsilon$
   - A must not be the start variable
   - Then for every rule with A on RHS, add new r… uv … appe… mo… rules: $A \to u$…
3. Remove all "unit" rules of the form $A \to$ …
   - Then, for every rule $B \to u$, add rule $A \to u$

$$
\begin{aligned}
S_0 &\to ASA \mid \mathtt{a}B \mid \mathtt{a} \mid SA \mid AS \\
S &\to ASA \mid \mathtt{a}B \mid \mathtt{a} \mid SA \mid AS \\
A &\to S \mid \mathtt{b} \mid \boldsymbol{ASA} \mid \boldsymbol{\mathtt{a}B} \mid \boldsymbol{\mathtt{a}} \mid \boldsymbol{SA} \mid \boldsymbol{AS} \\
B &\to \mathtt{b}
\end{aligned}
$$

$$
\begin{aligned}
S_0 &\to AA_1 \mid UB \mid \mathtt{a} \mid SA \mid AS \\
S &\to AA_1 \mid UB \mid \mathtt{a} \mid SA \mid AS \\
A &\to \mathtt{b} \mid AA_1 \mid UB \mid \mathtt{a} \mid SA \mid AS \\
A_1 &\to SA \\
U &\to \mathtt{a} \\
B &\to \mathtt{b}
\end{aligned}
$$

4. Split up rules with RHS longer than length 2
   - E.g., $A \to wxyz$ becomes $A \to wB$, $B \to xC$, $C \to yz$
5. Replace all terminals on RHS with new rule
   - E.g., for above, add $W \to w$, $X \to x$, $Y \to y$, $Z \to z$

Thm: A_{CFG} is a decidable language

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

- Create decider:

$S =$ "On input $\langle G, w\rangle$, where G is a CFG and w is a string:

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

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

- Recall:

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

T = "On input $\langle A\rangle$, where A 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

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

- 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:

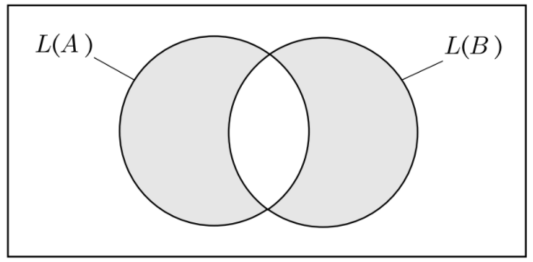
3. Mark any variable A where G has a rule $A \rightarrow U_1 U_2 \cdots U_k$ and each symbol $U_1, \ldots, U_k$ has already been marked.

4. If the start variable is not marked, *accept*; otherwise, *reject*."

Thm: EQ_{CFG} is a decidable language?

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

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

$$A = \{\mathtt{a}^m\mathtt{b}^n\mathtt{c}^n \mid m, n \geq 0\}$$

$$B = \{\mathtt{a}^n\mathtt{b}^n\mathtt{c}^m \mid m, n \geq 0\}$$

- But A ∩ B = $\{\mathtt{a}^n\mathtt{b}^n\mathtt{c}^n \mid n \geq 0\}$

- Not a CFL!

Complement of a CFL is not Closed!

- If CFLs closed under complement:

$$\text{if } G_1 \text{ and } G_2 \text{ context-free}$$

$$\overline{L(G_1)} \text{ and } \overline{L(G_2)} \text{ context-free}$$

$$\overline{L(G_1)} \cup \overline{L(G_1)} \text{ context-free}$$

$$\overline{\overline{L(G_1)} \cup \overline{L(G_1)}} \text{ context-free}$$

$$L(G_1) \cap L(G_2) \text{ context-free}$$

DeMorgan's Law!

Thm: $EQ_{\textsf{CFG}}$ is a decidable language?

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

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

Decidability of CFGs Recap

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

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

- $EQ_{\text{CFG}} = \{\langle G, H\rangle \mid G \text{ and } H \text{ are CFGs and } L(G) = L(H)\}$
  - We couldn't prove that this is decidable!
  - (Cant use this when creating a decider)

Next time: Thms: A_{TM} is Turing-recognizable

???

A_{TM} is undecidable

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

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