CSE 105
THEORY OF COMPUTATION

Spring 2018

http://cseweb.ucsd.edu/classes/sp18/cse105-ab/

Today's learning goals

Sipser Section 2.1

- Design a CFG generating a given language
- Prove the closure of class of CFLs under certain operations
- Identify when a CFG is ambiguous

Context-free grammar

Sipser Def 2.2, page 102

(V, Σ, R, S)

Variables: finite set of (usually upper case) variables **V**

Terminals: finite set of alphabet symbols **Σ**

Rules/Productions: finite set of allowed transformations **R**

$$A \to u \qquad A \in V, u \in (V \cup \Sigma)^*$$

Start variable: origination of each derivation **S**

The **language generated by a CFG** (V, Σ, R, S) is { w in Σ^* | starting with the Start variable and applying sequence of rules, can derive w on RHS}

Context-free languages

- L (00*)
- L((0U1)*)
- { abba }
- { $a^n b^n$ | n ≥ 0 }

Ex: { $0^i 1^j$ | j ≥ i ≥ 0 } recognizable by a PDA

PDAs and CFGs are equally expressive

Theorem 2.20: A language is context-free if and only if some nondeterministic PDA recognizes it.

Consequences

- Quick proof that every regular language is context free
- To prove closure of class of CFLs under a given operation, can choose two modes of proof (via CFGs or PDAs) depending on which is easier

Closure under union

If $G_1 = (V_1, \Sigma, R_1, S_1)$ and $G_2 = (V_2, \Sigma, R_2, S_2)$ are CFGs and G_1 generates L_1, G_2 generates L_2, how can we combine the grammars so we generate $L_1 \cup L_2$?

A. $G = (V_1 \cup V_2, \Sigma, R_1 \cup R_2, S_1 \cup S_2)$
B. $G = (V_1 \times V_2, \Sigma, R_1 \times R_2, (S_1, S_2))$
C. $G = (V1 \cup V2 \cup \{S0\}, \Sigma, R_1 \cup R_2 \cup \{S0 \rightarrow S1, S0 \rightarrow S2\}, S0)$
D. We might not always be able to: the class of CFG describable languages might not be closed under union.

Closure under concatenation

If $G_1 = (V_1, \Sigma, R_1, S_1)$ and $G_2 = (V_2, \Sigma, R_2, S_2)$ are CFGs

and G_1 generates L_1, G_2 generates L_2, how can we combine the grammars so we generate $L_1 L_2$?

CFGs in the wild

V = {E}, Σ = {2,+,x,(,)}, R = { **E → E+E | ExE | (E) | 2** }, S=E

Describing well-formed arithmetic expressions

Which of the followings strings is generated by this CFG?

A. E
B. 22
C. 2+2x2
D. ε
E. I don't know.

Derivations and parsing

E → E+E | ExE | (E) | 2

Lots of derivations for 2+2x2

$$E \Longrightarrow E+E \Longrightarrow E+E\times E \Longrightarrow 2+E\times E \Longrightarrow 2+2\times E \Longrightarrow 2+2\times 2$$

$$E \Longrightarrow E\times E \Longrightarrow E+E\times E \Longrightarrow 2+E\times E \Longrightarrow 2+2\times E \Longrightarrow 2+2\times 2$$

$$E \Longrightarrow E+E \Longrightarrow 2+E \Longrightarrow 2+E\times E \Longrightarrow 2+2\times E \Longrightarrow 2+2\times 2$$

Derivations and parsing

E → E+E | ExE | (E) | 2

Lots of derivations for 2+2x2

$$E \Longrightarrow E + E \Longrightarrow E + E \times E \Longrightarrow 2 + E \times E \Longrightarrow 2 + 2 \times E \Longrightarrow 2 + 2 \times 2$$

$$E \Longrightarrow E \times E \Longrightarrow E + E \times E \Longrightarrow 2 + E \times E \Longrightarrow 2 + 2 \times E \Longrightarrow 2 + 2 \times 2$$

$$E \Longrightarrow E + E \Longrightarrow 2 + E \Longrightarrow 2 + E \times E \Longrightarrow 2 + 2 \times E \Longrightarrow 2 + 2 \times 2$$

Understanding 2+2x2

Definition: A derivation is **leftmost** if at every step, the leftmost remaining variable is the one replaced.

Meaningful vs.	superficial differences in derivations
Different parsing	Same parsing, different order

Derivations and parsing

E → E+E | ExE | (E) | 2

Lots of derivations for 2+2x2

A string is **ambiguously derived** in a CFG if it has more than one leftmost derivation i.e. more than one parsing tree

More complicated language

$$\{a^n b^m \mid n \neq m\}$$

More complicated language

$$\{a^n b^m \mid n \neq m\}$$

Can rewrite as:

$$\{a^n b^m \mid n > m\} \cup \{a^n b^m \mid n < m\}$$

More complicated language

$$\{a^n b^m \mid n \neq m\}$$

Can rewrite as:

$$\{a^n b^m \mid n > m\} \cup \{a^n b^m \mid n < m\}$$

$G_1 = (\{S_1\}, \{a,b\}, R_1, S_1)$ with rules $S_1 \rightarrow aS_1 b \mid aS_1 \mid a$

$G_2 = (\{S_2\}, \{a,b\}, R_2, S_2)$ with rules $S_2 \rightarrow aS_2 b \mid S_2 b \mid b$

$G = (\{S, S_1, S_2\}, \{a,b\}, R, S)$ with rules $S \rightarrow S_1 \mid S_2$, …

Summary

A language L is

Regular	iff it is described by some regular expression
	iff it is recognized by some DFA
	iff it is recognized by some NFA
Context-free	iff it is recognized by some (nondet) PDA
	iff it is generated by some (ambiguous) CFG

What's left?

- **Fact**: Every regular language is a context-free language.
- **Fact**: There are context-free languages that are nonregular.
- **Fact**: There are countably infinitely many regular languages.
- **Fact**: There are countably infinitely many context-free languages.

Most languages are not context-free languages!

Examples of non-context-free languages

- $\{ a^n b^n c^n \mid 0 \leq n \}$ Sipser Ex 2.36
- $\{ a^i b^j c^k \mid 0 \leq i \leq j \leq k \}$ Sipser Ex 2.37
- $\{ w\, w \mid w \text{ is in } \{0,1\}^* \}$ Sipser Ex 2.38

To prove… Pumping lemma for CFLs (won't cover in CSE 105)

Closure properties of …

The class of regular languages is closed under

- Union
- Concatenation
- Star
- Complementation
- Intersection
- Difference
- Reversal
- FlipBits

The class of context-free languages is closed under

- Union
- Concatenation
- Star
- Reversal
- FlipBits

The class of context-free languages is not closed under

- Intersection
- Complementation
- Difference

For next time

IndivHW4 due Tuesday, February 13

For Monday, pre-class reading: pp. 166-167