CIS 770: Formal Language Theory

Pavithra Prabhakar

Kansas State University

Spring 2015

Union of CFLs

Let L_1 be language recognized by $G_1 = (V_1, \Sigma_1, R_1, S_1)$ and L_2 the language recognized by $G_2 = (V_2, \Sigma_2, R_2, S_2)$
Is $L_1 \cup L_2$ a context free language? Yes.
Just add the rule $S \to S_1 | S_2$
But make sure that $V_1 \cap V_2 = \emptyset$ (by renaming some variables).

Closure of CFLs under Union

$G = (V, \Sigma, R, S)$ such that $L(G) = L(G_1) \cup L(G_2)$:

- $V = V_1 \cup V_2 \cup \{S\}$ (the three sets are disjoint)
- $\Sigma = \Sigma_1 \cup \Sigma_2$
- $R = R_1 \cup R_2 \cup \{S \to S_1 | S_2\}$

Concatenation, Kleene Closure

Proposition

CFLs are closed under concatenation and Kleene closure

Proof.

Let L_1 be language generated by $G_1 = (V_1, \Sigma_1, R_1, S_1)$ and L_2 the language generated by $G_2 = (V_2, \Sigma_2, R_2, S_2)$

- Concatenation: $L_1 L_2$ generated by a grammar with an additional rule $S \to S_1 S_2$
- Kleene Closure: L_1^* generated by a grammar with an additional rule $S \to S_1 S | \epsilon$

As before, ensure that $V_1 \cap V_2 = \emptyset$. S is a new start symbol. (Exercise: Complete the Proof!) □

Intersection

Let L_1 and L_2 be context free languages. $L_1 \cap L_2$ is not necessarily context free!

Proposition

CFLs are not closed under intersection

Proof.

- $L_1 = \{a^i b^i c^j \mid i, j \geq 0\}$ is a CFL
  - Generated by a grammar with rules $S \rightarrow XY$; $X \rightarrow aXb|\epsilon$; $Y \rightarrow cY|\epsilon$.
- $L_2 = \{a^i b^j c^j \mid i, j \geq 0\}$ is a CFL.
  - Generated by a grammar with rules $S \rightarrow XY$; $X \rightarrow aX|\epsilon$; $Y \rightarrow bYc|\epsilon$.
- But $L_1 \cap L_2 = \{a^n b^n c^n \mid n \geq 0\}$ is not a CFL. $\square$

Complementation

Let L be a context free language. Is $\overline{L}$ context free? No!

Proof 1.

Suppose CFLs were closed under complementation. Then for any two CFLs L_1, L_2, we have

- $\overline{L_1}$ and $\overline{L_2}$ are CFL. Then, since CFLs closed under union, $\overline{L_1} \cup \overline{L_2}$ is CFL. Then, again by hypothesis, $\overline{\overline{L_1} \cup \overline{L_2}}$ is CFL.
- i.e., $L_1 \cap L_2$ is a CFL

i.e., CFLs are closed under intersection. Contradiction! □

Proof 2.

$L = \{x \mid x \text{ not of the form } ww\}$ is a CFL.

- L generated by a grammar with rules $X \rightarrow a|b$, $A \rightarrow a|XAX$, $B \rightarrow b|XBX$, $S \rightarrow A|B|AB|BA$

But $\overline{L} = \{ww \mid w \in \{a, b\}^*\}$ is not a CFL! (Why?) □

Set Difference

Proposition

If L_1 is a CFL and L_2 is a CFL then $L_1 \setminus L_2$ is not necessarily a CFL

Proof.

Because CFLs not closed under complementation, and complementation is a special case of set difference. (How?) □

Proposition

If L is a CFL and R is a regular language then $L \setminus R$ is a CFL

Proof.

$L \setminus R = L \cap \overline{R}$ □

Homomorphism

Proposition

Context free languages are closed under homomorphisms.

Proof.

Let $G = (V, \Sigma, R, S)$ be the grammar generating L, and let $h : \Sigma^* \to \Gamma^*$ be a homomorphism. A grammar $G' = (V', \Gamma, R', S')$ for generating $h(L)$:

- Include all variables from G (i.e., $V' \supseteq V$), and let $S' = S$
- Treat terminals in G as variables. i.e., for every $a \in \Sigma$
    - Add a new variable X_a to V'
    - In each rule of G, if a appears in the RHS, replace it by X_a
- For each X_a, add the rule $X_a \to h(a)$

G' generates $h(L)$. (Exercise!) $\square$

Homomorphism

Example

Let G have the rules $S \to 0S0|1S1|\epsilon$.
Consider the homorphism $h : \{0,1\}^* \to \{a,b\}^*$ given by $h(0) = aba$ and $h(1) = bb$.
Rules of G' s.t. $L(G') = h(L(G))$:

$$\begin{array}{rcl} S & \to & X_0SX_0|X_1SX_1|\epsilon \\ X_0 & \to & aba \\ X_1 & \to & bb \end{array}$$

Inverse Homomorphisms

Recall: For a homomorphism h, $h^{-1}(L) = \{w \mid h(w) \in L\}$

Proposition

If L is a CFL then $h^{-1}(L)$ is a CFL

Proof Idea

For regular language L: the DFA for $h^{-1}(L)$ on reading a symbol a, simulated the DFA for L on $h(a)$. Can we do the same with PDAs?

- Key idea: store $h(a)$ in a "buffer" and process symbols from $h(a)$ one at a time (according to the transition function of the original PDA), and the next input symbol is processed only after the "buffer" has been emptied.
- Where to store this "buffer"? In the state of the new PDA!

Inverse Homomorphisms

Let $P = (Q, \Delta, \Gamma, \delta, q_0, F)$ be a PDA such that $L(P) = L$. Let $h : \Sigma^* \to \Delta^*$ be a homomorphism such that $n = \max_{a \in \Sigma} |h(a)|$, i.e., every symbol of Σ is mapped to a string under h of length at most n. Consider the PDA $P' = (Q', \Sigma, \Gamma, \delta', q_0', F')$ where

- $Q' = Q \times \Delta^{\leq n}$, where $\Delta^{\leq n}$ is the collection of all strings of length at most n over Δ.
- $q_0' = (q_0, \epsilon)$
- $F' = F \times \{\epsilon\}$
- δ' is given by

$$\delta'((q, v), x, a) = \begin{cases} \{((q, h(x)), \epsilon)\} & \text{if } v = a = \epsilon \\ \{((p, u), b) \mid (p, b) \in \delta(q, y, a)\} & \text{if } v = yu,\ x = \epsilon \end{cases}$$

and $\delta'(\cdot) = \emptyset$ in all other cases.