CS304: Automata and Formal Languages

Lec 20

Decision Properties for CFLs and Context-Sensitive Languages

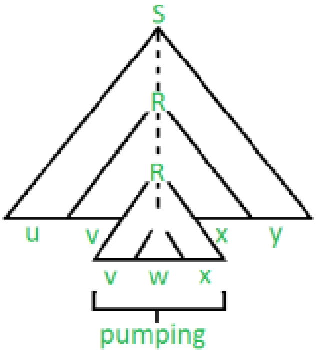
Rachit Nimavat

October 6, 2025

Outline

1. Recap
2. Decision Properties
3. Context-Sensitive Languages

Pumping Lemma for CFLs



Lemma 1.1

For each CFL L, there is a constant p (the pumping length) such that any string $s \in L$ with $|s| \geq p$ can be divided into five parts, $s = uvwxy$, such that

- *i.* $|vx| > 0$ *(at least 1 of v and x is not empty)*
- *ii.* $|vwx| \leq p$ *(middle portion is not too long)*
- *iii.* $\forall i \geq 0$, *the string* $uv^iwx^iy \in L$ *(we can pump v and x, one, or many times, and the resulting string is still in L)*

How to use Pumping Lemma?

Goal: Prove that a language L is not a CFL.
Assume that L is a CFL.

Pumping Lemma: Player-1

1. Provides 'pumping length' p
3. Provides a partition $s = uvwxy$ such that $|vx| > 0$ and $|vwx| < p$

Us: Player-2

2. Cleverly choose $s \in L$ such that $|s| \geq p$
4. WIN by choosing $i \geq 0$ such that $uv^iwx^iy \notin L$

If $uv^iwx^iy \notin L$, then we have a contradiction. Thus, our assumption that L is context-free must be false. Therefore, L is not a CFL.

Note: For this to work, we must have a winning strategy for each p and for each partition $s = uvwxy$ (that satisfies $|vx| > 0$ and $|vwx| < p$).

How to use the Pumping Lemma?

Prove that $L = \{0^n 1^n 2^n \mid n \geq 0\}$ is not context-free

Assume for contradiction that L is context-free

From pumping lemma, there is a constant p with "looping property"

Let's pick a string $s = 0^p 1^p 2^p$. Notice that $s \in L$

Consider its decomposition $s = uvwxy$ given by the pumping lemma

$0 < |vx| \leq |vwx| \leq p$ implies at least one of $\{0, 1, 2\}$ is not in vwx (since vwx is of length at most p, it cannot cover all three segments of s)

Choose $i = 0$. From pumping lemma, $s_0 = uv^0wx^0y = uwy$, which has unequal number of 0s, 1s and 2s.

Contradiction!

Closure Properties

Closure Properties: A set is closed under an operation if applying that operation to its elements results in elements that are also in the set.

- CFLs are closed under union (create a new start symbol with $S \mapsto S_1 \mid S_2$)
- CFLs are closed under concatenation (create a new start symbol with $S \mapsto S_1 S_2$)
- If L is a CFL, so is L^* (add rule $S \mapsto SS \mid \varepsilon$)
- Reversal of a CFL is context-free Reverse all production rules
- CFLs are closed under intersection **NO!** (Use $L_1 = \{a^i b^i c^j\}$ and $L_2 = \{a^j b^i c^i\}$)
- CFLs are closed under complementation **NO!** (Use DeMorgan's law)

Decision Properties of CFLs

Emptiness Problem: Given a CFG G, is $L(G) = \emptyset$? Can determine if the start symbol S is *generating* (i.e., can derive some terminal string).

Finiteness Problem: Given a CFG G, is $L(G)$ finite? Can determine if there is a cycle in the derivation from S to some terminal string. (You'll later learn graph algorithms to do this efficiently.)

Membership Problem: Given a CFG G and a string w, is $w \in L(G)$? Can be solved by simulating the corresponding PDA with input w. There is an efficient algorithm called the **CYK algorithm** that can solve this problem in $O(n^3)$ time, where n is the length of the string w.

Equivalence Problem: Given two CFGs G_1 and G_2, is $L(G_1) = L(G_2)$? This problem is **undecidable**, i.e., there is no algorithm that can solve this problem for all possible inputs. In the later part of the course, we'll learn how to even prove that a problem has **no algorithmic solution**.

Ambiguity: Given a CFG G, is G ambiguous? Undecidable!

Context-Sensitive Languages (CSLs)

In CFGs, the production rules are of the form $A \rightarrow \alpha$, where A is a single non-terminal and α is a string of terminals and/or non-terminals.

Relax this restriction and allow the left-hand side of a production rule to contain multiple symbols? **Context-Sensitive Grammars (CSGs)!**

A CSG is same as CFG, except the production rules are of the form $\alpha A \beta \rightarrow \alpha \gamma \beta$, where A is a non-terminal, α and β are strings of terminals and/or non-terminals (the context), and γ is a non-empty string of terminals and/or non-terminals.

Essentially, says, "Replace A with γ only when it appears in the **context** of α and β".

Languages that have CSGs are called **Context-Sensitive Languages (CSLs)**.

CSG: Alternative Definition

CSGs are **non-contracting grammars**:

- All production rules are of the form $\alpha \to \beta$, where α and β are strings of terminals and/or non-terminals, and $|\beta| \geq |\alpha|$.
- Except potentially the rule $S \to \epsilon$, where S is the start symbol, and S does not appear on the right-hand side of any production rule.

CSG Example

Consider $L = \{0^n 1^n 2^n \mid n \geq 0\}$ and its grammar:

$\mathsf{S} \to \mathsf{aSBC} \mid \varepsilon$

$\mathsf{CB} \to \mathsf{BC}$

$\mathsf{aB} \to \mathsf{ab}$

$\mathsf{bB} \to \mathsf{bb}$

$\mathsf{bC} \to \mathsf{bc}$

$\mathsf{cC} \to \mathsf{cc}$

How it works? On board!