Context-Free Language Properties

Closure, Pumping Lemma, and Decision Problems

Theory of Computation

Where We Are in the Course

So far:

- Regular languages and finite automata
- Context-free grammars (CFGs)
- Pushdown automata (PDAs)

Today:

- Closure properties of CFLs
- Limitations of CFLs
- Pumping lemma for CFLs
- Decision problems

What Does Closure Mean?

A class of languages is **closed** under an operation if:

$$L_1, L_2 \in \mathcal{C} \;\Rightarrow\; L_1 \circ L_2 \in \mathcal{C}$$

Closure tells us:

- which constructions are safe
- where expressive limits appear

CFLs Closed Under

Context-free languages are closed under:

- Union
- Concatenation
- Kleene star
- Homomorphism
- Inverse homomorphism

Closure Under Union

If L_1, L_2 are CFLs, then:

$$L_1 \cup L_2 \text{ is CFL}$$

Proof idea:

- Grammar view: new start symbol chooses G_1 or G_2
- PDA view: nondeterministically choose which PDA to simulate

Closure Under Concatenation

If L_1, L_2 are CFLs, then:

$$L_1 L_2 \text{ is CFL}$$

Proof idea:

- Grammar: $S \to S_1 S_2$
- PDA: run PDA for L_1, then switch to PDA for L_2

Closure Under Kleene Star

If L is CFL, then:

$$L^* \text{ is CFL}$$

Idea:

- Grammar recursion
- PDA looping with ε-moves

CFLs NOT Closed Under

CFLs are **not** closed under:

- Intersection
- Complement
- Difference

Non-Closure Under Intersection

Consider:

$$L_1 = \{a^i b^i c^j\}, \quad L_2 = \{a^i b^j c^j\}$$

Both are CFLs.

But:

$$L_1 \cap L_2 = \{a^n b^n c^n\}$$

Which is **not** context-free.

Why Complement Is Not Closed

If CFLs were closed under complement and union, then:

$$L_1 \cap L_2 = \overline{(\overline{L_1} \cup \overline{L_2})}$$

But CFLs are not closed under intersection.

Therefore:

CFLs are not closed under complement

Why We Need the Pumping Lemma

Closure properties are not enough to show:

$$L \notin \mathsf{CFL}$$

We need a **negative tool**.

That tool is the pumping lemma for CFLs.

Pumping Lemma for CFLs (Statement)

If L is context-free, then there exists $p > 0$ such that any $w \in L$ with $|w| \geq p$ can be written as:

$$w = uvxyz$$

such that:

- $|vxy| \leq p$
- $|vy| > 0$
- $uv^ixy^iz \in L$ for all $i \geq 0$

Why Pumping Works (Intuition)

- CFL derivations correspond to parse trees
- Long strings force repeated variables
- Repetition creates a "pumpable" subtree

This is an application of the pigeonhole principle.

How to Use the Pumping Lemma

Standard strategy:

1. Assume L is context-free
2. Let p be the pumping length
3. Choose a specific string w
4. Consider all valid decompositions
5. Show pumping breaks membership

Advanced Example C1: Equal Counts, Free Order

$$L = \{w \in \{a, b, c\}^* \mid \#a(w) = \#b(w)\}$$

Claim: L is context-free.

Reasoning:

- Order does not matter, only equality
- PDA can:
  - push for a
  - pop for b
  - push for b if stack empty
- c is ignored

Key insight: CFLs can enforce **one linear constraint**, even without fixed order.

Advanced Example C2: Two Independent Equalities

$$L = \{a^i b^j c^k d^\ell \mid i = k \text{ and } j = \ell\}$$

Claim: L is **not** context-free.

Intuition:

- Requires two independent counters
- Single stack cannot store two independent quantities

Formal proof uses the pumping lemma or intersection with a regular language.

Advanced Example C3: Center Marker Language

$$L = \{w\#w^R \mid w \in \{a, b\}^*\}$$

Claim: L is deterministic context-free.

Reason:

- The symbol $\#$ marks the midpoint
- DPDA:
  - push before $\#$
  - pop after $\#$

Contrast with:

$$\{ww^R\} \quad \text{(not DCFL)}$$

Advanced Example C4: Copy Language

$$L = \{ww \mid w \in \{a, b\}^*\}$$

Claim: L is not context-free.

Why it is tricky:

- Looks similar to palindromes
- Requires copying, not reversing

Proof requires a careful pumping-lemma argument.

Hard Proof D1: $\{ww\}$ Is Not CFL

Assume:

$$L = \{ww \mid w \in \{a, b\}^*\}$$

is context-free.

Let p be the pumping length.

Choose the String

Choose:

$$w = a^p b^p a^p b^p$$

Clearly $w \in L$ with:

$$w = (a^p b^p)(a^p b^p)$$

Structure of *vxy*

Because $|vxy| \leq p$, the substring vxy lies:

- entirely in the first half, or
- entirely in the second half

It cannot cover both copies of w.

Pumping Breaks Equality

Pumping v and y:

- changes only one copy of w
- leaves the other unchanged

Thus:

$$uv^ixy^iz \neq ww$$

Contradiction.

Conclusion

$$\{ww \mid w \in \{a, b\}^*\} \notin \mathsf{CFL}$$

Hard Proof D2: Using Intersection with Regular Languages

Consider:

$$L = \{a^i b^j c^k d^\ell \mid i = k \text{ and } j = \ell\}$$

Assume L is CFL.

Intersect with a Regular Language

Let:

$$R = a^* b^* c^* d^*$$

R is regular.

If CFLs were closed under intersection with regular languages:

$$L \cap R \text{ would be CFL}$$

Resulting Language

$$L \cap R = \{a^n b^m c^n d^m\}$$

This language requires:

- matching a with c
- matching b with d

Two independent equalities $\Rightarrow$ not CFL.

Contradiction

Single-stack PDA cannot enforce both equalities.

Thus:

$$L \notin \mathrm{CFL}$$

Hard Proof D3: CFLs Not Closed Under Difference

Assume CFLs are closed under difference.

Then for any CFLs L_1, L_2:

$$L_1 - L_2 \in \mathrm{CFL}$$

Deriving a Contradiction

Let:

$$L_1 = \Sigma^*, \quad L_2 = \{a^i b^j c^k \mid i = j = k\}$$

L_1 is regular (hence CFL).

If difference were closed:

$$\Sigma^* - L_2 = \overline{L_2}$$

would be CFL.

Contradiction

This would imply CFLs are closed under complement.

But they are not.

Therefore CFLs are not closed under difference.

Advanced Exam Problems

1. Prove $\{ww \mid w \in \{a, b\}^*\}$ is not context-free.
2. Show that adding a center marker makes palindromes deterministic.
3. Use closure properties to show a language is not CFL without pumping.
4. Explain why two independent equalities cannot be enforced by a PDA.
5. Compare DCFL and CFL closure properties.

Problem 1: $\{ww \mid w \in \{a, b\}^*\}$

Claim

$$L = \{ww \mid w \in \{a, b\}^*\} \notin \text{CFL}$$

Proof technique: Pumping Lemma for Context-Free Languages

Assumption and Pumping Setup

Assume, for contradiction, that L is context-free.

By the pumping lemma for CFLs, there exists a pumping length $p \geq 1$ such that any string $s \in L$ with $|s| \geq p$ can be written as:

$$s = uvxyz$$

satisfying:

- $|vxy| \leq p$
- $|vy| > 0$
- $uv^ixy^iz \in L$ for all $i \geq 0$

Choice of the String

Choose the string:

$$s = a^p b^p a^p b^p$$

Clearly,

$$s \in L$$

since:

$$s = (a^p b^p)(a^p b^p)$$

is of the form ww.

The string consists of four blocks:

$$a^p \;\; b^p \;\; a^p \;\; b^p$$

Structure of the Pumped Substring

Since $|vxy| \leq p$, the substring vxy must lie entirely within *one* of the four blocks.

Thus, vxy:

- cannot span both halves of the string
- affects symbols in only one copy of w

Pumping Leads to a Contradiction

Consider pumping with $i = 0$ or $i = 2$.

This changes the length of exactly one half of the string, while the other half remains unchanged.

Therefore, the resulting string is *not* of the form ww.

Hence:

$$uv^i xy^i z \notin L$$

which contradicts the pumping lemma.

Conclusion

$$\{ww \mid w \in \{a, b\}^*\} \notin \text{CFL}$$

Problem 2: $\{w\#w^R\}$

Claim

$$L = \{w\#w^R \mid w \in \{a, b\}^*\} \in \text{DCFL}$$

Deterministic PDA Construction

A deterministic PDA recognizes L as follows:

- While reading symbols before '#', push each symbol onto the stack
- Upon reading '#', switch to matching mode
- After '#', for each input symbol, pop and compare with stack top

At no point is a nondeterministic choice required.

Why Determinism Works

The symbol '#' uniquely determines:

- the midpoint of the string
- the moment to switch from push to pop

This removes the need to guess the midpoint.

Conclusion

$$\{w\#w^R \mid w \in \{a,b\}^*\} \text{ is deterministic context-free}$$

Problem 3: Language with Two Equalities

$$L = \{a^i b^j c^k d^\ell \mid i = k \text{ and } j = \ell\}$$

Claim

$$L \notin \mathrm{CFL}$$

Intersection with a Regular Language

Assume, for contradiction, that L is context-free.

Let:

$$R = a^* b^* c^* d^*$$

which is a regular language.

CFLs are closed under intersection with regular languages, so:

$$L \cap R \text{ is CFL}$$

Resulting Language

Compute the intersection:

$$L \cap R = \{a^n b^m c^n d^m\}$$

This language requires simultaneously:

- $\#a = \#c$
- $\#b = \#d$

Why This Is Impossible for a PDA

A pushdown automaton has:

- finite control
- exactly one stack

A single stack can enforce only one unbounded counting dependency.

Two independent equalities cannot be maintained.

Contradiction

$$L \notin \mathrm{CFL}$$

Problem 4: One Stack = One Dependency

A PDA can:

- push symbols to count
- pop symbols to match

This naturally enforces constraints such as:

$$\#a = \#b$$

Why Two Equalities Fail

Languages requiring:

$$\#a = \#c \quad \text{and} \quad \#b = \#d$$

need two independent counters.

A single stack:

- cannot store counters independently
- cannot access deeper information without destroying structure

Conclusion

$$\text{One stack} \Rightarrow \text{one unbounded dependency}$$

Problem 5: Closure Properties of CFLs

Context-Free Languages (CFLs)

- Closed under union
- Closed under concatenation
- Closed under Kleene star
- Not closed under intersection
- Not closed under complement

Closure Properties of DCFLs

Deterministic Context-Free Languages (DCFLs)

- Closed under complement
- Not closed under union
- Not closed under concatenation

Explanation

- Determinism eliminates nondeterministic branching
- Complementation preserves determinism
- Union and concatenation require nondeterministic choice