The Pumping Lemma for CFLs

In the following sections we will

- Give an analogous Pumping Lemma for context-free languages (CFLs)
  - Will follow Linz and Sipser. I don't like Cohen's treatment here.
- Give some properties of CFLs
  - The union or concatenation of two CFLs is a CFL
  - Two intersection of two CFLs may not be a CFL
- Discuss deterministic PDAs
  - Compiler theory follows naturally from this
- Follow Linz's treatment
  - May wish to check out Sipser's (pages 115-119) treatment for a different perspective

- Not everything is a context-free language
- Recall DFAs had memory proportional to the number of states in the graph
- For CFGs (so really NPDAs), we have the number of states and an infinite stack
  - Stack is infinite, but it's FILO
  - Thus, can guess languages like

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

    may not work since the second w is the "wrong way"
    * Recall ww^R is a CFL
  - May also guess that if needed to keep track of more than one match (e.g. number of a's = number of b's), then NPDA may not be able to handle it. For example

    $$L = \{a^n b^n c^n \mid n \geq 0\}$$

    After finishing with b's, stack is empty, so have "forgotten" how many a's we've had. Similarly, we'll prove

    $$L = \{a^i b^j c^k \mid 0 \leq i \leq j \leq k\}$$

    is also not a CFL

Below is the analog of the Pumping Lemma for CFLs:

Theorem 1 (Pumping Lemma for CFLs). *Let L be an infinite context-free language. Then there exists an integer $m \geq 1$ such that any $w \in L$ with $|w| \geq m$ can be decomposed as*

$$w = uvxyz$$

with

$$|vxy| \leq m$$

and

$$|vy| \geq 1$$

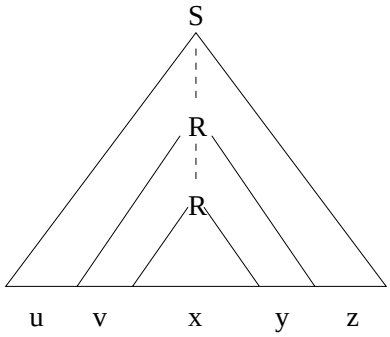
such that

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

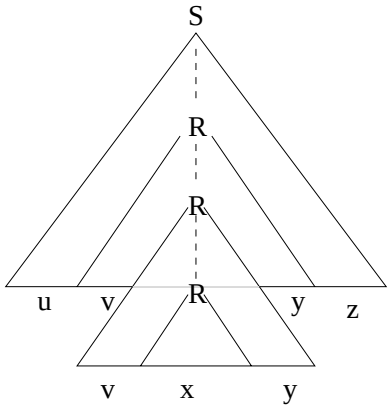
for all $i = 0, 1, 2, \ldots$.

The idea of the proof is

- Let w be a sufficiently long string in L
  - Thus, there's a long (tall) parse tree, T, for w
- There's a long path from S to a terminal (leaf) in T
- Along this path, some variable, R, must repeat due to the pigeonhole principle



- Can then copy the subtree under the first R to get a valid subtree



The above is a valid derivation tree (of a different string in the language)

- Can then copy (pump) as often as needed

Below is a more formal sketch of the proof:

Proof.

- Let $G = (V, T, S, P)$ be the CFG for L with λ-productions and unit-productions removed
- Let b be the maximum number of symbols in the right-hand side of a production.
  - $b \geq 2$ since there are no unit-productions
- In a parse tree of G
  - A vertex can have no more than b children
  - Thus, at any depth d, there is at most b^d nodes
- Let
$$m = b^{|V|+2}$$
  - Since $b \geq 2$, $m > b^{|V|+1}$
  - Thus, for a string of length $> m$, need height of at least $|V| + 2$

- Let $w \in L$ with $|w| \geq m$
  - Let τ be the parse tree for w with the smallest number of vertices
  - Due to above, the height of τ is $\geq |V| + 2$
    - Must have at least $|V| + 1$ variables with terminal leafs
  - Since G has only $|V|$ variables, some variable R, must appear more than once
    - Let's say R is the last variable (from the top) to be repeated
- Divide w as shown in the first figure above
  - Pump as discussed above

- $|vy| \geq 1$
  - If v and y where both λ then you could reduce the number of nodes in τ by substituting the subtree without v and y to get a smaller derivation tree. Contradiction since τ was assumed to be the smallest derivation tree.
- $|vxy| \leq m$
  - The parse subtree under R generates vxy
  - R is in the bottom $|V| + 1$ variables
  - Thus, the parse subtree is at most of height $|V| + 2$
    - That means word generated by R must be at most of length m

□

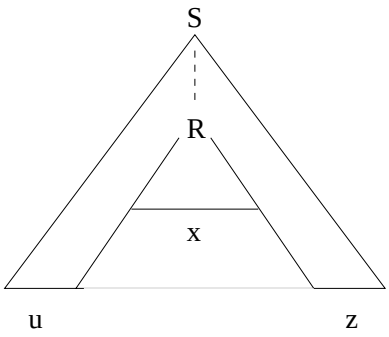
As before, you can pump zero times, $i = 0$, to get from the word

$$w = uvxyz$$

to

$$w' = uxz$$

given by the derivation tree



- Use exactly as you used previous pumping lemma
  - Assume L is a CFL
  - Find a string in L that when pumped is no longer in L
  - Contradiction
    - Thus L is not a CFL
- Same "adversary" argument
  - This time, the adversary breaks w into u, v, x, y, and z
- Now
  - Break word up into five pieces
  - Restriction $|vxy| \leq m$ less restrictive than previous pumping lemma's $|xy| \leq m$
  - Have u so vxy may not be at the beginning of w

  thus, proving L is not a CFL is somewhat trickier
- I will call the above theorem just the "pumping lemma" with the implication that we're doing the pumping lemma as applied to a CFL
- Again, I will provide a number of examples
  - Study them as templates for how to do others

PL for CFLs: Examples

- Prove
$$L = \{a^n b^n c^n \mid n \geq 0\}$$
is not context-free.
  - Assume L is a CFL
  - Let m be the pumping length for L that is guaranteed to exist by the pumping lemma
  - Let
$$w = a^m b^m c^m$$
Obviously, $|w| \geq m$
  - By the pumping lemma, either v or y is nonempty

– Two cases to consider:

1. Both v and y contain only one type of alphabet symbol. Then pumping once, $w' = uv^2xy^2z$, we increase the number of symbols in v and y. Since originally, w had an equal number of each symbol, pumping once implies there is at least more of than one symbol than one other. Thus, $w' \notin L$.
2. Either v and/or y contain more than one type of alphabet symbol. In a similar argument to above, can only conceive of pumping keeping the same number of symbols if v and y (combined) has the same number of a's, b's, and c's. This cannot happen since $|vxy| \leq m$, so at most, they can have two different symbols. Furthermore, even if v and y could have all three symbols in equal amounts, pumping would cause the a's b's and c's to come in the wrong order. Thus, pumping once, $w' \notin L$.

- Prove

$$L = \{a^i b^j c^k \mid 0 \leq i \leq j \leq k\}$$

is not a CFL. Again, letting $w = a^m b^m c^m$, we have the same two cases as above

1. v and y contain only one type of symbol. That means at least a, b, or c does not appear in v and y.
   1.1. If the a's do not appear, then pump down once to get $w = uxz$. Since we get rid of some b's and/or c's that means there's now more a's than b or c. Contradiction.
   1.2. If the b's do not appear, then
   - If a's appear, pump once to get more a's than b's
   - If c's appear, pump down to get more b's than c's

   1.3. If the c's do not appear, then pump once to get more a's and/or b's than c's
2. Either v or y contains more than one symbol. Then pumping once will produce a word that does not contain symbols in the correct order.

Your Turn Now

Prove

$$L = \{a^n b^j c^k \mid k > n, k > j, n \& j \geq 0\}$$

is not context-free.

Answer:

More Ex.'s of Pumping Lemma for CFLs

- Prove

$$L = \{ss \mid s \in \{a, b\}^*\}$$

is not a CFL. Let

$$w = uvxyz = a^m b^m a^m b^m$$

vxy must straddle the middle of the word, i.e. $vxy = b^i a^j$

  - If not, then pumping once will make the word not symmetric. For example, if $vxy = a^p b^q$ (vxy lies on the boundary between the first set of a's and b's), then pumping once gives

$$w' = a^{m+p} b^{m+q} a^m b^m$$

    which is clearly not of the form ss

If vxy straddles the middle of the word, then pumping down, we get $a^m b^k a^\ell b^m$ with k and ℓ cannot both be m. Thus, the string is no longer of the from ss.

- Prove

$$L = \{a^{n!} \mid n \geq 3\}$$

is not a CFL. We already showed this is not a regular language. In the case of one letter

  - Pumping lemma for regular languages can generate the same words (actually more) than the pumping lemma for CFL
  - Thus, since showed by pumping that the above fails for a regular language, can show pumping also fails for CFL
  - Can prove (Problem 20 on page 375 in Cohen) that

    Theorem 2. *If L is a language over the one-letter alphabet, $\Sigma = \{a\}$, and L can be shown to be non-regular using the pumping lemma for regular languages, then L can be shown to be non-context-free using the pumping lemma for context-free languages.*

Your Turn Now

Prove

$$L = \{a^{n^2} \mid n \geq 0\}$$

is not context-free. Do not use the theorem on the previous page. Instead, pick an appropriate string in L and pumping it using the CFG pumping lemma.

Answer:

More Ex.'s of Pumping Lemma for CFLs — II

- Prove

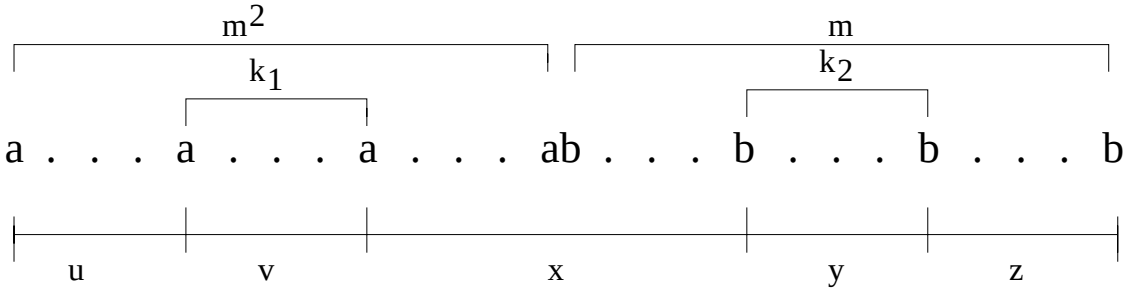
$$L = \{a^n b^j \mid n = j^2, j \geq 0\}$$

is not a CFL. Let

$$w = a^{m^2} b^m$$

If we pick anything other than

$$\begin{aligned} v &= a^{k_1} \\ x &= a^p b^q \\ y &= b^{k_2} \end{aligned}$$



then, pumping, it's easy to see the pumped word will no longer be in L.

Let's consider the above partition of w. Pump i times. The new string has

- $m^2 + (i-1)k_1$ a's
- $m + (i-1)k_2$ b's

There are several possibilities. The adversary can take

- $k_1 \neq 0$, $k_2 \neq 0$. Then with $i = 0$ we have

$$\begin{aligned}(m - k_2)^2 &\leq (m-1)^2 \\ &= m^2 - 2m + 1 \\ &< m^2 - k_1\end{aligned}$$

  so we do not have the number of a's as a square of the number of b's. Thus, the pumped word is not in L

- $k_1 = 0$ and $k_2 \neq 0$, OR $k_1 \neq 0$ and $k_2 = 0$. Let $i = 0$ so that the pumped string is no longer in L

PL for Linear Languages

Recall a linear grammar is a grammar with at most one variable on the right hand side of any production. We have the special case below for CFLs:

Definition 1. A context-free language L is said to be a *linear language* if there exists a linear context-free grammar G such that $L = L(G)$.

For example the grammar

$$\begin{aligned} S &\rightarrow Ab \\ A &\rightarrow aAb \\ A &\rightarrow \lambda \end{aligned}$$

for the language

$$L = \{a^n b^{n+1} \mid n \geq 0\}$$

is linear while the grammar

$$\begin{aligned} S &\rightarrow SS \\ S &\rightarrow \lambda \\ S &\rightarrow aSb \\ S &\rightarrow bSa \end{aligned}$$

for the language

$$L = \{w \mid n_a(w) = n_b(w)\}$$

is not linear.

The question to ask is:

All linear languages are CFLs, but are all CFLs linear languages?

The answer, as you may guess is NO.

To show not all CFLs are linear languages, let's introduce the pumping lemma for linear CFLs:

Theorem 3. *Let L be an infinite linear language. Then there exists some $m \geq 1$, such that for any $w \in L$ with $|w| \geq m$, it may be decomposed as $w = uvxyz$ with*

$$\begin{aligned} |uvyz| &\leq m \\ |vy| &\geq 1 \end{aligned}$$

such that

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

for all $i = 0, 1, 2, \ldots$

Do note that it's really $|uvyz| \leq m$ instead of previous $|vxy| \leq m$.

We're pumping the word only at the "ends" at most m characters in.

Please read Linz, page 217, for the proof of this theorem (Theorem 8.2).

Note that x in the above theorem can be of any length.

For example, the language

$$L = \{w \mid n_a(w) = n_b(w)\}$$

really isn't linear. (While we've shown a grammar for L which isn't linear, that doesn't mean we've shown there isn't a linear grammar for L.) To prove it, let

$$w = a^m b^{2m} a^m$$

Since $|uvyz| \leq m$, u, v, y, and z must consist entirely of a's only. Pumping the string, we get

$$w' = a^{m+k} b^{2m} a^{m+\ell}$$

with $k \geq 1$ and/or $\ell \geq 1$, which shows $w' \notin L$.

Thus we have shown

Not all CFLs are linear languages.

Properties of CFLs

We now turn our attention to some properties of CFLs:

- The union and concatenation of two CFLs is a CFL
- The Kleene star of a CFL is a CFL
- The intersection of two CFLs may *not* be a CFL
- The complement of a CFL may *not* be a CFL
- But, the intersection of a CFL and a regular language is a CFL
- Algorithm to show a CFL is not empty
- Algorithm to show if a CFL is finite or infinite

Theorem 4. *Let L_1 and L_2 be two context-free languages. Then*

- $L_1 \cup L_2$
- $L_1 L_2$
- ${L_1}^*$

are context-free languages.

We'll give a sketch of the proof using NPDAs (unlike Linz):

Proof. Since L_1 and L_2 are CFLs, there are NPDAs M_1 and M_2 associated with them. Assume that M_1 and M_2 accept a string only if the stack in the final state is empty.

- Let q_{01} and q_{02} be the initial states of M_1 and M_2 respectively. Then, create a new initial state q_0 with transitions

$$\begin{aligned} \delta(q_0, \lambda, \lambda) &= (q_{01}, \lambda) \\ \delta(q_0, \lambda, \lambda) &= (q_{02}, \lambda) \end{aligned}$$

  Let q_{01} and q_{02} no longer be initial states of the new machine. Then, this new NPDA accepts $L_1 \cup L_2$

- Let q_{fi}, $i = 0, 1, \ldots n = |F_{M1}|$, be the final states of M_1 and q_{02} be the initial state of M_2. Then add transitions

$$\begin{aligned} \delta(q_{f0}, \lambda, \lambda) &= (q_{02}, \lambda) \\ \delta(q_{f1}, \lambda, \lambda) &= (q_{02}, \lambda) \\ &\vdots \\ \delta(q_{fn}, \lambda, \lambda) &= (q_{02}, \lambda) \end{aligned}$$

  Then, let q_{fi} no longer be final states. Then, the new NPDA accepts $L_1 L_2$

- Let q_{fi}, $i = 0, 1, \ldots n = |F_{M1}|$, be the final states of M_1 and q_0 be the initial state of M_1. Then add a new final state q_f. Add transitions

$$\begin{aligned} \delta(q_{f0}, \lambda, \lambda) &= (q_f, \lambda) \\ \delta(q_{f1}, \lambda, \lambda) &= (q_f, \lambda) \\ &\vdots \\ \delta(q_{fn}, \lambda, \lambda) &= (q_f, \lambda) \\ \delta(q_0, \lambda, \lambda) &= (q_f, \lambda) \\ \delta(q_f, \lambda, \lambda) &= (q_0, \lambda) \end{aligned}$$

  The machine then accepts ${L_1}^*$

□

For the last two cases above, you may need to add an additional state to empty (POP) out the stack before going to the start state of M_2 or M_1 respectively.

The concatenation of two context-free languages implies that the two parts are independent of each other. For example, if

$$L_1 = L_2 = \{a^n b^n \mid n \geq 0\}$$

then

$$L_1 L_2 = \{a^m b^m a^n b^n \mid m, n \geq 0\}$$

and **NOT the below:**

$$L_1 L_2 = \{a^n b^n a^n b^n \mid n \geq 0\}$$

Similarly for the Kleene star.

Theorem 5. *The intersection of two context-free languages may not necessarily be a context-free language.*

Proof. Proof by counterexample. (We'll show where the intersection of two CFLs gives a languages which is not a CFL.) Let

$$\begin{aligned} L_1 &= \{a^n b^n c^m \mid n, m \geq 0\} \\ L_2 &= \{a^n b^m c^m \mid n, m \geq 0\} \end{aligned}$$

L_1 and L_2 are CFLs since they are the concatenation of two CFLs, i.e. $\{a^n b^n \mid n \geq 0\}$ and $\{c^m \mid m \geq 0\}$ for L_1 and similarly for L_2. However,

$$L_1 \cap L_2 = \{a^n b^n c^n \mid n \geq 0\}$$

which, as we've seen is not a CFL. Thus, we see a counterexample showing $L_1 \cap L_2$ may not necessarily be a CFL. □

Theorem 6. *Let L be a context-free language. Then $\overline{L}$ may not be a context-free language.*

Linz's (and whole lot of other people's) Proof. Recall

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

If the complement of a CFL is a CFL, then the above equality would imply that the intersection of CFLs is a CFL (since we've proven the union of two CFLs is a CFL). Contradiction. □

Cohen proves the above by giving a counterexample

- $L = \{a^n b^n a^n \mid n \geq 0\}$ is *not* a CFL
- Cohen shows $\overline{L}$ is a CFL — complex proof
- See Cohen, pages 389-392

The below property is used sometimes to show a language is or isn't a CFL.

Theorem 7. *Let L_1 be a context-free language. Let L_2 be a regular language. Then $L_1 \cap L_2$ is a context-free language.*

Proof. Let $M_1 = (Q, \Sigma, \Gamma, \delta_1, q_0, F_1)$ be the NPDA which accepts L_1. Let $M_2 = (P, \Sigma, \delta_2, p_0, F_2)$ be the DFA which accepts L_2. We create the NPDA

$$\hat{M} = (\hat{Q}, \Sigma, \Gamma, \hat{\delta}, \hat{q_0}, \hat{F})$$

which simulates the parallel action of M_1 and M_2. Let

$$\begin{aligned} \hat{Q} &= Q \times P \\ \hat{q_0} &= (q_0, p_0) \\ \hat{F} &= F_1 \times F_2 \end{aligned}$$

and define $\hat{\delta}$ such that

$$((q_k, p_\ell), x) \in \hat{\delta}((q_i, p_j), a, b)$$

if and only if

$$(q_k, x) \in \delta_1(q_i, a, b)$$

and

$$\delta_2(p_j, a) = p_\ell$$

and if $a = \lambda$ (M_1 needs to just PUSH or POP something), then we need $p_j = p_\ell$.

By induction, it can then be shown that

$$((q_0, p_0), w, \lambda) \vdash^*_{\hat{M}} ((q_r, p_s), x)$$

with $q_r \in F_1$ and $p_s \in F_2$, if and only if

$$(q_0, w, \lambda) \vdash^*_{M_1} (q_r, x)$$

and

$$\delta^*(p_0, w) = p_s$$

Thus, the $\hat{M}$ only accepts strings if they are accepted by both M_1 and M_2. □

We can use this property to prove some languages are or are not CFL. We give two examples:

1. The language

   $$L = \{a^n b^n \mid n \geq 0, n \neq 100\}$$

   is a CFL. We prove this by noting

   $$L_1 = \{a^{100} b^{100}\}$$

   is regular (so $\overline{L_1} = \{w \in \{a,b\}^* \mid w \neq a^{100} b^{100}\}$) and

   $$L_2 = \{a^n b^n \mid n \geq 0\}$$

   is a CFL. Thus, since

   $$L = \overline{L_1} \cap L_2$$

   L is a CFL

2. The language

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

   is not a CFL. We prove this by contradiction. Suppose L was a CFL. Then

   $$L' = L \cap L(a^* b^* c^*) = \{a^n b^n c^n \mid n \geq 0\}$$

   would also be a CFL since it's the intersection of a CFL and a regular language. But, we've already proven L' is not a CFL. Contradiction. Thus, L is not a CFL.

Your Turn Now

Prove

$$L = \{w \in \{a, b\}^* \mid n_a(w) = n_b(w), \text{ no substr } aab\}$$

is context-free.

Answer:

Properties of CFLs — II

Finally, we give algorithms for deciding two properties of CFLs:

Theorem 8. *Let G be a positive context-free grammar with starting variable S. Then there exists an algorithm for deciding whether or not $L(G)$ is empty*

Proof. Remove useless symbols and productions. (An algorithm for doing so was shown.) If S is found to be useless, then $L(G)$ is empty. □

Since we have already done examples of removing useless symbols (and the above is just an application of that), we shall skip an example of this.

Theorem 9. *Let $G = (V, T, S, P)$ be a positive context-free grammar. Then there exists an algorithm for determining whether or not $L(G)$ is infinite.*

Proof. • Let G have no λ-productions, unit-productions, or useless symbols. If G had a repeating variable, i.e.

$$A \stackrel{*}{\Rightarrow} xAy$$

where x and y cannot both be λ (since there's no λ-productions or unit-productions), then since A is not useless, we have

$$S \stackrel{*}{\Rightarrow} uAv \stackrel{*}{\Rightarrow} w$$

and

$$A \stackrel{*}{\Rightarrow} z$$

where u, v, and z are in T^*. Then, we have

$$S \stackrel{*}{\Rightarrow} uAv \stackrel{*}{\Rightarrow} ux^nAy^nv \stackrel{*}{\Rightarrow} ux^nzy^nv$$

for any n, thus $L(G)$ is infinite. (Otherwise, $L(G)$ is finite with the length of any derivation being $O(|V|)$.)

- To find if any variables repeat, create a dependency graph where the vertices are labeled by the variables and there is an edge (A, B) if there is a production

$$A \to xBy$$

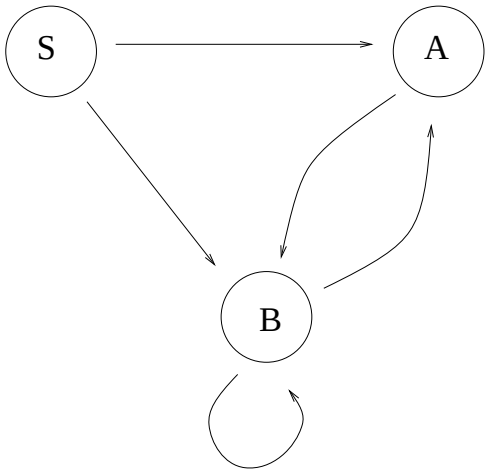
If there are any cycles in the dependency graph, then a variable repeats and thus $L(G)$ is infinite.

□

As an example, take the CFG

$$\begin{aligned} S &\to aABB \mid aAA \\ A &\to aBB \mid a \\ B &\to bBB \mid A \end{aligned}$$

The dependency graph is



Since there are cycles, $A - B - A$ and $B - B$, this language is infinite.

- No algorithm for showing two CFGs generate the same language
- A CFL under a homomorphism is still a CFL
- A linear CFL under a homomorphism is still a linear CFL
- The reverse of a CFL is still a CFL
  - Reverse edges in PDA
  - POPs become PUSHs and PUSHs become POPs
- The concatenation of linear CFLs need not be a linear CFL
  - For example, $L_1 = \{a^m b^m \mid m \geq 0\}$ and $L_2 = \{a^n b^n \mid n \geq 0\}$ are both linear CFLs, but

$$L_1 L_2 = \{a^m b^m a^n b^n \mid m, n \geq 0\}$$

    is *not* a linear CFL — you will prove this.