Properties of Context-Free Languages

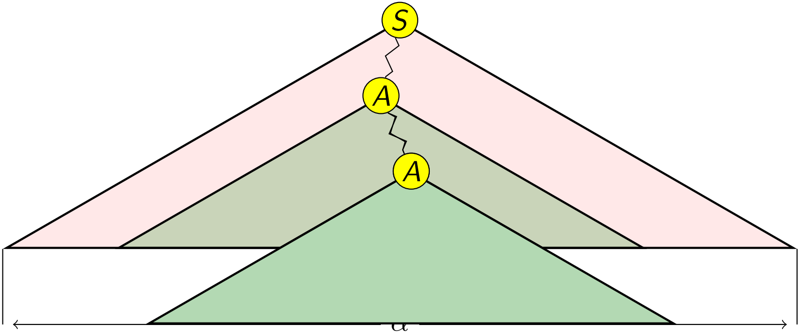
5DV037 — Fundamentals of Computer Science
Umeå University
Department of Computing Science

Stephen J. Hegner
hegner@cs.umu.se
http://www.cs.umu.se/~hegner

The Pumping Lemma for CFLs

Context: A CFG $G = (V, \Sigma, S, P)$.

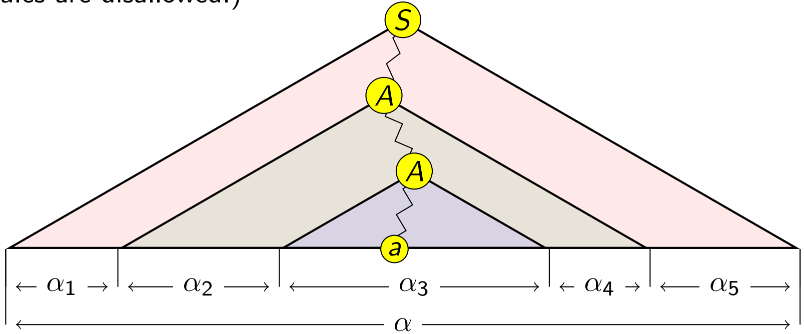
- Let α be a "sufficiently long" string in $\mathcal{L}(G)$.
- Then there is a path in a derivation tree for α in which some variable A occurs at least twice.
- By replacing the little subtree rooted at A with the big subtree rooted at A, the string may be "pumped" up to get a longer string in the language.
- Conversely for pumping down.



Details of the Pumping Lemma

Context: A CFG $G = (V, \Sigma, S, P)$.

- Choose A to be the variable whose second-lowest occurrence is lowest amongst all variables which occurs at at least twice on the path.
- Then the length of $\alpha_2 \cdot \alpha_3 \cdot \alpha_4$ will be bounded with a value dependent only upon G, independent of α. (Takes a little work to prove.)
- Also, either α_2 or α_4 will be nonempty. (Use the fact that chain and null rules are disallowed.)



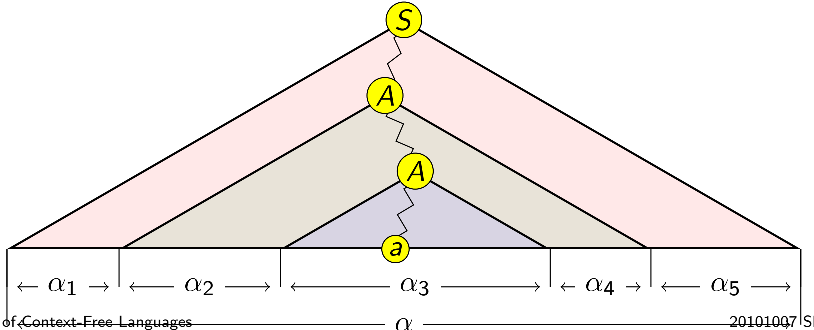
Formal Statement of the Pumping Lemma

Theorem (The Pumping Lemma for Context-Free Languages): Let L be a CFL. Then there is a constant $N \in \mathbb{N}$, depending only upon L, such that for any $\alpha \in L$ with Length(α) $\geq N$, there is a decomposition

$$\alpha = \alpha_1 \cdot \alpha_2 \cdot \alpha_3 \cdot \alpha_4 \cdot \alpha_5$$

with

- Length($\alpha_2 \cdot \alpha_4$) ≥ 1;
- Length($\alpha_2 \cdot \alpha_3 \cdot \alpha_4$) $\leq N$;
- $\alpha_1 \cdot (\alpha_2)^m \alpha_3 \cdot (\alpha_4)^m \cdot \alpha_5 \in L$ for all $m \in \mathbb{N}$. $\Box$



How to Use the Pumping Lemma

- In the pumping lemma for regular languages, the substring α_2 to be pumped always lies near the beginning of the string α to be tested.
- In the pumping lemma for context-free languages, the substrings α_2 and α_4 to be pumped may lie anywhere in α, although they must be "close" to each other.
- Otherwise, the strategy for use is the same.
- Suppose that $L \subseteq \Sigma^*$ is a language which is to be proven not context free.
- Assume that N is fixed, but you may not set it to any particular value.
- You choose the string $\alpha \in L$ to "pump".
- It must be the case that $\text{Length}(\alpha) \geq N$.
- Use N as a parameter of the string α.
- You must take into account *all* decompositions of α into $\alpha_1 \cdot \alpha_2 \cdot \alpha_3 \cdot \alpha_4 \cdot \alpha_5$ which satisfy the conditions of the pumping lemma.
- In general, the pumping lemma can only be used to show that a language is not context free; it cannot be used to show that a language is context free.

An Example of the Use of the Pumping Lemma

Example: $L = \{a^k b^k c^k \mid k \in \mathbb{N}\}$ (with the alphabet $\Sigma = \{a, b, c\}$).

- Show that L is not context free.
- Let N be the constant guaranteed for L by the pumping lemma.
- Choose $\alpha = a^N b^N c^N$.
- There are five possible forms for a PL decomposition:

	α_1	$\alpha_2 \cdot \alpha_3 \cdot \alpha_4$	α_5	
(a)	a^i	a^j	$a^k b^N c^N$	$j > 0,\ i + j + k = N$
(b)	a^i	$a^j b^k$	$b^\ell c^N$	$i + j = k + \ell = N;\ j + k > 0$
(c)	$a^N b^i$	b^j	$b^k c^N$	$i + j + k = N;\ j > 0$
(d)	$a^N b^i$	$b^j c^k$	c^ℓ	$i + j = k + \ell = N;\ j + k > 0$
(e)	$a^N b^N c^i$	c^j	c^k	$i + j + k = N;\ j > 0$

- $\alpha_2 \cdot \alpha_3 \cdot \alpha_4$ contains at most two of $\{a, b, c\}$.
- Thus, when pumping up or down, the number of occurrences of some letter will not change, while that of at least one other must change.
- Hence, the result cannot have an equal number of each letter.
- Thus, the language is not context free.

Further Examples of the Pumping Lemma for CFLs

- The same or very similar strings may be used to prove that related languages are not context free.

Example: $L = \{a^{k_1} b^{k_1} c^{k_2} \mid k_1 > k_2\}$.

- Let N be the constant guaranteed for L by the Pumping Lemma for this language.
- The string $a^{N+1} b^{N+1} c^{N} \in L$ may be used to show that this language is not context free, in exactly the same way, except that in certain cases it may be necessary to pump in a single direction (up or down) in order to obtain a string not in L.

Example: $L = \{a^k b^k a^k \mid k \in \mathbb{N}\}$.

- This is essentially the same example as on the previous slide. Choose $\alpha = a^N b^N a^N$.

Example: $L = \{w \in \{a, b, c\}^* \mid \mathrm{Count}\langle a, w\rangle = \mathrm{Count}\langle b, w\rangle = \mathrm{Count}\langle c, w\rangle\}$.

- Choose $\alpha = a^N b^N c^N$, and proceed as on the previous slide.

Further Examples of the Pumping Lemma for CFLs — 2

Example: $L = \{w \cdot w \mid w \in \{a, b\}^*\}$.

- Let N be the constant guaranteed for L by the Pumping Lemma for this language.
- Choose $\alpha = a^N b^N a^N b^N$ and pump up or down.

Example: $L = \{w \cdot \beta \cdot w \mid w, \beta \in \{a, b\}^* \text{ and Length(w)} > 0\}$.

- Recall that $L' = \{w \cdot \beta \cdot w^R \mid w, \beta \in \{a, b\}^* \text{ and Length(w)} > 0\}$ is a regular language.
- Is L context free for similar reasons?
- No, it is not.
- Think about pumping the string $\alpha = a^N b^N a^N b^N$.
- In all cases, it can be pumped out of the language.

Further Examples of the Pumping Lemma for CFLs — 3

Example: $L = \{a^{n^2} \mid n \in \mathbb{N}\}$.

- Let N be the constant guaranteed for L by the Pumping Lemma for this language.
- Choose $\alpha = a^{(N+1)^2}$.
- Any decomposition $\alpha = \alpha_1 \cdot \alpha_2 \cdot \alpha_3 \cdot \alpha_4 \cdot \alpha_5$ satisfying the conditions of the pumping lemma must look like

$$\alpha_1 = a^{n_1},\ \alpha_2 = a^{n_2},\ \alpha_3 = a^{n_3},\ \alpha_4 = a^{n_4},\ \alpha_5 = a^{n_5},$$

$$\text{with } n_1 + n_2 + n_3 + n_4 + n_5 = (N+1)^2$$

$$\text{and } n_2 + n_3 + n_4 \leq N \text{ and } n_2 + n_4 > 0.$$

- Pump down to $\alpha_1 \cdot \alpha_2^0 \cdot \alpha_3 \cdot \alpha_4^0 \cdot \alpha_5 = \alpha_1 \cdot \alpha_3 \cdot \alpha_5 = a^{n_1+n_3+n_5}$.
- $N^2 < n^2 + N + 1 = (N+1)^2 - N \leq n_1 + n_3 + n_5 < (N+1)^2$.
- Thus $n_1 + n_3 + n_5$ is not the square of any integer.
- Hence L is not a CFL.

A Simplifying Result

Theorem: Let Σ be an alphabet consisting of a single letter (*e.g.*, $\Sigma = \{a\}$). Then if $L \subseteq \Sigma^*$ is a CFL, it is also a regular language.

- In other words, for a single-letter alphabet, the context-free and regular languages are the same.

Proof: Consult a more advanced textbook. $\square$

Application: To show that

$$L = \{a^{n^2} \mid n \in \mathbb{N}\}$$

is not context free, it suffices to show that it is not regular.

- Thus, the (simpler) pumping lemma for regular languages may be applied.

Are Programming Languages Context Free?

- Consider the following infinite sequence of perfectly legal C programs:

```
main(){int ab;ab=0;}
main(){int aabb;aabb=0;}
⋮
```

  $\texttt{main()\{int a}^n\texttt{b}^n\texttt{;a}^n\texttt{b}^n\texttt{=0\}}$

  ⋮

- Suppose that C is context free.
- Let N be the constant guaranteed by the pumping lemma for CFLs.
- It is easy to pump $\texttt{main()\{int a}^N\texttt{b}^N\texttt{;a}^N\texttt{b}^N\texttt{=0\}}$ out of C.
- This argument requires that arbitrarily long identifiers be allowed.
- Otherwise, a really ugly (and impractical) grammar could be used to generate all of the finite number of possibilities.
- So, it is a reasonable model of reality.

Are Programming Languages Context Free? — 2

Question: Does this mean that CFLs are not useful for the specification of programming languages?

Answer: On the contrary, they are the standard means of such specification.

- The solution to the above issue is to:
  - *overgenerate* the language (by allowing more than just the legal programs) with the CFL, and then
  - to use other means to filter out the illegal programs.
- In the specific case illustrated above, this means that the CFL will not rule out programs with undeclared variables.
- This form of checking must be done in other ways.
- This issue will be discussed in more detail on a following set of slides.

Basic Closure Properties of Context-Free Languages

Algorithm: Let $G_1 = (V_1, \Sigma, S_1, P_1)$ and $G_2 = (V_2, \Sigma, S_2, P_2)$ be CFGs. Construct a CFG $G_{1+2} = (V_{1+2}, \Sigma, S_{1+2}, P_{1+2})$ with

$$\mathcal{L}(G_{1+2}) = \mathcal{L}(G_1) \cup \mathcal{L}(G_2).$$

Construction: Without loss of generality, assume that $V_1 \cap V_2 = \emptyset$. Rename variables if necessary. Define $P_{1+2} = P_1 \cup P_2 \cup \{S_{1+2} \to S_1 \mid S_2\}$. $\square$

Algorithm: Let $G_1 = (V_1, \Sigma, S_1, P_1)$ and $G_2 = (V_2, \Sigma, S_2, P_2)$ be CFGs. Construct a CFG $G_{1\cdot 2} = (V_{1\cdot 2}, \Sigma, S_{1\cdot 2}, P_{1\cdot 2})$ with

$$\mathcal{L}(G_{1\cdot 2}) = \mathcal{L}(G_1) \cdot \mathcal{L}(G_2).$$

Construction: Without loss of generality, assume that $V_1 \cap V_2 = \emptyset$. Rename variables if necessary. Define $P_{1\cdot 2} = P_1 \cup P_2 \cup \{S_{1\cdot 2} \to S_1 S_2\}$. $\square$

Algorithm: Let $G = (V, \Sigma, S, P)$ be a CFG. Construct a CFG $G_* = (V_*, \Sigma, S_*, P_*)$ with $\mathcal{L}(G_*) = (\mathcal{L}(G))^*$.

Construction: Just let $V_* = V \cup \{S_*\}$ with $S_* \notin V$, and Define

$$P_* = P \cup \{S_* \to S S_* \mid \lambda\}. \square$$

Basic Closure Properties of Context-Free Languages — 2

Theorem: The class of CFLs over a given finite alphabet Σ is closed under union, intersection, and Kleene star. $\square$

- However. ...

Theorem: The class of DCFLs (deterministic CFLs) is NOT closed under any of these operations.

Proof: Consult a more advanced textbook. $\square$

Example: Let $\Sigma = \{a, b, c\}$.

$L_1 = \{a^i b^j c^k \mid i = j\}$ and $L_2 = \{a^i b^j c^k \mid j = k\}$.

It is easy to show that both L_1 and L_2 are DCFLs.

- However, $L_1 \cup L_2 = \{a^i b^j c^k \mid i = j \text{ or } j = k\}$ is a standard example of an inherently ambiguous (and hence nondeterministic) CFL.

Basic Non-Closure Properties of Context-Free Languages

Theorem: Let Σ be a finite alphabet consisting of at least two distinct elements. Then the class of context-free languages over Σ is not closed under intersection or complement.

Proof: Define $L_1 = \{a^i b^j a^k \mid i = j\}$ and $L_2 = \{a^i b^j a^k \mid j = k\}$. It is easy to show that L_1 and L_2 are CFLs. Yet
$L_1 \cap L_2 = \{a^i b^j a^k \mid i = j = k\} = \{a^k b^k a^k \mid k \in \mathbb{N}\}$ is not a CFL, as is easily established using the pumping lemma.
Since $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$, and since the class of CFLs is closed under union, it follows that it cannot be closed under complement, since $L_1 \cap L_2$ is not a CFL. $\square$

Fact: For $\mathrm{Card}(\Sigma) \geq 2$, the class of deterministic CFLs is closed under complement, but not intersection $\square$

The Intersection of a CFL and a Regular Language

Algorithm: Let Σ be a finite alphabet, and let

$M_1 = (Q_1, \Sigma, \Gamma_1, \delta_1, q_{0_1}, z_1, F_1)$ be an NPDA.

and $M_2 = (Q_2, \Sigma, \delta_2, q_{0_2}, F_2)$ be an NFA

Construct an NPDA $M = (Q, \Sigma, \Gamma, \delta, q_0, z, F)$ with

$$\mathcal{L}_A(M) = \mathcal{L}_A(M_1) \cap \mathcal{L}(M_2).$$

Construction: The idea is to build a "product" machine in which the NPDA and NFA run in parallel, on the same input elements.

- Define
  - $Q = Q_1 \times Q_2$
  - $F = F_1 \times F_2$
  - $q_0 = (q_{0_1}, q_{0_2})$
  - $\delta : Q \times \Sigma^* \cup \{\lambda\} \times \Gamma \to 2^{Q \times \Gamma^*}_{\text{finite}}$ by

$$((q_1, q_2), x, y) \mapsto \{((q_1', q_2'), \beta) \mid (q_1', \beta) \in \delta_1(q_1, x, y) \text{ and } q_2' \in \delta_2^*(q_2, x)\}$$

for all $(q_1, q_2) \in Q_1 \times Q_2$, $x \in \Sigma^* \cup \{\lambda\}$, $y \in \Gamma$. $\square$

Theorem: Let L_1 be a CFL and let L_2 be a regular language, over the same alphabet. Then $L_1 \cap L_2$ is also a CFL. $\square$