1 Introduction

Parenthesis Matching

Problem
Describe the set of arithmetic expressions with correctly matched parenthesis.

Arithmetic expressions with correctly matched parenthesis cannot be described by a regular expression

- Let L be the language of correct expressions
- Suppose h maps number and variables to ϵ, and opening parenthesis to 0 and closing parenthesis to 1 then $h(L) \subseteq \{0,1\}^*$
- $h(L) \cap 0^*1^* = \{0^n1^n \mid n \geq 0\}$ which is not regular.

This is an example of a *context-free language*, that we study.

Parenthesis Matching
Inductive Definition

Ignoring numbers and variables, and focussing only on parenthesis, correctly matched expressions can be defined as

- The ϵ is a valid expression
- A valid string ($\neq \epsilon$) must either be
  - The concatenation of two correctly matched expressions, or
  - It must begin with (and end with) and moreover, once the first and last symbols are removed, the resulting string must correspond to a valid expression.

Parenthesis Matching
Grammar

Taking E to be the set of correct expressions, the inductive definition can be succinctly written as

$$E \to \epsilon$$
$$E \to EE$$
$$E \to (E)$$

English Sentences

English sentences can be described as

$$\begin{aligned}
&\langle S\rangle \to \langle NP\rangle\langle VP\rangle\\
&\langle NP\rangle \to \langle CN\rangle \mid \langle CN\rangle\langle PP\rangle\\
&\langle VP\rangle \to \langle CV\rangle \mid \langle CV\rangle\langle PP\rangle\\
&\langle PP\rangle \to \langle P\rangle\langle CN\rangle\\
&\langle CN\rangle \to \langle A\rangle\langle N\rangle\\
&\langle CV\rangle \to \langle V\rangle \mid \langle V\rangle\langle NP\rangle\\
&\langle A\rangle \to \text{a} \mid \text{the}\\
&\langle N\rangle \to \text{boy} \mid \text{girl} \mid \text{flower}\\
&\langle V\rangle \to \text{touches} \mid \text{likes} \mid \text{sees}\\
&\langle P\rangle \to \text{with}
\end{aligned}$$

English Sentences
Examples

$$\overbrace{\underbrace{\text{a}}_{\text{article}}\ \underbrace{\text{boy}}_{\text{noun}}}^{\text{noun-phrs}}\ \overbrace{\underbrace{\text{sees}}_{\text{verb}}}^{\text{verb-phrs}}$$

$$\overbrace{\underbrace{\text{the}}_{\text{article}}\ \underbrace{\text{boy}}_{\text{noun}}}^{\text{noun-phrs}}\ \overbrace{\underbrace{\text{sees}}_{\text{verb}}\ \underbrace{\text{a flower}}_{\text{noun-phrs}}}^{\text{verb-phrs}}$$

Applications

Such rules (or grammars) play a key role in

- Parsing programming languages
- Markup Languages like HTML and XML.
- Modelling software

---

2 Formal Definition

2.1 Grammars

Context-Free Grammars

Definition 1. A *context-free grammar* (CFG) is $G = (V, \Sigma, R, S)$ where

- V is a finite set of *variables* also called *nonterminals* or *syntactic categories*. Each variable represents a language.

- Σ is a finite set of symbols, disjoint from V, called *terminals*, that form the strings of the language.
- R is a finite set of *rules* or *productions*. Each production is of the form $A \to \alpha$ where $A \in V$ and $\alpha \in (V \cup \Sigma)^*$
- $S \in V$ is the *start symbol*; it is the variable that represents the language being defined. Other variables represent auxiliary languages that are used to define the language of the start symbol.

Examples

Example 2. Let $G_{\text{par}} = (V, \Sigma, R, S)$ be

- $V = \{E\}$
- $\Sigma = \{(,)\}$
- $R = \{E \to \epsilon, E \to EE, E \to (E)\}$
- $S = E$

Example 3. A string w is a *palindrome* if $w = w^R$.

$G_{\text{pal}} = (\{S\}, \{0,1\}, R, S)$ defines palindromes over $\{0,1\}$, where R is

$$\begin{aligned} S &\to \epsilon \\ S &\to 0 \\ S &\to 1 \\ S &\to 0S0 \\ S &\to 1S1 \end{aligned}$$

Or more briefly, $R = \{S \to \epsilon \mid 0 \mid 1 \mid 0S0 \mid 1S1\}$

Example 4. Consider the language of all arithmetic expressions (E) built out of integers (N) and identifiers (I), using only $+$ and $*$

$G_{\text{exp}} = (\{E, I, N\}, \{a, b, 0, 1, (,), +, *, -\}, R, E)$ where R is

$$\begin{aligned} E &\to I \mid N \mid E + E \mid E * E \mid (E) \\ I &\to a \mid b \mid Ia \mid Ib \\ N &\to 0 \mid 1 \mid N0 \mid N1 \mid -N \mid +N \end{aligned}$$

Derivations

Informal Overview

Expand the start symbol using one of its rules. Further expand the resulting string by expanding one of the variables in the string, by the RHS of one of its rules. Repeat until you get a string of terminals.

For the grammar $G_{\text{pal}} = (\{S\}, \{0,1\}, \{S \to \epsilon \mid 0 \mid 1 \mid 0S0 \mid 1S1\}, S)$ we have

$$S \Rightarrow 0S0 \Rightarrow 00S00 \Rightarrow 001S100 \Rightarrow 0010100$$

Definition 5. Let $G = (V, \Sigma, R, S)$ be a CFG. We say $\alpha A \beta \Rightarrow_G \alpha\gamma\beta$, where $\alpha, \beta, \gamma \in (V \cup \Sigma)^*$ and $A \in V$ if $A \to \gamma$ is a rule of G.

We say $\alpha \overset{*}{\Rightarrow}_G \beta$ if either $\alpha = \beta$ or there are $\alpha_0, \alpha_1, \ldots \alpha_n$ such that

$$\alpha = \alpha_0 \Rightarrow_G \alpha_1 \Rightarrow_G \alpha_2 \Rightarrow_G \cdots \Rightarrow_G \alpha_n = \beta$$

Notation
When G is clear from the context, we will write $\Rightarrow$ and $\overset{*}{\Rightarrow}$ instead of $\Rightarrow_G$ and $\overset{*}{\Rightarrow}_G$.

Context-Free Language

Definition 6. The *language of CFG* $G = (V, \Sigma, R, S)$, denoted $\mathbf{L}(G)$ is the collection of strings over the terminals derivable from S using the rules in R. In other words,

$$\mathbf{L}(G) = \{w \in \Sigma^* \mid S \overset{*}{\Rightarrow} w\}$$

Definition 7. A language L is said to be *context-free* if there is a CFG G such that $L = \mathbf{L}(G)$.

2.2 Proving Properties

Palindromes Revisited

Recall, $L_{\text{pal}} = \{w \in \{0,1\}^* \mid w = w^R\}$ is the language of palindromes.

Consider $G_{\text{pal}} = (\{S\}, \{0,1\}, R, S)$ defines palindromes over $\{0,1\}$, where $R = \{S \to \epsilon|0|1|0S0|1S1\}$

Proposition 8. $\mathbf{L}(G_{\text{pal}}) = L_{\text{pal}}$

Proof. To prove the proposition, we need to show that $L_{\text{pal}} \subseteq \mathbf{L}(G_{\text{pal}})$ and $\mathbf{L}(G_{\text{pal}}) \subseteq L_{\text{pal}}$.

$L_{\text{pal}} \subseteq \mathbf{L}(G_{\text{pal}})$**:** Let $w \in L_{\text{pal}}$. We prove that $S \overset{*}{\Rightarrow} w$ by induction on $|w|$.

- *Base Cases:* If $|w| = 0$ or $|w| = 1$ then $w = \epsilon$ or 0 or 1. And $S \to \epsilon \mid 0 \mid 1$.
- *Induction Step:* If $|w| \geq 2$ and $w = w^R$ then it must begin and with the same symbol. Let $w = 0x0$. Now, $w^R = 0x^R0 = w = 0x0$; thus, $x^R = x$. By induction hypothesis, $S \overset{*}{\Rightarrow} x$. Hence $S \Rightarrow 0S0 \overset{*}{\Rightarrow} 0x0$. If $w = 1x1$ the argument is similar. □

$\mathbf{L}(G_{\text{pal}}) \subseteq L_{\text{pal}}$**:** Let $w \in \mathbf{L}(G)$, i.e., $S \overset{*}{\Rightarrow} w$. We will show $w \in L_{\text{pal}}$ by induction on the number of derivation steps.

- *Base Case:* If the derivation has only one step then the derivation must be $S \Rightarrow \epsilon$, $S \Rightarrow 0$ or $S \Rightarrow 1$. Thus $w = \epsilon$ or 0 or 1 and is in L_{Pal}.
- *Induction Step:* Consider an $(n+1)$-step derivation of w. It must be of the form $S \Rightarrow 0S0 \overset{*}{\Rightarrow} 0x0 = w$ or $S \Rightarrow 1S1 \overset{*}{\Rightarrow} 1x1 = w$. In either case $S \overset{*}{\Rightarrow} x$ in n-steps. Hence $x \in L_{\text{Pal}}$ and so $w = w^R$.

Proving correctness
Example II

Let $L_{j\geq 2i} = \{a^i b^j \mid j \geq 2i\}$. Consider the following grammar $G_{i\geq 2i} = (\{S, B\}, \{a, b\}, R, S)$ where

$$R = \{S \to aSbb \mid B;\ B \to \epsilon \mid bB\}$$

Proposition 9. $\mathbf{L}(G_{j\geq 2i}) = L_{j\geq 2i}$.

Proof. Like in the previous example, to prove correctness, we will show that $\mathbf{L}(G_{j\geq 2i}) \supseteq L_{j\geq 2i}$ and $\mathbf{L}(G_{j\geq 2i}) \subseteq L_{j\geq 2i}$. And as in the previous example, the easiest way to do this, is to prove the first part by induction on the length of the string $w \in L_{j\geq 2i}$, and the second part by induction on the length of the derivation. However, in order for these proofs to work out, we also need to strengthen the claim: the correctness only states that the set of strings derivable from S is exactly those that belong to the set $L_{j\geq 2i}$, but for the induction proof to go through, we also need to describe the set of strings that are derivable from B. This is very similar to the way induction proofs needed to be strengthened for DFAs. In this case, the strengthened statement we will establish is as follows:

$$\begin{aligned} \forall w \in \{a, b\}^*. \quad & S \stackrel{*}{\Rightarrow} w \text{ iff } w \in L_{j\geq 2i} \text{ and} \\ & B \stackrel{*}{\Rightarrow} w \text{ iff } w \in \mathbf{L}(b^*) \end{aligned}$$

$\Leftarrow$**:** Need to show that

$$\begin{aligned} \forall w \in \{a, b\}^*. \quad & w \in L_{j\geq 2i} \ \Rightarrow\ S \stackrel{*}{\Rightarrow} w \text{ and} \\ & w \in \mathbf{L}(b^*) \ \Rightarrow\ B \stackrel{*}{\Rightarrow} w \end{aligned}$$

We will prove this by induction on $|w|$.

- **Base Case:** Consider $w = \epsilon$. Now, we have derivations, $S \Rightarrow B \Rightarrow \epsilon$ and $B \Rightarrow \epsilon$, which establish the base case.
- **Ind. Hyp.:** Assume that the claim holds for all w such that $|w| < n$.
- **Ind. Step:** Consider $w = b^n \in \mathbf{L}(b^*)$, where $n > 0$. Then, $w = bu$, where $u = b^{n-1} \in \mathbf{L}(b^*)$. By induction hypothesis, we have $B \stackrel{*}{\Rightarrow} b^{n-1} = u$. Thus, we have a derivation, $B \Rightarrow bB \stackrel{*}{\Rightarrow} bu = w$. This establishes one part of the induction step.

  Consider $w = a^i b^j \in L_{j\geq 2i}$, where $j \geq 2i$ and $i + j = n$. Now, if $i = 0$, then $w = b^j$, where $j > 0$. In other words, $w \in \mathbf{L}(b^*)$. Now by the first part of the induction step, we have $B \stackrel{*}{\Rightarrow} b^j = w$. And so we have a derivation, $S \Rightarrow B \stackrel{*}{\Rightarrow} b^j = w$. On the other hand, if $i > 0$, then $w = aubb$, where $u = a^{i-1}b^{j-2}$. Now if $j \geq 2i$ then $j - 2 \geq 2i - 2 = 2(i - 1)$. Thus, $u \in L_{j\geq 2i}$. By induction hypothesis, we have $S \stackrel{*}{\Rightarrow} u$. Hence, we have a derivation, $S \Rightarrow aSbb \stackrel{*}{\Rightarrow} aubb = w$. This finished the induction step.

$\Rightarrow$**:** Need to show that

$$\begin{aligned} \forall w \in \{a, b\}^*. \quad & S \stackrel{*}{\Rightarrow} w \ \Rightarrow\ w \in L_{j\geq 2i} \text{ and} \\ & B \stackrel{*}{\Rightarrow} w \ \Rightarrow\ w \in \mathbf{L}(b^*) \end{aligned}$$

We will prove this by induction on the number of derivation steps for w.

- **Base Case:** Suppose $B \Rightarrow w$. Since the only rule that has only terminals on the right-hand side is $B \to \epsilon$, we have $w = \epsilon \in \mathbf{L}(b^*)$. This establishes one part of the base case. On the other hand, there is no rule with left-hand side S and right-hand side only terminals. Hence, no string of terminals can be derived from S in 1 step, and so the base case for this condition holds vaccuously.
- **Ind. Hyp.:** We assume that if $B \stackrel{*}{\Rightarrow} w$ in $< n$ steps then $w \in \mathbf{L}(b^*)$, and if $S \stackrel{*}{\Rightarrow} w$ in $< n$ steps then $w \in L_{j \geq 2i}$.
- **Ind. Step:** Let $B \stackrel{*}{\Rightarrow} w$ in $n (> 1)$ steps. Because of the rules, we know that this derivation is going to be of the form, $B \Rightarrow bB \stackrel{*}{\Rightarrow} bu = w$, where $B \stackrel{*}{\Rightarrow} u$ in $n-1$ steps. By induction hypothesis, $u \in \mathbf{L}(b^*)$, and thus, $w = bu \in \mathbf{L}(b^*)$. This finishes part one of the induction step.

  Let $S \stackrel{*}{\Rightarrow} w$ in n steps. Then there are two possibilities for the form of derivation. First case is, $S \Rightarrow B \stackrel{*}{\Rightarrow} w$. Since $B \stackrel{*}{\Rightarrow} w$ in $n-1$ steps, it means that (from induction hypothesis) $w \in \mathbf{L}(b^*) \subseteq L_{j \geq 2i}$. The second case is that $S \Rightarrow aSbb \stackrel{*}{\Rightarrow} aubb = w$, where $S \stackrel{*}{\Rightarrow} u$ in $n-1$ steps. By induction hypothesis, $u \in L_{j \geq 2i}$ and so $u = a^i b^j$, for some i, j such that $j \geq 2i$. Now $w = aubb = a(a^i b^j)bb = a^{i+1} b^{j+2}$. Since $j \geq 2i$, we have $j + 2 \geq 2i + 2 = 2(i+1)$. Thus, $w \in L_{j \geq 2i}$ which establishes the second part of the induction step.

□

2.3 Parse Trees

Parse Trees

For CFG $G = (V, \Sigma, R, S)$, a *parse tree* (or *derivation tree*) of G is a tree satisfying the following conditions:

- Each interior node is labeled by a variable in V
- Each leaf is labeled by either a variable, a terminal or ϵ; a leaf labeled by ϵ must be the only child of its parent.
- If an interior node labeled by A with children labeled by $X_1, X_2, \ldots X_k$ (from the left), then $A \to X_1 X_2 \cdots X_k$ must be a rule.

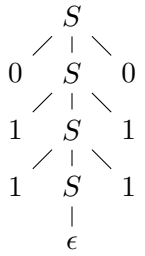


Figure 1: Example Parse Tree with yield 011110

Yield of a parse tree is the concatenation of leaf labels (left–right)

Parse Trees and Derivations

Proposition 10. *Let $G = (V, \Sigma, R, S)$ be a CFG. For any $A \in V$ and $\alpha \in (V \cup \Sigma)^*$, $A \stackrel{*}{\Rightarrow} \alpha$ iff there is a parse tree with root labeled A and whose yield is α.*

Proof. ($\Rightarrow$)*:* Proof by induction on the number of steps in the derivation.

- *Base Case:* If $A \Rightarrow \alpha$ then $A \to \alpha$ is a rule in G. There is a tree of height 1, with root A and leaves the symbols in α.

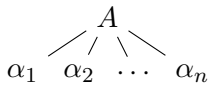


Figure 2: Parse Tree for Base Case

- *Induction Step:* Let $A \stackrel{*}{\Rightarrow} \alpha$ in $k+1$ steps.
- Then $A \stackrel{*}{\Rightarrow} \alpha_1 X \alpha_2 \Rightarrow \alpha_1 \gamma \alpha_2 = \alpha$, where $X \to X_1 \cdots X_n = \gamma$ is a rule
- By ind. hyp., there is a tree with root A and yield $\alpha_1 X \alpha_2$.
- Add leaves $X_1, \ldots X_n$ and make them children of X. New tree is a parse tree with desired yield.

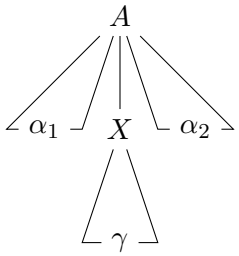


Figure 3: Parse Tree for Induction Step

($\Leftarrow$)*:* Assume that there is a parse tree with root A and yield α. Need to show that $A \stackrel{*}{\Rightarrow} \alpha$. Proof by induction on the number of internal nodes in the tree.

- *Base Case:* If tree has only one internal node, then it has the form as in picture
- Then, $\alpha = X_1 \cdots X_n$ and $A \to \alpha$ is a rule. Thus, $A \stackrel{*}{\Rightarrow} \alpha$.

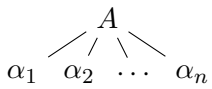


Figure 4: Parse Tree with one internal node

Induction Step: Suppose α is the yield of a tree with $k+1$ interior nodes. Let $X_1, X_2, \ldots X_n$ be the children of the root ordered from the left. Not all X_i are leaves, and $A \to X_1 X_2 \cdots X_n$ must be a rule.

- Let α_i be the yield of the tree rooted at X_i; so X_i is a leaf $\alpha_i = X_i$
- Now if $j < i$ then all the descendents of X_j are to the left of the descendents of X_i. So $\alpha = \alpha_1 \alpha_2 \cdots \alpha_n$.
- Each subtree rooted at X_i has at most k internal nodes. So if X_i is a leaf $X_i \stackrel{*}{\Rightarrow} \alpha_i$ and if X_i is not a leaf then $X_i \stackrel{*}{\Rightarrow} \alpha_i$ (ind. hyp.).
- Thus $A \Rightarrow X_1 X_2 \cdots X_n \stackrel{*}{\Rightarrow} \alpha_1 X_2 \cdots X_n \stackrel{*}{\Rightarrow} \alpha_1 \alpha_2 \cdots X_n \stackrel{*}{\Rightarrow} \alpha_1 \cdots \alpha_n = \alpha$ □

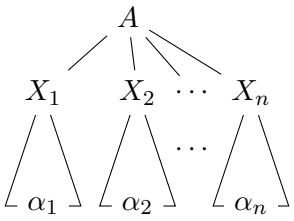


Figure 5: Tree with $k+1$ internal nodes

Recap ...

For a CFG G with variable A the following are equivalent

1. $A \stackrel{*}{\Rightarrow} w$
2. There is a parse tree with root A and yield w

Context-free-ness
CFGs have the property that if $X \stackrel{*}{\Rightarrow} \gamma$ then $\alpha X \beta \stackrel{*}{\Rightarrow} \alpha \gamma \beta$

---

3 Ambiguity

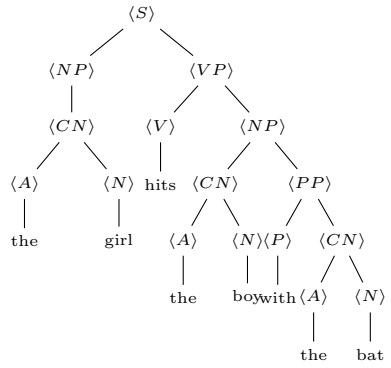
3.1 The Concept

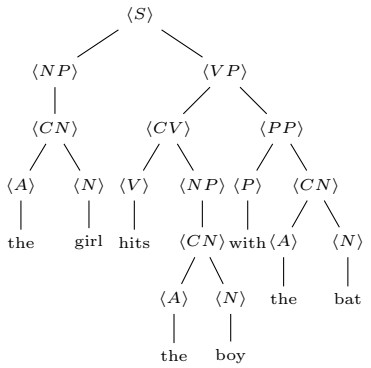
Ambiguity through examples

Example 11. English sentences can be described as

$$\begin{aligned}
&\langle S\rangle \to \langle NP\rangle\langle VP\rangle\\
&\langle NP\rangle \to \langle CN\rangle \mid \langle CN\rangle\langle PP\rangle\\
&\langle VP\rangle \to \langle CV\rangle \mid \langle CV\rangle\langle PP\rangle\\
&\langle PP\rangle \to \langle P\rangle\langle CN\rangle\\
&\langle CN\rangle \to \langle A\rangle\langle N\rangle\\
&\langle CV\rangle \to \langle V\rangle \mid \langle V\rangle\langle NP\rangle\\
&\langle A\rangle \to \text{a} \mid \text{the}\\
&\langle N\rangle \to \text{boy} \mid \text{girl} \mid \text{bat}\\
&\langle V\rangle \to \text{hits} \mid \text{likes} \mid \text{sees}\\
&\langle P\rangle \to \text{with}
\end{aligned}$$

The sentence "the girl hits the boy with the bat" has the following parse trees



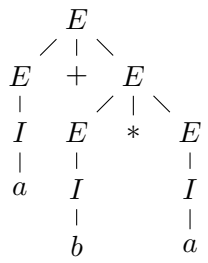


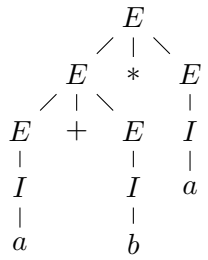
Example 12. Consider the language of all arithmetic expressions (E) built out of integers (N) and identifiers (I), using only + and $*$

$G_{\text{exp}} = (\{E, I, N\}, \{a, b, 0, 1, (,), +, *, -\}, R, E)$ where R is

$$E \to I \mid N \mid -N \mid E + E \mid E * E \mid (E)$$
$$I \to a \mid b \mid Ia \mid Ib$$
$$N \to 0 \mid 1 \mid N0 \mid N1$$

The parse trees for expression $a + b * a$ in the grammar G_{exp} is





Ambiguity

Definition 13. A grammar $G = (V, \Sigma, R, S)$ is said to be *ambiguous* if there is $w \in \Sigma^*$ for which there are two different parse trees.

Warning!
Existence of two derivations for a string does not mean the grammar is ambiguous!

3.2 Removing Ambiguity

Removing Ambiguity

Ambiguity maybe removed either by

- Using the semantics to change the rules. For example, if we knew who had the bat (the girl or the boy) from the context, we would know which is the right interpretation.
- Adding precedence to operators. For example, $*$ binds more tightly than $+$, or "else" binds with the innermost "if".

An Example

Recall, G_{exp} has the following rules

$$E \to I \mid N \mid -N \mid E + E \mid E * E \mid (E)$$
$$I \to a \mid b \mid Ia \mid Ib$$
$$N \to 0 \mid 1 \mid N0 \mid N1$$

New CFG G'_{exp} has the rules

$$I \to a \mid b \mid Ia \mid Ib$$
$$N \to 0 \mid 1 \mid N0 \mid N1$$
$$F \to I \mid N \mid -N \mid (E)$$
$$T \to F \mid T * F$$
$$E \to T \mid E + T$$

Ambiguity: Computational Problems

Removing Ambiguity
Problem: Given CFG G, find CFG G' such that $\mathbf{L}(G) = \mathbf{L}(G')$ and G' is unambiguous.
There is no algorithm that can solve the above problem!

Deciding Ambiguity
Problem: Given CFG G, determine if G is ambiguous.
There is no algorithm that can solve the above problem!

Inherently Ambiguous Languages

Problem: Is it the case that for every CFG G, there is a grammar G' such that $\mathbf{L}(G) = \mathbf{L}(G')$ and G' is unambiguous, *even if G' cannot be constructed algorithmically*?
No! There are context-free languages L such that every grammar for L is ambiguous.

Definition 14. A context-free language L is said to be *inherently ambiguous* if every grammar G for L is ambiguous.

Inherently Ambiguous Languages
An Example

Consider

$$L = \{a^i b^j c^k \mid i = j \text{ or } j = k\}$$

One can show that any CFG G for L will have two parse trees on $a^n b^n c^n$, for all but finitely many values of n

- One that checks that number of a's = number of b's
- Another that checks that number of b's = number of c's
