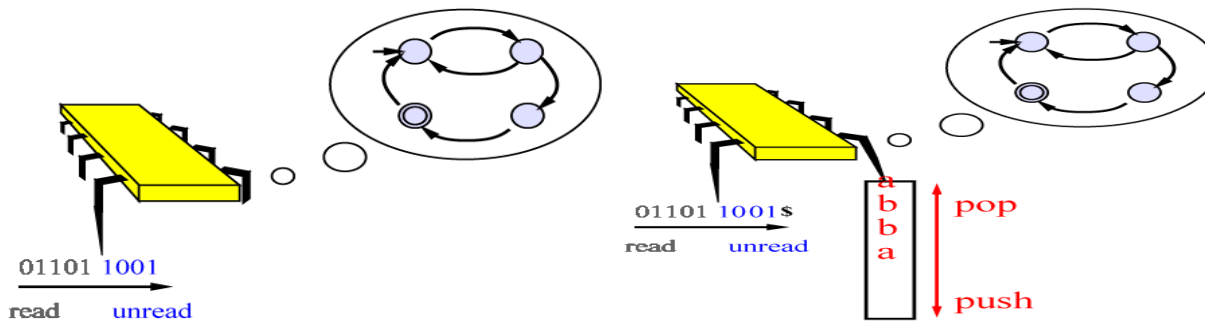


Computational Models - Lectures 4

- **Context Free** Grammars/Languages (CFG/CFL)
- Algorithmic issues for CFL
- **Pumping Lemma** for context free languages

Next week:

- Push Down Automata (**PDA**)
- **Equivalence** of CFGs and PDAs



- Sipser's book, 2.1, 2.2 & 2.3

Short Overview of the Course

So far we saw

- finite automata,
- regular languages,
- regular expressions,
- Myhill-Nerode theorem
- pumping lemma for regular languages.

We now introduce stronger machines and languages with more expressive power:

- pushdown automata,
- context-free languages,
- context-free grammars,
- pumping lemma for context-free languages.

Context Free Grammars (CFG)

An example of a context free grammar, G_1 :

- $A \rightarrow 0A1$
- $A \rightarrow B$
- $B \rightarrow \#$

Terminology:

- Each line is a **substitution rule** or **production**.
- Each rule has the form: **symbol** \rightarrow **string**.
The left-hand symbol is a **variable** (usually upper-case).
- A **string** consists of **variables** and **terminals**.
- One variable is the **start variable** (**lhs** of top rule).

Rules for Generating Strings

- Write down the start variable.
- Pick a variable written down in current string and a derivation that starts with that variable.
- Replace that variable with right-hand side of that derivation.
- Repeat until no variables remain.
- Return final string (concatenation of terminals).

Process is inherently **non deterministic**.

Example

Grammar G_1 :

• $A \rightarrow 0A1$

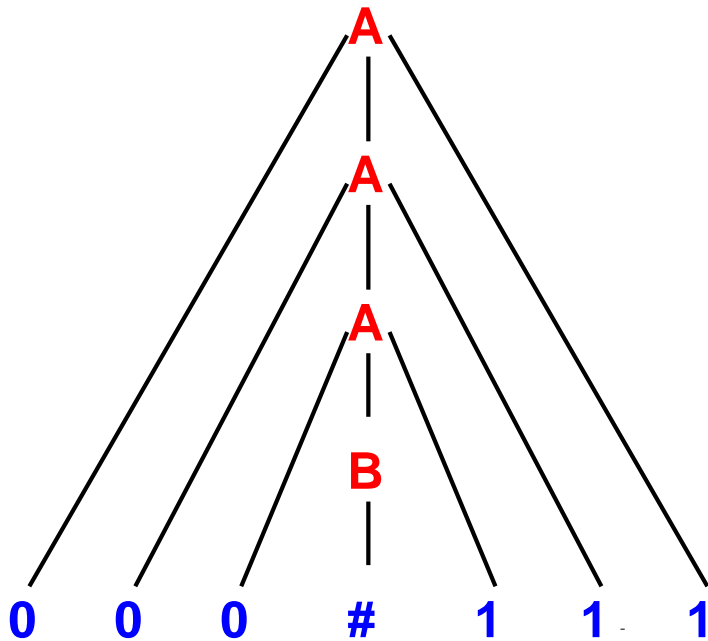
• $A \rightarrow B$

• $B \rightarrow \#$

Derivation with G_1 :

$$\begin{aligned} A &\Rightarrow 0A1 \\ &\Rightarrow 00A11 \\ &\Rightarrow 000A111 \\ &\Rightarrow 000B111 \\ &\Rightarrow 000\#111 \end{aligned}$$

A Parse Tree



Indifferent to derivation **order**.

Question: What strings can be generated in this way from the grammar G_1 ?

Answer: Exactly those of the form $0^n \# 1^n$ ($n \geq 0$).

Context-Free Languages (CFL)

The language generated in this way is called the **language of the grammar**.

For example, $L(G_1)$ is $\{0^n \# 1^n \mid n \geq 0\}$.

Any language generated by a context-free grammar is called a **context-free language**.

A Useful Abbreviation

Rules with same variable on left hand side

$$A \rightarrow 0A1$$

$$A \rightarrow B$$

are written as:

$$A \rightarrow 0A1 \mid B$$

English-like Sentences

A grammar G_2 to describe a few English sentences:

< SENTENCE > \rightarrow < NOUN-PHRASE > < VERB >

< NOUN-PHRASE > \rightarrow < ARTICLE > < NOUN >

< NOUN > \rightarrow boy | girl | flower

< ARTICLE > \rightarrow a | the

< VERB > \rightarrow touches | likes | sees

Deriving English-like Sentences

A specific derivation in G_2 :

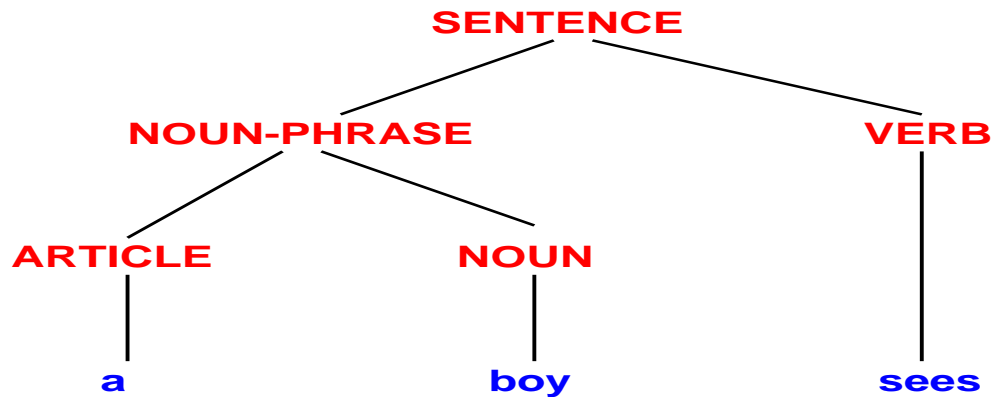
< SENTENCE > \Rightarrow < NOUN-PHRASE > < VERB >
 \Rightarrow < ARTICLE > < NOUN > < VERB >
 \Rightarrow a < NOUN > < VERB >
 \Rightarrow a boy < VERB >
 \Rightarrow a boy sees

More strings generated by G_2 :

a flower sees
the girl touches

Derivation and Parse Tree

< SENTENCE > ⇒ < NOUN-PHRASE >< VERB >
⇒ < ARTICLE >< NOUN >< VERB >
⇒ a < NOUN >< VERB >
⇒ a boy < VERB >
⇒ a boy sees



Formal Definitions

A context-free grammar is a 4-tuple (V, Σ, R, S) where

- V is a finite set of variables,
- Σ is a finite set of terminals,
- R is a finite set of rules: each rule is a variable and a finite string of variables and terminals.
- S is the start symbol.

Formal Definitions (2)

- If u and v are strings of variables and terminals,
- and $A \rightarrow w$ is a rule of the grammar, then
- we say uAv **yields** uww , written $uAv \Rightarrow uww$.

We write $u \xRightarrow{*} v$ if $u = v$ or

$$u \Rightarrow u_1 \Rightarrow \dots \Rightarrow u_k \Rightarrow v.$$

for some sequence u_1, u_2, \dots, u_k .

Definition: The language of the grammar is

$$\left\{ w \in \Sigma^* \mid S \xRightarrow{*} w \right\} .$$

Example 1

$G_4 = (\{S\}, \{a, b\}, R, S)$.

R (Rules): $S \rightarrow aSb \mid SS \mid \varepsilon$.

Some words in the language: $aabb$, $aababb$.

Q.: But what **is** this language?

Hint: Think of parentheses.

Example 2

$G_5 = (\{S\}, \{a, b\}, R, S)$.

R (Rules): $S \rightarrow aSa \mid bSb \mid \varepsilon$

Some words in the language: $abba$, $aabaabaa$.

Q.: But what **is** this language?

$L(G_5) = \{ww^R \mid w \in \{a, b\}^*\}$

Example 3

$G_6 = (\{S, A, B\}, \{a, b\}, R, S)$.

R (Rules):

$S \rightarrow aB \mid bA \mid \varepsilon$

$A \rightarrow a \mid aS \mid bAA$

$B \rightarrow b \mid bS \mid aBB$

Some words in the language: *aababb*, *baabba*.

Q.: But what **is** this language?

$L(G_6) = \{w \in \{a, b\}^* \mid \#_a(w) = \#_b(w)\}$

Proving $L(G_6) = \{w \in \{a, b\}^* \mid \#_a(w) = \#_b(w)\}$

- What do we need to show?!
- (1) If w generated by G_6 then $\#_a(w) = \#_b(w)$.
- (2) If $\#_a(w) = \#_b(w)$ then w generated by G_6 .
- **Claim:** If $S \xrightarrow{*} \alpha$ then $\#_a(\alpha) + \#_A(\alpha) = \#_b(\alpha) + \#_B(\alpha)$.
- **Claim:** For $w \in \{a, b\}^*$, let $k = \#_a(w) - \#_b(w)$. Then:
 - If $k = 0$, then $S \xrightarrow{*} wS$
 - If $k > 0$, then $S \xrightarrow{*} wB^k$
 - If $k < 0$, then $S \xrightarrow{*} wA^k$

Proof by induction on $|w|$

- Are we done?!

Proving $L(G_6) = \{w \in \{a, b\}^* \mid \#_a(w) = \#_b(w)\}$

- Proof by induction on $|w|$.
- Basis, $w = \epsilon$, then $k = 0$ and $S \xrightarrow{*} \epsilon S = S$.
- Induction step. Let $w = \sigma_1 \dots \sigma_n$. Then $w = w'\sigma$ and $|w'| = n - 1$.
- For example suppose that $\#_a(w') - \#_b(w') = k > 0$ and $\sigma = a$.
- By the induction hypothesis $S \xrightarrow{*} w'B^k$,
- Since $B \Rightarrow aBB$, then $S \xrightarrow{*} wB^{k+1}$
- To complete the proof, need to show for $k = 0$ and $k < 0$, and also for $\sigma = b$.

Designing Context-Free Grammars

No recipe in general, but few rules-of-thumb

- If CFG is the **union** of several CFGs, rename variables (**not terminals**) so they are disjoint, and add new rule $S \rightarrow S_1 \mid S_2 \mid \dots \mid S_i$.
- For languages (like $\{0^n \# 1^n \mid n \geq 0\}$), with **linked** substrings, a rule of form $R \rightarrow uRv$ is helpful to force desired relation between substrings.
- For a regular language, grammar “follows” a DFA for the language (see next slide).

CFG for Regular Languages

DFA: $M = (Q, \Sigma, \delta, q_0, F)$

CFG for $L(M)$:

R_0 is the starting variable.

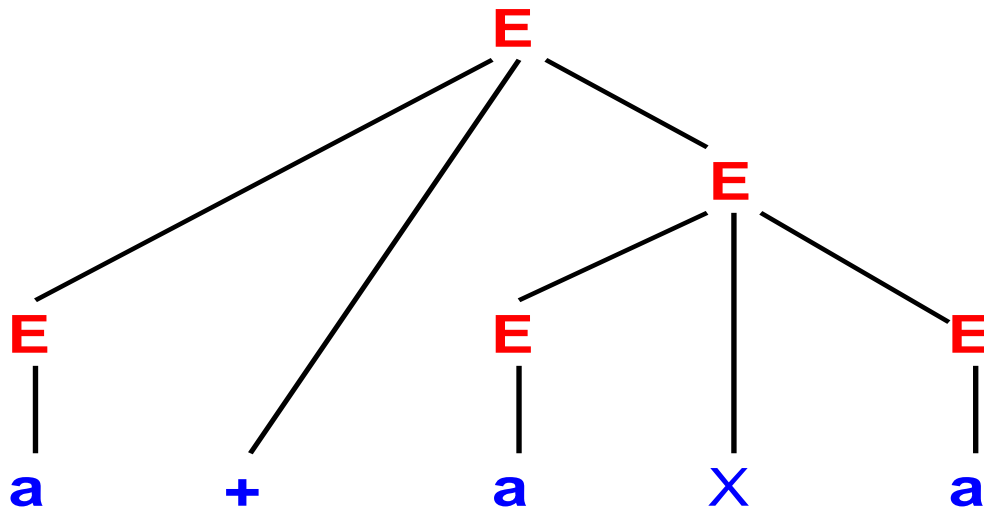
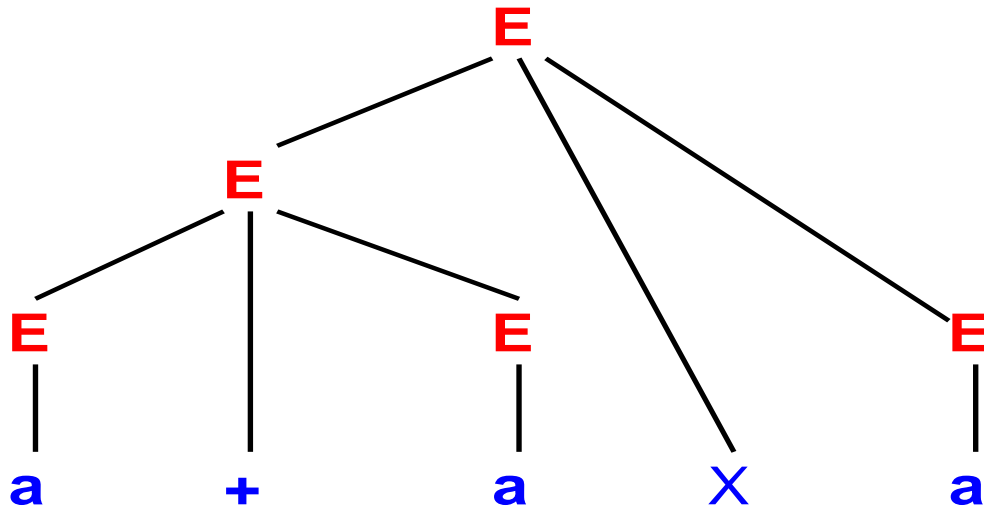
$$R_i \rightarrow aR_j, \text{ if } \delta(q_i, a) = q_j$$

$$R_i \rightarrow \varepsilon, \text{ if } q_i \in F$$

Next class we'll see alternative proof via "Push-Down Automata"

Ambiguity in CFLs

Grammar: $E \rightarrow E + E \mid E \times E \mid (E) \mid a$



Arithmetic Example

Consider (V, Σ, R, E) , where

- $V = \{E, T, F\}$

- $\Sigma = \{a, +, \times, (,)\}$

$$E \rightarrow E + T \mid T$$

Rules: $T \rightarrow T \times F \mid F$

$$F \rightarrow (E) \mid a$$

Some strings generated by the grammar:

$a + a \times a$ and $(a + a) \times a$.

Same arithmetic expressions, just not ambiguous.

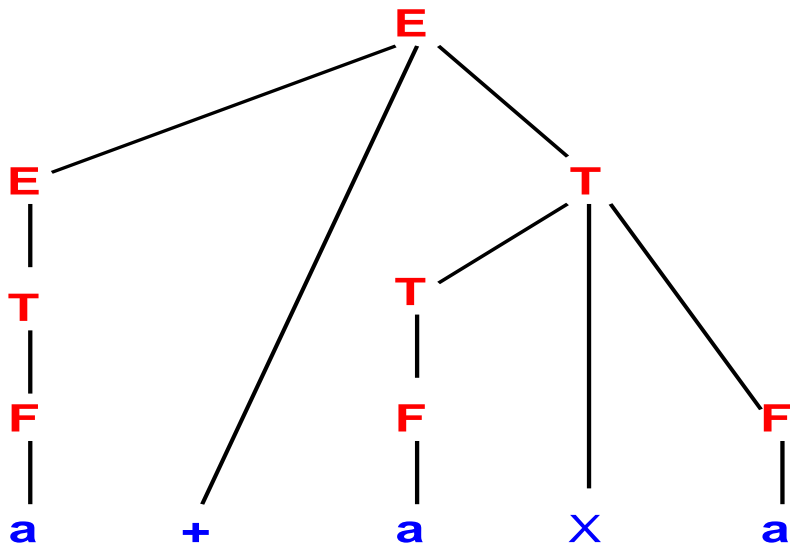
Proof by induction on the string length.

Parse Tree for $a + a \times a$

$$E \rightarrow E + T \mid T$$

$$T \rightarrow T \times F \mid F$$

$$F \rightarrow (E) \mid a$$

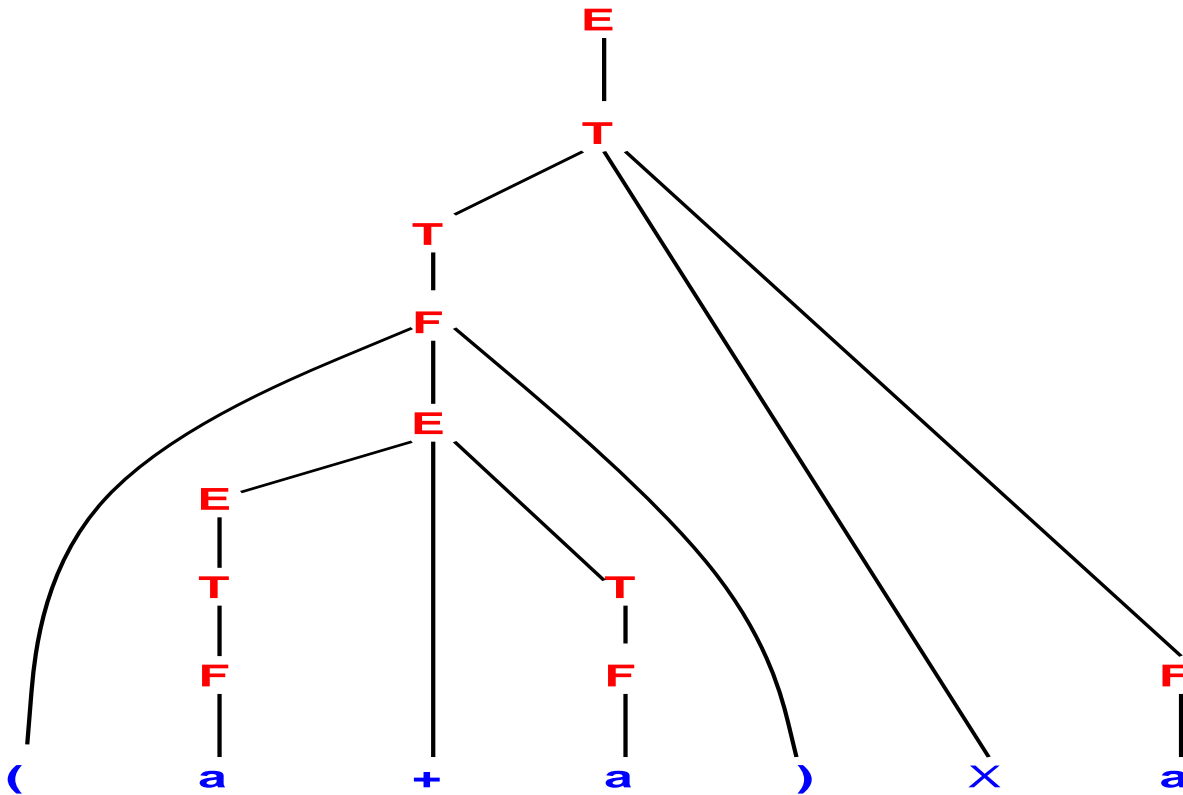


Parse Tree for $(a + a) \times a$

$$E \rightarrow E + T \mid T$$

$$T \rightarrow T \times F \mid F$$

$$F \rightarrow (E) \mid a$$



Ambiguity

We say that a string w is derived **ambiguously** from grammar G if w has two or more parse trees that generate it from G .

Ambiguity is usually not only a syntactic notion but also **semantic**, implying multiple meanings for the same string. Think of $a + a \times a$ from last grammar.

It is **sometime** possible to **eliminate** ambiguity by finding a different context free grammar generating the same language. This is true for the **arithmetic expressions** grammar.

Some languages, e.g. $\{1^i 2^j 3^k \mid i = j \text{ or } j = k\}$ are **inherently** ambiguous. **Proof?**

Checking Membership

Checking Membership in a CFL

Given a CFG, G , and a string w , does G generate w ?

Initial Idea: Design an algorithm that tries **all derivations**.

Problem: If G does **not** generate w , we'll never stop.

Chomsky Normal Form

A simplified, canonical form of context free grammars.
Every rule has the form

$$A \rightarrow BC$$

$$A \rightarrow a$$

$$S \rightarrow \varepsilon$$

- S is the **start** symbol
- B and C are **not** the start symbol
- A is **any** variable, and a is **any** terminal

Simpler to analyze: each derivation adds (at most) a single terminal, S only appears once, ε appears only at the empty word

CNF: Theorem

Theorem: Any context-free language is generated by a context-free grammar in Chomsky Normal Form.

Basic Idea:

- Add new start symbol S_0 .
- Eliminate all ε rules of the form $A \rightarrow \varepsilon$.
- Eliminate all “unit” rules of the form $A \rightarrow B$.
- Patch up rules so that grammar generates the same language.
- Convert remaining “long rules” to proper form.

CNF: Example

$$S \rightarrow ASA \mid aB$$

$$A \rightarrow B \mid S$$

$$B \rightarrow b \mid \varepsilon$$

Transformed into:

$$S_0 \rightarrow AA_1 \mid UB \mid a \mid SA \mid AS$$

$$S \rightarrow AA_1 \mid UB \mid a \mid SA \mid AS$$

$$A \rightarrow b \mid AA_1 \mid UB \mid a \mid SA \mid AS$$

$$A_1 \rightarrow SA$$

$$U \rightarrow a$$

$$B \rightarrow b$$

CNF: Bounded Derivation Length

Lemma: If G is in Chomsky Normal Form, $|w| = n$, and w is generated by G , then w has a derivation of length $2n - 1$.
Proof?

Advantage: Easier to check whether $w \in L(G)$

Checking Membership for Grammars in CNF Form

Given a Chomsky normal form grammar $G = (V, \Sigma, R, S)$, we build a function $\text{Derive}(A, x)$ that returns **TRUE** if $A \xRightarrow{*} x$.

Function $\text{Derive}(A, x)$:

- If $x = \varepsilon$, then if $A \rightarrow \varepsilon \in R$ (i.e., $A = S$) return **TRUE**, otherwise return **FALSE**.
- If $|x| = 1$ then if $A \rightarrow x \in R$ return **TRUE**, otherwise return **FALSE**.
- For each $A \rightarrow BC$ and each partition $x = x_1x_2$,
 - Call $\text{Derive}(B, x_1)$ and $\text{Derive}(C, x_2)$.
 - If both return true, exit and return **TRUE**.
- Return **FALSE**.

Test whether $w \in L(G)$ by calling $\text{Derive}(S, w)$.

- Correctness?

Checking Membership for Grammars in CNF Form, cont

- Procedure Derive can also output a **parse tree** for w
- Have we **critically** used that G is in CNF?

Time Complexity

- What is the time complexity of $\text{Derive}(S, w)$?
- Simple Recursive implementation analysis:
 - each time test $|R|$ rules and n partitions.
 - $T(n) \leq |R|n \cdot 2T(n - 1)$
 - $T(n) = O((|R|n)^n)$.
- Still exponential

Efficient Algorithm

- Keep in memory the results of $\text{Derive}(A, x)$.
 - Number of different inputs: $|V| \cdot n^2$.
 - Only $|V| \cdot n^2$ calls, each takes $O(n|R|)$.
 - $T(n) = O(|R|n^3|V|)$.
- Polynomial time!

Dynamic Programming

- This approach is called **Dynamic Programming**
- Basic idea:
- If number of different inputs is limited, say $I(n)$.
- Each run (excluding recursive calls) takes at most $R(n)$ time
- Total running time is bounded by $T(n) \leq R(n)I(n)$.

Non-Context-Free Languages

Proving a Language is not a CFL

- The **pumping lemma** for finite automata and **Myhill-Nerode** theorem are our tools for showing that languages are **not regular**.
- We will now show a similar **pumping lemma** for context-free languages.
- It is slightly more complicated . . .

Pumping Lemma for CFL

Also known as the $uvxyz$ Theorem.

Theorem: If L is a CFL, there is an ℓ (critical length), such that if $w \in L$ and $|w| \geq \ell$, then $w = uvxyz$ where

- for every $i \geq 0$, $uv^i xy^i z \in L$
- $|vy| > 0$, (non-triviality)
- $|vxy| \leq \ell$.

Basic Intuition

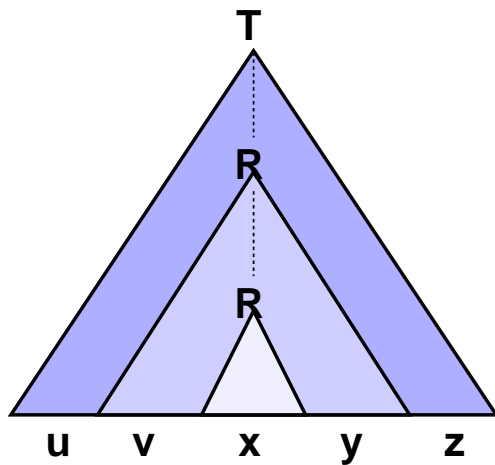
Let L be a CFL and G its CFG.

Let w be a “very long” string in L .

Then w must have a “tall” parse tree.

And some root-to-leaf path must repeat a symbol.

ahmmm,... why is that so?



We have: $T \xRightarrow{*} uRz$, $R \xRightarrow{*} vRy$, and $R \xRightarrow{*} x$.

Proof

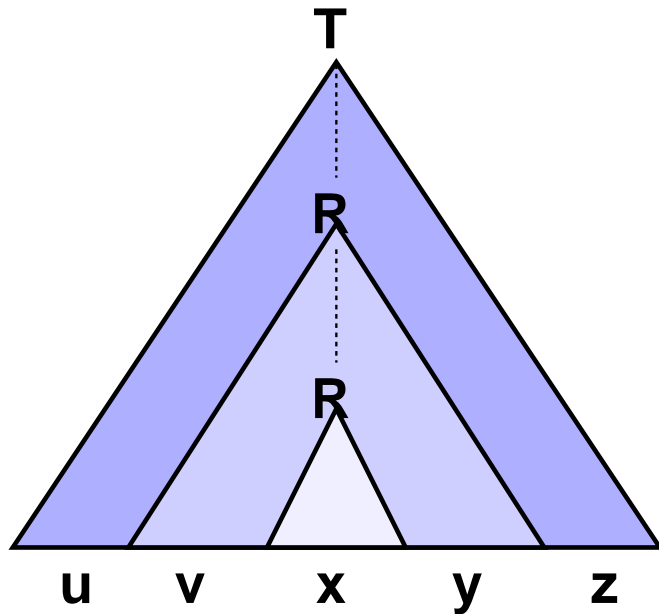
- let G be a CFG for CFL L .
- let b be the max number of symbols in right-hand-side of any rule (what is b for CNF grammars?).
- no node in parse tree has $> b$ children.
- at depth d , can have at most b^d leaves.
- let $|V|$ be the number of variables in G .
- set $\ell = b^{|V|+2}$.

Proof (2)

- let w be a string with $|w| \geq \ell$
- let T be parse tree for w with **fewest** nodes
- T has height $\geq |V| + 2$
- some path in T has length $\geq |V| + 2$
- that path **repeats** a variable R

Proof (3)

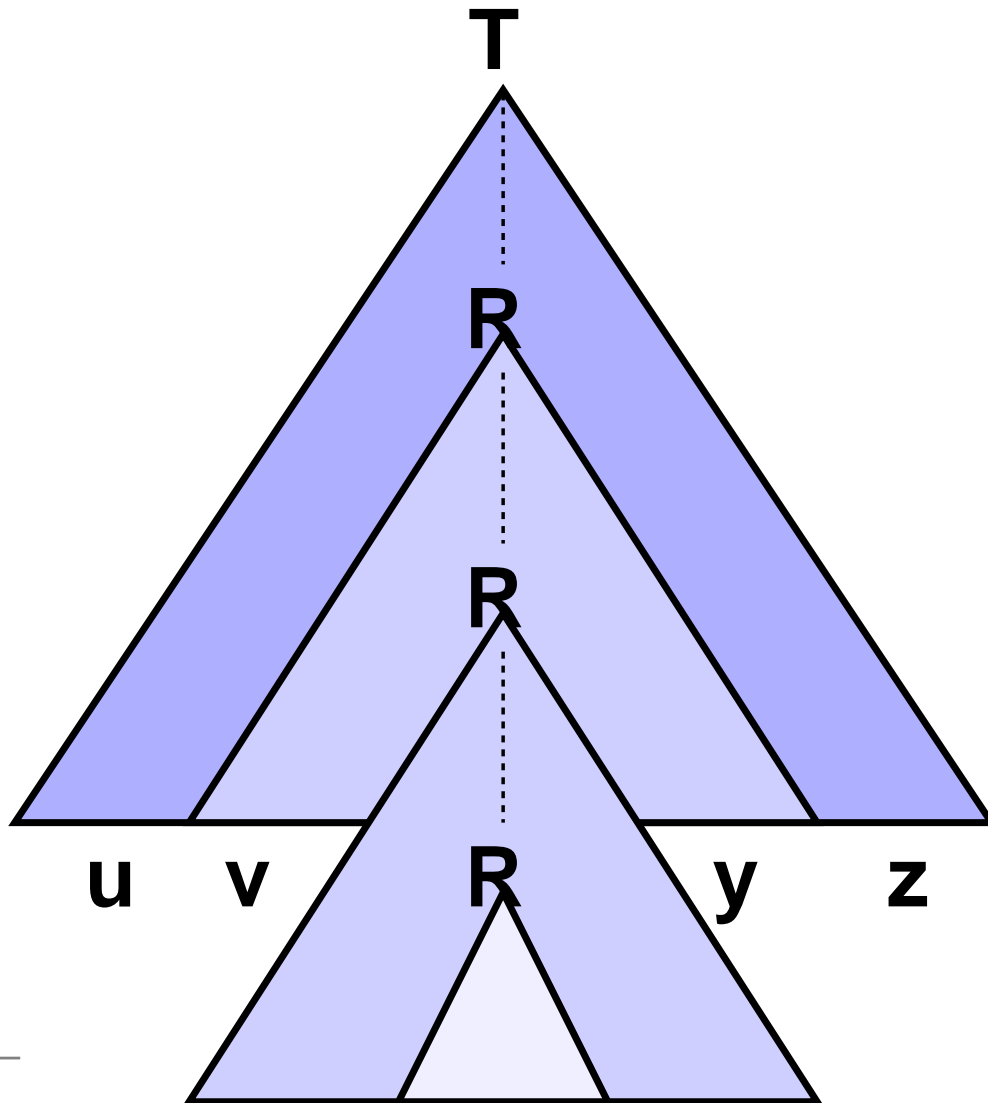
Split $w = uvxyz$



- each occurrence of R produces a string
- upper produces string vxy
- lower produces string x

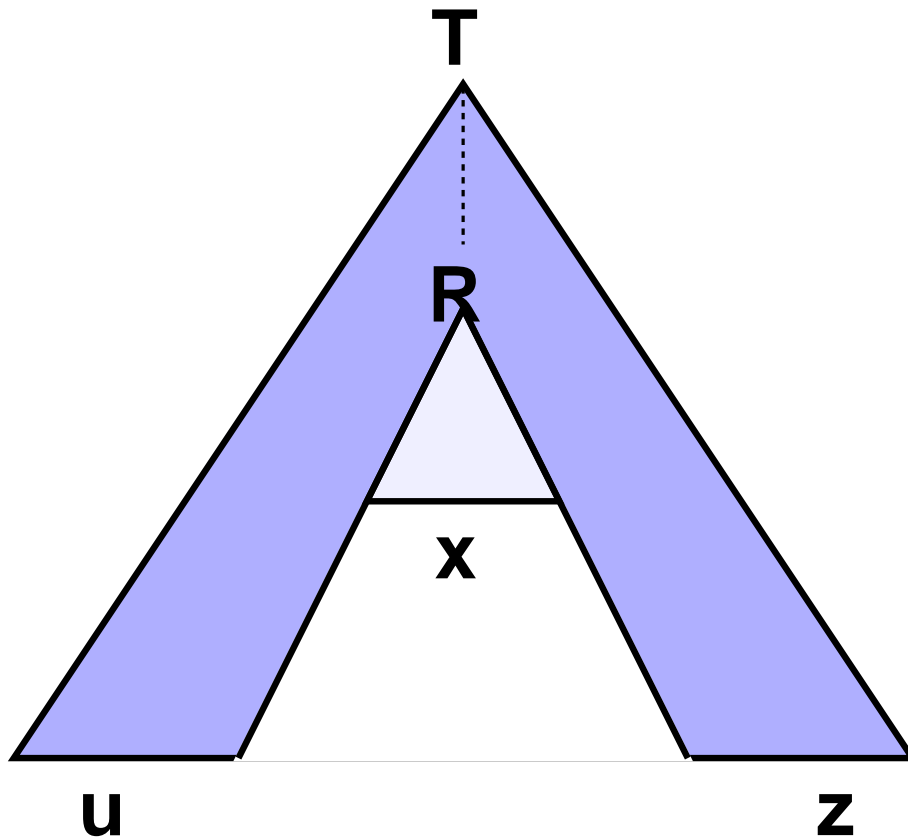
Proof

Replacing smaller by larger yields $uv^i xy^i z$, for $i > 0$.



Proof (4)

Replacing larger by smaller yields uxz .



Together, they establish:

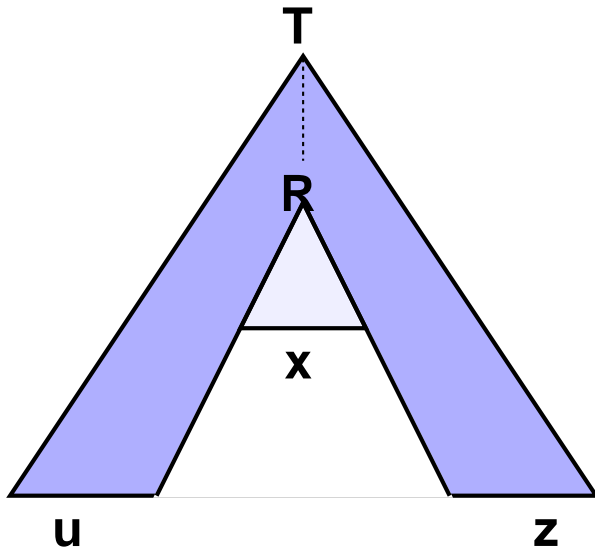
- for all $i \geq 0$, $uv^i xy^i z \in L$

Proof (5)

Next condition is:

- $|vy| > 0$ (out of $uvxyz$)

If v and y are both ε , then



is a parse tree for w with **fewer nodes** than T , contradiction.

Non CFL Example (1)

Claim: $B = \{a^n b^n c^n\}$ is not a CFL.

Proof: By contradiction.

Let ℓ be the critical length, and consider $w = a^\ell b^\ell c^\ell$.

If $w = uvxyz$, neither v nor y can contain

- both a 's and b 's, or
- both b 's and c 's,

because otherwise uv^2xy^2z would have out-of-order symbols.

But if v and y contain only one letter, then uv^2xy^2z has an imbalance!

Non CFL Example (2)

Claim: $C = \{a^i b^j c^k \mid 0 \leq i \leq j \leq k\}$ is not context free.

Let ℓ be the critical length, and $w = a^\ell b^\ell c^\ell$.

Let $w = uvxyz$

- Neither v nor y contains two distinct symbols, because otherwise uv^2xy^2z would have out-of-order symbols.
- vxy cannot be all the same letter (why?)
- $|vxy| \leq \ell$, so either
 - v contains only a 's and y contains only b 's, but then uv^2xy^2z has too few c 's.
 - v contains only b 's and y contains only c 's, but then uv^0xy^0z has too many a 's.

Non CFL Example (3)

Claim: $D = \{ww \mid w \in \{0, 1\}^*\}$ is not context-free.

Let $w = 0^\ell 1^\ell 0^\ell 1^\ell$. As before, suppose $w = uvxyz$

Recall that $|vxy| \leq \ell$.

- if vxy is in the first half of w , uv^2xy^2z “moves” a 1 into the first position in second half.
- if vxy is in the second half, uv^2xy^2z “moves” a 0 into the last position in first half.
- if vxy straddles the midpoint, then pumping *down* to uxz yields $0^\ell 1^i 0^j 1^\ell$ where i and j cannot both be ℓ .

Note that $D = \{ww^R \mid w \in \{0, 1\}^*\}$ is CFL.