CONTEXT-FREE GRAMMARS:

A more powerful way to specify languages.

A CFG ~~has~~ consists of ~~a~~ (1) A set of symbols called terminals (T or Σ) (2) A set of symbols called non-terminals or variables V
(3) A set of rules for generating sequences: productions.
(4) A special variable called the start symbol: S.

Example T or $\Sigma = \{a, b\}$ $\quad V = \{S\}$

$$S \xrightarrow{1} \varepsilon \qquad S \xrightarrow{2} aSb$$

How does this produce a string in Σ^*?

$$S \xrightarrow{2} a\underline{S}b \xrightarrow{2} a\,\underline{aSb}\,b \xrightarrow{2} aa\,aSb\,bb \xrightarrow{1} aaa\,bbb.$$

When you produce a string without variables you stop. The sequence is called a derivation.

This grammar produces the language $\{a^n b^n / n \geq 0\}$ which is not regular.

Used in linguistics to model sentence generation:

$\langle SENTENCE \rangle \rightarrow \langle N.P \rangle \cancel{\&} \langle V.P. \rangle$

$\langle N.P \rangle \rightarrow \langle C.N \rangle \mid \langle C.N. \rangle \langle PREP\text{-}PHRASE \rangle$

$\langle V.P \rangle \rightarrow \langle C.V. \rangle \mid \langle C.V. \rangle \langle PREP\text{-}P. \rangle$

$\langle P.\ PHRASE \rangle \rightarrow \langle PREP \rangle \langle C.N \rangle$

ARITH~~E~~METIC EXPRESSIONS: $\Sigma = \{0, 1, 2, \cdots 9, +, \times, (,)\}$.

$V = \{ \langle EXP \rangle, \langle NUM \rangle, \langle NZ \rangle, \langle N \rangle \}$

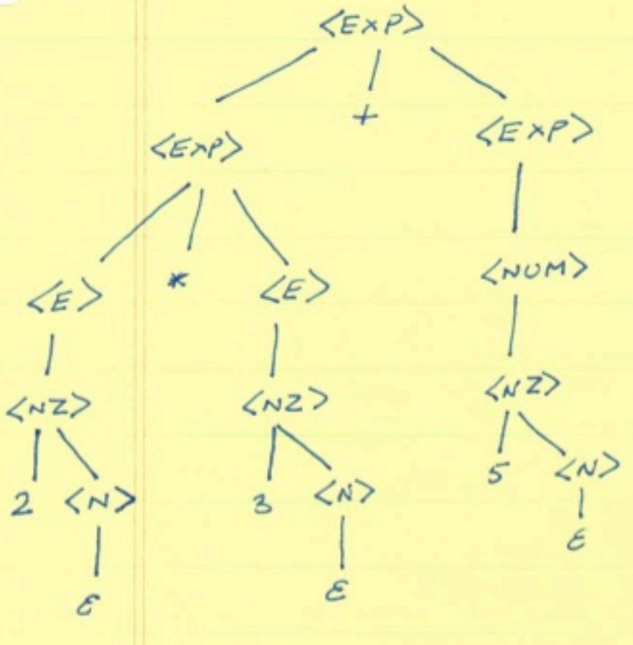
$\langle EXP \rangle \rightarrow \langle EXP \rangle + \langle EXP \rangle \mid \langle EXP \rangle * \langle EXP \rangle \mid (\langle EXP \rangle) \mid \langle NUM \rangle$

$\langle NUM \rangle \rightarrow 0 \mid \langle NZ \rangle \mid$

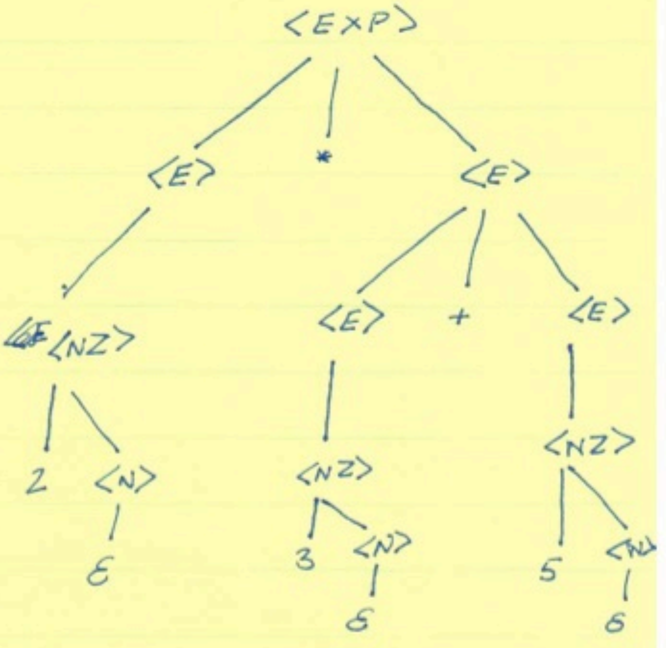
$\langle NZ \rangle \rightarrow \cancel{1 \langle N \rangle}\ 1\langle N \rangle \mid 2 \langle N \rangle \mid \cdots \mid 9 \langle N \rangle$

$\langle N \rangle \rightarrow 0 \langle N \rangle \mid 1 \langle N \rangle \mid \cdots \mid 9 \langle N \rangle \mid \varepsilon.$

To display derivations it is far better to use a tree: parse tree of a string. CFG's capture tree structure



2 * 3 + 5



2 * 3+5

AMBIGUOUS!

Def A grammar is said to be <u>ambiguous</u> if there are two <u>distinct</u> <u>parse</u> trees for the same string.

NOTE This is a property of the grammar not of the language.

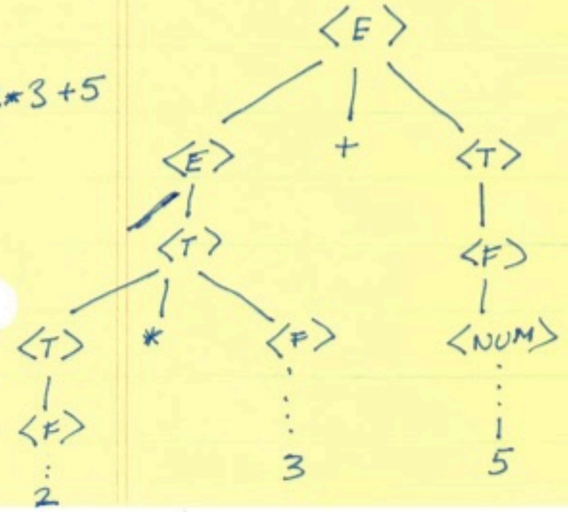
$V = \{\langle E\rangle, \langle T\rangle, \langle F\rangle\}$

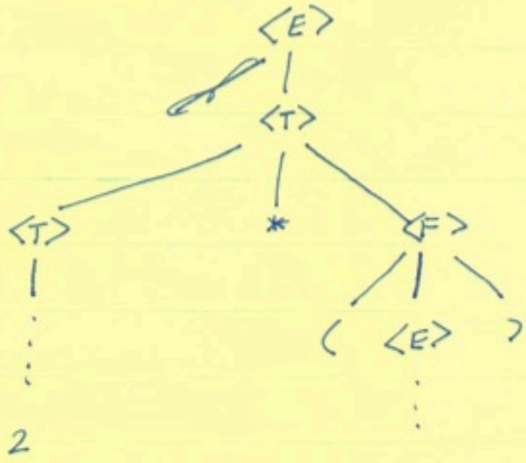
$\langle E\rangle \to \langle E\rangle + \langle T\rangle \mid \langle T\rangle$

$\langle T\rangle \to \langle T\rangle * \langle F\rangle \mid \langle F\rangle$

$\langle F\rangle \to (\langle E\rangle) \mid \langle NUM\rangle$

2*3+5





2 * (3+5) PARENS ARE FORCED

Can we do everything? No! $\{a^n b^n c^n \mid n \geq 0\}$.

~~Closure properties:~~

A <u>language</u> is <u>context-free</u> if there is a CFG for it.

$\{a^n b^n c^n \mid n \geq 0\}$ is not a CFL.

Closure properties: If L_1, L_2 are CFLs so is $L_1 \cup L_2$

$S_1 \cdots L_1 \quad S_2 \cdots L_2$

New grammar $V = V_1 \cup V_2 \cup \{S\}$ $\quad \Sigma = \Sigma_1 \cup \Sigma_2$

$S \to S_1 \mid S_2$ $\quad$ $L_1 = a^n b^n c^*$ is CFL, $L_2 = a^* b^n c^n$ is a CFL

$L_1 \cap L_2$ is <u>NOT</u>.

Any regular language is a CFL:

$M = (Q, q_0, \delta, F)$

$G = (V = Q, S = q_0,$

if $\delta(q, a) = q'$ add the rule

$\langle q \rangle = a \langle q' \rangle$

If $q \in F$ add the rule $\langle q \rangle \to \varepsilon$.

A grammar in which every rule has the form

$A \to aB$ or $A \to \varepsilon$ is called a <u>regular grammar</u>.

<u>Chomsky Normal Form</u>

A CFG is in CNF if every rule has the form

$A \to BC$ or $A \to a$

B, C are not allowed to be the start variable

& we do allow $S \to \varepsilon$.

<u>Thm</u> Every CFL is generated by a CFG in CNF.

<u>Proof.</u> First add a new start S_0 & add the rule $S_0 \to S$. This ensures that S_0 does not occur on the RHS of a rule. However we will have to deal with this new rule later.

We remove all rules of the form $A \to \varepsilon$: For any rule with A on the RHS we add a new rule with each <u>occurrence</u> of A removed.

e.g. $X \to uAvAw$ will be ~~replaced by~~ have added
$X \to uvw \mid uAvw \mid uvAw$.

If there is a rule $X \to A$ we add $X \to \varepsilon$ unless we had previously removed $X \to \varepsilon$. We keep going until we remove all rules of the form $A \to \varepsilon$.

Third we remove rules like $A \to B$. If we have $B \to u$ we add $A \to u$ unless this is a unit rule previously removed.

Fourth: Now we have no ε-rules or unit rules we need to get rid of "long" RH sides.

$$A \to u_1 u_2 \cdots u_k \quad \text{where } u_i \in \Sigma \cup V.$$

We replace this with

$$A \to u_1 A_1 \quad A_1 \to u_2 A_2 \cdots A_{k-2} \to u_{k-1} u_k.$$

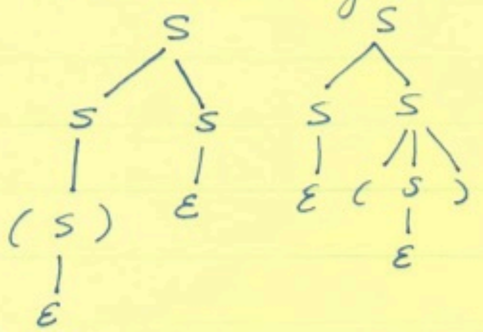
The A_i's are new variables. Now any ~~non-term~~ terminal u_i is replaced by a new variable U_i & we add $U_i \to u_i$.

CFG: parsing, HTML, XML

Grammar for nested parens

$S \to (S) \mid SS \mid \varepsilon$

This is ambiguous



Two different parse trees for ()()().

$S \to (S)S \mid \varepsilon$

Unambiguous.

Can we show that everything generated by this grammar is a collection of properly nested parens?

Can we show that every properly nested string of parens is generated by this grammar?

<u>Def</u> A sequence of parens is <u>properly nested</u> if ~~the~~ (1) the total number of ~~left~~ (&) are equal & (2) in any <u>prefix</u> the number of ($\geq$ number of).

<u>Prop 1</u> Any string generated by G_2 is properly nested.

<u>Proof</u> By induction on the length of the derivation.

$$\begin{array}{l} S \to (S)S \xrightarrow{*} (\alpha)\beta . \\ \quad\;\;\; \downarrow_* \;\; \downarrow_* \\ \quad\;\;\; \alpha \;\;\;\; \beta \end{array}$$

By IH : ~~α, β are~~ α, β are both properly nested.

So $(\alpha)\beta$ must have equal number of (&).

The prefixes of $(\alpha)\beta$ are:

(i) ((ii) $(\alpha'$ where $\alpha' \leq \alpha$ (iii) (α) (iv) $(\alpha)\beta'$ where $\beta' \leq \beta$.

In each case the property (2) holds.

<u>Prop 2</u> ~~So~~ If α is properly nested then $S \xrightarrow{*} \alpha$.

<u>Proof</u> By induction on $|\alpha|$

If α is properly nested then the first symbol must be (so $\alpha = (\alpha'$

& The leading (must be matched by a) somewhere.

Two possibilities $\alpha = (\beta)$ or $\alpha = (\gamma)\delta$

~~Now it must be~~ Now the matching) is the first point where $N_L - N_R = 0$ so γ must be properly matched & (γ) is properly matched so must δ be then

$$\begin{array}{l} S \to (S)S \qquad S \Rightarrow \gamma \;\&\; S \Rightarrow \delta \text{ must be derivable} \\ \quad\;\;\; \downarrow \;\;\; \downarrow \\ \quad\;\;\; \gamma \;\;\; \delta \text{ gives a derivation of } (\gamma)\delta . \end{array}$$

~~CFLs~~ Closure properties:

(1) L_1, L_2 are CFLs then $L_1 \cup L_2$ is a CFL

$$S \rightarrow S_1 / S_2$$

(2) $L_1 \cdot L_2$ is a CFL $\qquad S \rightarrow S_1 S_2$

(3) L^* is a CFL Add new start symbol S' & new rules

$$S' \rightarrow SS' / \varepsilon.$$

CFLs are <u>NOT</u> closed under intersection:

~~$a^*b^nc^n$~~ $L_1 = \{ a^n b^n c^m / n, m \geq 0 \}$

$L_2 = \{ a^m b^n c^n / n, m \geq 0 \}$

$L_1 \cap L_2$ is not context-free.

CFLs are not closed under complement. If they were they would have to be closed under $\cap$.

$L = \{ a^n b^n c^n / n \geq 0 \}$ is not a CFL but $\overline{L}$ is a CFL.

If L is a CFL & R is a regular language then $L \cap R$ is a CFL.