Context-free Grammars and Languages

Context-free languages

$\Sigma = \{a, +, *, (,)\}$

Rules:

$E \to P + E$

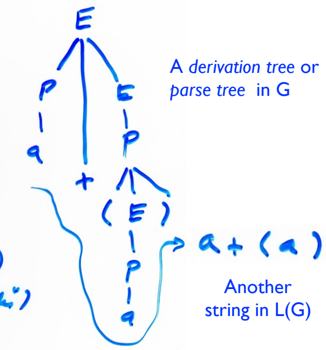
$E \to P * E$

$E \to P$

$P \to (E)$

$P \to a$

Example strings in L(G): a, (a)



A CFG $G = (V, \Sigma, R, S)$

V a finite set ("variables")

$V \cap \Sigma = \emptyset$

("start" or "sentence") $S \in V$

("rules") R is a finite subset of $V \times (V \cup \Sigma)^*$

Notation

$\to$ in rules (only; "produces" or "may be rewritten as")

$\Rightarrow$ "yields": relation on strings in $(V \cup \Sigma)^*$

$\alpha A \beta \Rightarrow \alpha \gamma \beta$

if $A \to \gamma$ is a rule

for all $\alpha, \beta \in (V \cup \Sigma)^*$ → i.e., "context-free"

$\Rightarrow^*$ "derives" (reflexive, transitive closure of $\Rightarrow$; "0 or more steps")

$\alpha \Rightarrow^* \beta$ means $\exists\ \alpha_0\ \alpha_1 \cdots \alpha_k$ $k \geq 0$

$\alpha = \alpha_0 \Rightarrow \alpha_1 \Rightarrow \alpha_2 \cdots \alpha_k = \beta$

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

Example

$G = (V, \Sigma, R, S)$

$V = \{S\}$

$\Sigma = \{a, b\}$

R:

$S \to aSb \mid \varepsilon$

$S \Rightarrow \varepsilon$

$S \Rightarrow aSb \Rightarrow ab$

$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aabb$

$\vdots$

$L(G) = \{ a^n b^n \mid n \geq 0 \}$

Note that L(G) is non-regular

Example

$G_2 = (V, \Sigma, R, S)$

$V = \{S\}$

$\Sigma = \{a, b\}$

R:

$S \to aSa \mid bSb \mid \varepsilon$

$S \Rightarrow \varepsilon$

$S \Rightarrow a\varepsilon a \Rightarrow aa$

$S \Rightarrow bSb \Rightarrow bb$

$S \Rightarrow aSa \Rightarrow abSba \Rightarrow abba$

$L(G_2)$ = even length palindromes $\{ww^R \mid w \in \Sigma^*\}$

We'll see later that

$L_{two} = \{ww \mid w \in \Sigma^*\}$

is *not* context free. At first glance, you might think that adding a new start symbol S' and a rule

$S' \to SS$

to G_2 would generate L_{two}, but it doesn't; it generates all strings in L_{two} *plus* many others, since derivations from the two S's are *not coordinated*. (Why not? It's *context-free*; what happens to one S can't influence the other.)

Example

G_3: as above but add
$S \to a \mid b$

$L(G_3)$ all palindromes
$\{w \in \Sigma^* \mid w = w^R\}$

Trees, Derivations and Ambiguity

A grammar

$E \to P + E$

$E \to P * E$

$E \to P$

$P \to (E)$

$P \to a$

A tree

3 derivations correspond to same tree (same rules being used in the same places, just written in different orders in the linear derivation)

1) E => P+E => a+E => a+P => a+a

2) E => P+E => P+P => a+P => a+a

3) E => P+E => P+P => P+a => a+a

But only one *leftmost* derivation corresponds to it

(and *vice versa*). (more in HW?)

Another grammar for the same language:

E → E+E | E*E | (E) | a

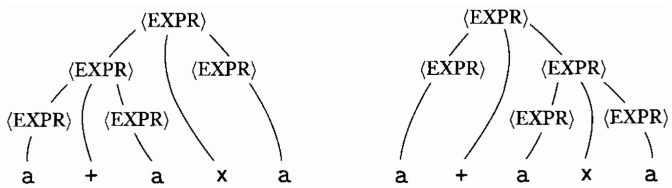


Fig 2.6: Two parse trees for a+a×a in grammar G_5

This grammar is *ambiguous:* there is a string in L(G_5) with two *different* parse trees, or, equivalently, with 2 different leftmost derivations. Note the pragmatic difference: in general, (a+a)*a != a+(a*a); which is "right"?

E → E+E | E*E | a

ambiguous

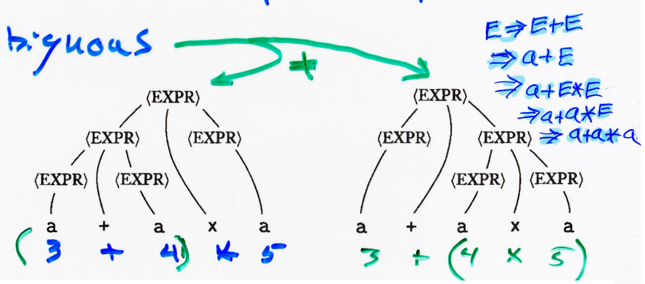


Fig 2.6: Two parse trees for a+a×a in grammar G_5

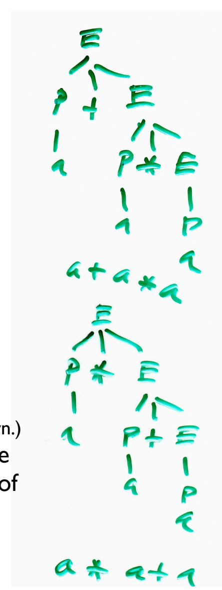
Leftmost deriv

$E \Rightarrow_L E*E \Rightarrow_L E+E*E \Rightarrow_L a+E*E \Rightarrow_L a+a*E \Rightarrow_L a+a*a$

non-leftmost deriv: E*E ⇒ E*a ⇒ E+E*a ⇒ E+a*a

The "E, P" grammar again

E → P+E
E → P*E
E → P
P → (E)
P → a



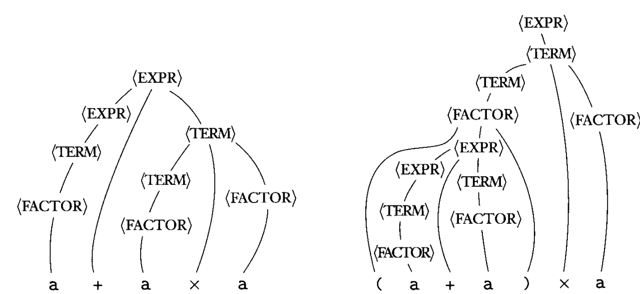
This grammar is *un*ambiguous.
(Why? Very informally, the 3 E rules generate P(('+'∪'*')P)* and only via a parse tree that "hangs to the right", as shown.)
But it has another undesirable feature: Parse tree structure does not reflect the usual precedence of * over +. E.g., tree at lower right suggests "a * a + a == a * (a + a)"

EXAMPLE 2.4

Consider grammar $G_4 = (V, \Sigma, R, \langle\text{EXPR}\rangle)$.
V is $\{\langle\text{EXPR}\rangle, \langle\text{TERM}\rangle, \langle\text{FACTOR}\rangle\}$ and Σ is $\{a, +, \times, (,)\}$. The rules are

$$\langle\text{EXPR}\rangle \to \langle\text{EXPR}\rangle + \langle\text{TERM}\rangle \mid \langle\text{TERM}\rangle$$
$$\langle\text{TERM}\rangle \to \langle\text{TERM}\rangle \times \langle\text{FACTOR}\rangle \mid \langle\text{FACTOR}\rangle$$
$$\langle\text{FACTOR}\rangle \to (\langle\text{EXPR}\rangle) \mid a$$

The two strings a+a×a and (a+a)×a can be generated with grammar G_4. The parse trees are shown in the following figure.



A more complex grammar, again the same language. This one is unambiguous *and* its parse trees reflect usual precedence/associativity of plus and times.

$$L = \{ \overbrace{\underbrace{a^i b^j}_{A}}^{} \overbrace{b^j c^k}^{B} \mid i=j \text{ or } j=k \}$$

$$\left.\begin{aligned} S &\to AC \mid DB \\ A &\to aAb \mid \varepsilon \\ C &\to cC \mid \varepsilon \\ D &\to aD \mid \varepsilon \\ B &\to bBc \mid \varepsilon \end{aligned}\right\} G$$

$a^{10} b^{10} c^{22}$

$a^{10} b^{22} c^{22}$

$a^{10} b^{10} c^{10}$

Can we always tweak the grammar to make it unambiguous?

No! Language L is a CFL; grammar at left. Easy to see this G is ambiguous–strings of the form $a^n b^n c^n$ can come from the i=j (AC) or j=k (DB) path. Hard to prove, but true, that *every G for this L is also ambiguous*. Intuitively, a grammar can only match a's & b's or b's & c's, not both. As a related point, $\{ a^n b^n c^n \mid n>0 \}$ is *not* CFL.

G is ambiguous

L is *inherently ambiguous*, meaning every G for L is ambiguous

Some closure results for CFLs

<u>Theorem:</u>

The set of context-free languages is closed under ∪, •, and *

<u>Corollary:</u>

All regular languages are CFLs

Proof Sketch:

Directly give simple CFLs for ∅, {ε}, and {a} for each a ∈ Σ. Combine them using the above theorem.

(Aside:

We'll later prove that CFLs are *not* closed under intersection or complementation.)

Proof: Closure under Concatenation

$G_i = (V_i, \Sigma, R_i, S_i)$

be 2 CFG's

with $V_1 \cap V_2 = \phi$

let $S \notin V_1 \cup V_2$

Build new grammar

$G = (V, \Sigma, R, S)$

$V = V_1 \cup V_2 \cup \{S\}$

$R = R_1 \cup R_2 \cup \{ S \to S_1 S_2 \}$

$\forall x \in L_1 \ \forall y \in L_2$

$S_1 \overset{*}{\underset{G_1}{\Rightarrow}} x$ & $S_2 \overset{*}{\underset{G_2}{\Rightarrow}} y$

$\therefore S \underset{G}{\Rightarrow} S_1 S_2 \overset{*}{\underset{G}{\Rightarrow}} x S_2 \overset{*}{\underset{G}{\Rightarrow}} xy$

$\therefore L_1 \cdot L_2 \subseteq L(G)$

Suppose $S \overset{*}{\Rightarrow}_G w$

* $\boxed{S \Rightarrow_G S_1 S_2 \overset{*}{\Rightarrow}_G w}$ Then, for some x, y ε Σ*

$S_1 S_2 \overset{*}{\Rightarrow}_L x S_2$

using only rules from G_1

$x S_2 \overset{*}{\Rightarrow}_L x y = w$

using only G_2 rules

$S_1 \overset{*}{\Rightarrow} x$ in G_1
$S_2 \Rightarrow y$ in G_2 **

$L(G) \subseteq L_1 \cdot L_2$

A key issue in this direction of the proof is that, since $V_1 \cap V_2 = \varnothing$, there is no "crosstalk" between the two sub-grammars: any derivation in G from S_1 is also a derivation in G_1, and likewise S_2/G_2, so derivation (*) above in G can be split into (**) in G_1 & G_2.