Why Use Grammars?

Precise specification of language structure

Ex. Begin-End hierarchy

Certain classes of grammars have efficient automatic tools to construct parsers for languages

Ex. Bison

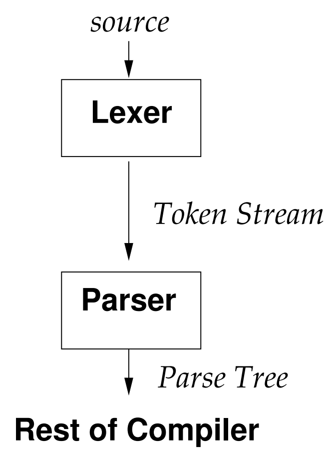
Provides feedback to language designer

See if new construct difficult to handle

Adaptable

Easy to add new language constructs

Front End of Compiler



Main Goal of Parser: Produce parse tree via grammar

Example: Context-free Grammar

Rules

E → a
E → EAE
A → +
A → *

Terminals + * a

Variables E A

Start Variable E

Context-free since can apply rule to variable in any context.
Use rules starting from start variable to derive strings .

Using Context-free Grammar

E → EAE → aAE → a+E → a+EAE → a+aAE → a+a*E → a+a*a

Derivation of terminal string **a+a*a**

Can abbreviate the rules:

E → a **E → EAE**	*written as*	**E → a | EAE**

Can derive string of variables and terminals

Language of grammar is the set of terminal strings it can derive

Formal Definition of Context-free Grammar

A *CFG G* is a 4-tuple (V, Σ, R , S) where,

1. V is a finite set of *variables (non-terminals)*
2. Σ is a finite set of *terminals*
3. R is a set of *rules (productions)* of the form

   variable $\rightarrow$ string of terminals, variables
4. S in V is the *start variable*

Ex. V = { E, T, F} *Ex. 2.3 p 95*

Σ = {a, +, x, (,) }

R E $\rightarrow$ E+T | T

T $\rightarrow$ TxF | F

F $\rightarrow$ (E) | a

Start E *Derive ?*

Derivations and Languages

Given CFG G with start var. S

uAv $\Longrightarrow$ uwv if A $\rightarrow$ w is a rule of G and

u, v, w are strings of variables and terminals.

u $\overset{*}{\Longrightarrow}$ v if u = v, or there is a sequence u1...uk

u = u1 $\Longrightarrow$ u2 .... uk-1 $\Longrightarrow$ uk = v

L(G) = { w in Σ^*| S $\overset{*}{\Longrightarrow}$ w}

G1 is *equivalent* to G2 if L(G1) = L(G2)

Leftmost Derivation:

Derivation in which always replace leftmost var.

Ex. a+a $*$a

May be many derivations of same string!

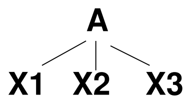
Rightmost, leftmost, other

Parse Trees

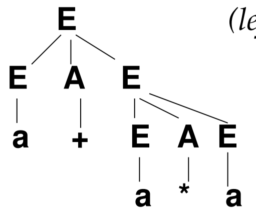
Similar to derivation, ignores order of replacement

Rule: A → X1 X2 X3

Parse tree:



Ex. *(leftmost deriv. a+a*a)*

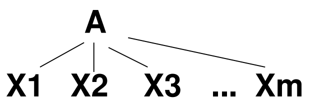


Unique leftmost (rightmost) derivation ⟷ *Parse tree*

Formal Definition of Parse Tree

Given a CFG G, *a parse tree* **is a labelled tree with**

1. **a single root labelled with the start variable**
2. **Leaves that are labelled by terminal symbols**
3. **Interior nodes labelled by variables**
4. **If the parse tree contains**



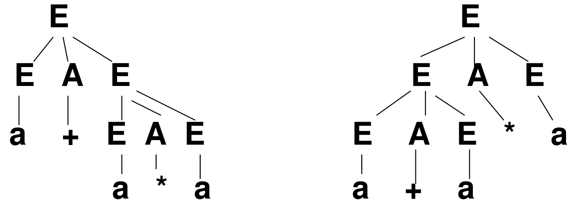
variable

all A's children

Then A → X1 X2 X3 ... Xm is a rule of G.

Ambiguity of a Grammar

Ex. a+a*a



Leaves the same: derive same string

*Correspond to a + (a*a) and (a + a) * a*

May yield different answers if evaluated!

Def. G is *ambiguous* **if there is some string w in Σ^* with two different parse trees.**

Ambiguity of a Grammar?

Ex. $S \rightarrow (S) \mid SS \mid \varepsilon$ where Σ is {ε, (,)}

What is L(G)?

Is G ambiguous?

Ex. $E \rightarrow E+E \mid ExE \mid (E) \mid a$

Is G ambiguous?

Ex. S → if E then S | if E then S else S | other
where Σ is { if, then, else, other}

Is G ambiguous?

Disambiguation of CFG

E → E+E | E x E |(E) | a *Problem: Precedence of +, x??*

Rewrite: E → E+T | T *1 new variable for each*
T → TxF | F
F → (E) | a *level of precedence*

Expression is a list of terms (T) separated by +
Term is a list of factors (F) separated by x

Ex. S → if E then S | if E then S else S | other
where Σ is { if, then, else, other}

Want to match else to closest previous unmatched else (C)
Add 2 new variables, MS (matched) and US (unmatched)

S → MS | US
MS → if E then MS else MS
US → if E then S | if E then MS else US

Disambiguation of CFG

Ex. S → (S) | SS | ε *What is the problem?*

Ex. B → B and B | a | b

Problem: How associate--to left or right?
Make right associative:

Designing CFG's

Ex. $\{ 0^n 1^n \mid n \geq 0\}$

Ex. $\{ 0^n 1^n \mid n \geq 0\} \bigcup \{1^n 0^n \mid n \geq 0\}$

Ex. $\{ w w^r \mid w \text{ in } \{0,1\}^* \}$ *where r is the reverse op.*