## 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

```
source

Lexer

Token Stream

Parser

Parse Tree

Rest 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

\[ E \rightarrow EAE \rightarrow aAE \rightarrow a+E \rightarrow a+EAE \rightarrow a+aAE \rightarrow \]
\[ a+aE \rightarrow a+a*a \]

Derivation of terminal string \( a+a*a \)

Can abbreviate the rules:

\[ E \rightarrow a \quad \text{written as} \quad E \rightarrow a \mid EAE \]

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 $(V, \Sigma, R, S)$ where,

1. $V$ is a finite set of variables (non-terminals)
2. $\Sigma$ 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. $V = \{E, T, F\}$  
$\Sigma = \{a, +, x, (, )\}$  
$R$  
$E \rightarrow E + T \mid T$  
$T \rightarrow T x F \mid F$  
$F \rightarrow (E) \mid a$  
Start $E$ 

*Derive?*

---

**Derivations and Languages**

Given CFG $G$ with start var. $S$

$uA v \Rightarrow uvw$ if $A \rightarrow w$ is a rule of $G$ and $u, v, w$ are strings of variables and terminals.

$u \Rightarrow^* v$ if $u = v$, or there is a sequence $u 1 ... u k$

$u = u 1 \Rightarrow u 2 \ldots u k$ if $u = v$

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

$G_1$ is equivalent to $G_2$ if $L(G_1) = L(G_2)$

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: \( A \rightarrow X_1 X_2 X_3 \)

Parse tree:

```
          A
         / \
        X1   X2   X3
```

Ex.  

```
      E
     / \   (leftmost deriv. \( a+a^*a \))
    E   A   E
   / \   /   \
  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

```
          A
         / \
        X_1  X_2  X_3  ...  X_m
```

Then \( A \rightarrow X_1 X_2 X_3 ... X_m \) 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 $\Sigma^*$ with two different parse trees.

Ambiguity of a Grammar?

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

What is $L(G)$?

Is $G$ ambiguous?

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

Is $G$ ambiguous?

Ex. $S \rightarrow$ if $E$ then $S$ $|$ if $E$ then $S$ else $S$ $|$ other
where $\Sigma$ is $\{\text{if, then, else, other}\}$

Is $G$ ambiguous?
Disambiguation of CFG

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

Problem: Precedence of $+, \times$?

Rewrite:

$E \rightarrow E + T \mid T$
$T \rightarrow T \times F \mid F$
$F \rightarrow (E) \mid a$

1 new variable for each level of precedence

Expression is a list of terms ($T$) separated by $+$
Term is a list of factors ($F$) separated by $\times$

Ex. $S \rightarrow$ if $E$ then $S$ | if $E$ then $S$ else $S$ | other
where $\Sigma$ is { if, then, else, other}

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

$S \rightarrow MS \mid US$
$MS \rightarrow$ if $E$ then $MS$ else $MS$
$US \rightarrow$ if $E$ then $S$ | if $E$ then $MS$ else $US$

Ex. $S \rightarrow (S) \mid SS \mid \varepsilon$

What is the problem?

Ex. $B \rightarrow B$ and $B \mid a \mid b$

Problem: How associate—to left or right?
Make right associative:
**Designing CFG's**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Ex.</td>
<td>{ 0^n 1^n</td>
</tr>
<tr>
<td>Ex.</td>
<td>{ 0^n 1^n</td>
</tr>
<tr>
<td>Ex.</td>
<td>{ w w'^r</td>
</tr>
</tbody>
</table>