

Griffith University

**3130CIT Theory of Computation**

(Based on slides by Harald Søndergaard of  
The University of Melbourne)

**Context-Free Languages**

## Context-Free Grammars

... were invented in the fifties, when Chomsky proposed different formalisms for describing natural language syntax.

They were popularised by Naur with the Algol 60 report, and are often referred to as Backus-Naur Formalism (BNF).

Standard tools for parsing owe much to this formalism, which indirectly has helped make parsing a routine task.

It is extensively used to specify syntax of programming languages, and now also document formats (XML's document-type definition).

## Context-Free Grammars (cont.)

We have already used the formalism of context-free grammars. To specify the syntax of regular expressions we gave a *grammar*, much like

$$R \rightarrow 0$$

$$R \rightarrow 1$$

$$R \rightarrow \epsilon$$

$$R \rightarrow \emptyset$$

$$R \rightarrow R \cup R$$

$$R \rightarrow R \circ R$$

$$R \rightarrow R^*$$

Hence a grammar is a set of *substitution rules*, or *productions*. We have the shorthand notation

$$R \rightarrow 0 \mid 1 \mid \epsilon \mid \emptyset \mid R \cup R \mid R \circ R \mid R^*$$

## Sentences

A simpler example is this grammar  $G$ :

$$A \rightarrow 0 A 1 1$$

$$A \rightarrow \epsilon$$

Using the two rules as a rewrite system, we get *derivations* such as

$$A \Rightarrow 0A11$$

$$\Rightarrow 00A1111$$

$$\Rightarrow 000A111111$$

$$\Rightarrow 000111111$$

$A$  is called a *variable*. Other symbols (here 0 and 1) are *terminals*.

Compiler writers refer to a valid string of terminals (such as 000111111) as a *sentence*.

The intermediate strings that mix variables and terminals are *sentential forms*.

## Context-Free Languages

Clearly a grammar determines a formal language.

The language of  $G$  is written  $L(G)$ .

$$L(G) = \{0^n 1^{2n} \mid n > 0\}$$

A language which can be generated by some context-free grammar is a *context-free language* (CFL).

It should be clear that some of the languages that we found not to be regular *are* context-free, for example

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

## Context-Free Grammars Formally

A context-free grammar (CFG)  $G$  is a 4-tuple  $(V, \Sigma, R, S)$ , where

1.  $V$  is a finite set of *variables*,
2.  $\Sigma$  is a finite set of *terminals*,
3.  $R$  is a finite set of *rules*, each consisting of a variable (the left-hand side) and a sentential form (the right-hand side),
4.  $S$  is the *start variable*.

The binary relation  $\Rightarrow$  on sentential forms is defined as follows.

Let  $u$ ,  $v$ , and  $w$  be sentential forms. Then  $uAw \Rightarrow uvw$  iff  $A \rightarrow v$  is a rule in  $R$ .

So  $\Rightarrow$  captures a single derivation step.

Let  $\Rightarrow^*$  be the reflexive transitive closure of  $\Rightarrow$ .

$$L(G) = \{s \in \Sigma^* \mid S \xRightarrow{*} s\}$$

## Parse Trees

Here is a grammar with three variables, 14 terminals, and 15 rules:

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

$$T \rightarrow F \mid F * T$$

$$F \rightarrow 0 \mid 1 \mid \dots \mid 9 \mid ( E )$$

When the start variable is unspecified, it is assumed to be the variable of the first rule.

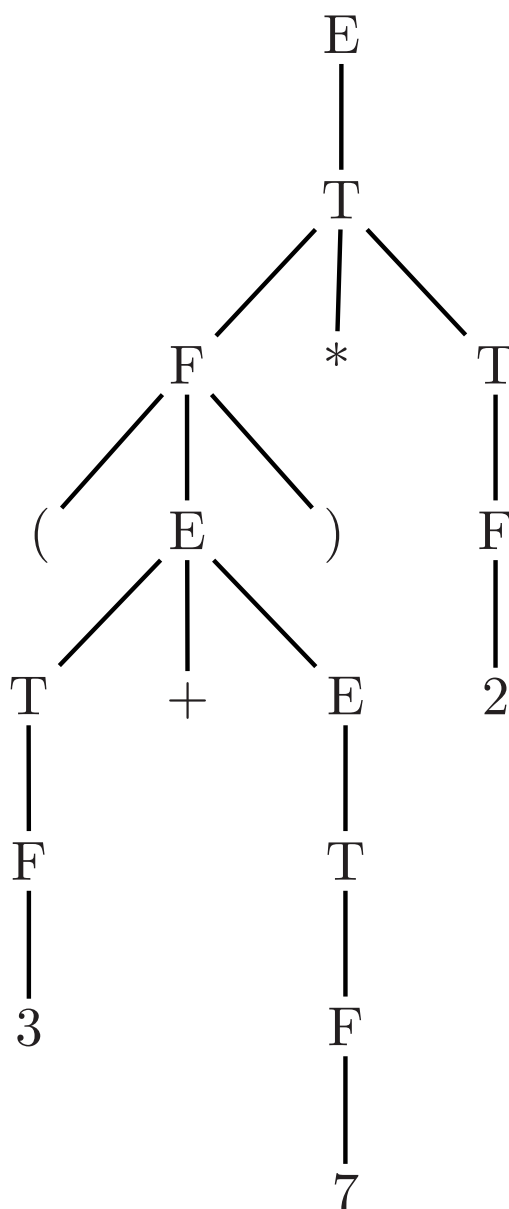
An example sentence in the language is

$$(3 + 7) * 2$$

The grammar ensures that  $*$  binds tighter than  $+$ .

# Parse Trees (cont.)

Here is a *parse tree* for  $(3 + 7) * 2$ :



## Parse Trees (cont.)

There are different derivations leading to the sentence  $(3 + 7) * 2$ , all corresponding to the parse tree above. They differ in the order in which we choose to replace variables.

Here is the *leftmost* derivation:

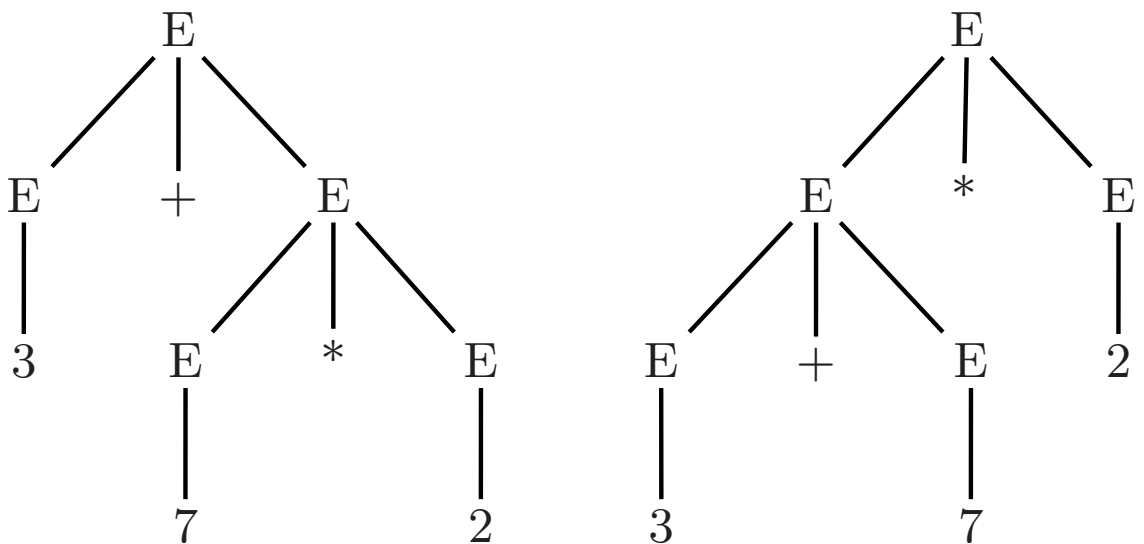
$$\begin{aligned} E &\Rightarrow T \\ &\Rightarrow F * T \\ &\Rightarrow ( E ) * T \\ &\Rightarrow ( T + E ) * T \\ &\Rightarrow ( F + E ) * T \\ &\Rightarrow ( 3 + E ) * T \\ &\Rightarrow ( 3 + T ) * T \\ &\Rightarrow ( 3 + F ) * T \\ &\Rightarrow ( 3 + 7 ) * T \\ &\Rightarrow ( 3 + 7 ) * F \\ &\Rightarrow ( 3 + 7 ) * 2 \end{aligned}$$

# Ambiguity

Consider the grammar

$$E \rightarrow E + E \mid E * E \mid ( E ) \mid 0 \mid 1 \mid \dots \mid 9$$

This grammar allows not only different derivations, but different parse trees for  $3 + 7 * 2$ :



## Ambiguity (cont.)

A grammar that has different parse trees for some sentence is *ambiguous*.

Sometimes we can find a better grammar (as in our example) which is not ambiguous, and which generates the same language.

However, this is not always possible: There are CFLs that are *inherently ambiguous*, for example,

$$L = \{ a^i b^j c^k \mid i = j \text{ or } j = k \}.$$

(Consider parse trees for  $a^3 b^3 c^3$ .)

## Chomsky Normal Form

It is sometimes convenient to bring a CFG into a normal form.

A simple normal form is *Chomsky normal form* where every rule is of one of these forms:

$$A \rightarrow BC$$

$$A \rightarrow a$$

$$S \rightarrow \epsilon$$

where  $S$  is the start variable,  $A$  may be the start variable,  $B$  and  $C$  are (non-start) variables, and  $a$  is a terminal.

**Theorem:** Every CFL has a CFG in Chomsky normal form.

## Conversion to Chomsky Form

The method for converting a grammar to Chomsky normal form is this:

1. Add a new start variable  $S_0$  and rule  $S_0 \rightarrow S$ .
2. Eliminate epsilon rules  $A \rightarrow \epsilon$ .
3. Eliminate unit rules  $A \rightarrow B$ .
4. Eliminate useless symbols.
5. Ensure that right-hand sides with length greater than 1 consist of variables only.
6. Break right-hand sides of length 3 or more into several rules by introducing fresh variables.

## Eliminating Epsilon Rules

If we have  $A \rightarrow \epsilon$  then we replace every occurrence of  $A$  on right-hand sides by  $\epsilon$ . For example,

$$\begin{array}{l}
 S_0 \rightarrow S \\
 S \rightarrow A S A B \\
 S \rightarrow \epsilon \\
 A \rightarrow \epsilon \\
 B \rightarrow C \\
 C \rightarrow a
 \end{array}
 \quad \Rightarrow \quad
 \begin{array}{l}
 S_0 \rightarrow S \\
 S_0 \rightarrow \epsilon \\
 S \rightarrow A S A B \\
 S \rightarrow A A B \\
 S \rightarrow S A B \\
 S \rightarrow A B \\
 S \rightarrow A S B \\
 S \rightarrow S B \\
 S \rightarrow B \\
 B \rightarrow C \\
 C \rightarrow a
 \end{array}$$

A rule  $E \rightarrow A$  gets replaced by  $E \rightarrow \epsilon$  unless we already removed *that* rule.  $A \rightarrow \epsilon$  is removed.

## Eliminating Unit Rules

We replace a rule  $B \rightarrow C$  by  $B \rightarrow u$  for each rule  $C \rightarrow u$ , *unless*  $B \rightarrow u$  is a unit rule we already removed.

$S_0 \rightarrow S$		$S_0 \rightarrow$ (same as $S$ )
$S_0 \rightarrow \epsilon$		$S_0 \rightarrow \epsilon$
$S \rightarrow A S A B$		$S \rightarrow A S A B$
$S \rightarrow A A B$		$S \rightarrow A A B$
$S \rightarrow S A B$		$S \rightarrow S A B$
$S \rightarrow A B$	$\Rightarrow$	$S \rightarrow A B$
$S \rightarrow A S B$		$S \rightarrow A S B$
$S \rightarrow S B$		$S \rightarrow S B$
$S \rightarrow B$		$S \rightarrow a$
$B \rightarrow C$		$B \rightarrow a$
$C \rightarrow a$		$C \rightarrow a$

## Eliminating Useless Symbols

There are two kinds of useless variables.

First remove rules with a symbol such as  $A$  which is not *generating*:

$$S_0 \rightarrow S B$$

$$S_0 \rightarrow a$$

$$S_0 \rightarrow \epsilon$$

$$S \rightarrow S B$$

$$S \rightarrow a$$

$$B \rightarrow a$$

$$C \rightarrow a$$

## Eliminating Useless Symbols (cont.)

Then remove rules with a symbol such as  $C$  which is not *reachable*:

$$S_0 \rightarrow S B$$

$$S_0 \rightarrow a$$

$$S_0 \rightarrow \epsilon$$

$$S \rightarrow S B$$

$$S \rightarrow a$$

$$B \rightarrow a$$

This grammar is now in Chomsky normal form, but sometimes we need another two steps ...

## Another Example

Consider the grammar (with new start variable)

$$S \rightarrow E$$

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

$$T \rightarrow F \mid F * T$$

$$F \rightarrow 0 \mid 1 \mid ( E )$$

There are no epsilon rules, but several unit rules to eliminate:

$$S \rightarrow 0 \mid 1 \mid ( E ) \mid F * T \mid T + E$$

$$E \rightarrow 0 \mid 1 \mid ( E ) \mid F * T \mid T + E$$

$$T \rightarrow 0 \mid 1 \mid ( E ) \mid F * T$$

$$F \rightarrow 0 \mid 1 \mid ( E )$$

## Another Example (cont.)

Now make right-hand sides of length more than 1 consist of variables:

$$S \rightarrow 0 \mid 1 \mid L E R \mid F M T \mid T P E$$

$$E \rightarrow 0 \mid 1 \mid L E R \mid F M T \mid T P E$$

$$T \rightarrow 0 \mid 1 \mid L E R \mid F M T$$

$$F \rightarrow 0 \mid 1 \mid L E R$$

$$L \rightarrow ($$

$$R \rightarrow )$$

$$M \rightarrow *$$

$$P \rightarrow +$$

## Another Example (cont.)

Finally cascade the rules as needed, introducing more variables:

$$S \rightarrow 0 \mid 1 \mid L' R \mid F' T \mid T' E$$

$$E \rightarrow 0 \mid 1 \mid L' R \mid F' T \mid T' E$$

$$T \rightarrow 0 \mid 1 \mid L' R \mid F' T$$

$$F \rightarrow 0 \mid 1 \mid L' R$$

$$L \rightarrow ($$

$$R \rightarrow )$$

$$M \rightarrow *$$

$$P \rightarrow +$$

$$L' \rightarrow L E$$

$$F' \rightarrow F M$$

$$T' \rightarrow T P$$