3 Defining Languages with Context-free Grammars

Context-free Grammars Defined

A production is a replacement rule. For example,

```
A → aAb
```

says `A` can be replaced with `aAb`.

Allowable forms for productions in a context-free grammar:

`1. A → bc`	right side all terminals
`2. A → BCD`	right side all nonterminals
`3. A → B`	right side one nonterminal
`4. A → a`	right side one terminal
`5. A → aBc`	right side terminals and nonterminals
`6. A → aABAb`	left side also on right side
`7. A → λ`	right side lambda

Capital letters are *nonterminals*. Small letters are *nonterminals*. Replacement rules replace the nonterminals.

A *terminal string* is either λ or a string containing only terminals.

Derivation

1) S → AB
2) A → aA
3) A → λ
4) B → bb

Figure 3.1

S
S ⇒ AB
S ⇒ AB ⇒ aAB
S ⇒ AB ⇒ aAB ⇒ aB
S ⇒ AB ⇒ aAB ⇒ aB ⇒ abb (this is a *derivation* of abb from S)

Language defined: set of terminal strings derivable from S.

Examples of Context-free Grammars

1) $S \rightarrow aSb$
2) $S \rightarrow \lambda$

1) $S \rightarrow aS$	1) $S \rightarrow Sa$
2) $S \rightarrow \lambda$	2) $S \rightarrow \lambda$

1) $S \rightarrow aS$	1) $S \rightarrow Sa$	1) $S \rightarrow aA$	1) $S \rightarrow Aa$
2) $S \rightarrow a$	2) $S \rightarrow a$	2) $A \rightarrow aA$	2) $A \rightarrow Aa$
		3) $A \rightarrow \lambda$	3) $A \rightarrow \lambda$

Grammars for Arithmetic Expressions

```
a
(a)
a+b
a*a
a*(b+c)
```

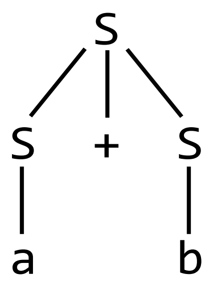
1) S → S+S
2) S → S*S
3) S → (S)
4) S → a
5) S → b
6) S → c

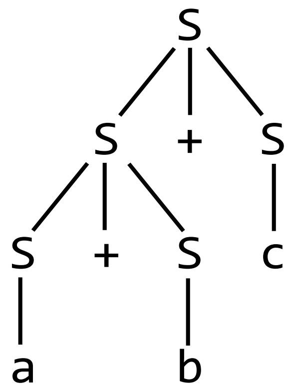
Figure 3.2

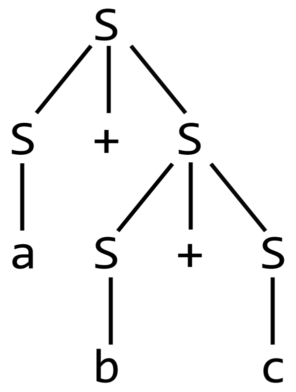
S → S+S | S*S | (S) | a | b | c

S ⇒ S+S ⇒ a+S ⇒ a+b

Parse tree







Ambiguous grammar

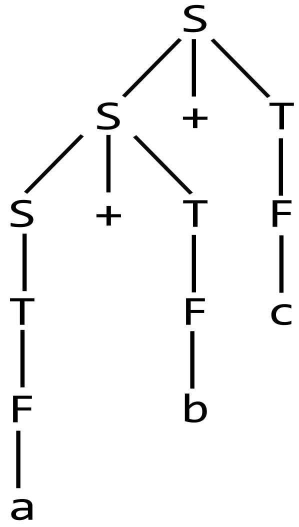
Equivalent unambiguous grammar

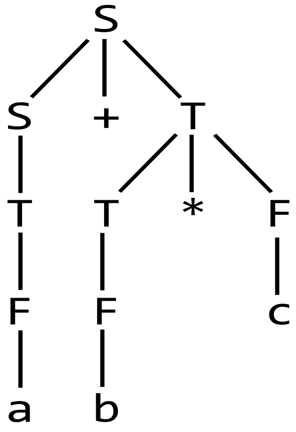
1) S → S+T
2) S → T
3) T → T*F
4) T → F
5) F → a | b | c | (S)

Figure 3.3

-

Here are the only possible parse trees for a+b+c and a+b*c using the grammar in Fig. 3.3:

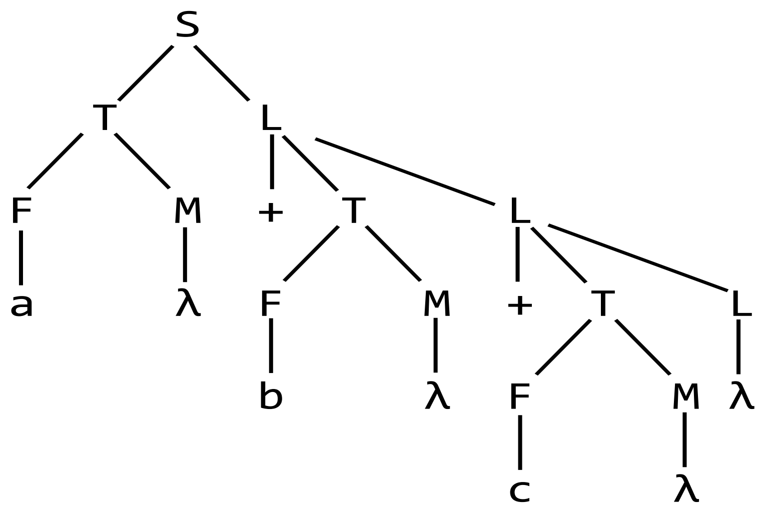




Another unambiguous grammar for arithmetic expressions:

1) S → TL
2) L → +TL
3) L → λ
4) T → FM
5) M → *FM
6) M → λ
7) F → a | b | c | (S)

Figure 3.4



Using star operator in context-free grammars

1) S → TL (generates a leading T)
2) L → +TL (each time used, generates one occcurence of "+T")
3) L → λ (eliminates L)

L ⇒ λ	(use production 3 once)
L ⇒ +TL ⇒ +T	(use production 2 once then production 3 once)
L ⇒ +TL ⇒ +T+TL ⇒ +T+T	(use production 2 twice then production 3 once)

Represent lists using the star operator

S → T(+T)*

T → F(*F)*

1) S → T(+T)*
2) T → F(*F)*
3) F → a | b | c | (S)

Figure 3.5

Use quotes to distinguish terminals

```
1) S → T('+' T)*
2) T → F('*' F)*
3) F → 'a' | 'b' | 'c' | '(' S ')'
```

Figure 3.6

Use angle brackets to allow multi-character terminals

```
1) <expr>      → <term> ('+' <term>)*
2) <term>      → <factor> ('*' <factor>)*
3) <factor>    → 'a' | 'b' | 'c' | '(' <expr> ')'
```

Figure 3.7