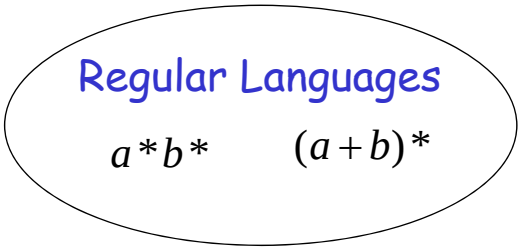
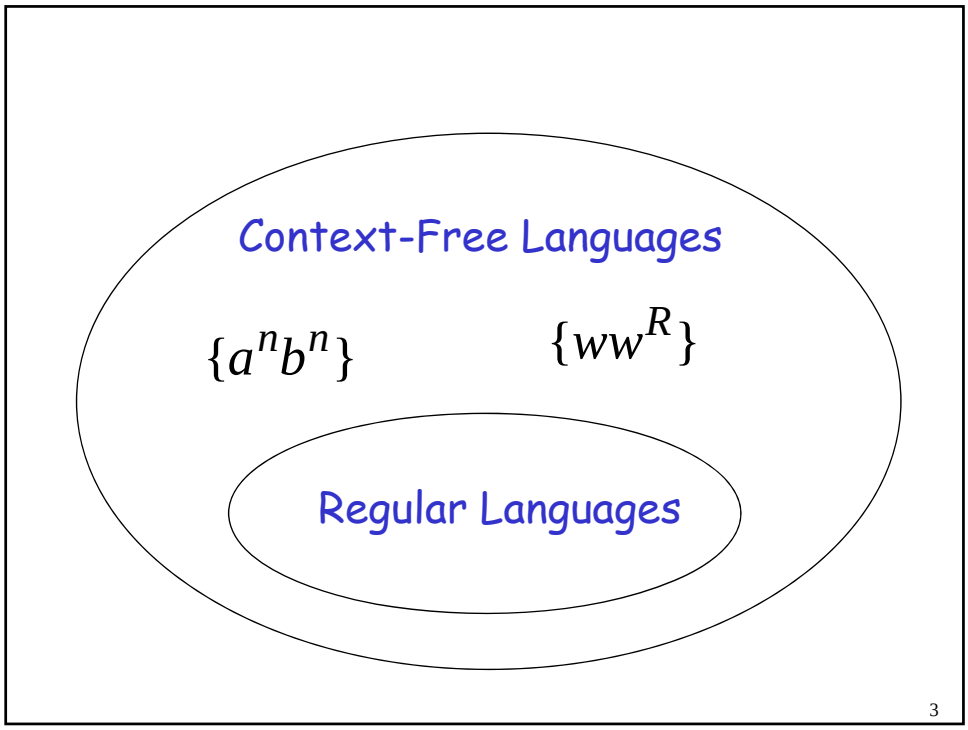
Context-Free Languages

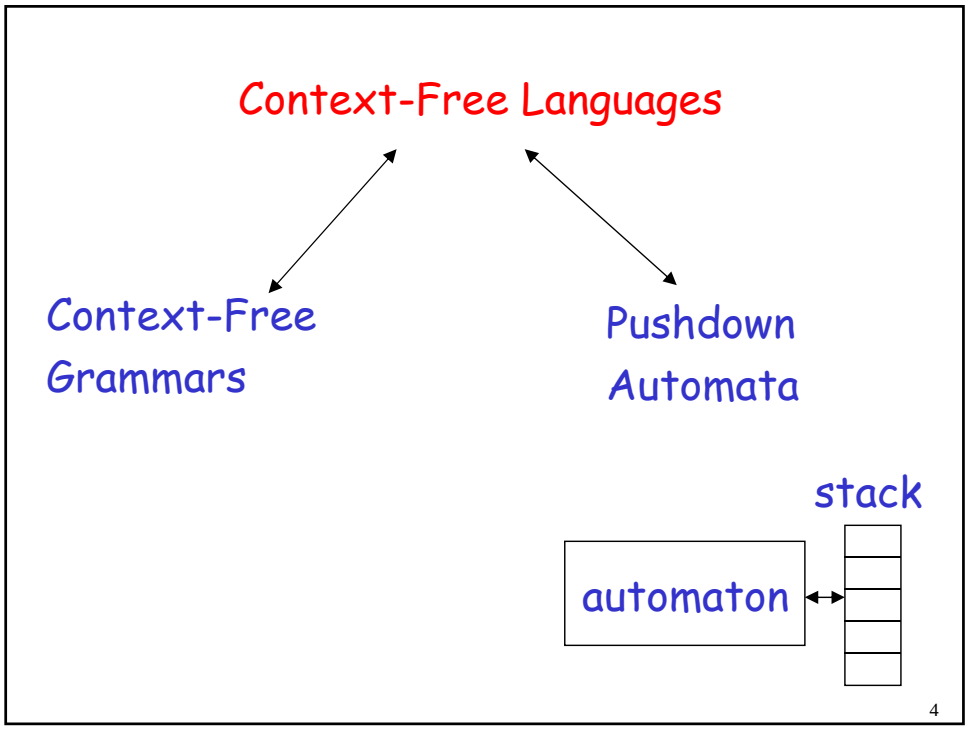
1

$\{a^n b^n : n \geq 0\}$ $\{ww^R\}$



2


Context-Free Languages
$\{a^n b^n\}$
$\{ww^R\}$
Regular Languages
3


Context-Free Languages
Context-Free Grammars
Pushdown Automata
stack
automaton
4

Context-Free Grammars

5

Example

A context-free grammar G:

$$S \to aSb$$

$$S \to \lambda$$

A derivation:

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

6

A context-free grammar G:

$$S \to aSb$$
$$S \to \lambda$$

Another derivation:

$$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb \Rightarrow aaabbb$$

7

$$S \to aSb$$
$$S \to \lambda$$

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

Describes parentheses: (((( ))))

8

Example

A context-free grammar G:

$$S \to aSa$$
$$S \to bSb$$
$$S \to \lambda$$

A derivation:

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

9

A context-free grammar G:

$$S \to aSa$$
$$S \to bSb$$
$$S \to \lambda$$

Another derivation:

$$S \Rightarrow aSa \Rightarrow abSba \Rightarrow abaSaba \Rightarrow abaaba$$

10

$$S \to aSa$$
$$S \to bSb$$
$$S \to \lambda$$

$$L(G) = \{ww^R : \quad w \in \{a,b\}^*\}$$

11

Example

A context-free grammar G: $S \to aSb$

$S \to SS$

$S \to \lambda$

A derivation:

$$S \Rightarrow SS \Rightarrow aSbS \Rightarrow abS \Rightarrow ab$$

12

A context-free grammar G: $S \to aSb$

$S \to SS$

$S \to \lambda$

A derivation:

$$S \Rightarrow SS \Rightarrow aSbS \Rightarrow abS \Rightarrow abaSb \Rightarrow abab$$

13

$S \to aSb$

$S \to SS$

$S \to \lambda$

$$L(G) = \{w \ : n_a(w) = n_b(w), \text{ and } n_a(v) \geq n_b(v) \text{ in any prefix } v\}$$

Describes matched parentheses: () ((( ))) (( ))

14

Definition: Context-Free Grammars

Grammar $G = (V, T, S, P)$

- Variables
- Terminal symbols
- Start variable
- Productions of the form:

$$A \to x$$

Variable — String of variables and terminals

15

$$G = (V, T, S, P)$$

$$L(G) = \{w: \quad S \overset{*}{\Rightarrow} w, \quad w \in T^*\}$$

16

Definition: Context-Free Languages

A language L is context-free

if and only if

there is a context-free grammar G
with $L = L(G)$

17

Derivation Order

1. $S \to AB$
2. $A \to aaA$
3. $A \to \lambda$
4. $B \to Bb$
5. $B \to \lambda$

Leftmost derivation:

$$S \overset{1}{\Rightarrow} AB \overset{2}{\Rightarrow} aaAB \overset{3}{\Rightarrow} aaB \overset{4}{\Rightarrow} aaBb \overset{5}{\Rightarrow} aab$$

Rightmost derivation:

$$S \overset{1}{\Rightarrow} AB \overset{4}{\Rightarrow} ABb \overset{5}{\Rightarrow} Ab \overset{2}{\Rightarrow} aaAb \overset{3}{\Rightarrow} aab$$

18

$$S \to aAB$$

$$A \to bBb$$

$$B \to A \mid \lambda$$

Leftmost derivation:

$$S \Rightarrow aAB \Rightarrow abBbB \Rightarrow abAbB \Rightarrow abbBbbB$$
$$\Rightarrow abbbbB \Rightarrow abbbb$$

Rightmost derivation:

$$S \Rightarrow aAB \Rightarrow aA \Rightarrow abBb \Rightarrow abAb$$
$$\Rightarrow abbBbb \Rightarrow abbbb$$

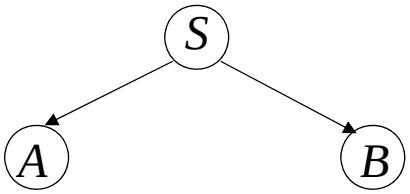
19

Derivation Trees

20

$S \to AB$ $A \to aaA \mid \lambda$ $B \to Bb \mid \lambda$

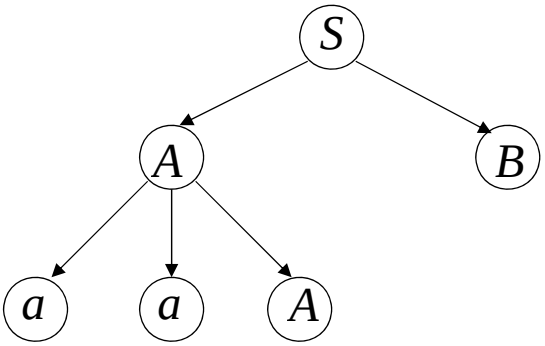
$S \Rightarrow AB$



21

$S \to AB$ $A \to aaA \mid \lambda$ $B \to Bb \mid \lambda$

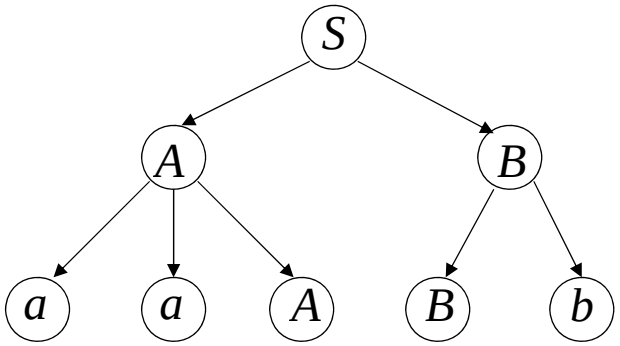
$S \Rightarrow AB \Rightarrow aaAB$



22

$S \to AB \qquad A \to aaA \mid \lambda \qquad B \to Bb \mid \lambda$

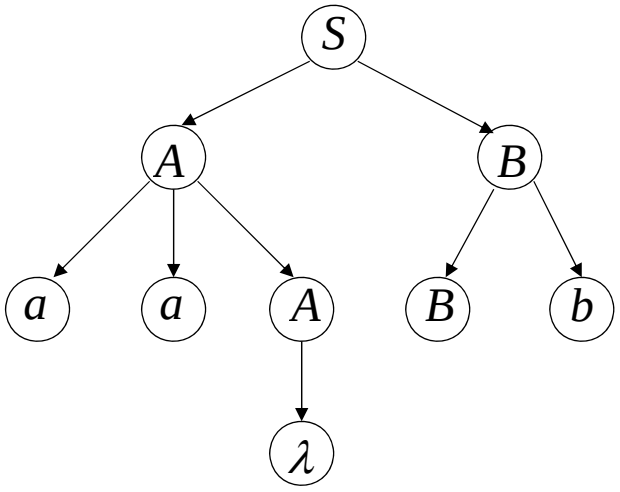
$S \Rightarrow AB \Rightarrow aaAB \Rightarrow aaABb$



23

$S \to AB \qquad A \to aaA \mid \lambda \qquad B \to Bb \mid \lambda$

$S \Rightarrow AB \Rightarrow aaAB \Rightarrow aaABb \Rightarrow aaBb$

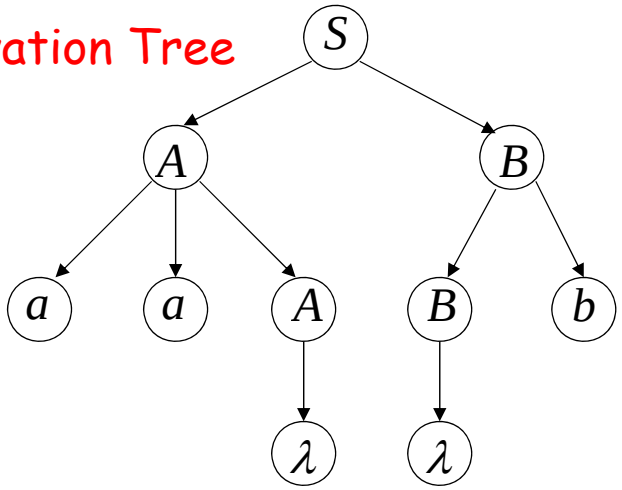


24

$S \to AB \qquad A \to aaA \mid \lambda \qquad B \to Bb \mid \lambda$

$S \Rightarrow AB \Rightarrow aaAB \Rightarrow aaABb \Rightarrow aaBb \Rightarrow aab$

Derivation Tree

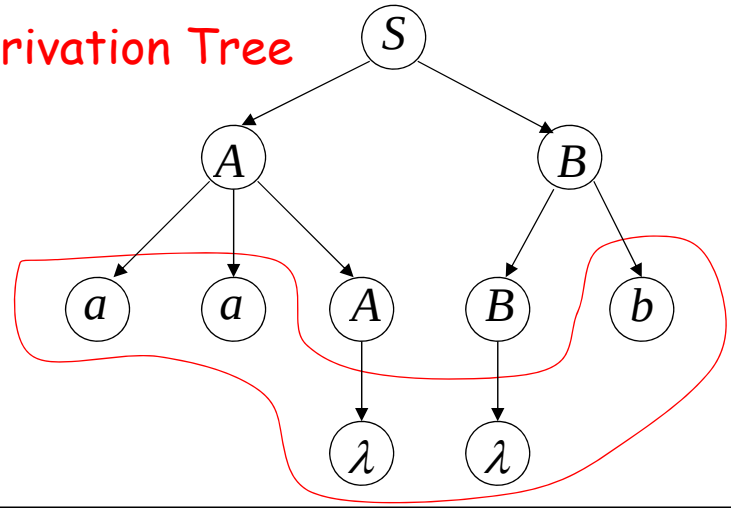


25

$S \to AB \qquad A \to aaA \mid \lambda \qquad B \to Bb \mid \lambda$

$S \Rightarrow AB \Rightarrow aaAB \Rightarrow aaABb \Rightarrow aaBb \Rightarrow aab$

Derivation Tree



yield

$aa\lambda\lambda b$

$= aab$

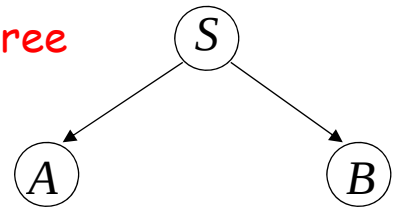
26

Partial Derivation Trees

$S \to AB$ $A \to aaA \mid \lambda$ $B \to Bb \mid \lambda$

$S \Rightarrow AB$

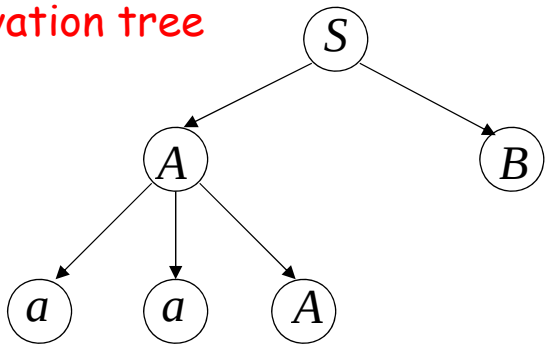
Partial derivation tree



27

$S \Rightarrow AB \Rightarrow aaAB$

Partial derivation tree

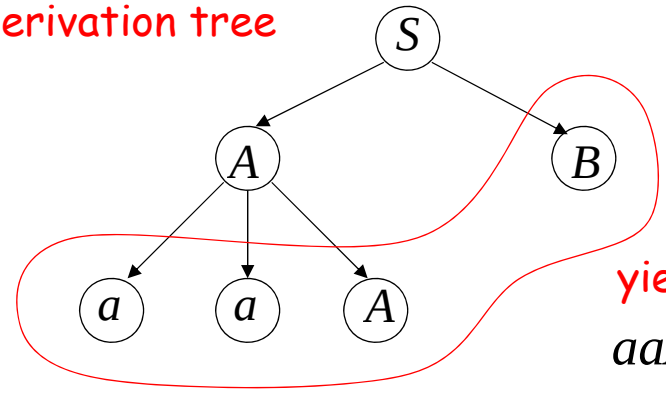


28

$$S \Rightarrow AB \Rightarrow aaAB$$

sentential form

Partial derivation tree



yield

$aaAB$

29

Sometimes, derivation order doesn't matter

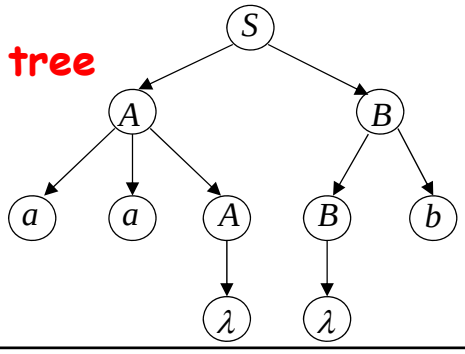
Leftmost:

$$S \Rightarrow AB \Rightarrow aaAB \Rightarrow aaB \Rightarrow aaBb \Rightarrow aab$$

Rightmost:

$$S \Rightarrow AB \Rightarrow ABb \Rightarrow Ab \Rightarrow aaAb \Rightarrow aab$$

Same derivation tree

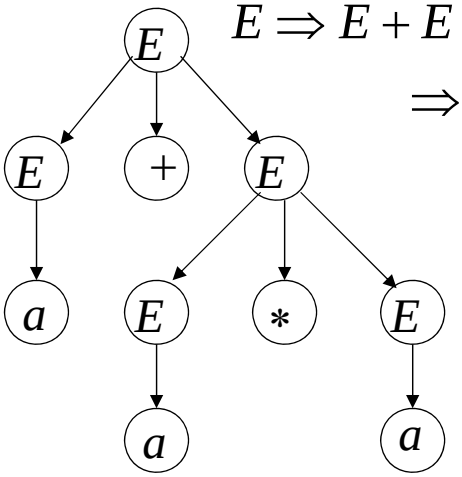


30

Ambiguity

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

$a + a * a$



$E \Rightarrow E + E \Rightarrow a + E \Rightarrow a + E * E$

$\Rightarrow a + a * E \Rightarrow a + a * a$

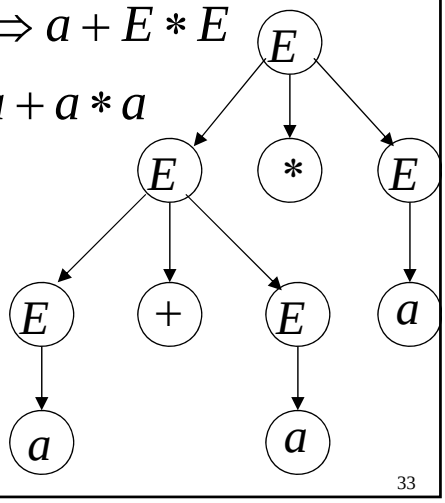
leftmost derivation

$$E \to E + E \mid E * E \mid (E) \mid a$$

$$a + a * a$$

$$E \Rightarrow E * E \Rightarrow E + E * E \Rightarrow a + E * E$$
$$\Rightarrow a + a * E \Rightarrow a + a * a$$

leftmost derivation

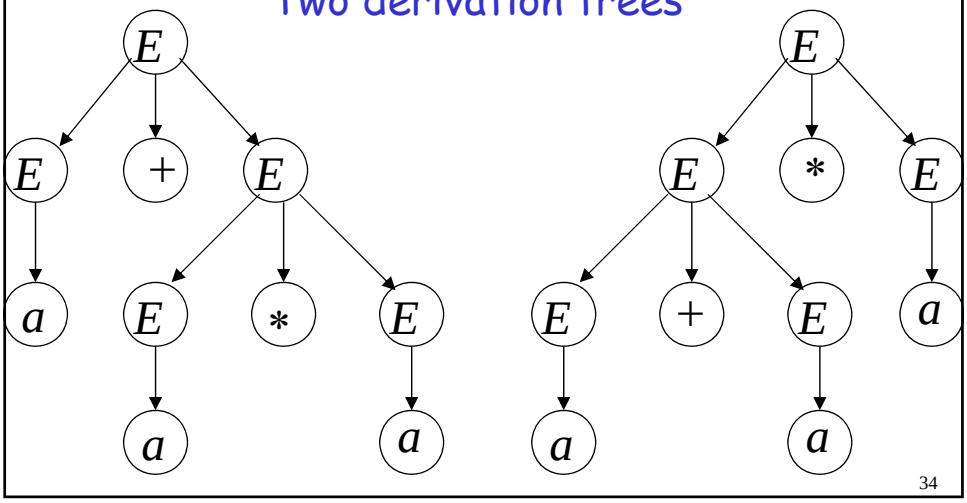


33

$$E \to E + E \mid E * E \mid (E) \mid a$$

$$a + a * a$$

Two derivation trees

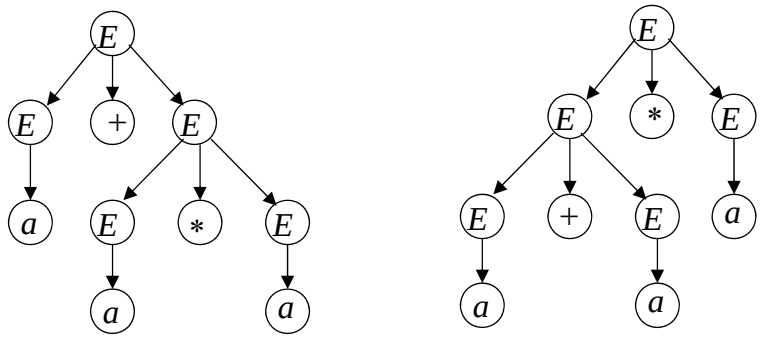


34

The grammar $E \to E + E \mid E * E \mid (E) \mid a$

is ambiguous:

string $a + a * a$ has two derivation trees



35

The grammar $E \to E + E \mid E * E \mid (E) \mid a$

is ambiguous:

string $a + a * a$ has two leftmost derivations

$$E \Rightarrow E + E \Rightarrow a + E \Rightarrow a + E * E$$
$$\Rightarrow a + a * E \Rightarrow a + a * a$$

$$E \Rightarrow E * E \Rightarrow E + E * E \Rightarrow a + E * E$$
$$\Rightarrow a + a * E \Rightarrow a + a * a$$

36

Definition:

A context-free grammar G is **ambiguous**

if some string $w \in L(G)$ has:

two or more derivation trees

37

In other words:

A context-free grammar G is **ambiguous**

if some string $w \in L(G)$ has:

two or more leftmost derivations

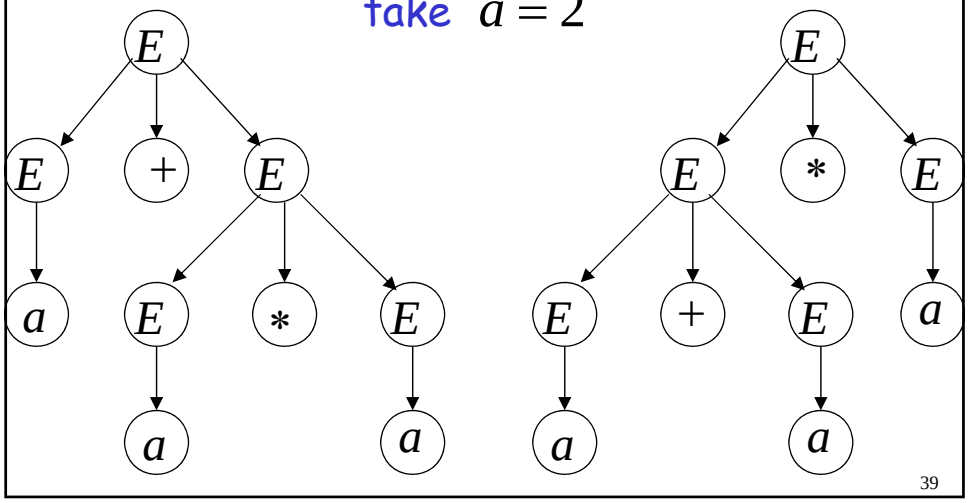
(or rightmost)

38

Why do we care about ambiguity?

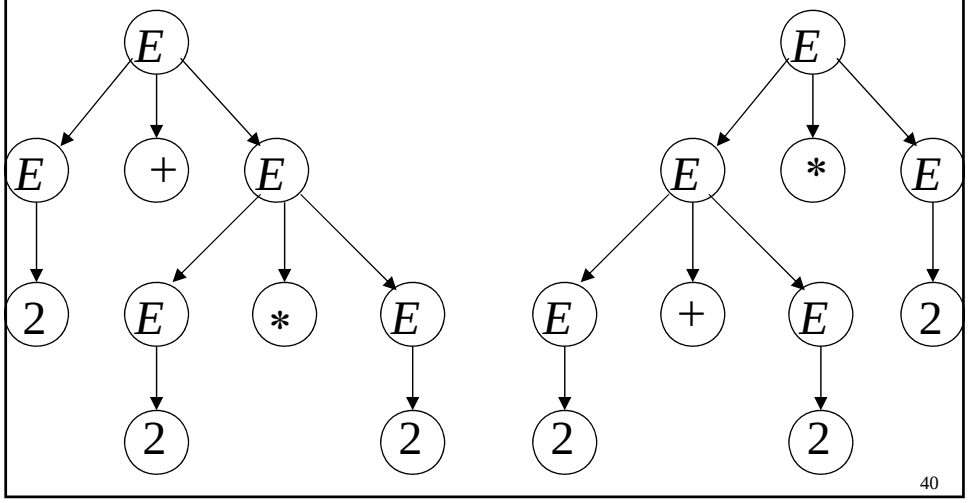
$$a + a * a$$

take $a = 2$



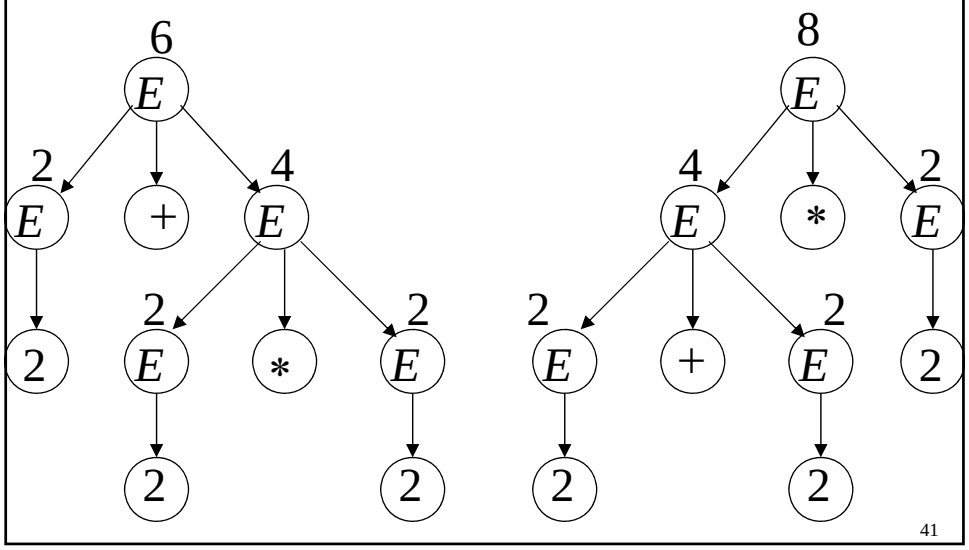
39

$$2 + 2 * 2$$



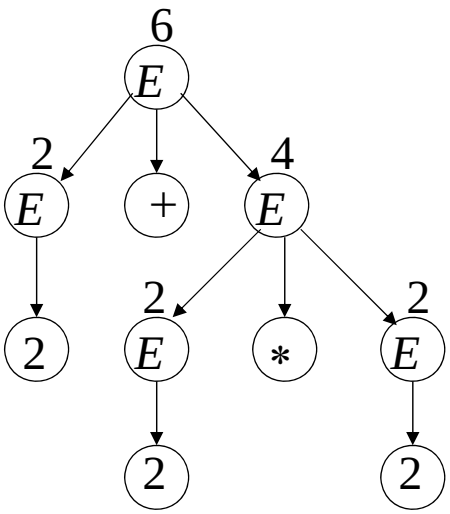
40

$2+2*2=6$ $2+2*2=8$



41

Correct result: $2+2*2=6$



42

- Ambiguity is **bad** for programming languages

- We want to remove ambiguity

43

We fix the ambiguous grammar:

$$E \to E + E \mid E * E \mid (E) \mid a$$

New non-ambiguous grammar:

$$E \to E + T$$
$$E \to T$$
$$T \to T * F$$
$$T \to F$$
$$F \to (E)$$
$$F \to a$$

44

$$E \Rightarrow E+T \Rightarrow T+T \Rightarrow F+T \Rightarrow a+T \Rightarrow a+T*F$$
$$\Rightarrow a+F*F \Rightarrow a+a*F \Rightarrow a+a*a$$

$E \rightarrow E+T$

$E \rightarrow T$

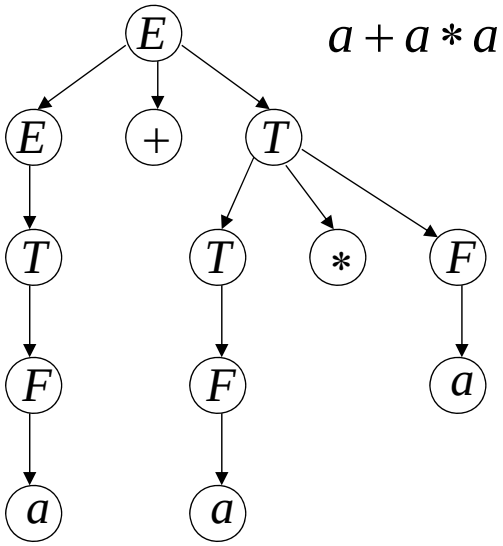
$T \rightarrow T*F$

$T \rightarrow F$

$F \rightarrow (E)$

$F \rightarrow a$

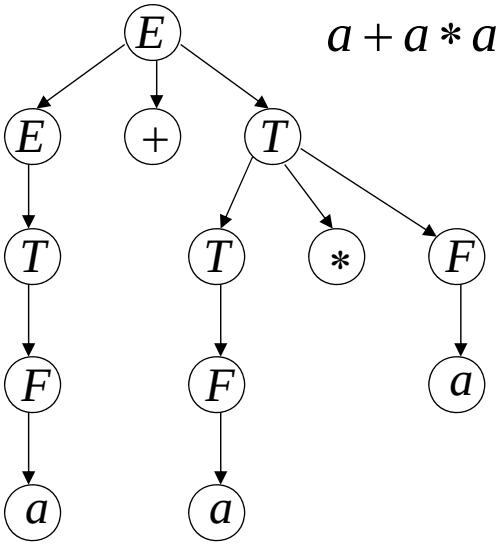
$a+a*a$



45

Unique derivation tree

$a+a*a$



46

The grammar G:

$$E \to E + T$$
$$E \to T$$
$$T \to T * F$$
$$T \to F$$
$$F \to (E)$$
$$F \to a$$

is non-ambiguous:

Every string $w \in L(G)$ has a unique derivation tree

47

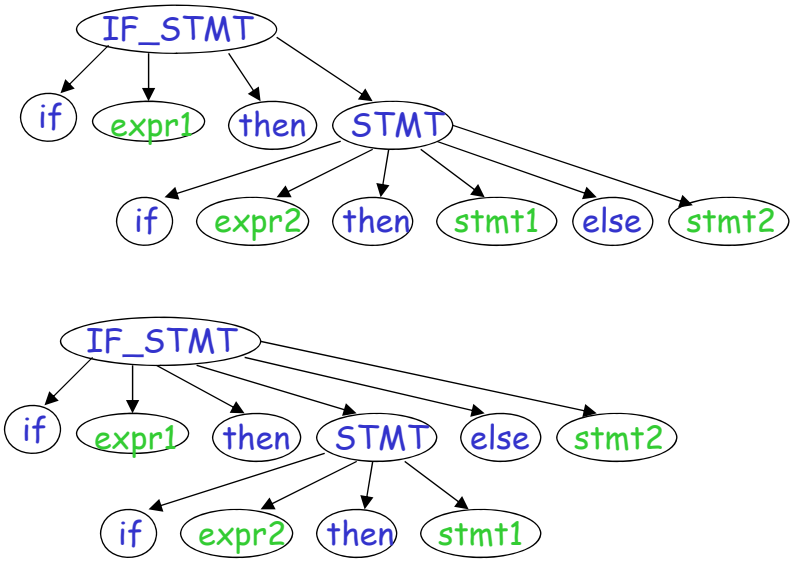
Another Ambiguous Grammar

IF_STMT → if EXPR then STMT

| if EXPR then STMT else STMT

48

If expr1 then if expr2 then stmt1 else stmt2



49

Inherent Ambiguity

Some context free languages
have only ambiguous grammars

Example: $L = \{a^n b^n c^m\} \cup \{a^n b^m c^m\}$

$S \to S_1 \mid S_2$

$S_1 \to S_1 c \mid A$

$A \to aAb \mid \lambda$

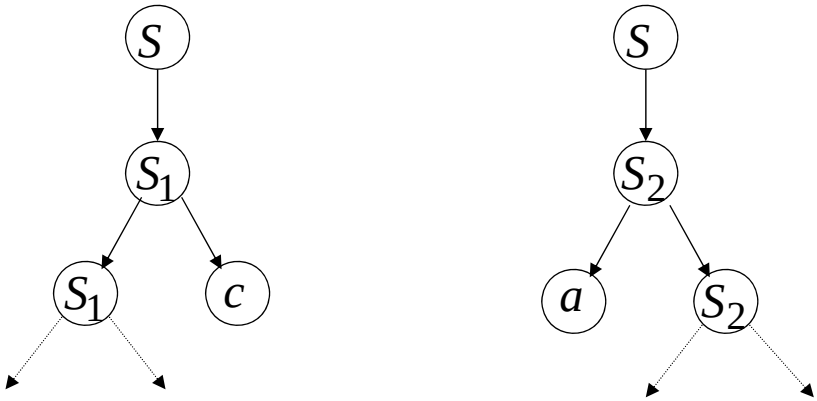
$S_2 \to aS_2 \mid B$

$B \to bBc \mid \lambda$

50

The string $a^n b^n c^n$

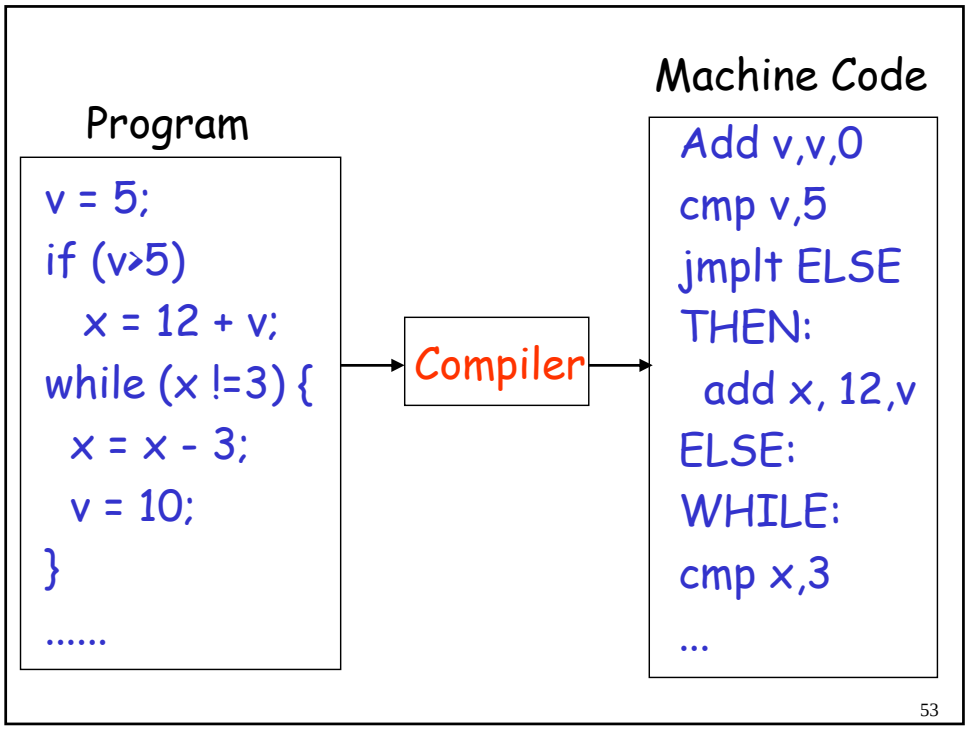
has two derivation trees

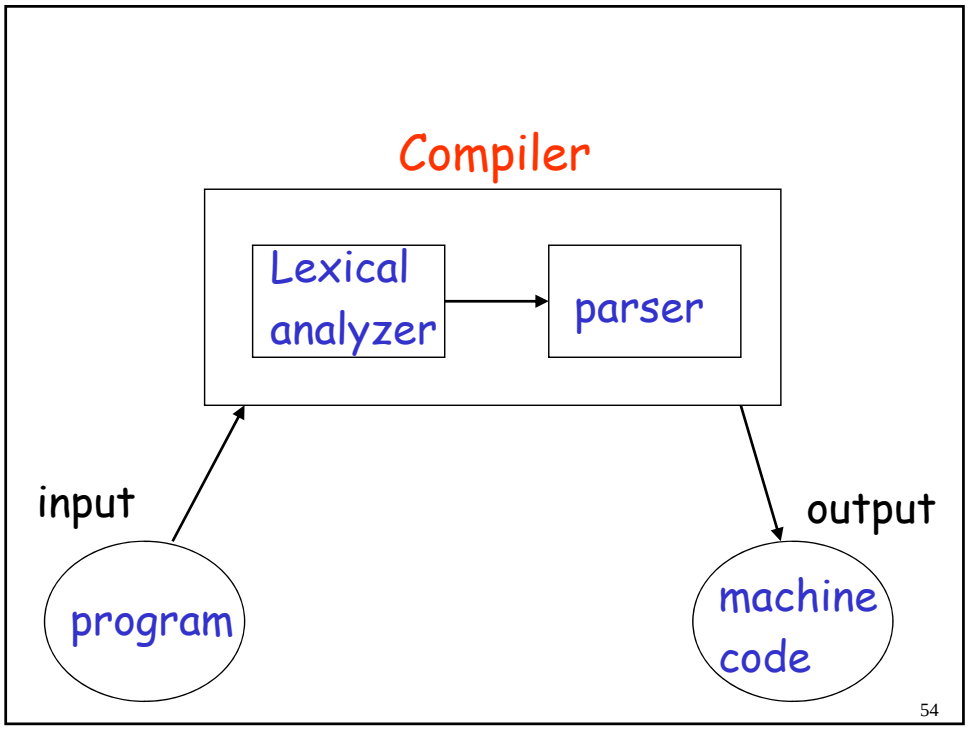


51

Compilers

52


Program
v = 5;
if (v>5)
x = 12 + v;
while (x !=3) {
x = x - 3;
v = 10;
}
.....
Compiler
Machine Code
Add v,v,0
cmp v,5
jmplt ELSE
THEN:
add x, 12,v
ELSE:
WHILE:
cmp x,3
...
53


Compiler
Lexical analyzer
parser
input
output
program
machine code
54

A parser knows the grammar
of the programming language

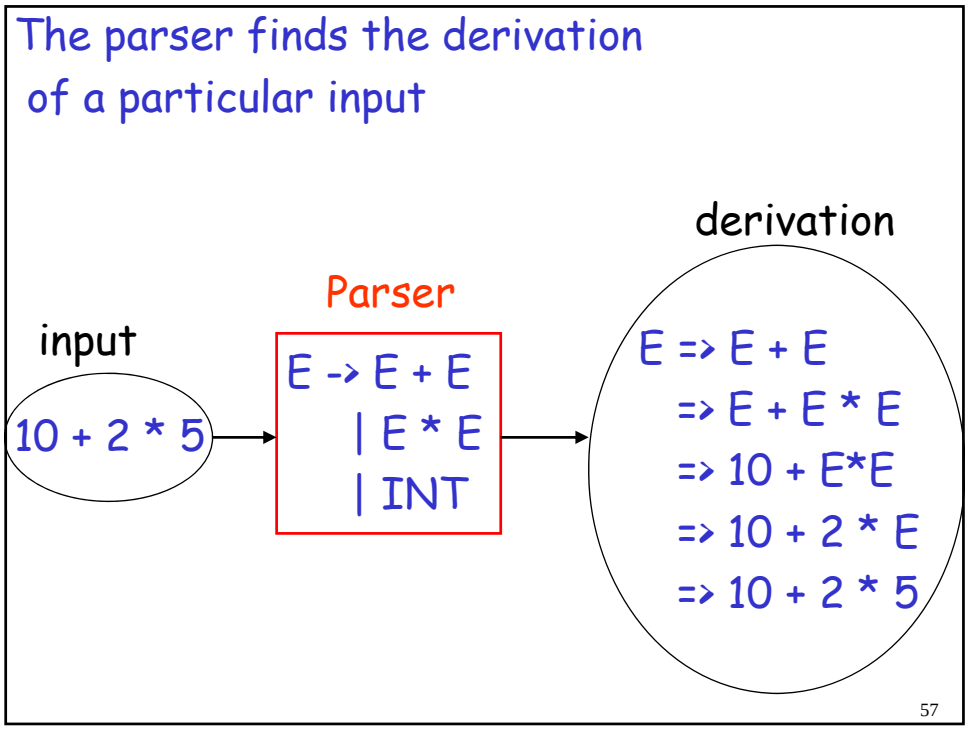
55

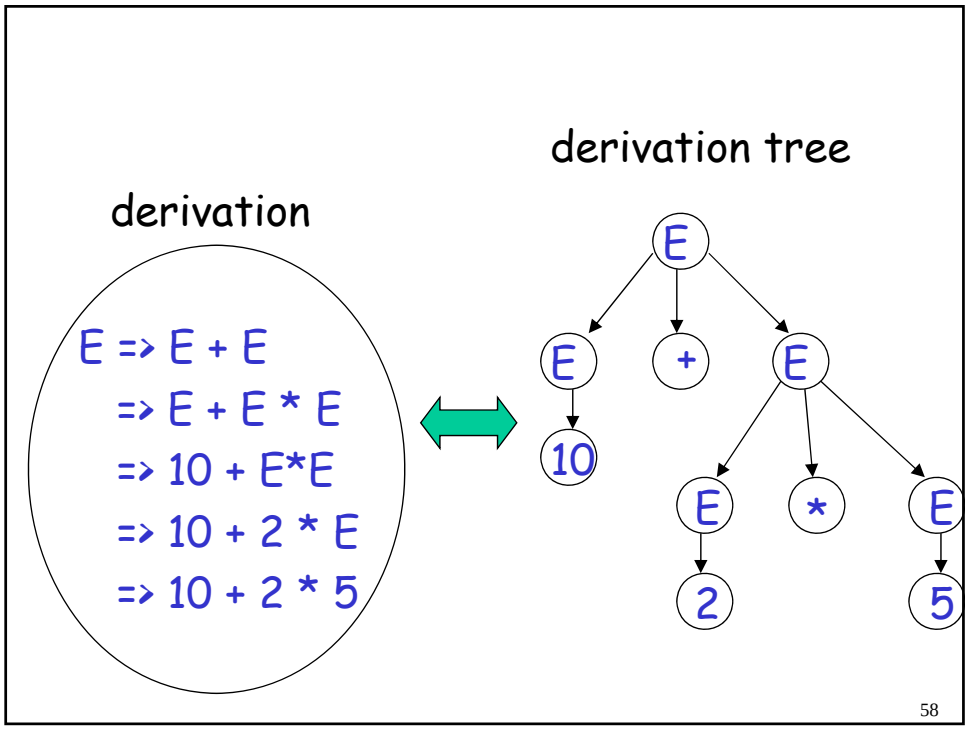
Parser

PROGRAM → STMT_LIST
STMT_LIST → STMT; STMT_LIST | STMT;
STMT → EXPR | IF_STMT | WHILE_STMT
| { STMT_LIST }

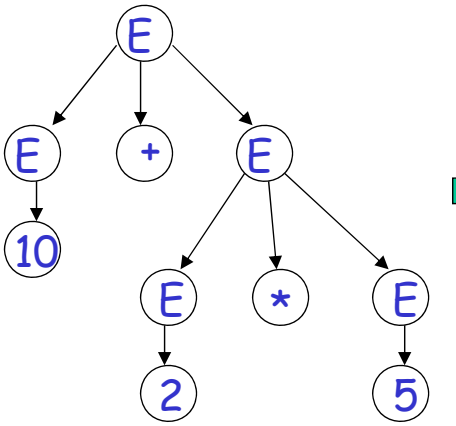
EXPR → EXPR + EXPR | EXPR - EXPR | ID
IF_STMT → if (EXPR) then STMT
| if (EXPR) then STMT else STMT
WHILE_STMT → while (EXPR) do STMT

56


The parser finds the derivation
of a particular input
input
10 + 2 * 5
Parser
E -> E + E
| E * E
| INT
derivation
E => E + E
=> E + E * E
=> 10 + E*E
=> 10 + 2 * E
=> 10 + 2 * 5
57


derivation
E => E + E
=> E + E * E
=> 10 + E*E
=> 10 + 2 * E
=> 10 + 2 * 5
derivation tree
E
E
+
E
10
E
*
E
2
5
58

derivation tree



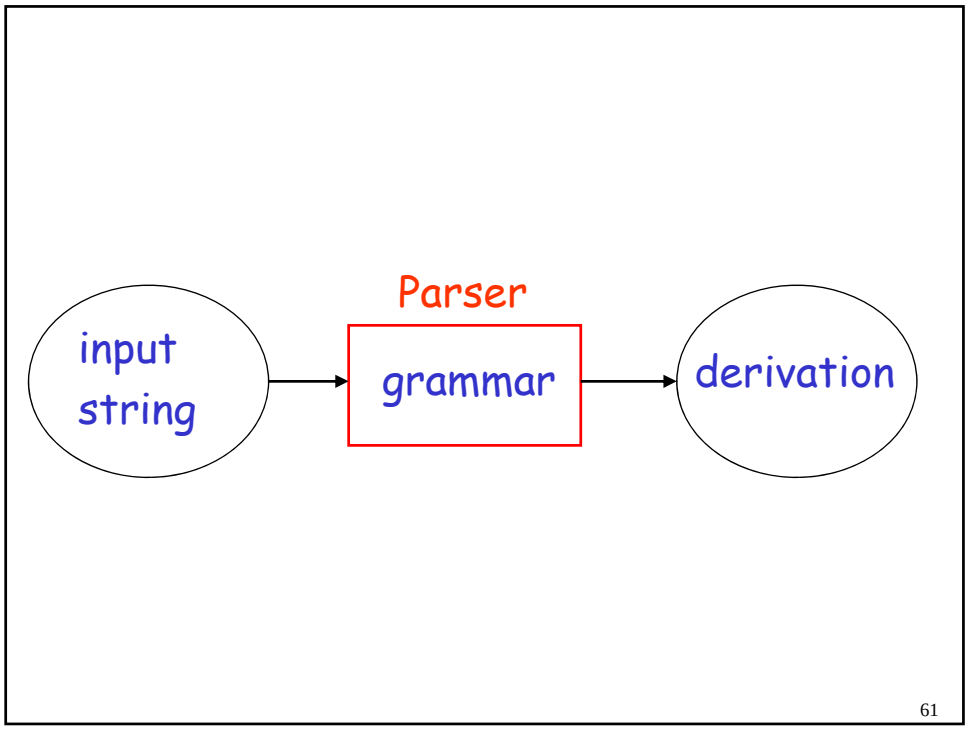
machine code

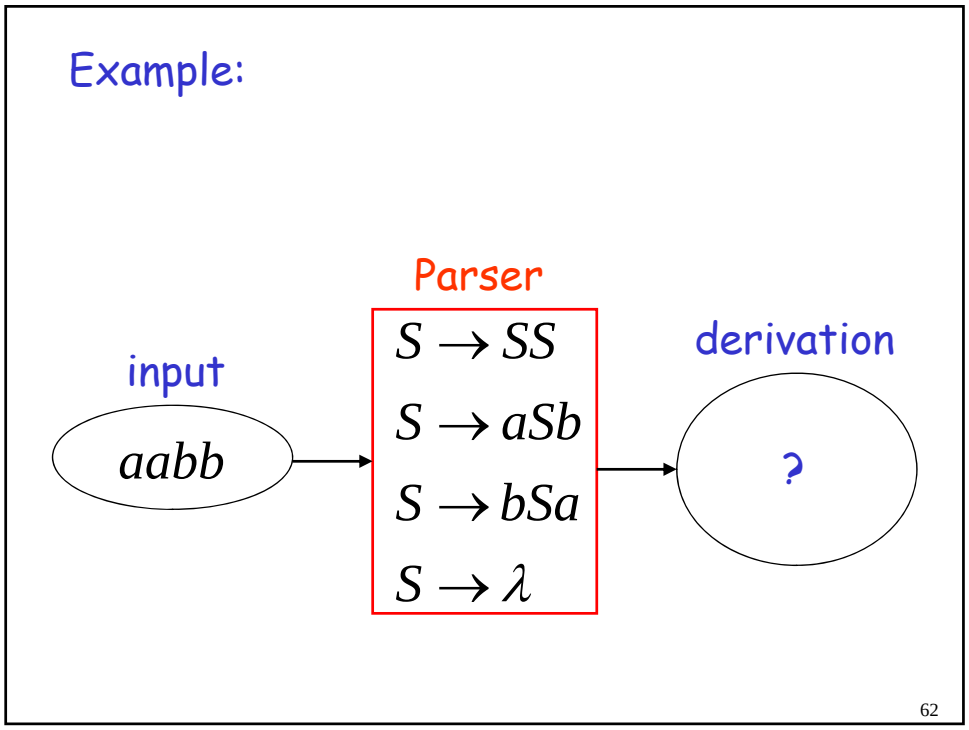
mult a, 2, 5
add b, 10, a

59

Parsing

60


Parser
input string
grammar
derivation
61


Example:
Parser
input
aabb
$S \to SS$
$S \to aSb$
$S \to bSa$
$S \to \lambda$
derivation
?
62

Exhaustive Search

$$S \to SS \mid aSb \mid bSa \mid \lambda$$

Phase 1:

$S \Rightarrow SS$

$S \Rightarrow aSb$

$S \Rightarrow bSa$

$S \Rightarrow \lambda$

Find derivation of

$aabb$

All possible derivations of length 1

63

$S \Rightarrow SS$

$S \Rightarrow aSb$

~~$S \Rightarrow bSa$~~

~~$S \Rightarrow \lambda$~~

$aabb$

64

Phase 2 $S \to SS \mid aSb \mid bSa \mid \lambda$

$S \Rightarrow SS \Rightarrow SSS$

$S \Rightarrow SS \Rightarrow aSbS$ $aabb$

Phase 1 ~~$S \Rightarrow SS \Rightarrow bSaS$~~

$S \Rightarrow SS$ $S \Rightarrow SS \Rightarrow S$

$S \Rightarrow aSb$ $S \Rightarrow aSb \Rightarrow aSSb$

$S \Rightarrow aSb \Rightarrow aaSbb$

~~$S \Rightarrow aSb \Rightarrow abSab$~~

~~$S \Rightarrow aSb \Rightarrow ab$~~

65

$S \to SS \mid aSb \mid bSa \mid \lambda$

Phase 2

$S \Rightarrow SS \Rightarrow SSS$

$S \Rightarrow SS \Rightarrow aSbS$ $aabb$

$S \Rightarrow SS \Rightarrow S$

$S \Rightarrow aSb \Rightarrow aSSb$

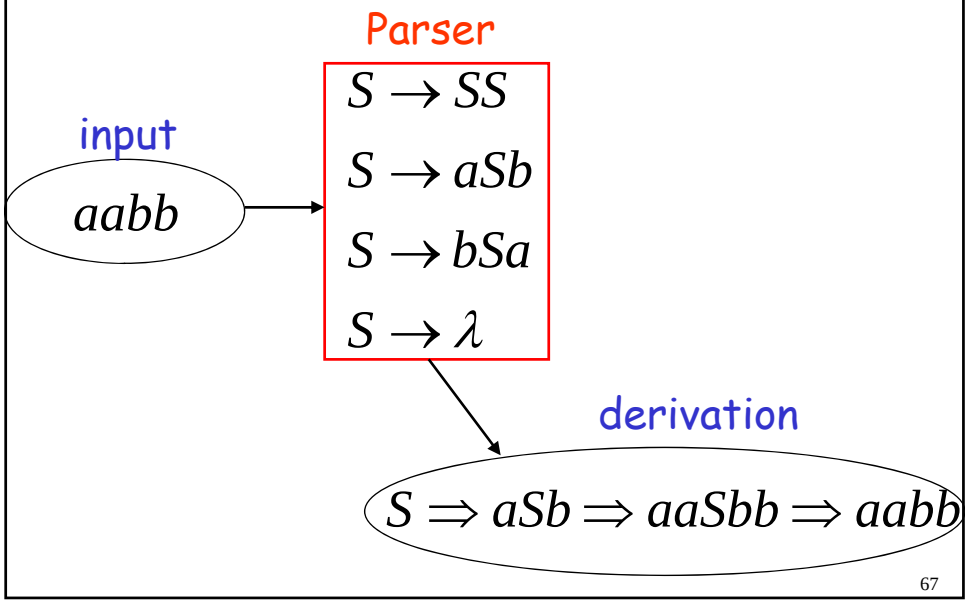
$S \Rightarrow aSb \Rightarrow aaSbb$

Phase 3

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

66

Final result of exhaustive search (top-down parsing)



67

Time complexity of exhaustive search

Suppose there are no productions of the form

$$A \to \lambda$$

$$A \to B$$

Number of phases for string w : $2|w|$

68

For grammar with k rules

Time for phase 1: k

k possible derivations

69

Time for phase 2: k^2

k^2 possible derivations

70

Time for phase $2|w|$: $k^{2|w|}$

$k^{2|w|}$ possible derivations

71

Total time needed for string w:

$$k + k^2 + \cdots + k^{2|w|}$$

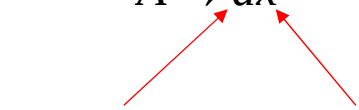
phase 1 phase 2 phase 2|w|

Extremely bad!!!

72

There exist faster algorithms
for specialized grammars

S-grammar: $A \to ax$



Pair (A, a) appears once

73

S-grammar example:

$$S \to aS$$
$$S \to bSS$$
$$S \to c$$

Each string has a unique derivation

$$S \Rightarrow aS \Rightarrow abSS \Rightarrow abcS \Rightarrow abcc$$

74

For S-grammars:

In the exhaustive search parsing
there is only one choice in each phase

Time for a phase: 1

Total time for parsing string w: $|w|$

75

For general context-free grammars:

There exists a parsing algorithm
that parses a string $|w|$
in time $|w|^3$

(we will show it in the next class)

76