Context-Free Languages

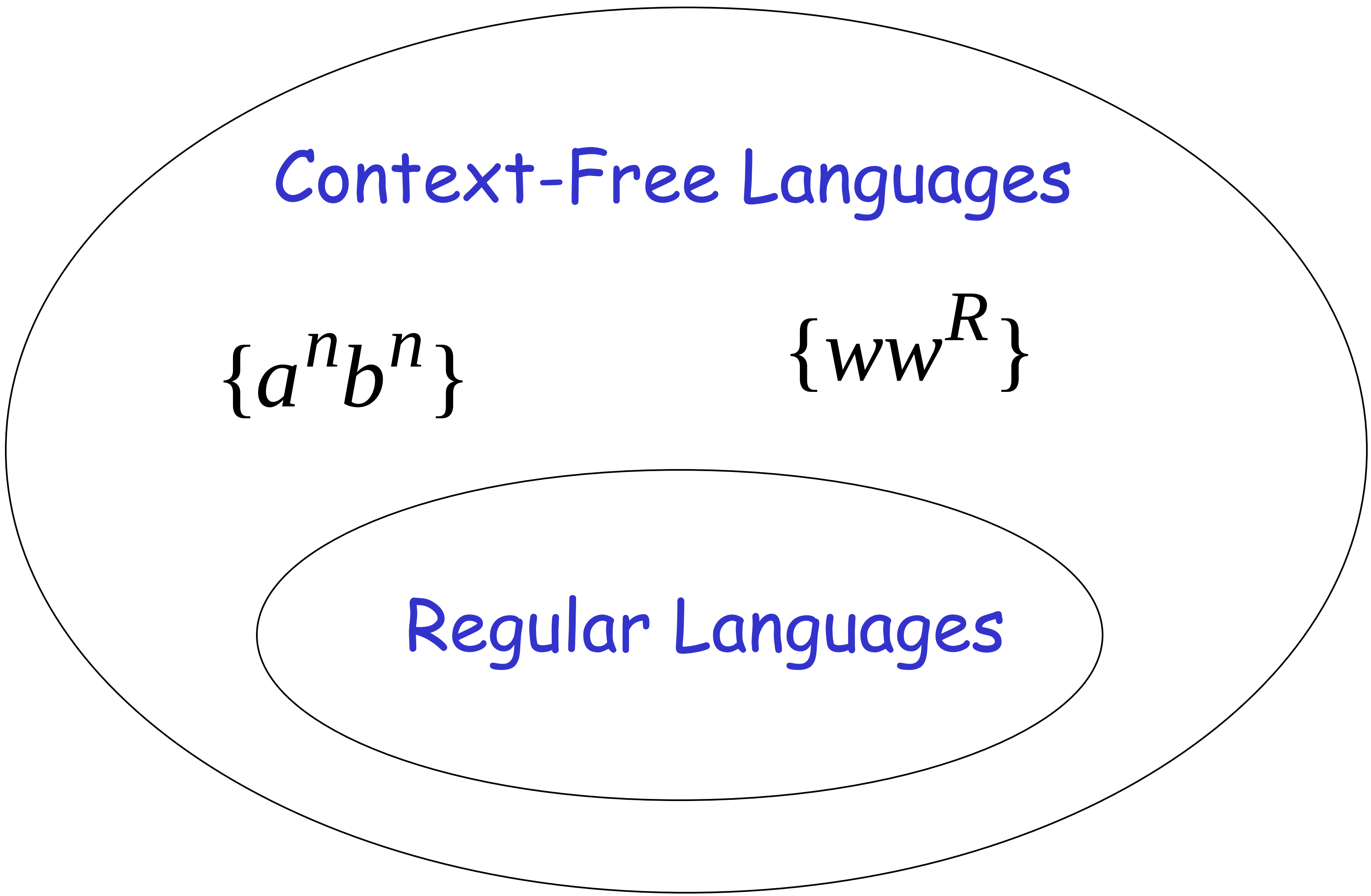
$\{a^n b^n : n \geq 0\}$

$\{ww^R\}$

Regular Languages

a^*b^*

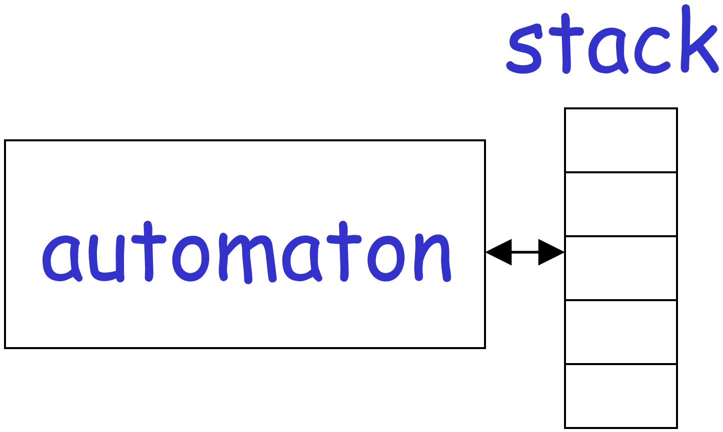
$(a+b)^*$


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

Context-Free Languages

Context-Free Grammars

Pushdown Automata



Context-Free Grammars

Example

A context-free grammar G:

$$S \rightarrow aSb$$

$$S \rightarrow \lambda$$

A derivation:

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

A context-free grammar G:

$$S \to aSb$$

$$S \to \lambda$$

Another derivation:

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

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

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

Describes parentheses: (((( ))))

Example

A context-free grammar G:

$$S \rightarrow aSa$$

$$S \rightarrow bSb$$

$$S \rightarrow \lambda$$

A derivation:

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

A context-free grammar G:

$$S \to aSa$$

$$S \to bSb$$

$$S \to \lambda$$

Another derivation:

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

$$S \to aSa$$

$$S \to bSb$$

$$S \to \lambda$$

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

Example

A context-free grammar G:

$$S \rightarrow aSb$$

$$S \rightarrow SS$$

$$S \rightarrow \lambda$$

A derivation:

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

A context-free grammar G:

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

A derivation:

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

$$S \to aSb$$

$$S \to SS$$

$$S \to \lambda$$

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

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

Definition: Context-Free Grammars

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

Variables

Terminal symbols

Start variable

Productions of the form:

$$A \rightarrow x$$

Variable

String of variables and terminals

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

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

Definition: Context-Free Languages

A language L is context-free

if and only if

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

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:

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

$$S \to aAB$$

$$A \to bBb$$

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

Leftmost derivation:

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

$$\Rightarrow abbbbB \Rightarrow abbbb$$

Rightmost derivation:

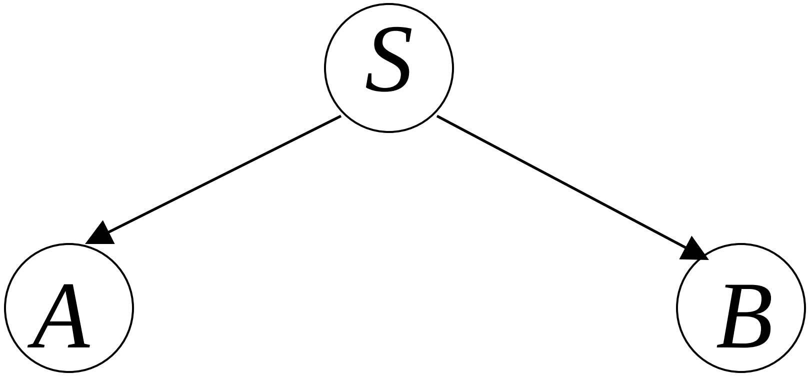
$$S \Rightarrow aAB \Rightarrow aA \Rightarrow abBb \Rightarrow abAb$$

$$\Rightarrow abbBbb \Rightarrow abbbb$$

Derivation Trees

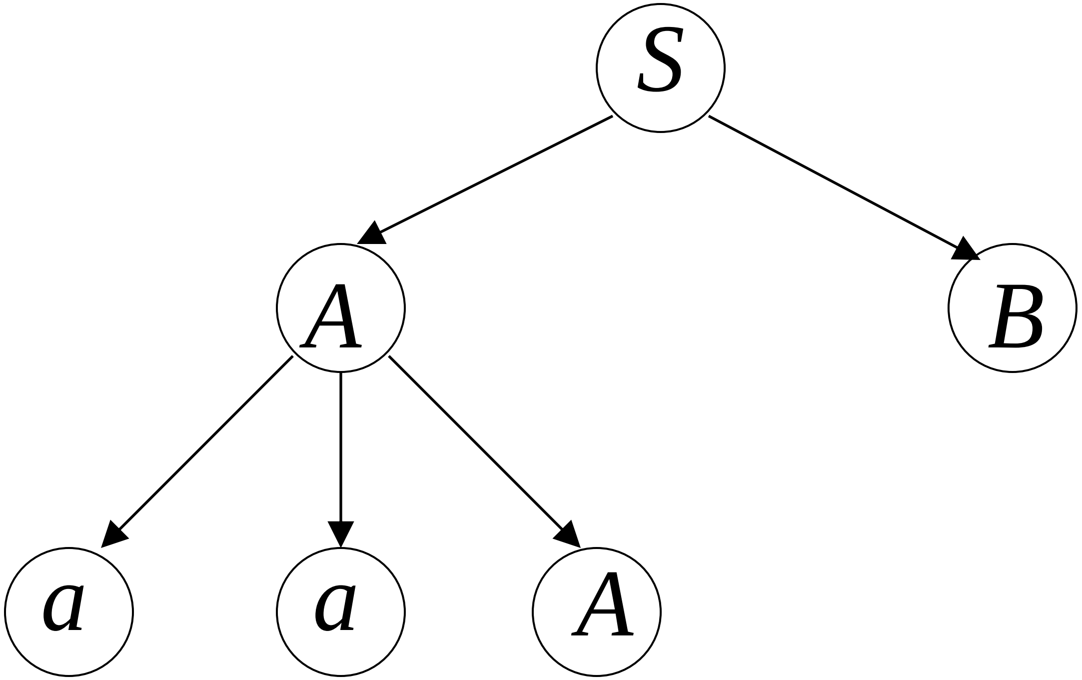
$$S \to AB \qquad A \to aaA \mid \lambda \qquad B \to Bb \mid \lambda$$

$$S \Rightarrow AB$$



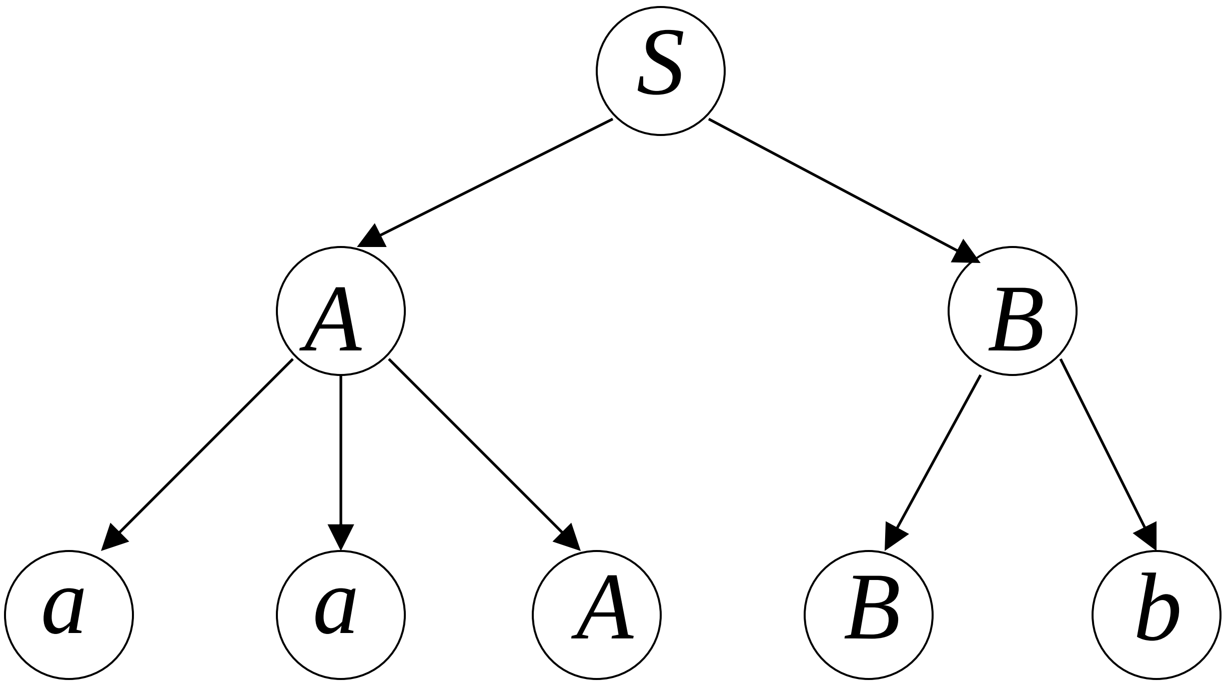
$$S \to AB \qquad A \to aaA \mid \lambda \qquad B \to Bb \mid \lambda$$

$$S \Rightarrow AB \Rightarrow aaAB$$



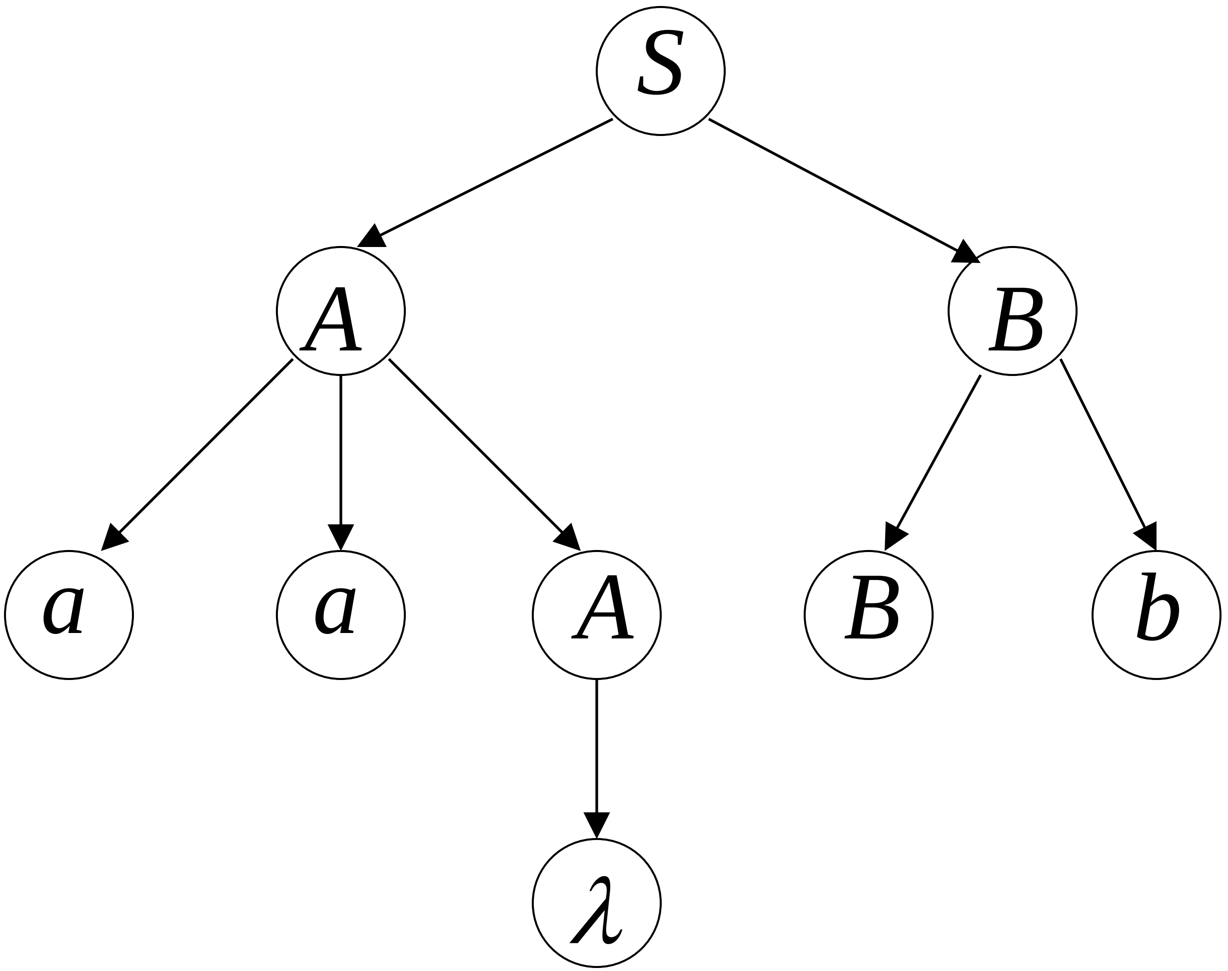
$$S \to AB \qquad A \to aaA \mid \lambda \qquad B \to Bb \mid \lambda$$

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



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

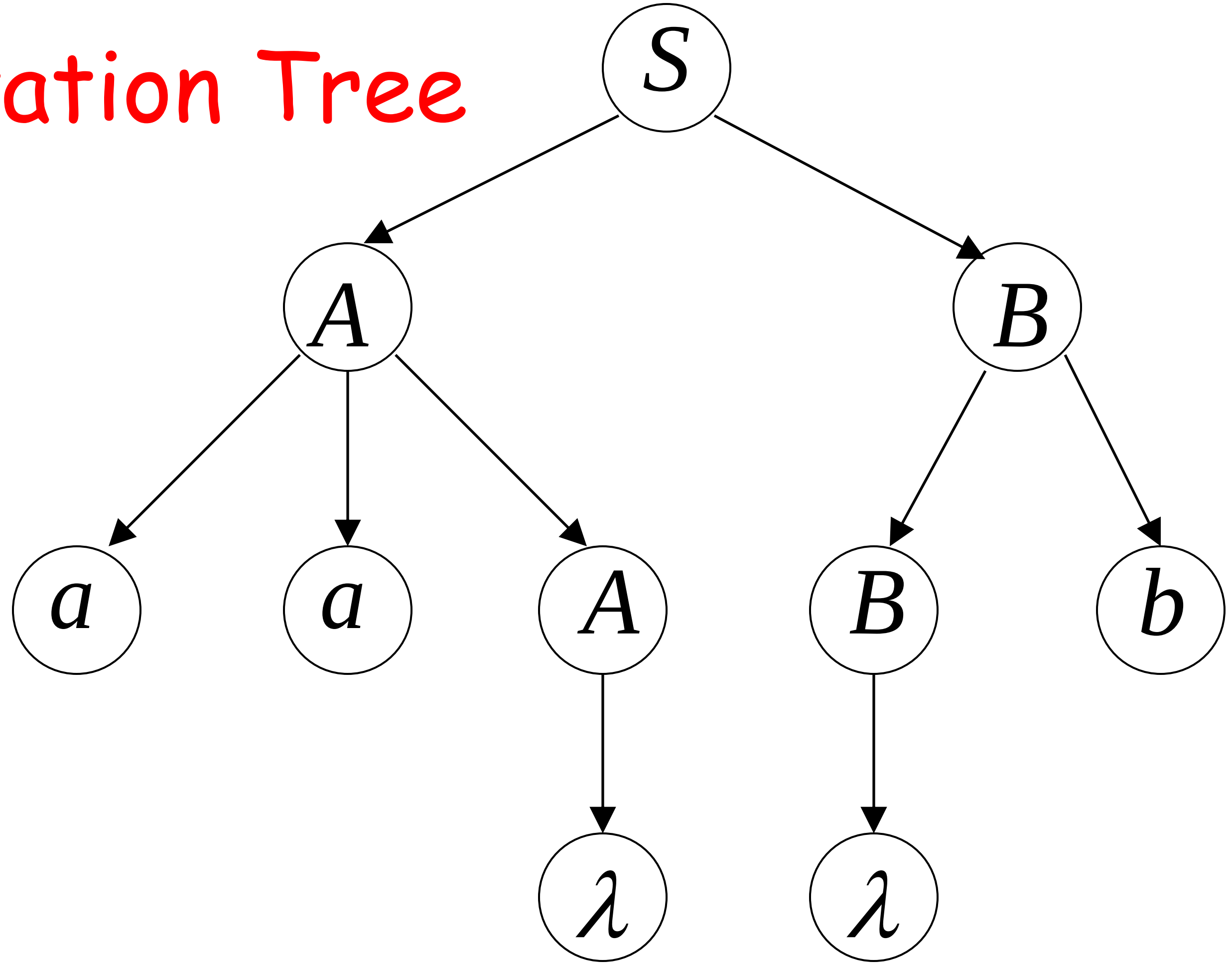
$$S \Rightarrow AB \Rightarrow aaAB \Rightarrow aaABb \Rightarrow aaBb$$



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

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

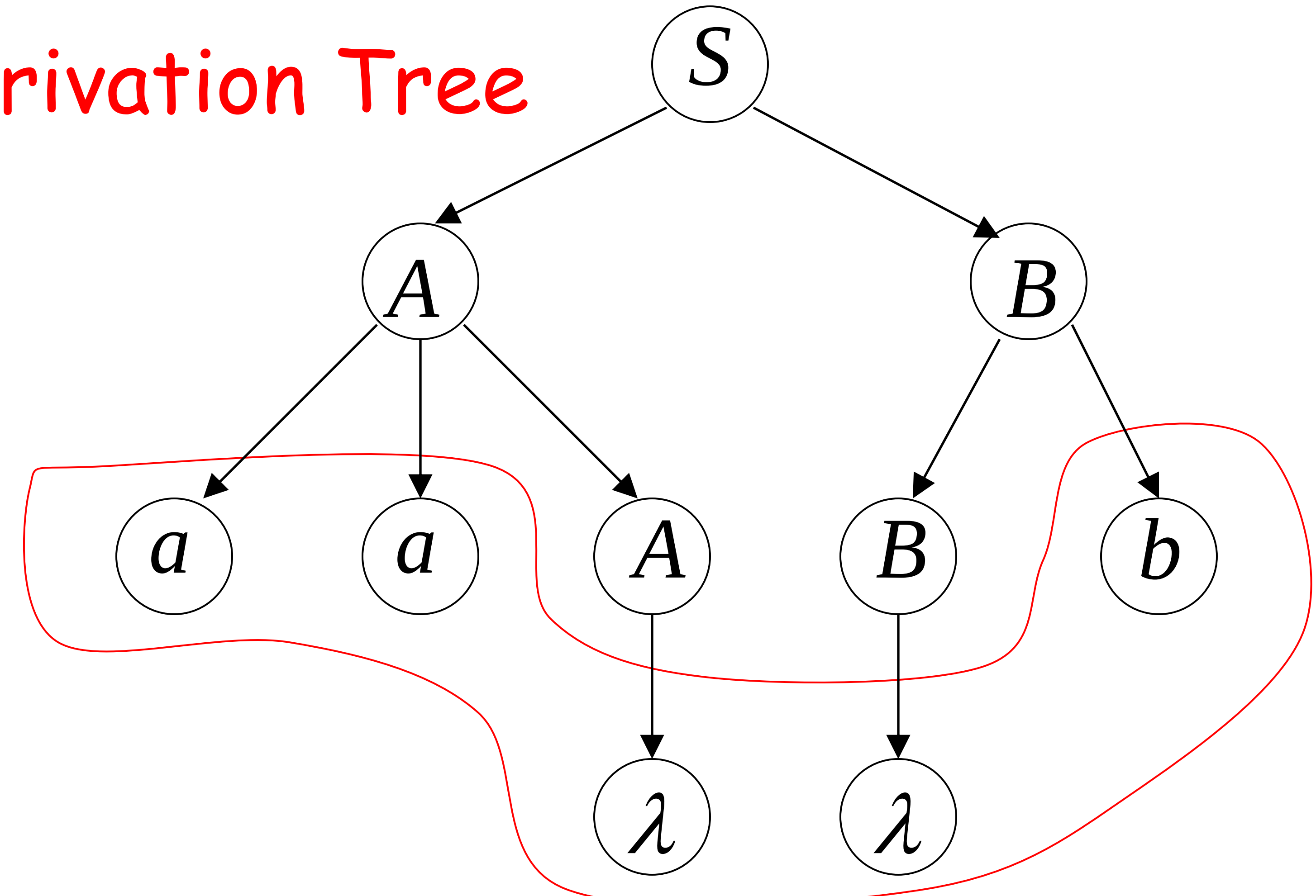
Derivation Tree



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

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

Derivation Tree



yield

$aa\lambda\lambda b$

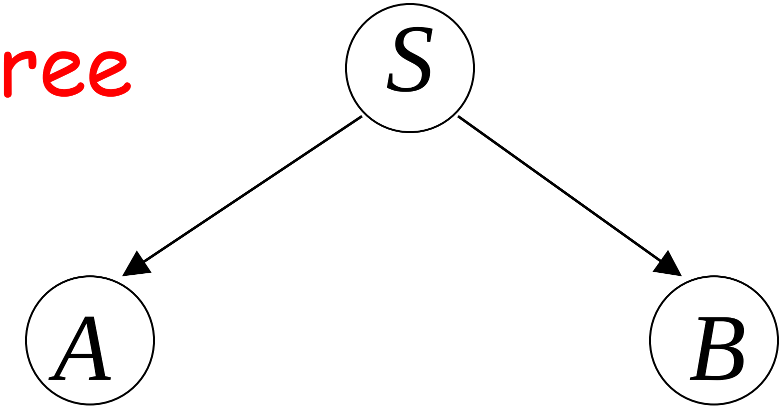
$= aab$

Partial Derivation Trees

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

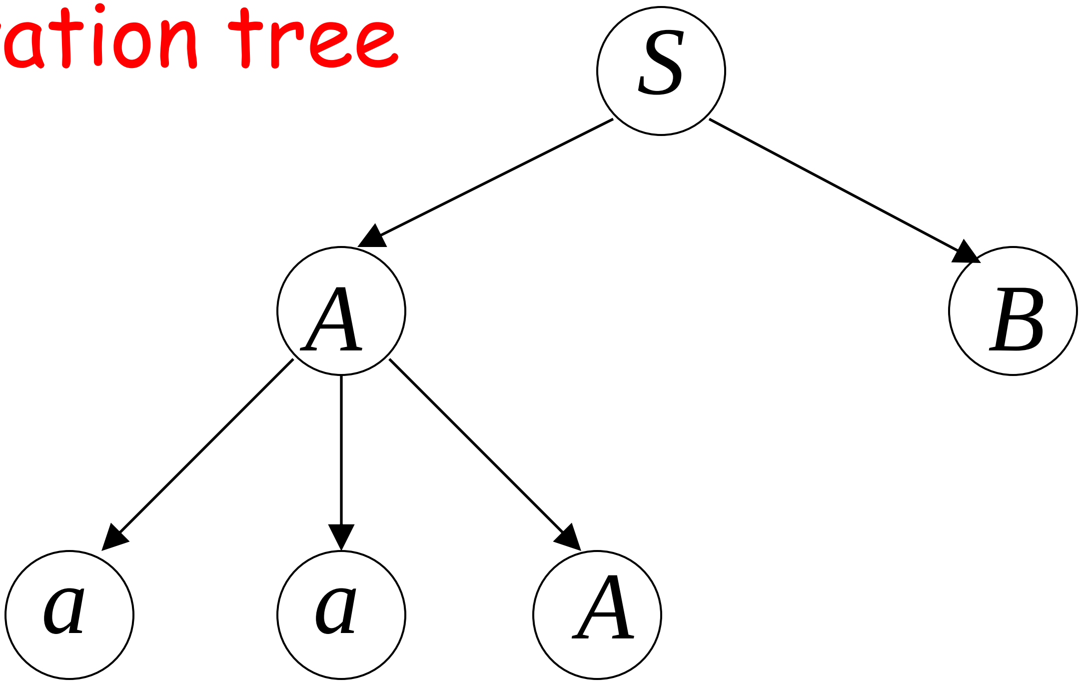
$$S \Rightarrow AB$$

Partial derivation tree



$$S \Rightarrow AB \Rightarrow aaAB$$

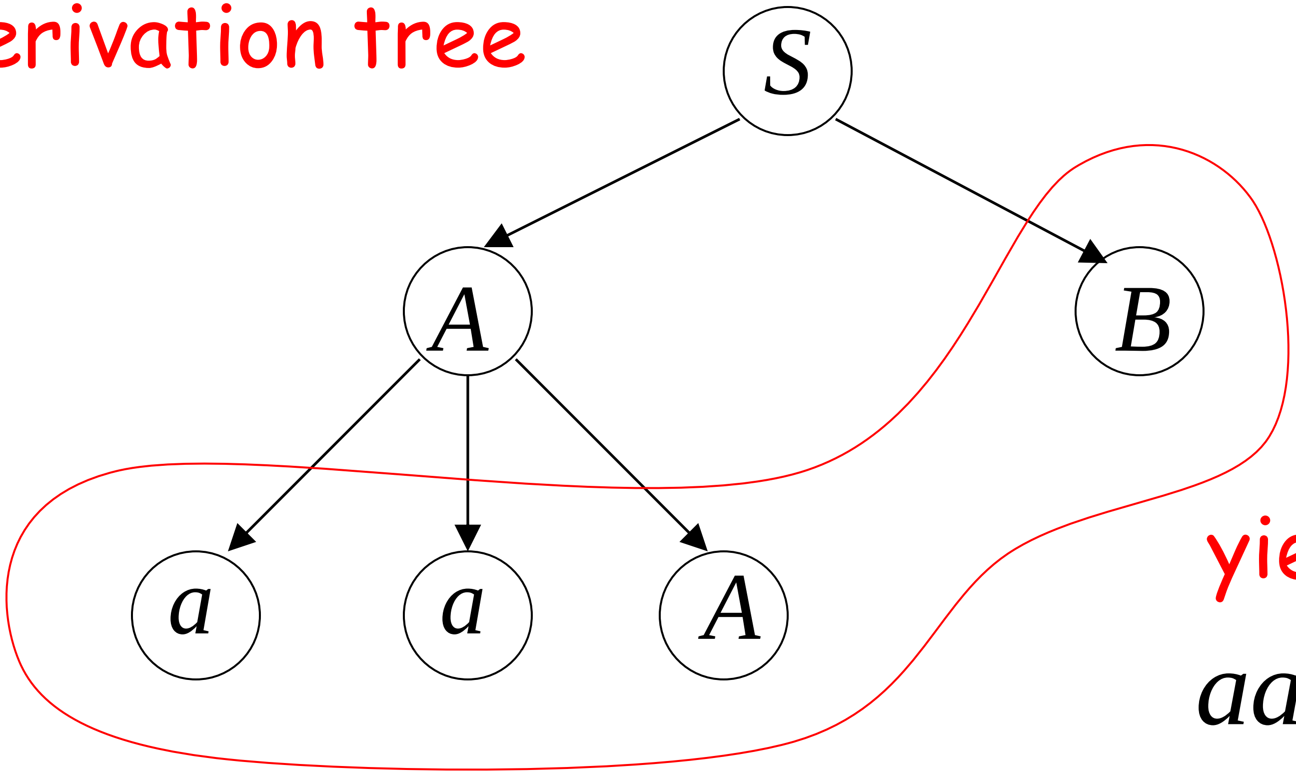
Partial derivation tree



$$S \Rightarrow AB \Rightarrow aaAB$$

sentential form

Partial derivation tree



yield

$aaAB$

Sometimes, derivation order doesn't matter

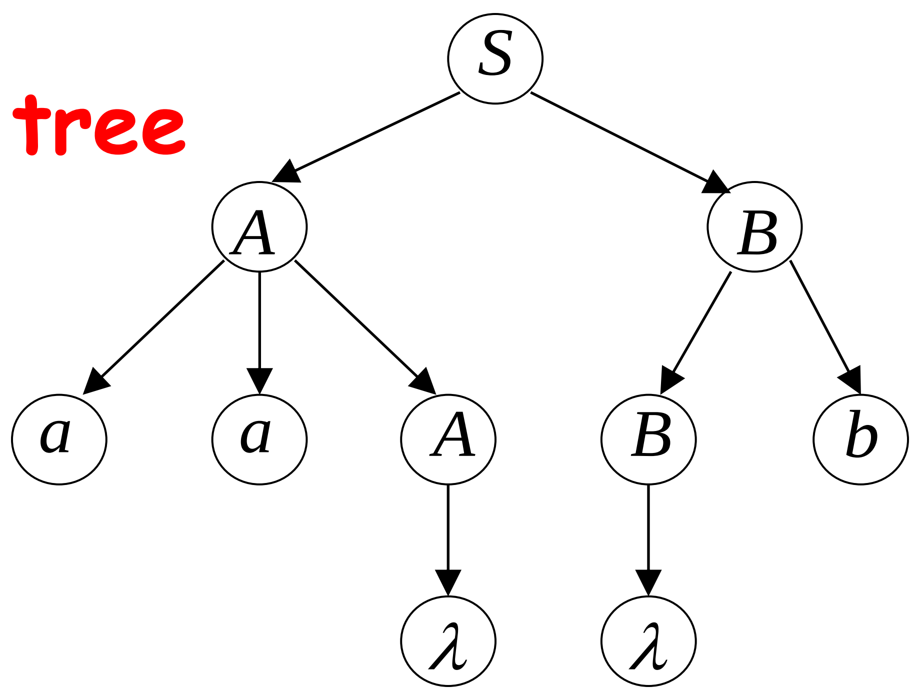
Leftmost:

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

Rightmost:

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

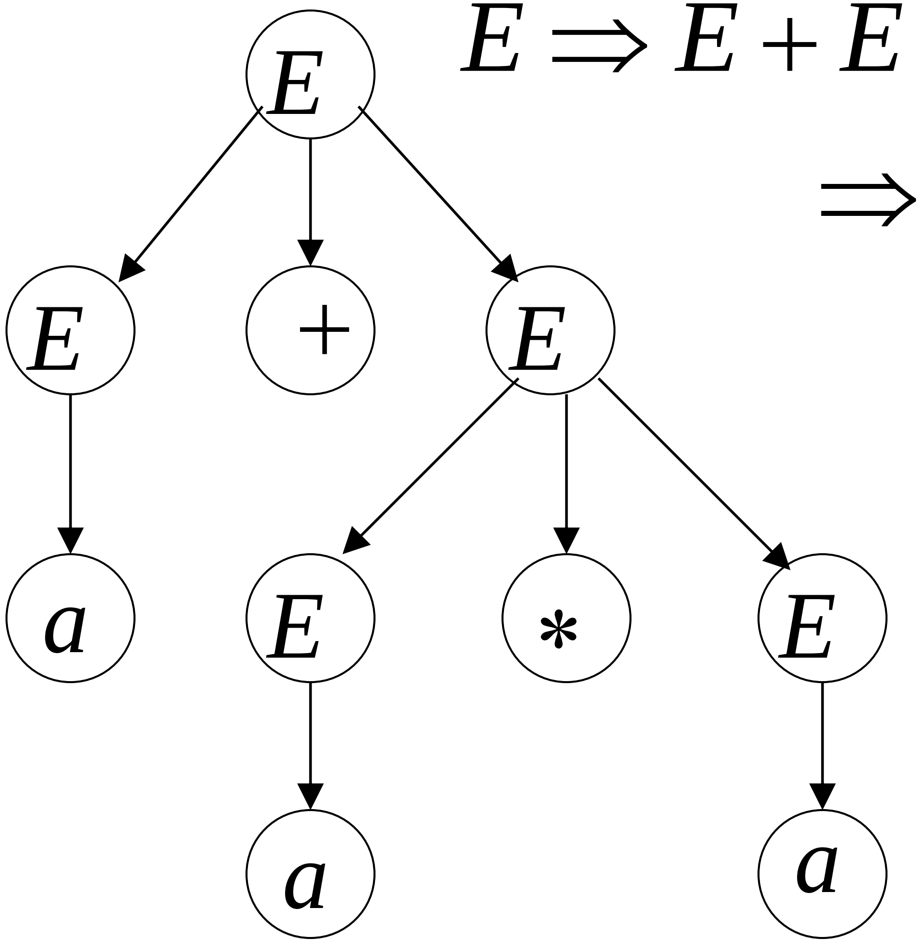
Same derivation tree



Ambiguity

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

$$a+a*a$$



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

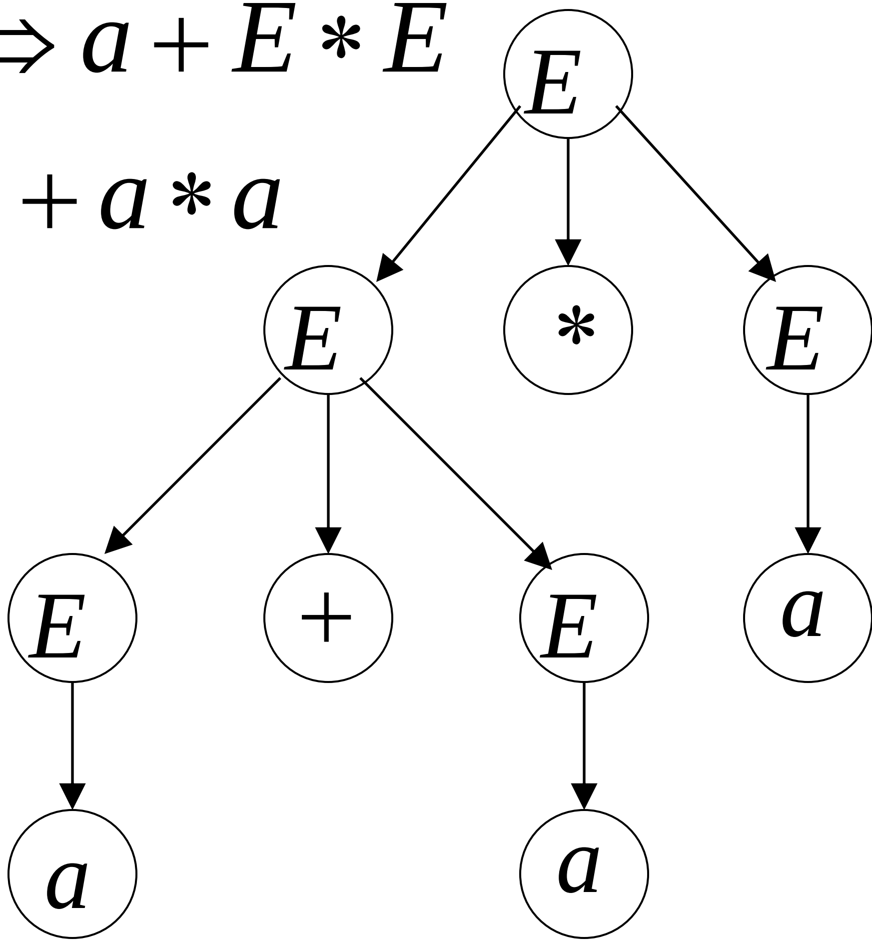
leftmost derivation

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

$$a + a * a$$

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

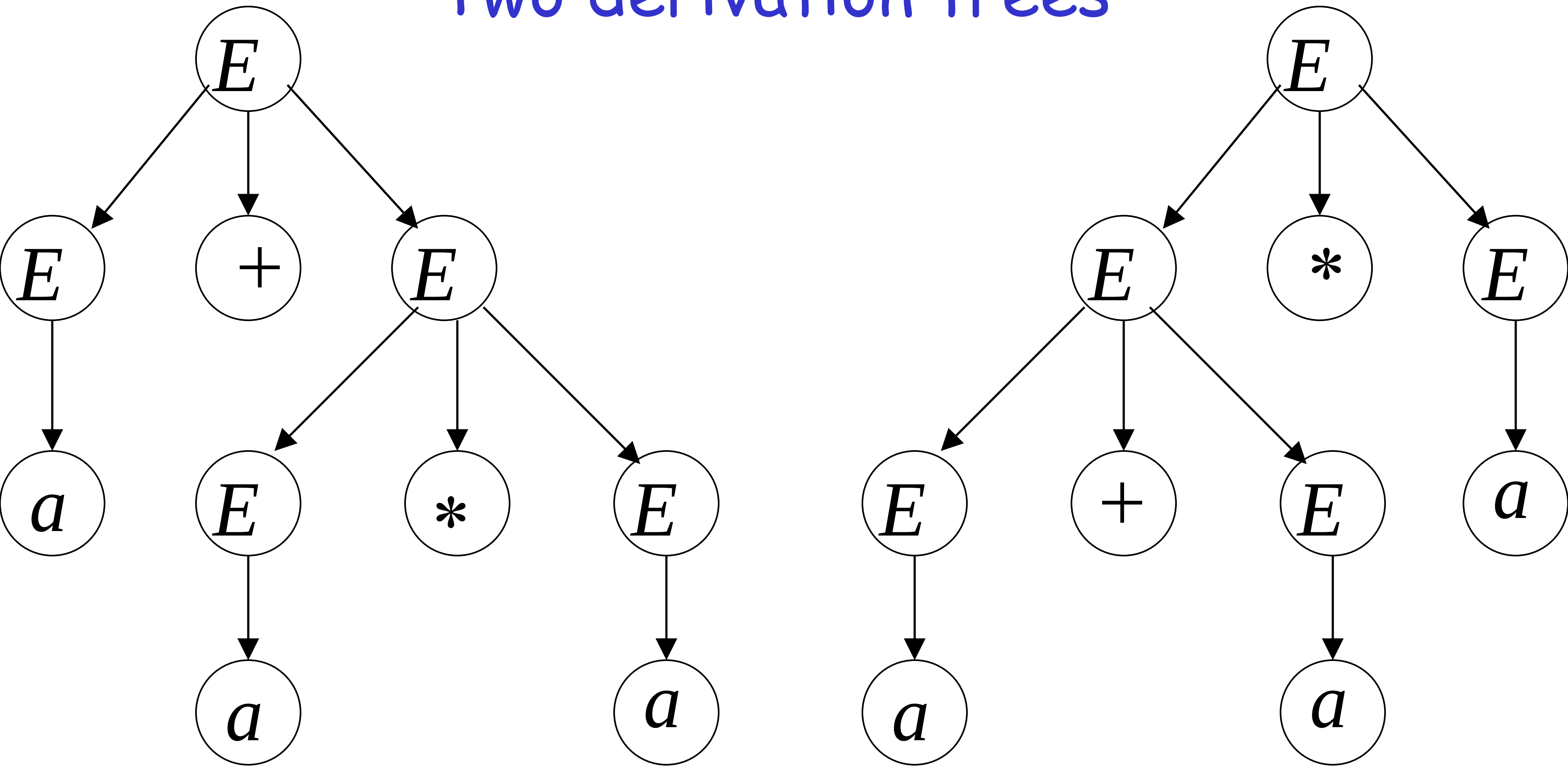
leftmost derivation



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

$$a + a * a$$

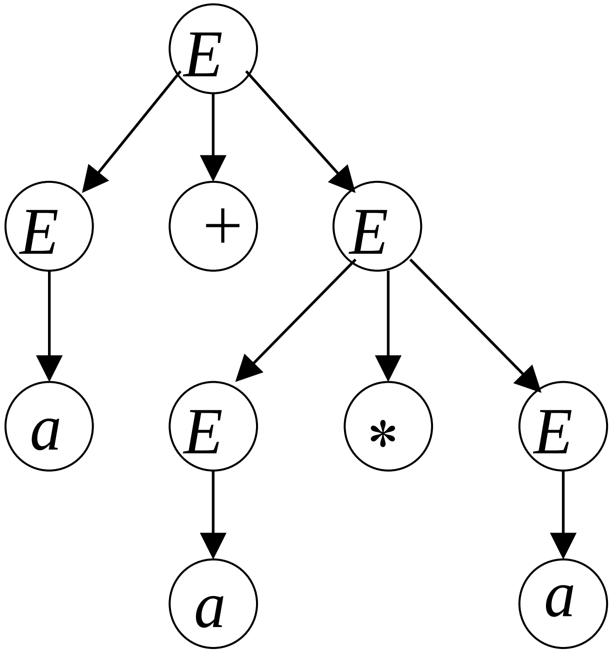
Two derivation trees

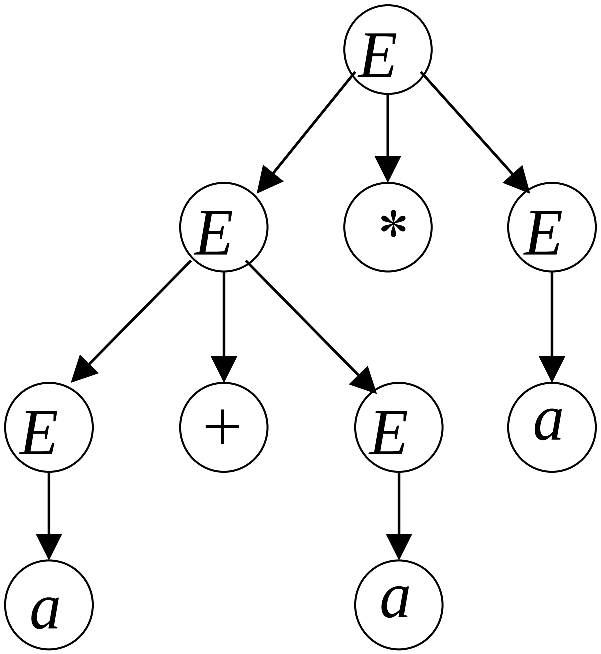


The grammar $E \to E + E \mid E * E \mid (E) \mid a$

is ambiguous:

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





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

is ambiguous:

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

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

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

Definition:

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

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

two or more derivation trees

In other words:

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

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

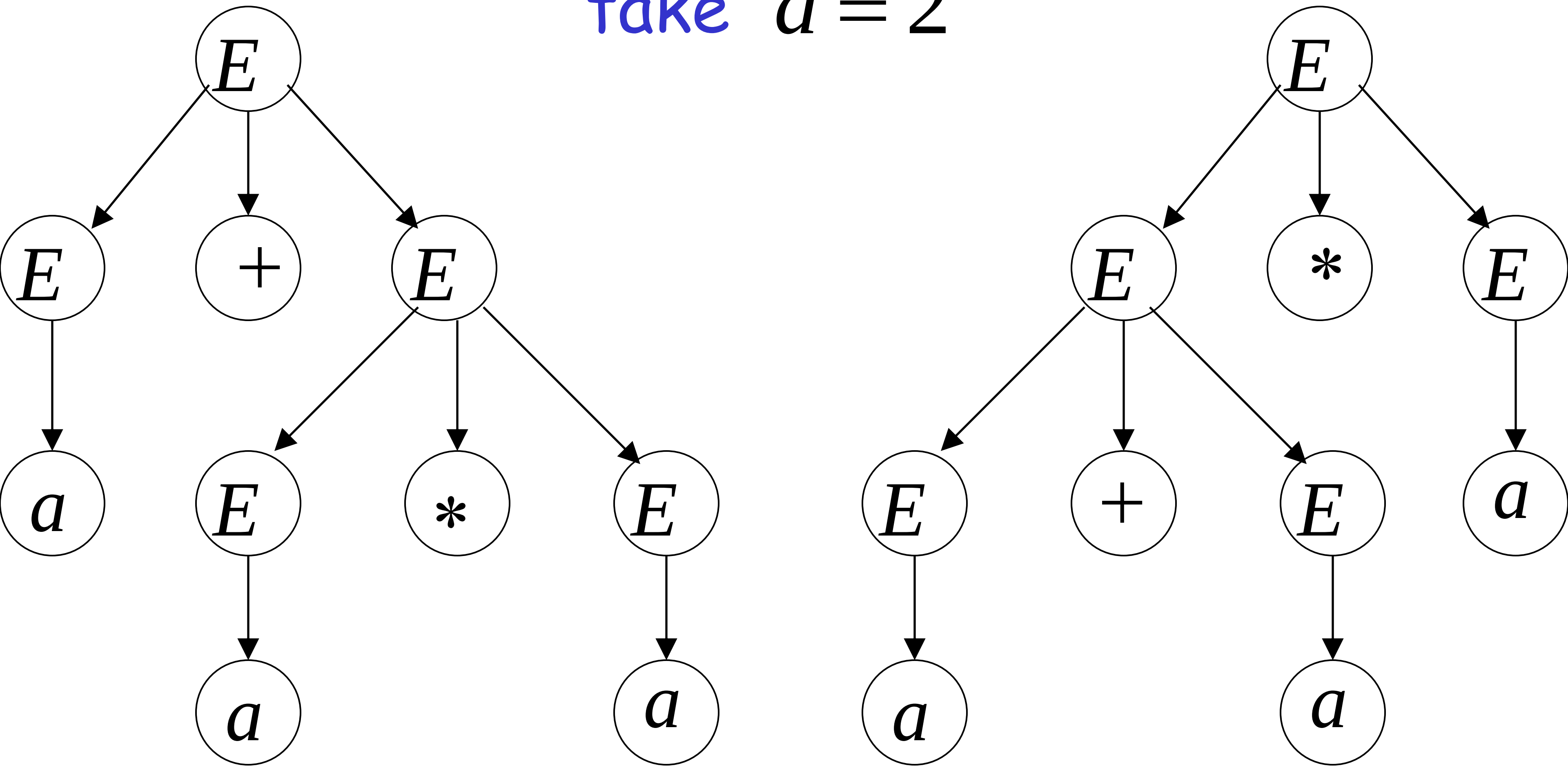
two or more leftmost derivations

(or rightmost)

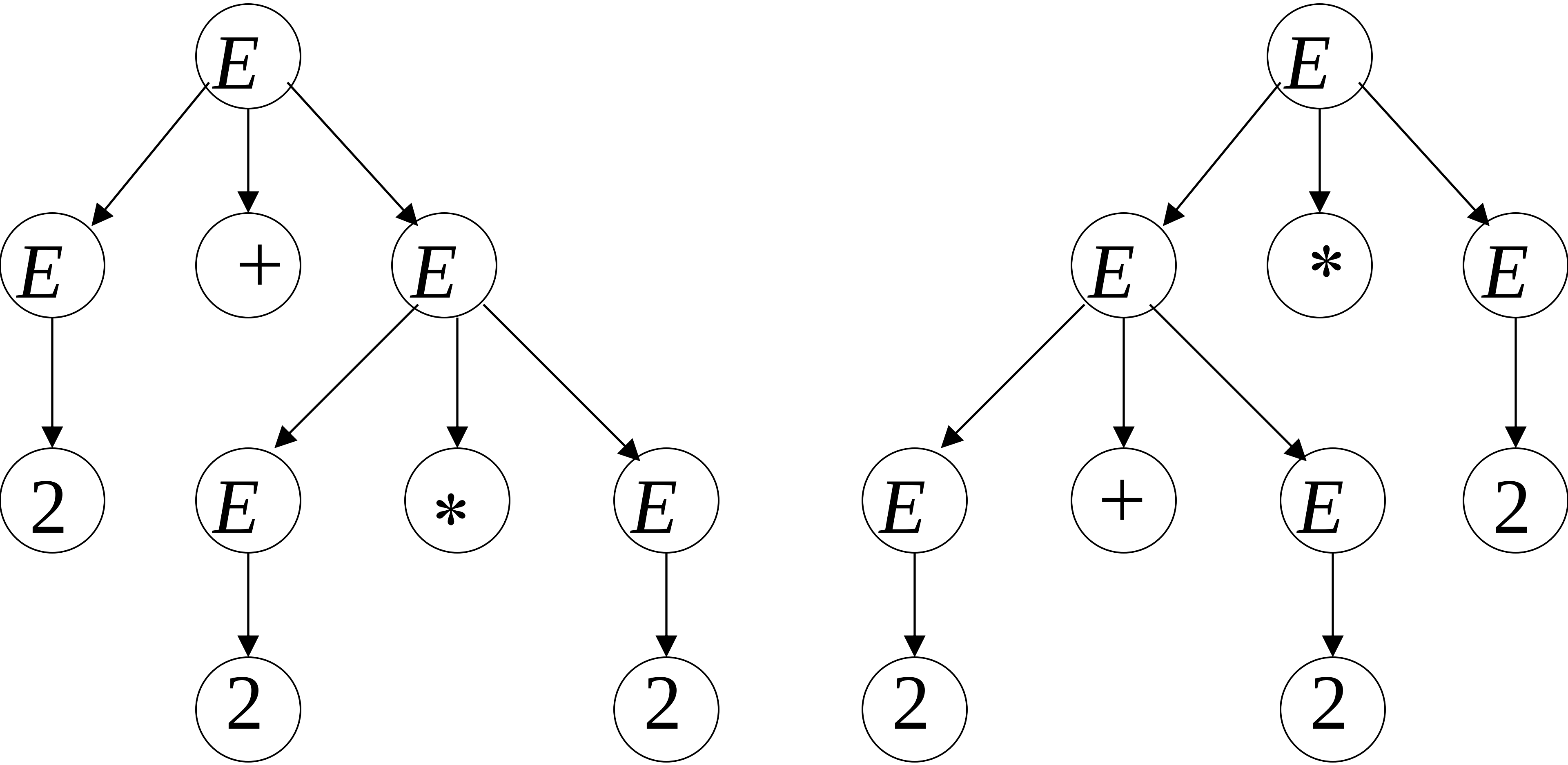
Why do we care about ambiguity?

$$a + a * a$$

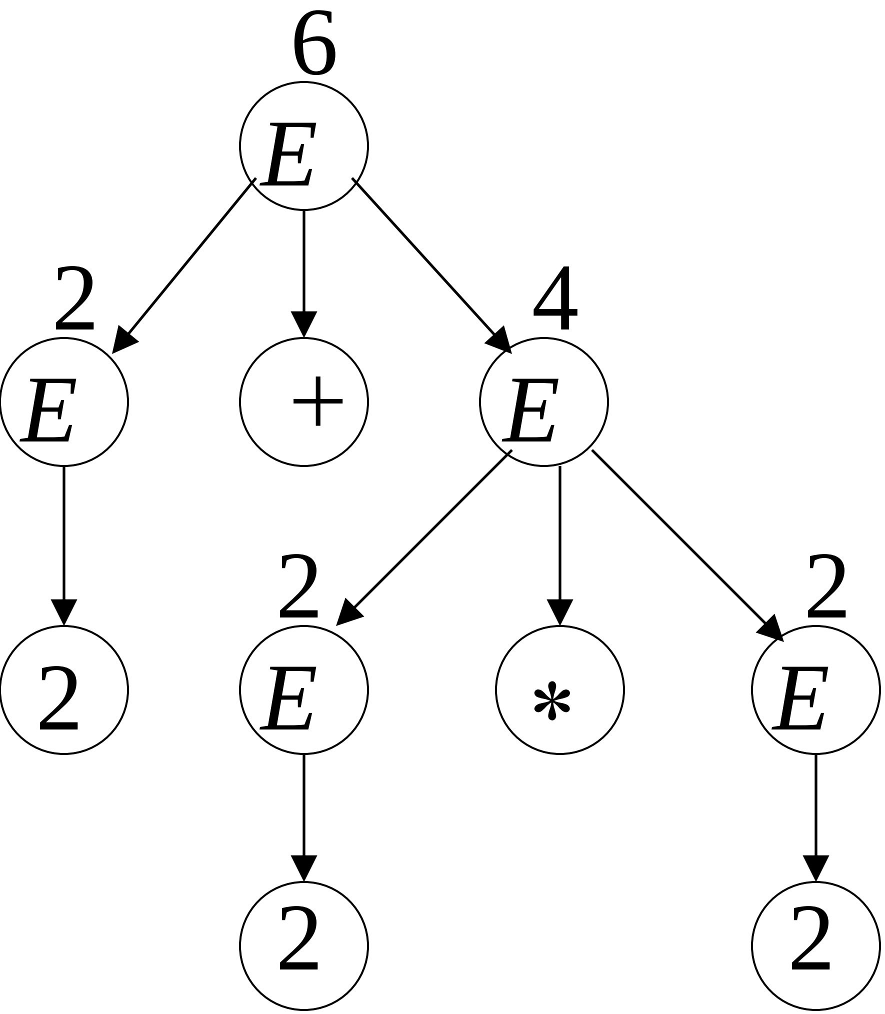
take $a = 2$

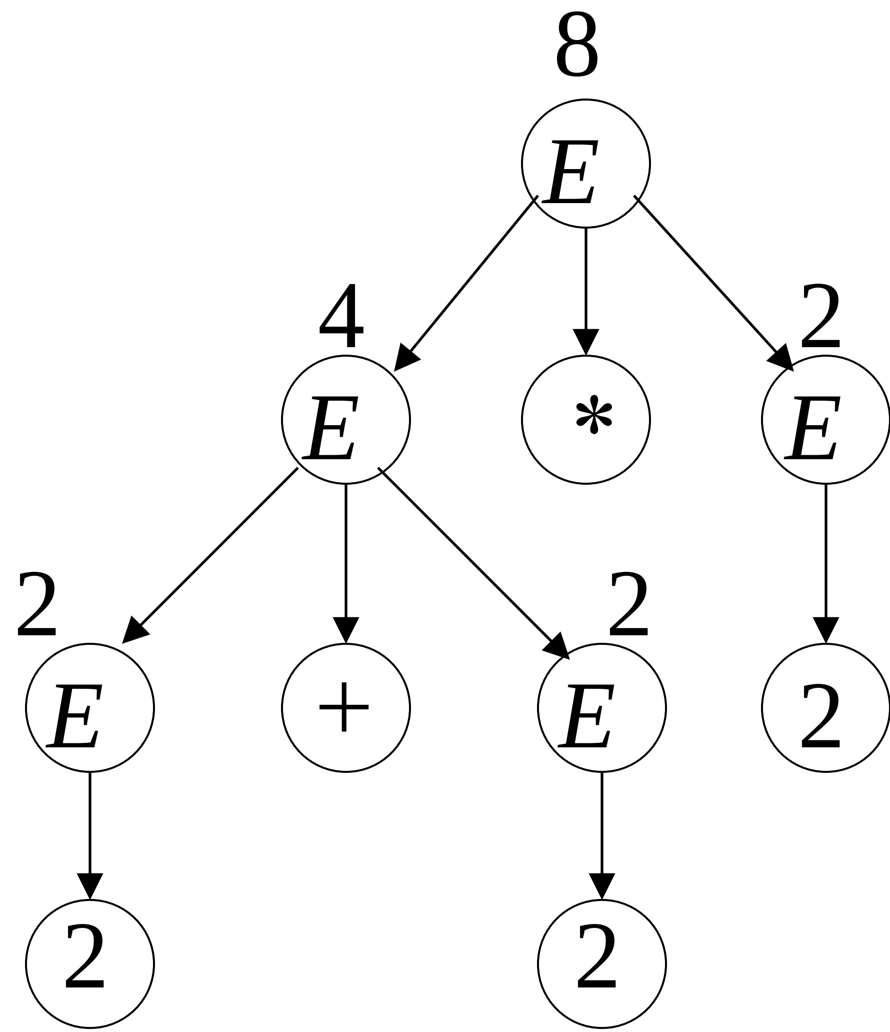


$$2+2*2$$

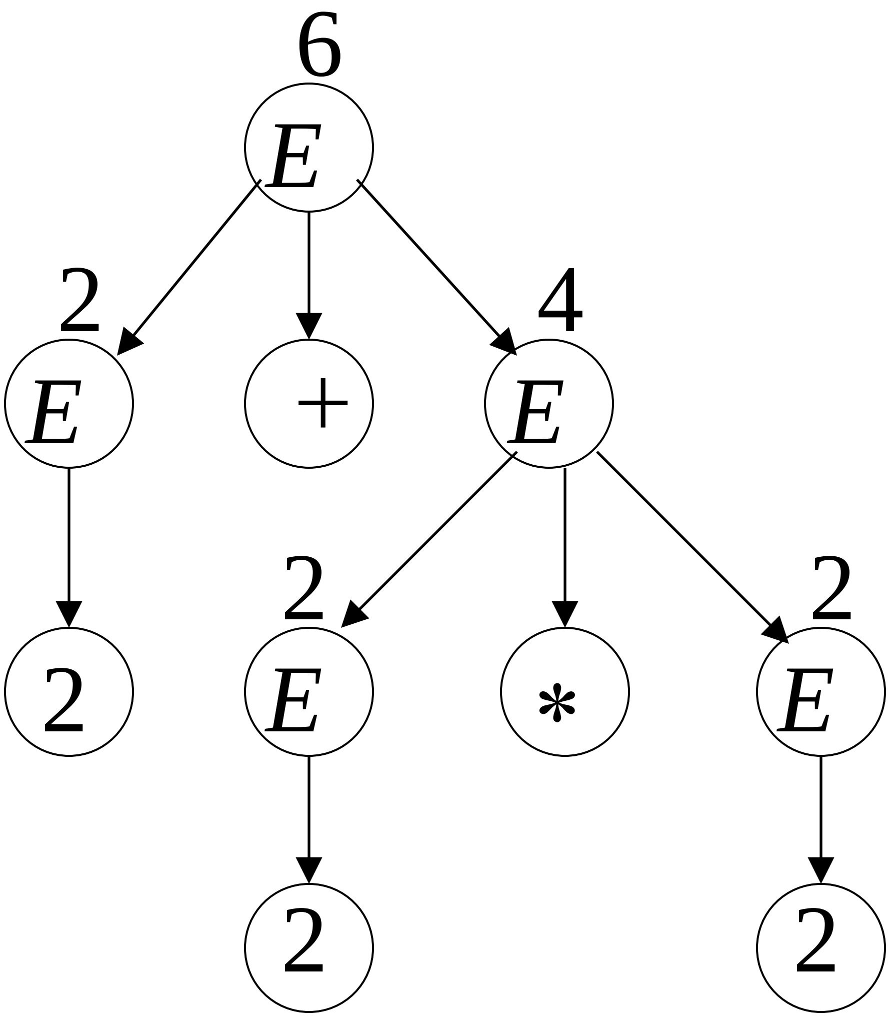


$2+2*2=6$ $\qquad\qquad$ $2+2*2=8$





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



- Ambiguity is **bad** for programming languages

- We want to remove ambiguity

We fix the ambiguous grammar:

$$E \rightarrow E + E \quad | \; E * E \; | \; (E) \; | \; a$$

New non-ambiguous grammar:

$$E \rightarrow E + T$$

$$E \rightarrow T$$

$$T \rightarrow T * F$$

$$T \rightarrow F$$

$$F \rightarrow (E)$$

$$F \rightarrow a$$

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

$$\Rightarrow a+F*F \Rightarrow a+a*F \Rightarrow a+a*a$$

$$E \rightarrow E+T$$

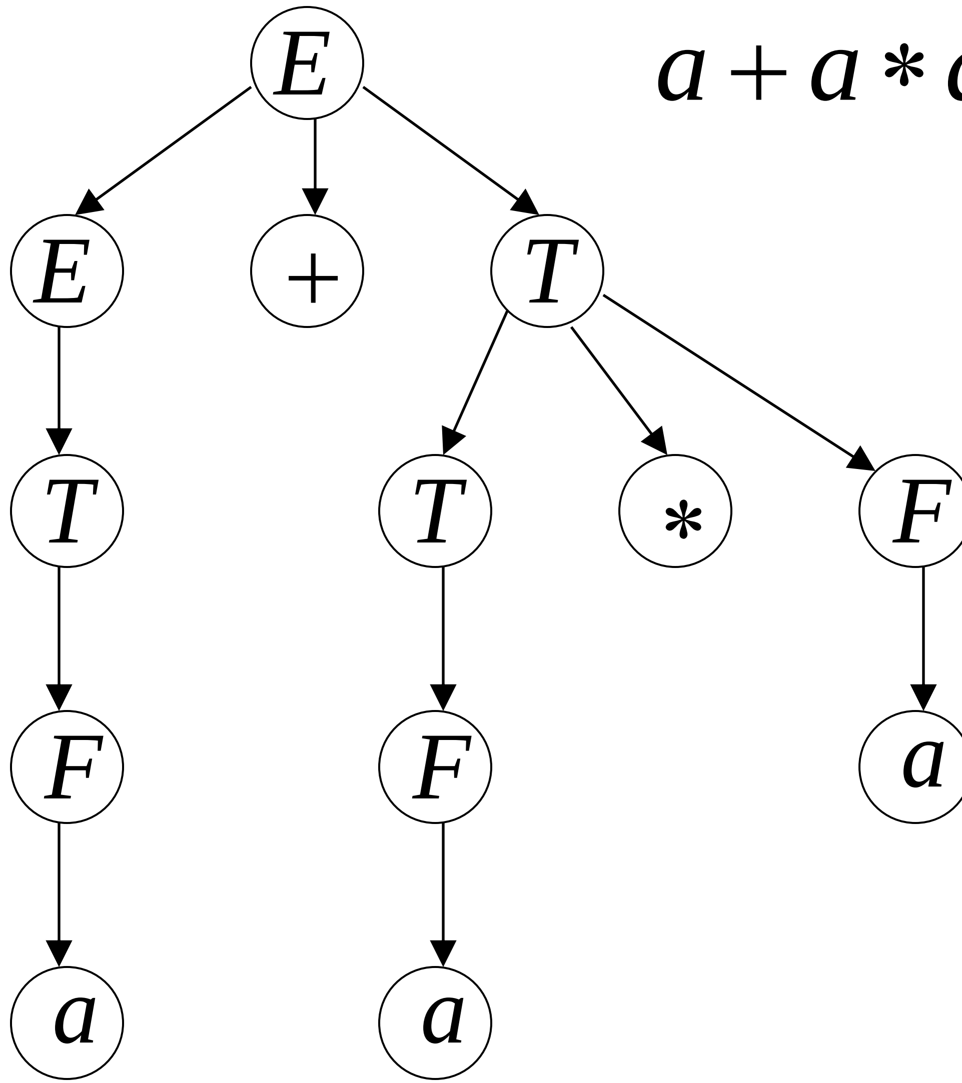
$$E \rightarrow T$$

$$T \rightarrow T*F$$

$$T \rightarrow F$$

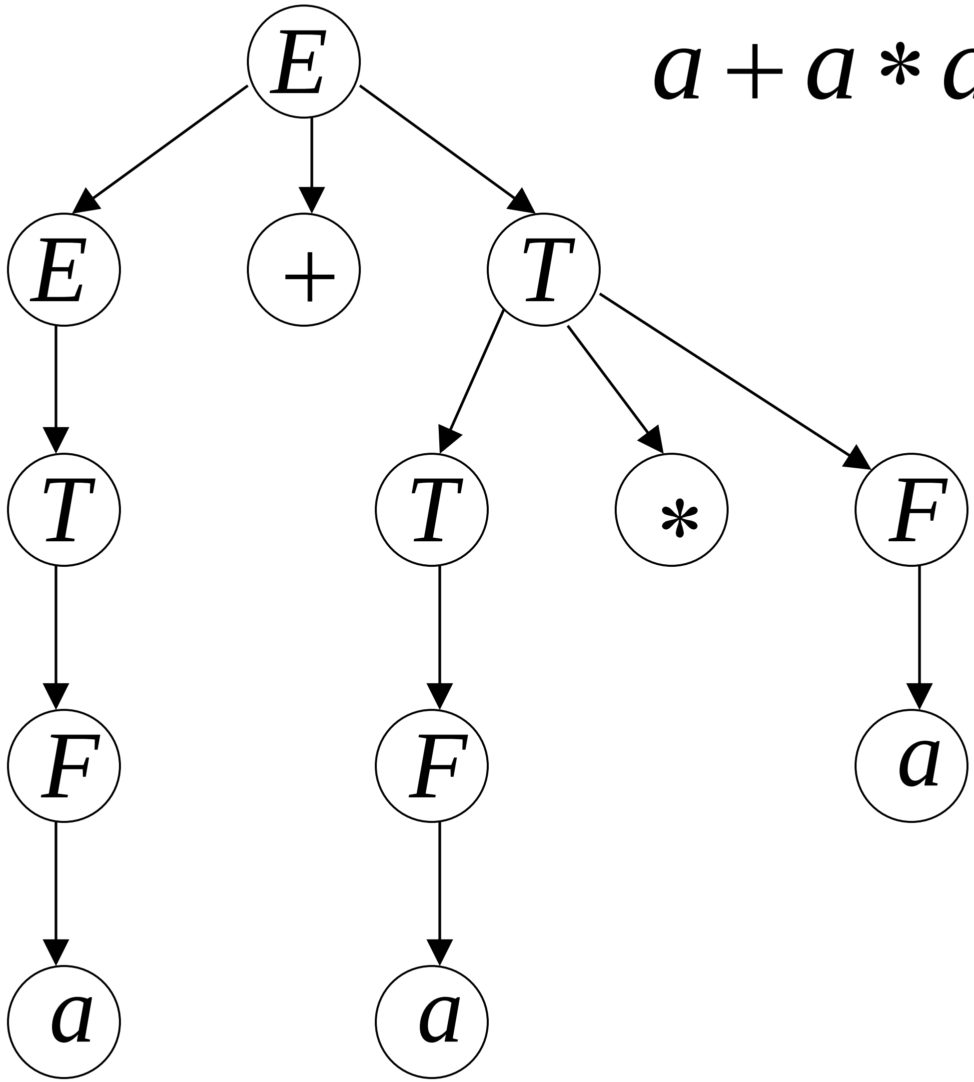
$$F \rightarrow (E)$$

$$F \rightarrow a$$



$$a+a*a$$

Unique derivation tree



$a + a * a$

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:

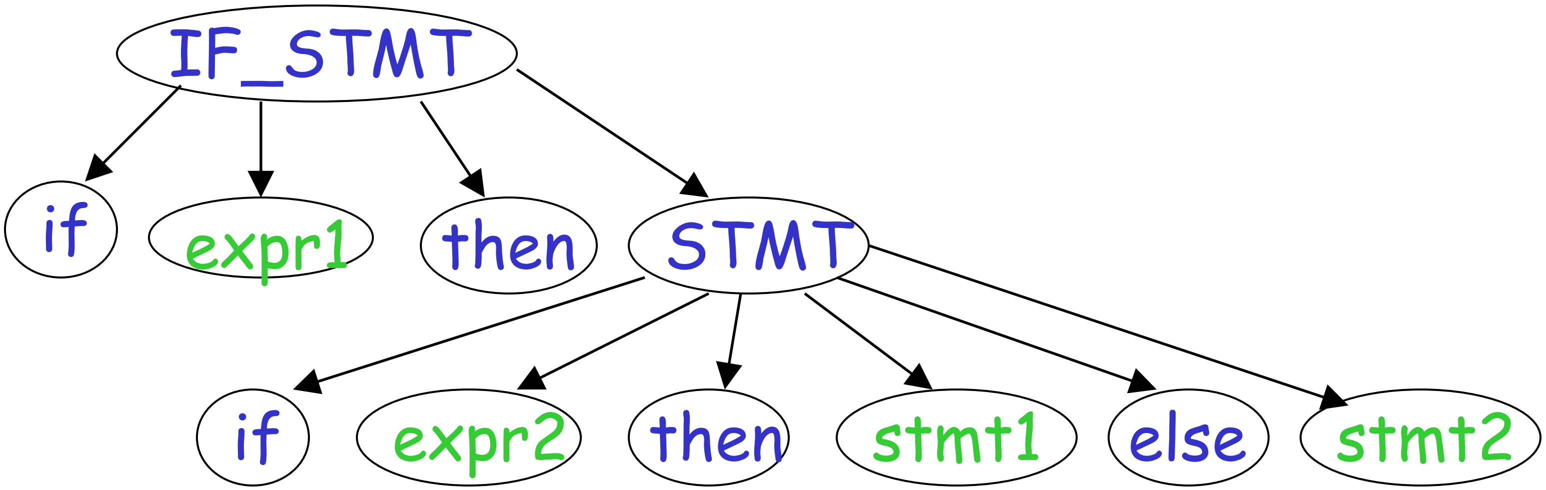
Every string $w \in L(G)$ has a unique derivation tree

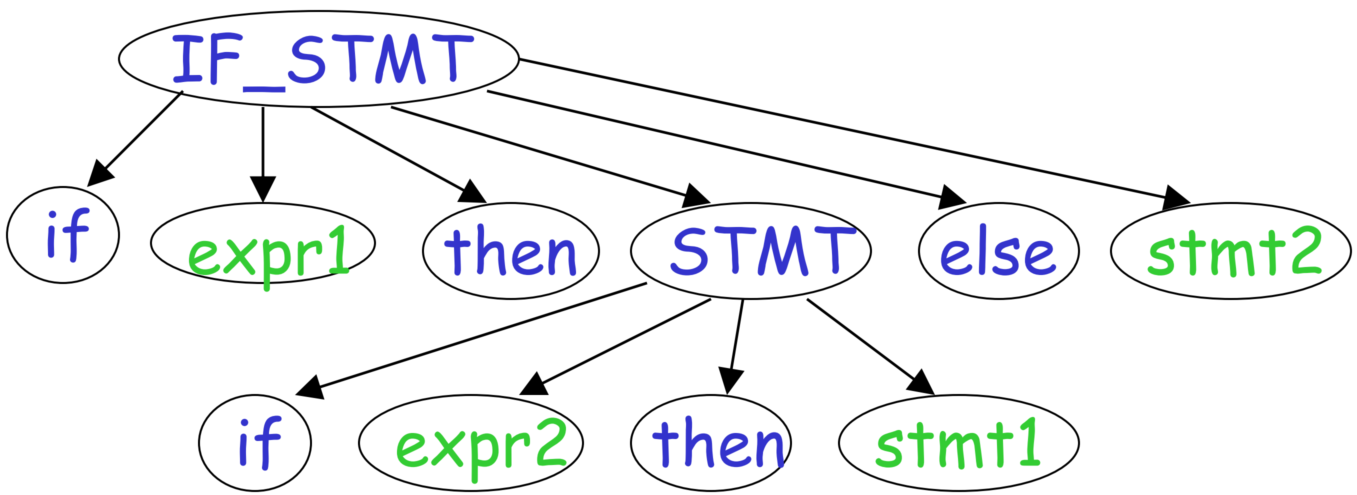
Another Ambiguous Grammar

IF_STMT $\rightarrow$ if EXPR then STMT

$\mid$ if EXPR then STMT else STMT

If expr1 then if expr2 then stmt1 else stmt2





Inherent Ambiguity

Some context free languages
have only ambiguous grammars

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

$S \to S_1 \mid S_2$

$S_1 \to S_1 c \mid A$

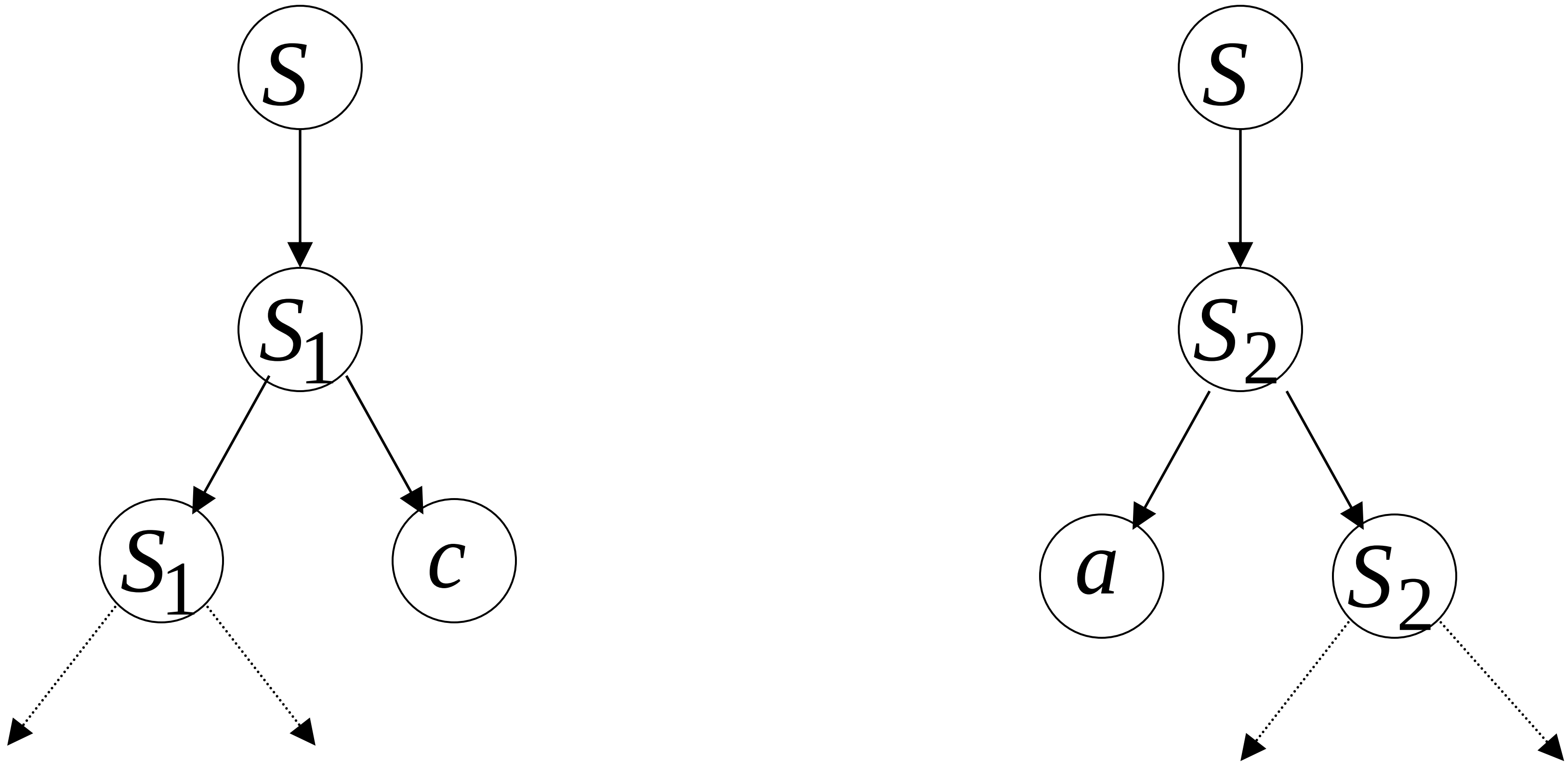
$A \to aAb \mid \lambda$

$S_2 \to aS_2 \mid B$

$B \to bBc \mid \lambda$

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

has two derivation trees



Compilers

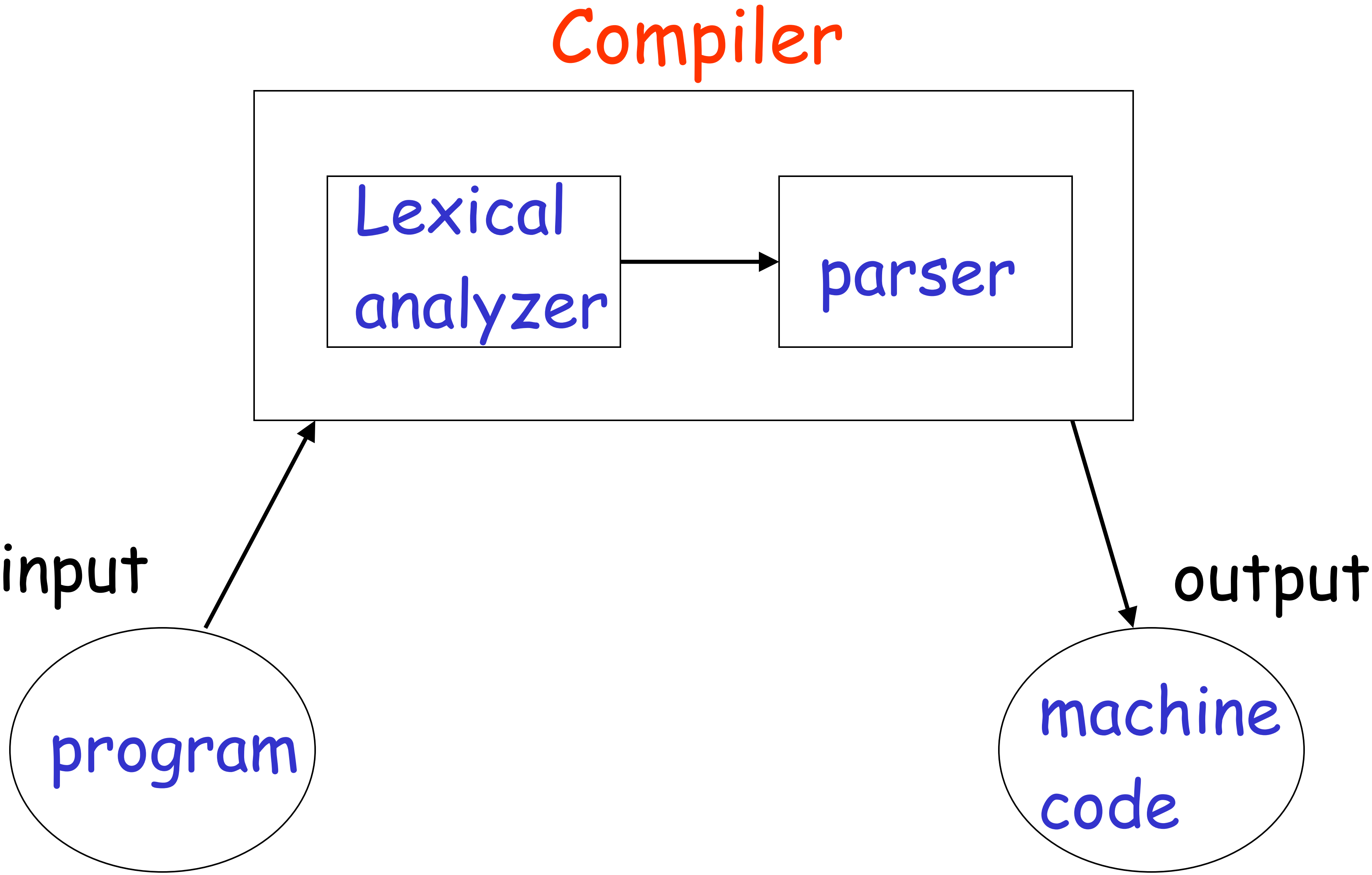
Program

```
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
...
```


Compiler
Lexical analyzer
parser
input
program
output
machine code

A parser knows the grammar of the programming language

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

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

derivation

E => E + E

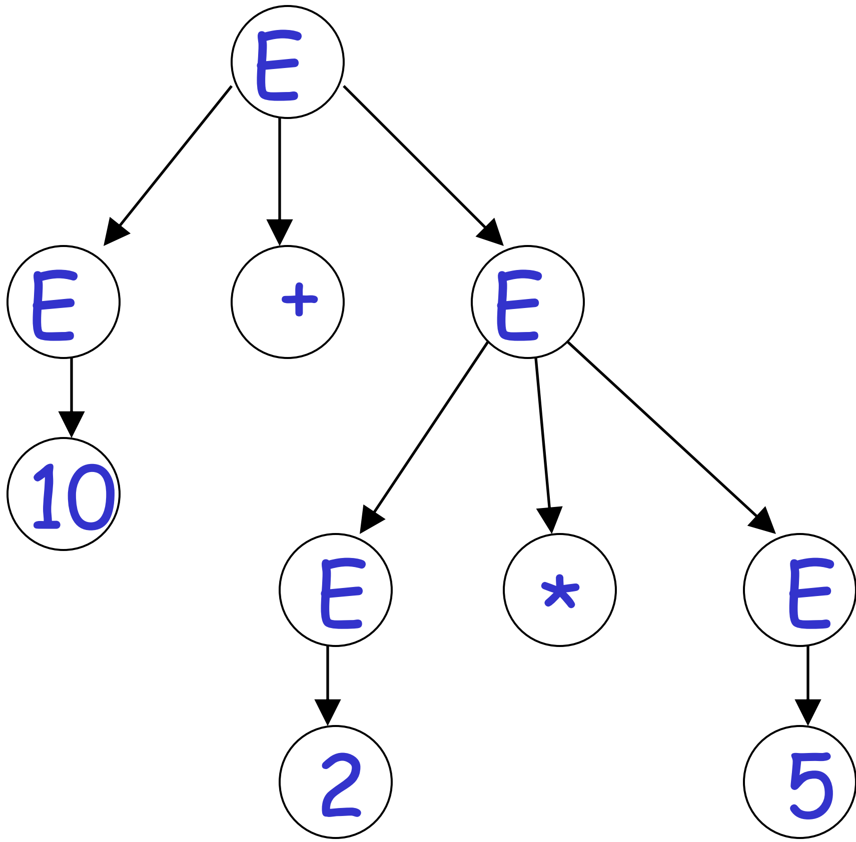
=> E + E * E

=> 10 + E*E

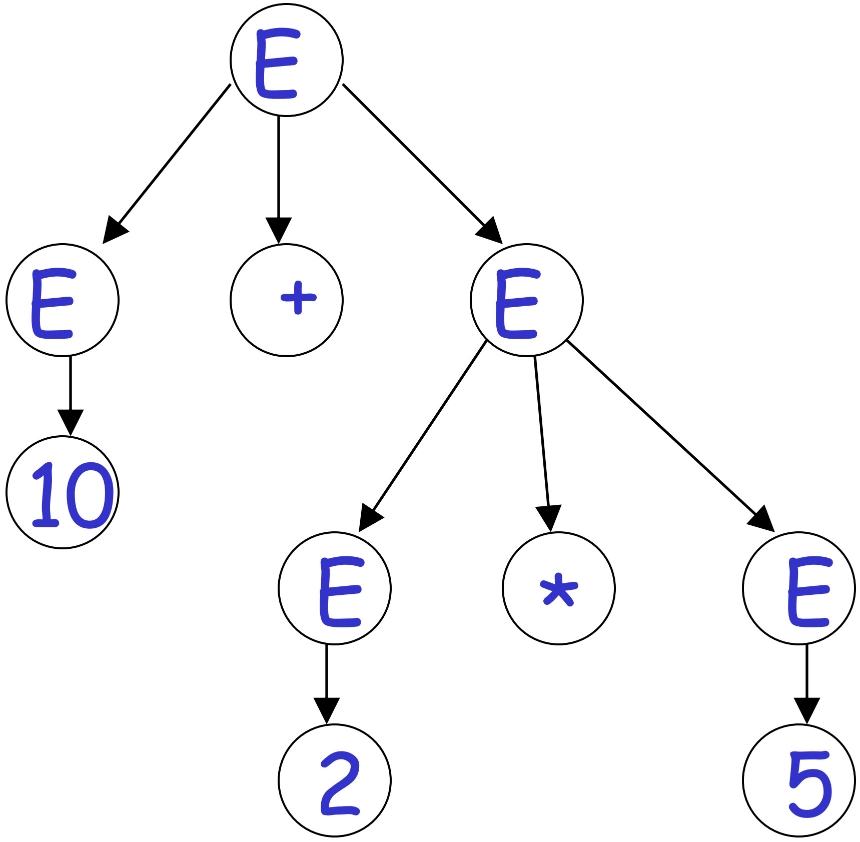
=> 10 + 2 * E

=> 10 + 2 * 5

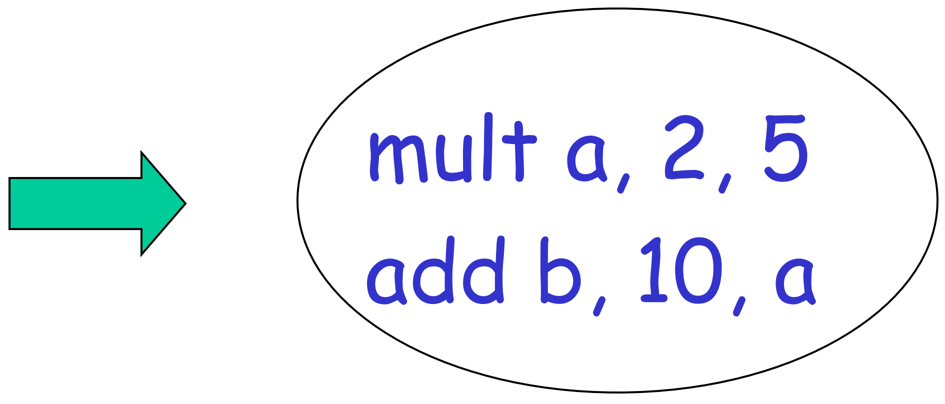
derivation tree



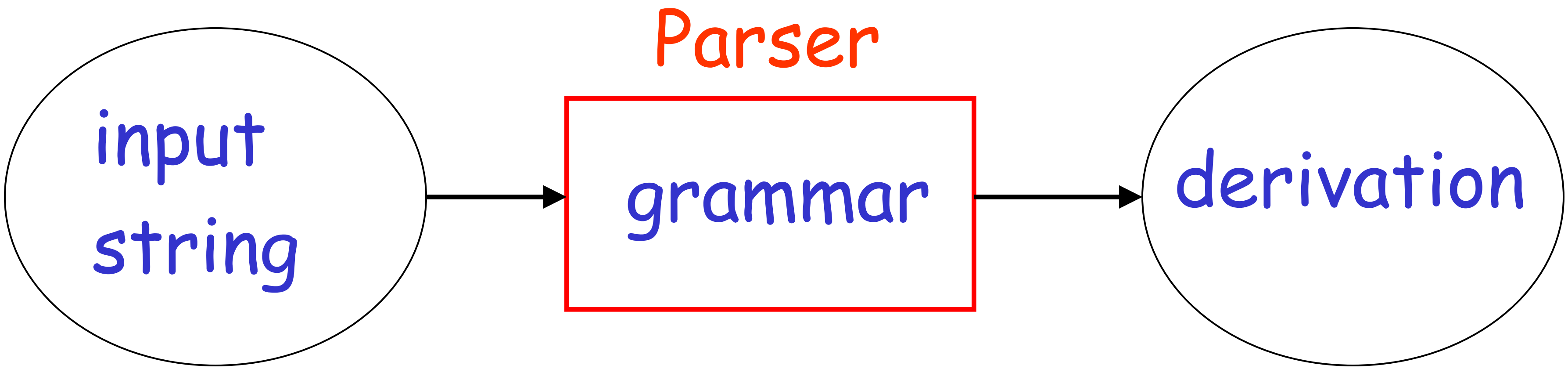
derivation tree



machine code



Parsing


Parser
input
string
grammar
derivation

Example:

input

$aabb$

Parser

$$S \to SS$$

$$S \to aSb$$

$$S \to bSa$$

$$S \to \lambda$$

derivation

?

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

$$S \Rightarrow SS$$

$$aabb$$

$$S \Rightarrow aSb$$

$$\cancel{S \Rightarrow bSa}$$

$$\cancel{S \Rightarrow \lambda}$$

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

$S \Rightarrow SS \Rightarrow SSS$

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

$\cancel{S \Rightarrow SS \Rightarrow bSaS}$

Phase 1

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

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

$S \Rightarrow aSb \Rightarrow aaSbb$

$\cancel{S \Rightarrow aSb \Rightarrow abSab}$

$\cancel{S \Rightarrow aSb \Rightarrow ab}$

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

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

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

Final result of exhaustive search (top-down parsing)

input

$aabb$

Parser

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

derivation

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

Time complexity of exhaustive search

Suppose there are no productions of the form

$$A \rightarrow \lambda$$

$$A \rightarrow B$$

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

For grammar with k rules

Time for phase 1: k

k possible derivations

Time for phase 2: k^2

k^2 possible derivations

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

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

Total time needed for string w:

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

phase 1

phase 2

phase 2|w|

Extremely bad!!!

There exist faster algorithms
for specialized grammars

S-grammar: $A \to ax$

symbol

string
of variables

Pair (A, a) appears once

S-grammar example:

$$S \rightarrow aS$$

$$S \rightarrow bSS$$

$$S \rightarrow c$$

Each string has a unique derivation

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

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: $|w|$

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)