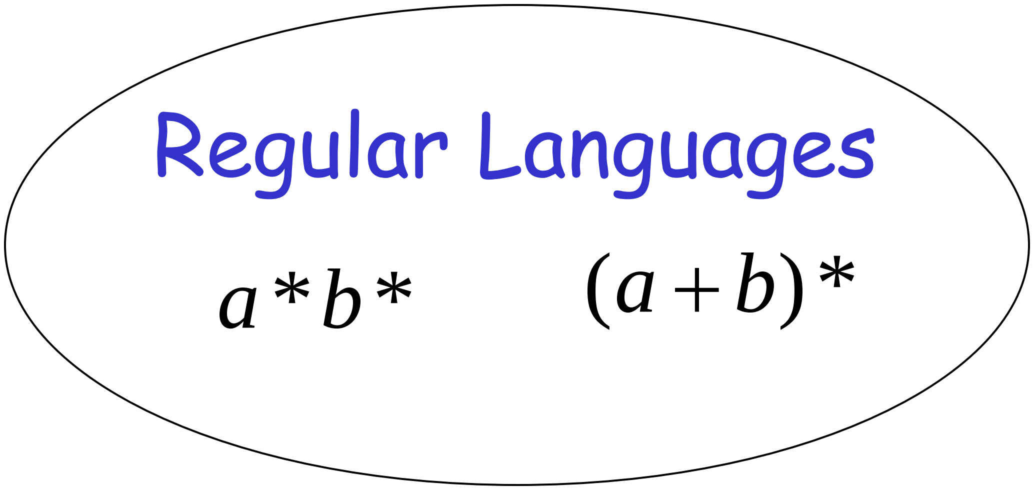
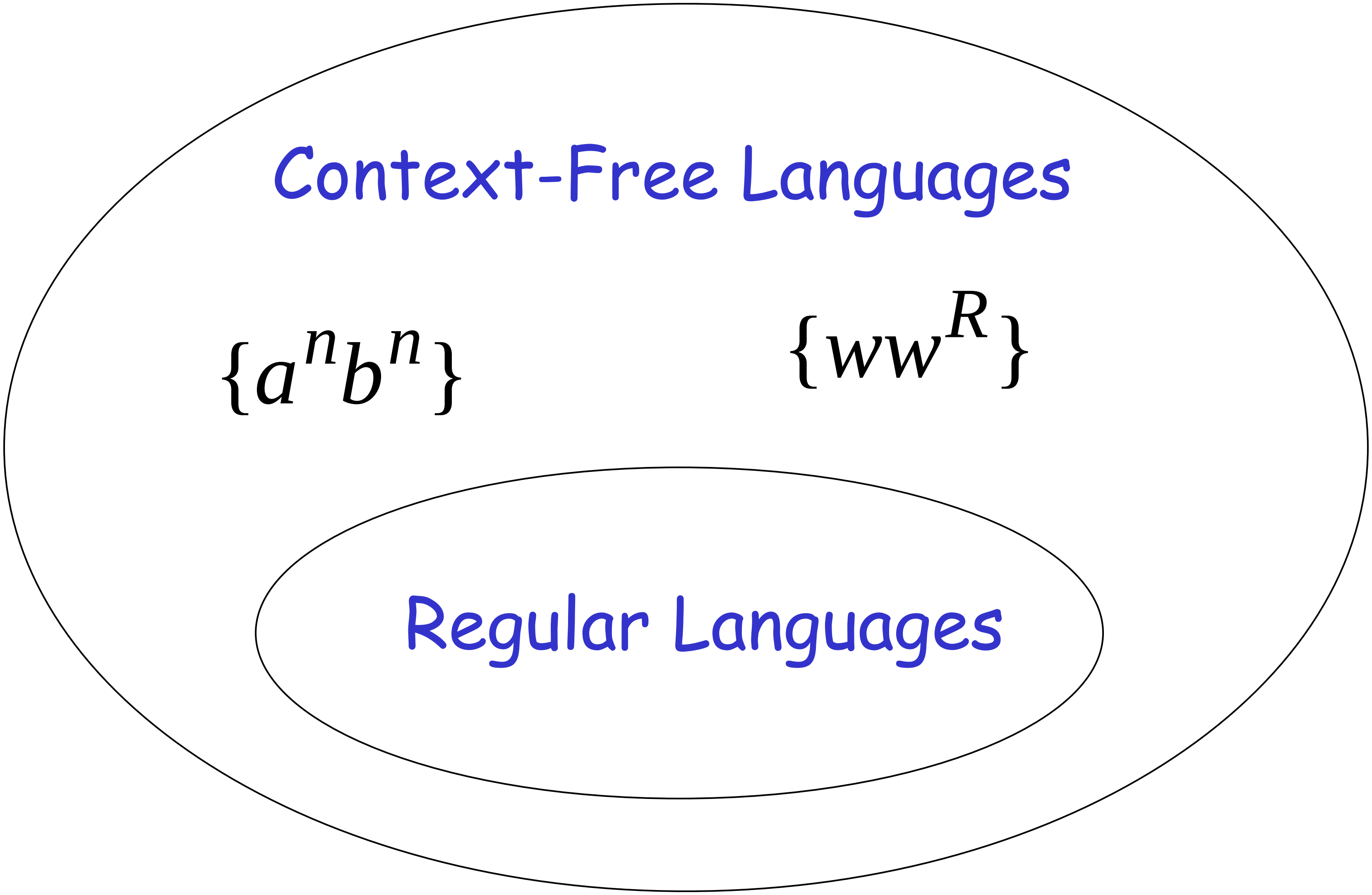
Context-Free Languages

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



Regular Languages

a^*b^* $(a+b)^*$

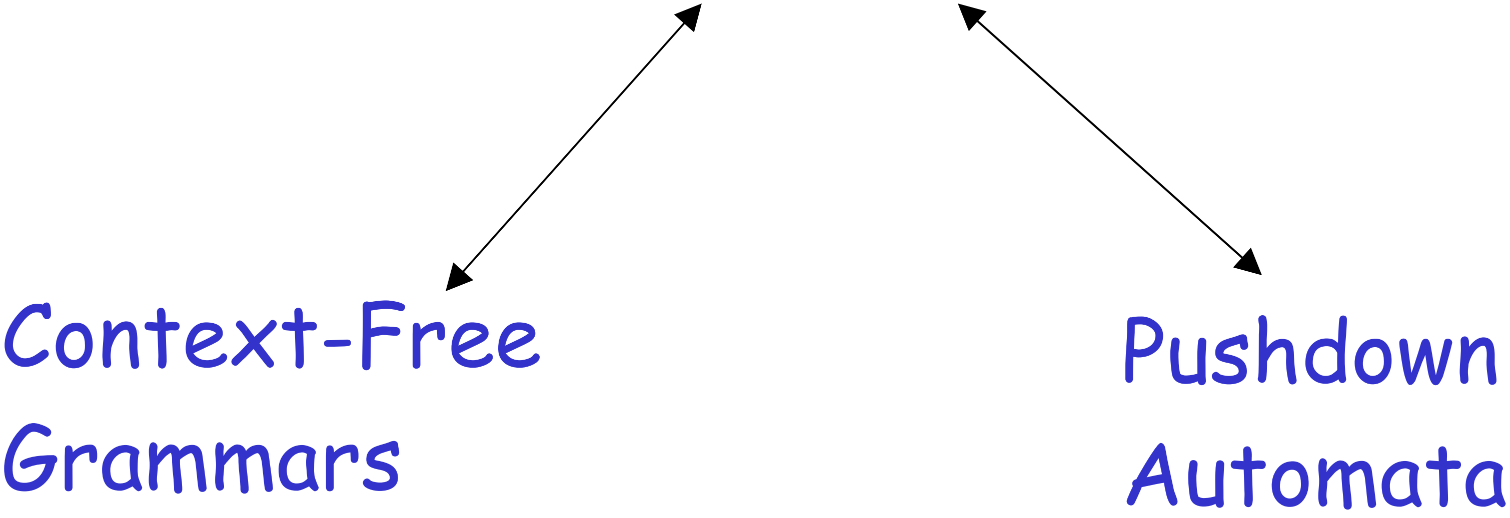


Context-Free Languages

$\{a^n b^n\}$ $\{ww^R\}$

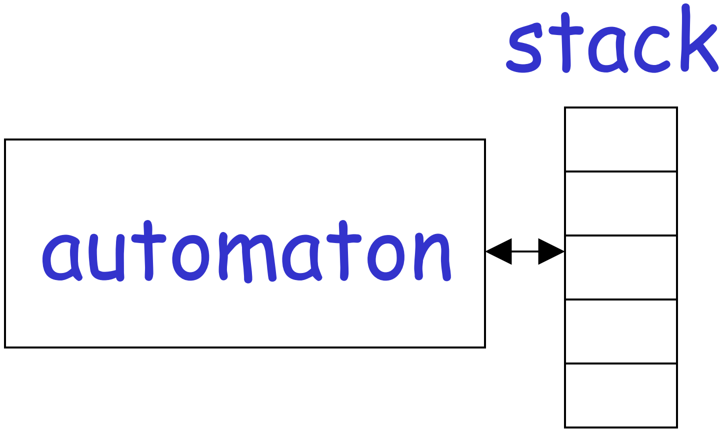
Regular Languages

Context-Free Languages



Context-Free Grammars

Pushdown Automata



stack

automaton

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:

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

$$S \rightarrow aSb$$

$$S \rightarrow \lambda$$

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

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:

$$S \rightarrow aSa$$

$$S \rightarrow bSb$$

$$S \rightarrow \lambda$$

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:

$$S \to aSb$$

$$S \to SS$$

$$S \to \lambda$$

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:

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

$$S \to aSb$$

$$S \to SS$$

$$S \to \lambda$$

$$L(G) = \{w \quad : n_a(w) = n_b(w),$$

$$\text{and } \; n_a(v) \geq n_b(v)$$

$$\text{in any } \; \text{prefix } \; v\}$$

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

Definition: Context-Free Grammars

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

Variables

Terminal symbols

Start variable

Productions of the form:

$$A \to x$$

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

$1.\ S \to AB$ $\qquad 2.\ A \to aaA$ $\qquad 4.\ B \to Bb$

$3.\ A \to \lambda$ $\qquad 5.\ B \to \lambda$

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:

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

$$S \rightarrow aAB$$

$$A \rightarrow bBb$$

$$B \rightarrow A \mid \lambda$$

Leftmost derivation:

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

$$\Rightarrow abbbbB \Rightarrow abbbb$$

Rightmost derivation:

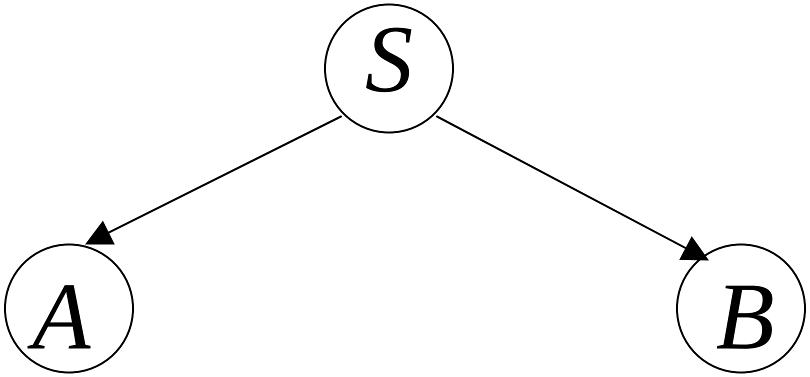
$$S \Rightarrow aAB \Rightarrow aA \Rightarrow abBb \Rightarrow abAb$$

$$\Rightarrow abbBbb \Rightarrow abbbb$$

Derivation Trees

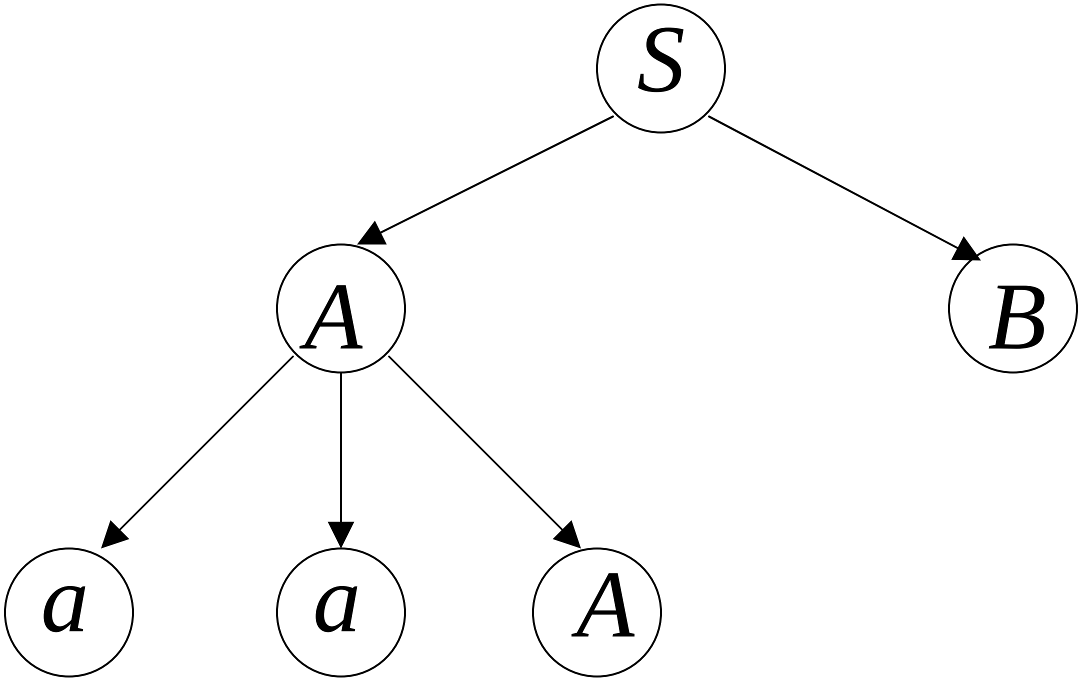
$S \rightarrow AB$ $\quad A \rightarrow aaA \mid \lambda$ $\quad B \rightarrow Bb \mid \lambda$

$$S \Rightarrow AB$$



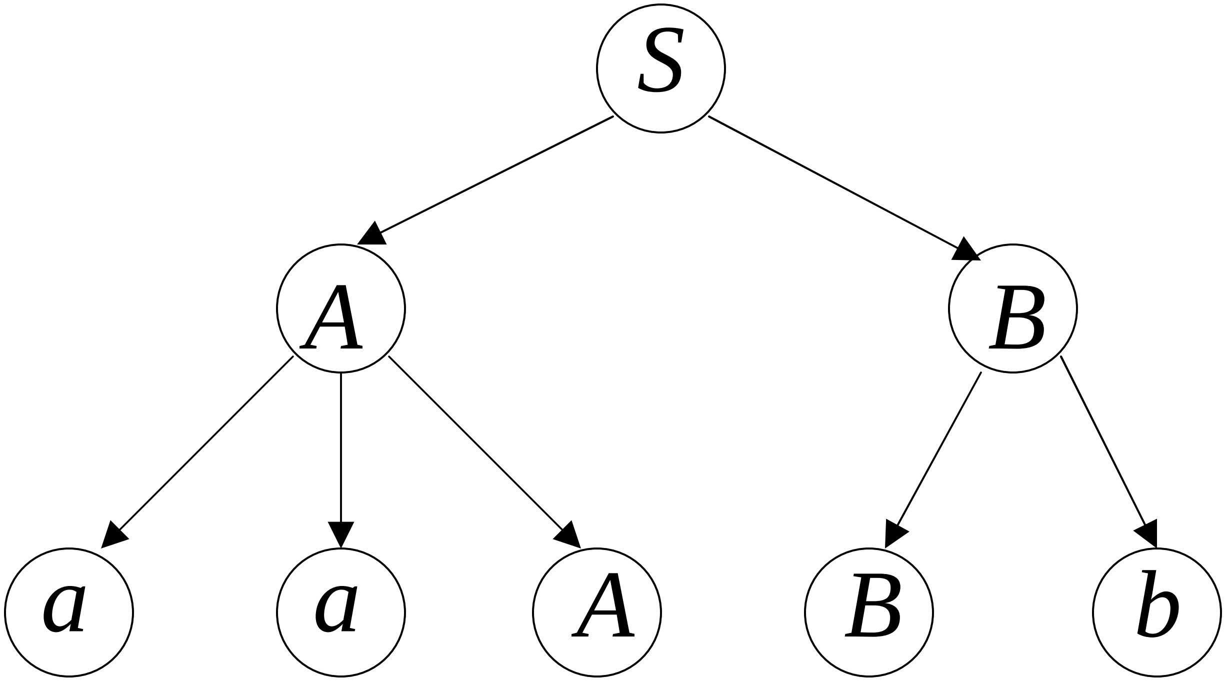
$S \to AB \qquad A \to aaA \mid \lambda \qquad B \to Bb \mid \lambda$

$$S \Rightarrow AB \Rightarrow aaAB$$



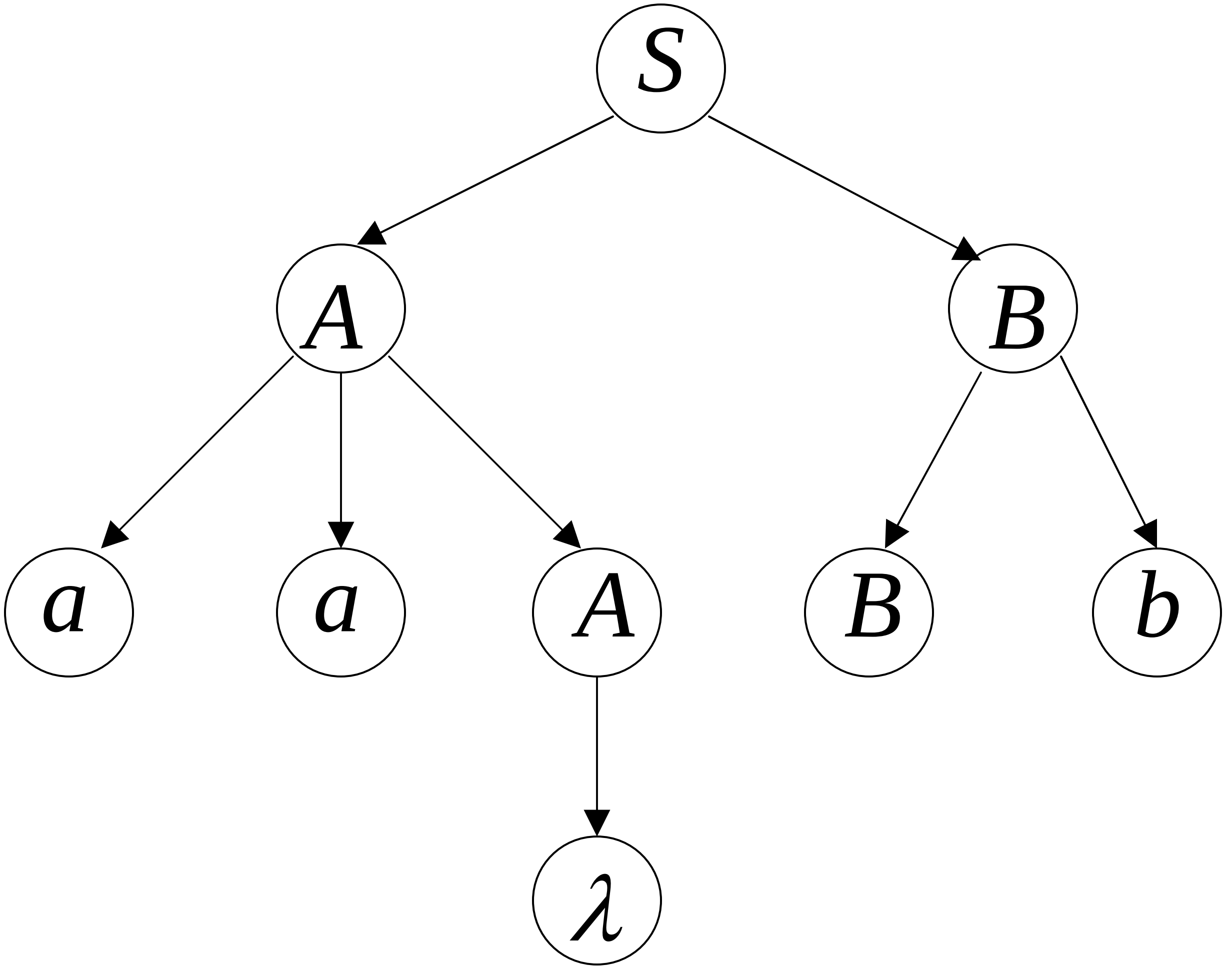
$S \to AB \qquad A \to aaA \mid \lambda \qquad B \to Bb \mid \lambda$

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



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

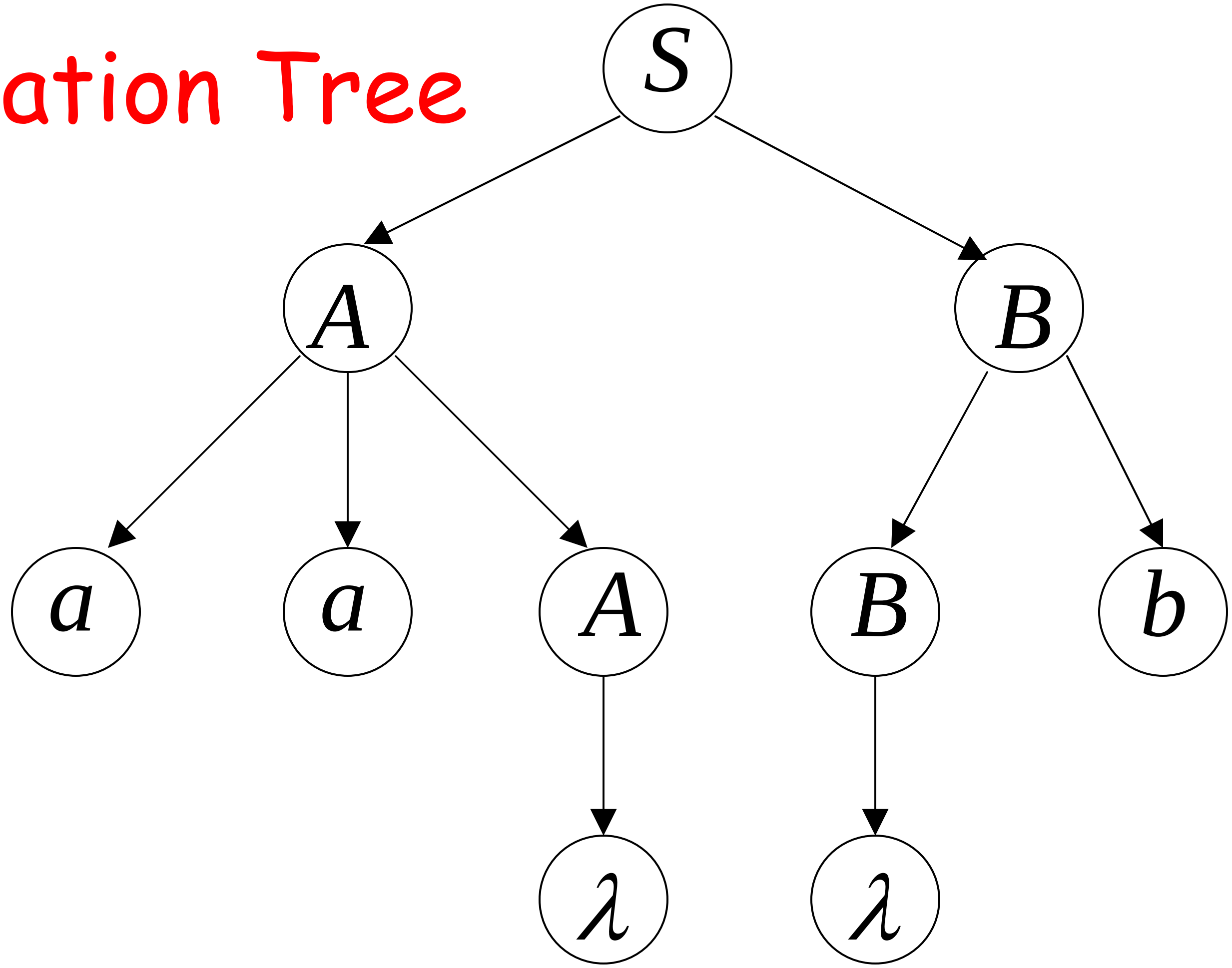
$$S \Rightarrow AB \Rightarrow aaAB \Rightarrow aaABb \Rightarrow aaBb$$



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

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

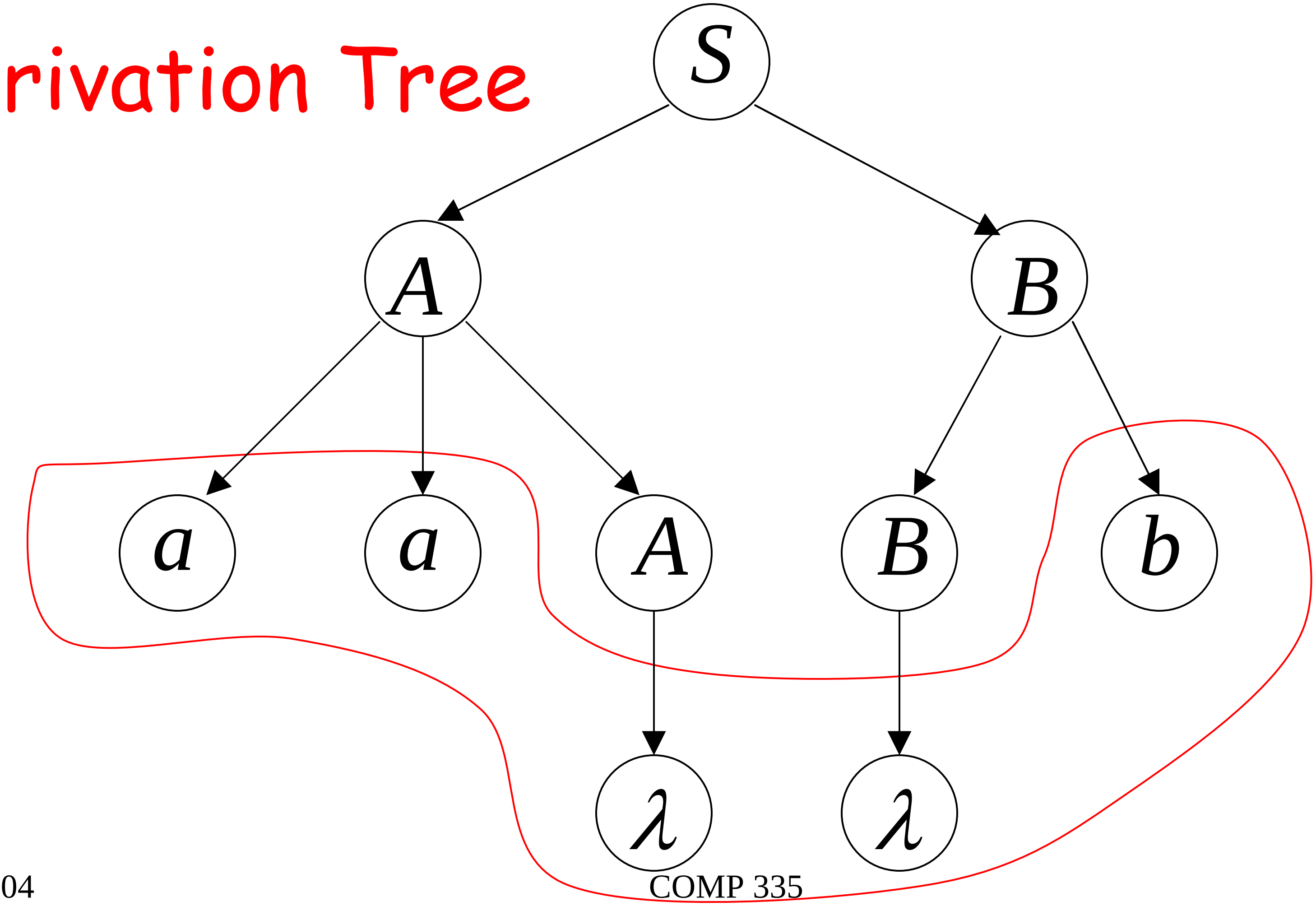
Derivation Tree



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

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

Derivation Tree



yield

$aa\lambda\lambda b$

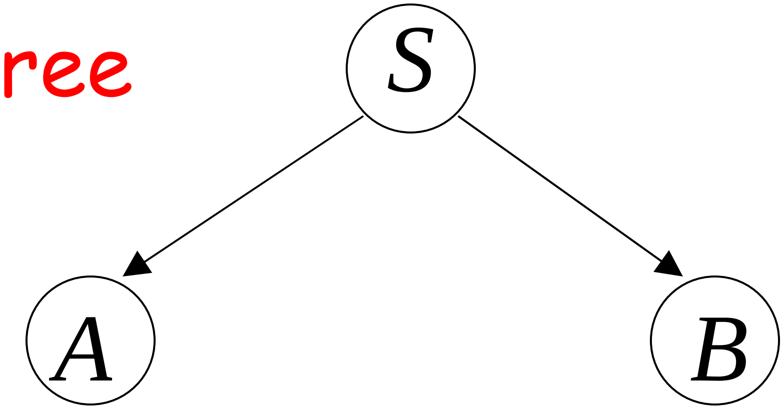
$= aab$

Partial Derivation Trees

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

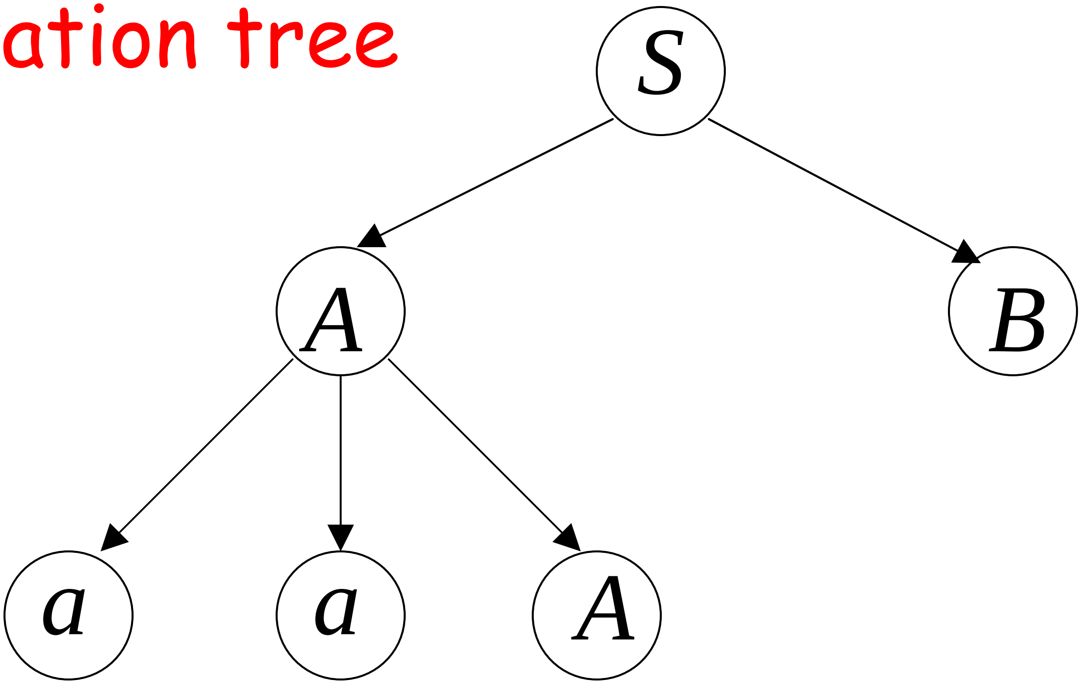
$$S \Rightarrow AB$$

Partial derivation tree



$$S \Rightarrow AB \Rightarrow aaAB$$

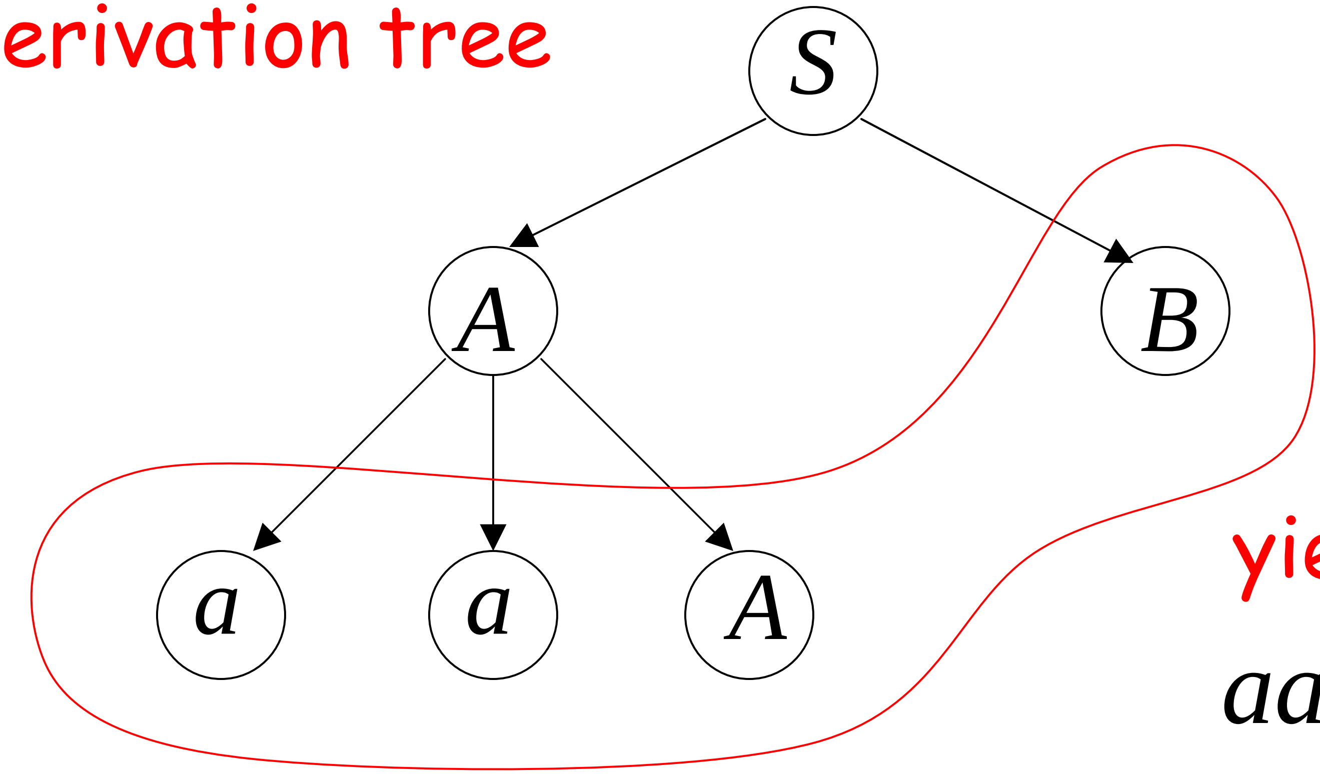
Partial derivation tree



$$S \Rightarrow AB \Rightarrow aaAB$$

sentential form

Partial derivation tree



yield

$aaAB$

Sometimes, derivation order doesn't matter

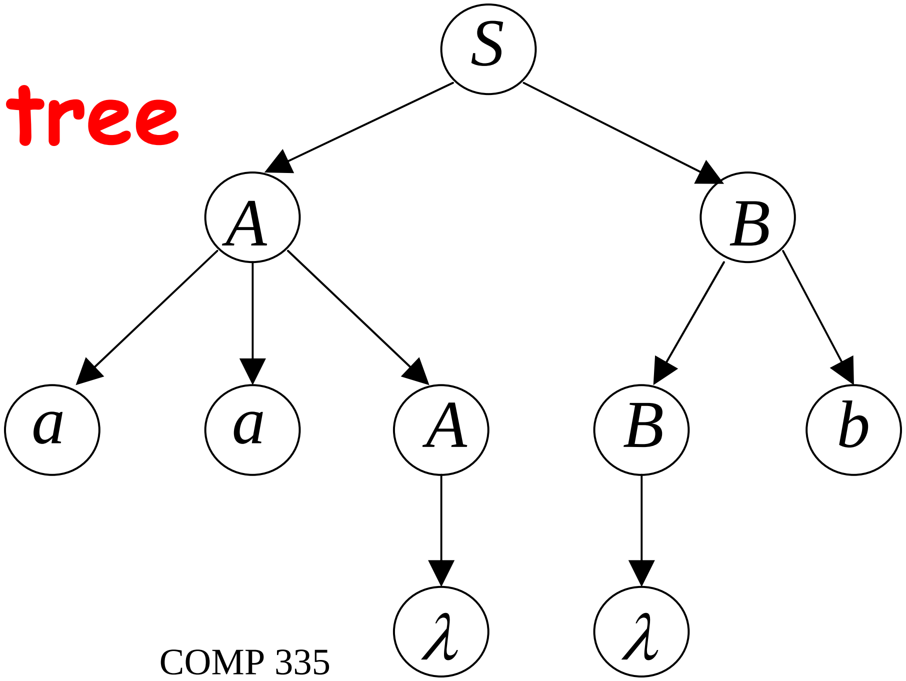
Leftmost:

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

Rightmost:

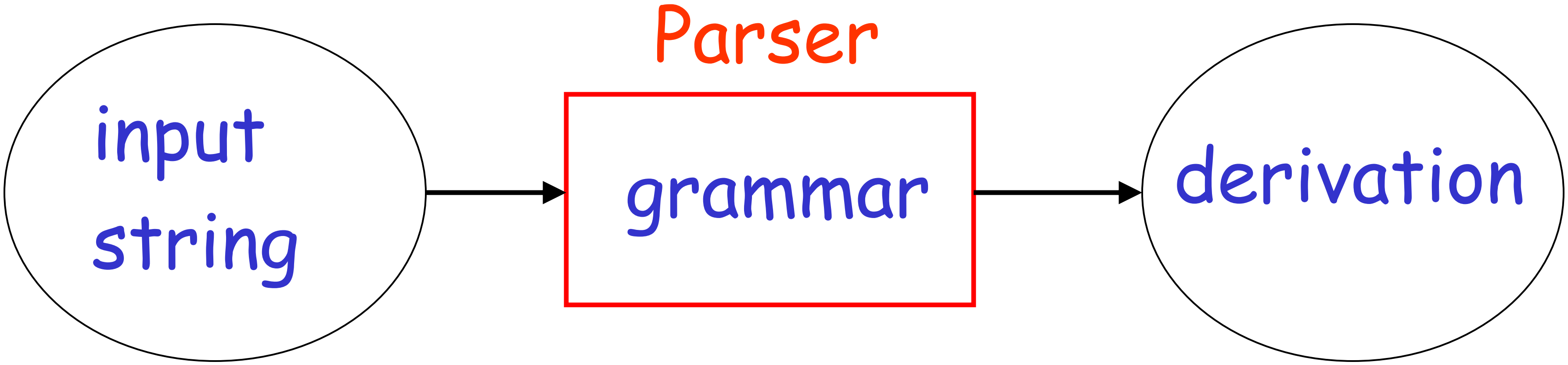
$$S \Rightarrow AB \Rightarrow ABb \Rightarrow Ab \Rightarrow aaAb \Rightarrow aab$$

Same derivation tree



So far, we concentrated on generative aspect of grammars.

How about analytical aspect?
Parsing.....



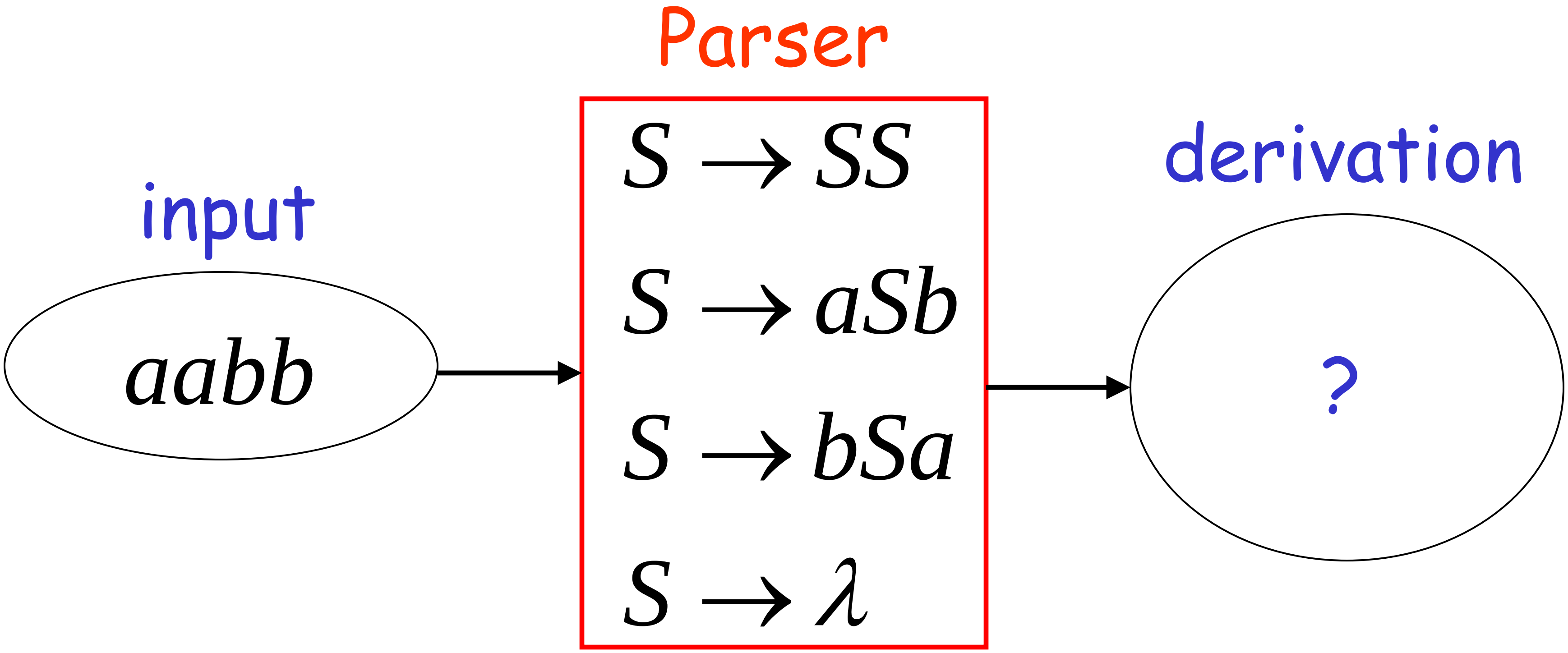
Parser

input string

grammar

derivation

Example:



input

$aabb$

Parser

$$S \rightarrow SS$$

$$S \rightarrow aSb$$

$$S \rightarrow bSa$$

$$S \rightarrow \lambda$$

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

$$S \Rightarrow SS$$

$$S \Rightarrow aSb$$

$$\xcancel{S \Rightarrow bSa}$$

$$\xcancel{S \Rightarrow \lambda}$$

$aabb$

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

$aabb$

Phase 1

$S \Rightarrow SS$

$S \Rightarrow aSb$

Phase 2

$S \Rightarrow SS \Rightarrow SSS$

$S \Rightarrow SS \Rightarrow aSbS$

~~$S \Rightarrow SS \Rightarrow bSaS$~~

$S \Rightarrow SS \Rightarrow S$

$S \Rightarrow aSb \Rightarrow aSSb$

$S \Rightarrow aSb \Rightarrow aaSbb$

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

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

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

Phase 2

$$S \Rightarrow SS \Rightarrow SSS$$

$$S \Rightarrow SS \Rightarrow aSbS$$

$$aabb$$

$$S \Rightarrow SS \Rightarrow S$$

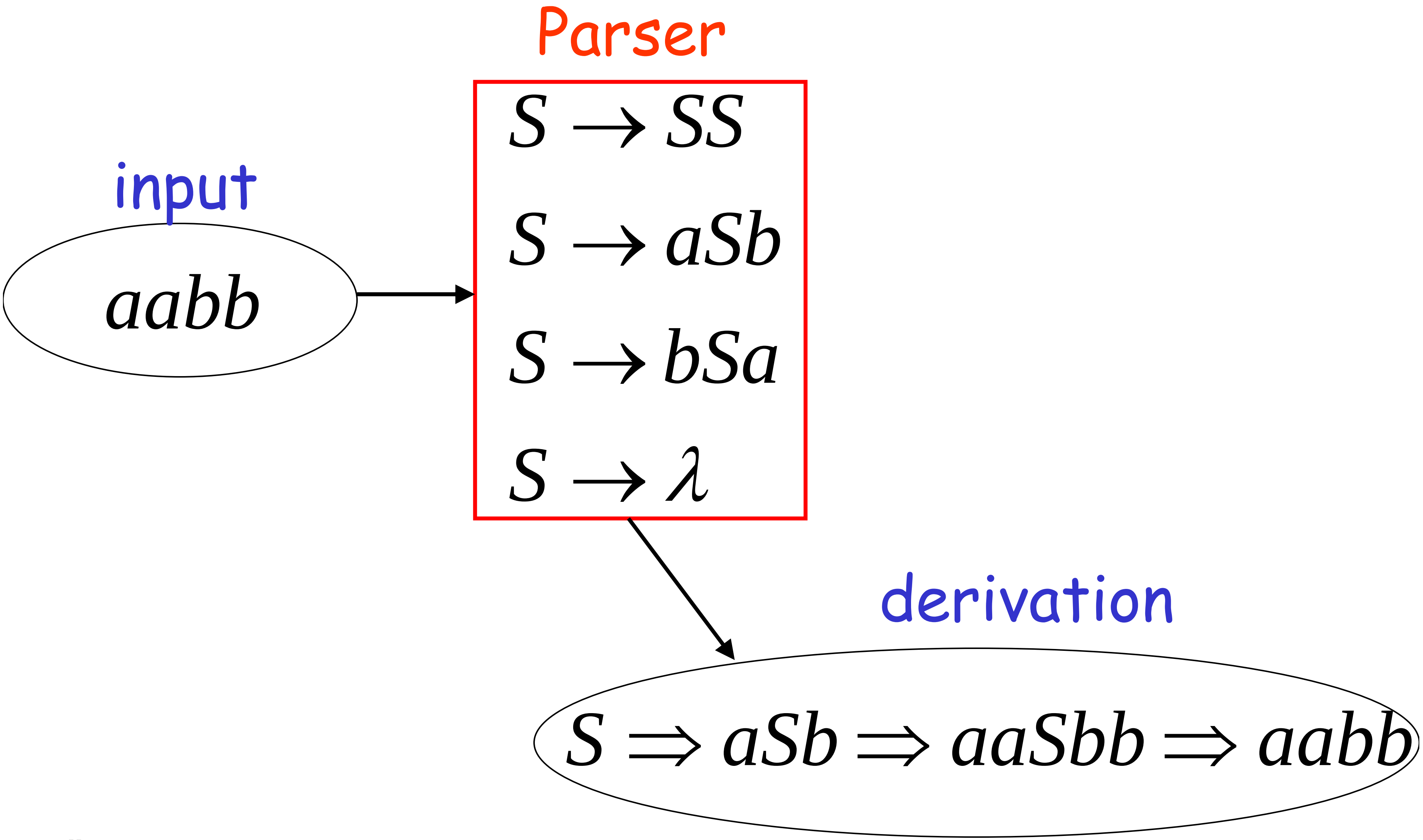
$$S \Rightarrow aSb \Rightarrow aSSb$$

$$S \Rightarrow aSb \Rightarrow aaSbb$$

Phase 3

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

Final result of exhaustive search (top-down parsing)



input

$aabb$

Parser

$S \rightarrow SS$

$S \rightarrow aSb$

$S \rightarrow bSa$

$S \rightarrow \lambda$

derivation

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

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 is $2|w|$

Why?

For grammar with k rules

Time for phase 1: k

k possible derivations

Time for phase 2: k^2

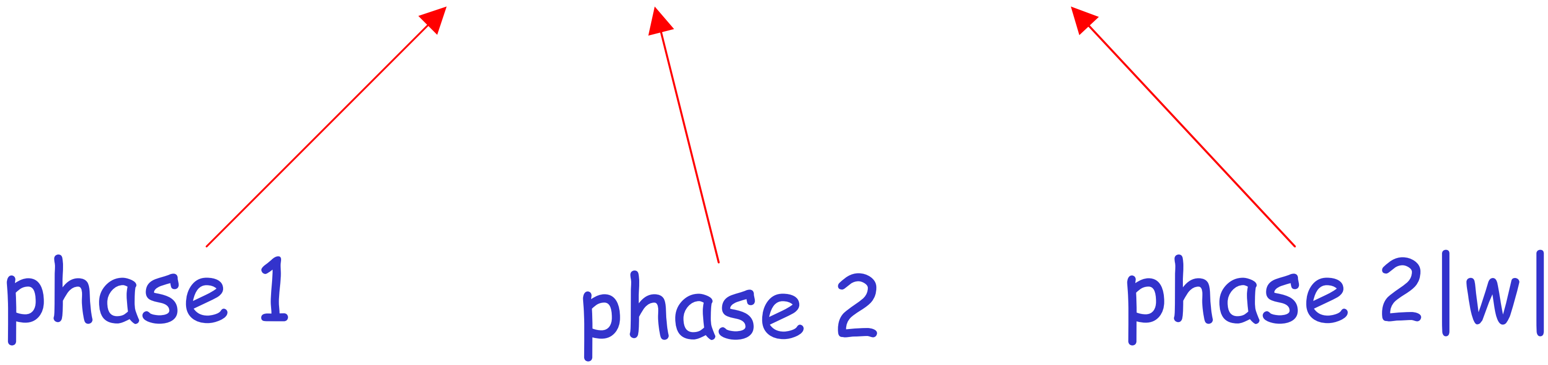
k^2 possible derivations

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

possible derivations: $k^{2|w|}$
which is exponential in the length of w

Total time needed for parsing w :

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



phase 1

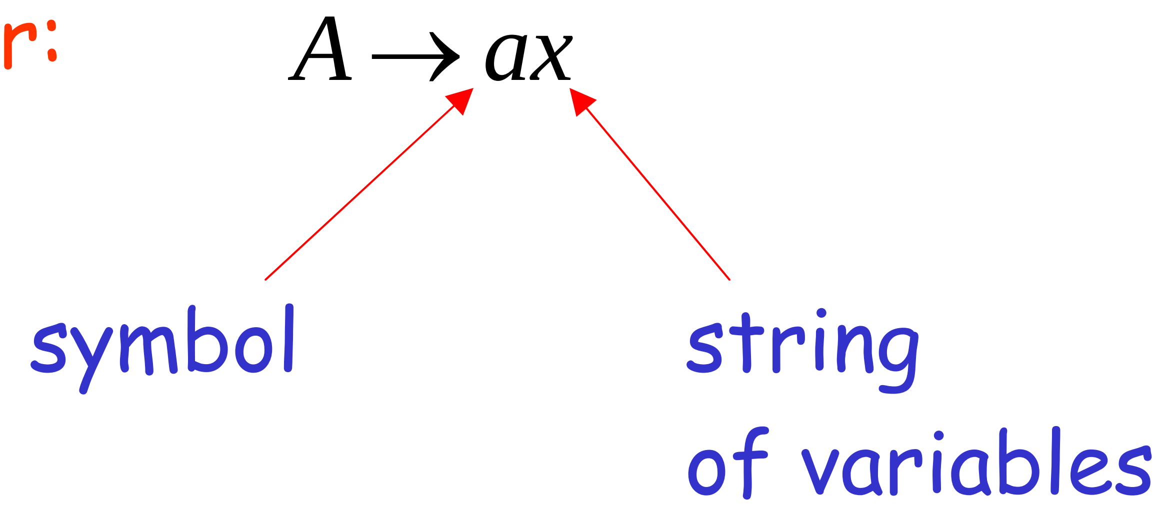
phase 2

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)

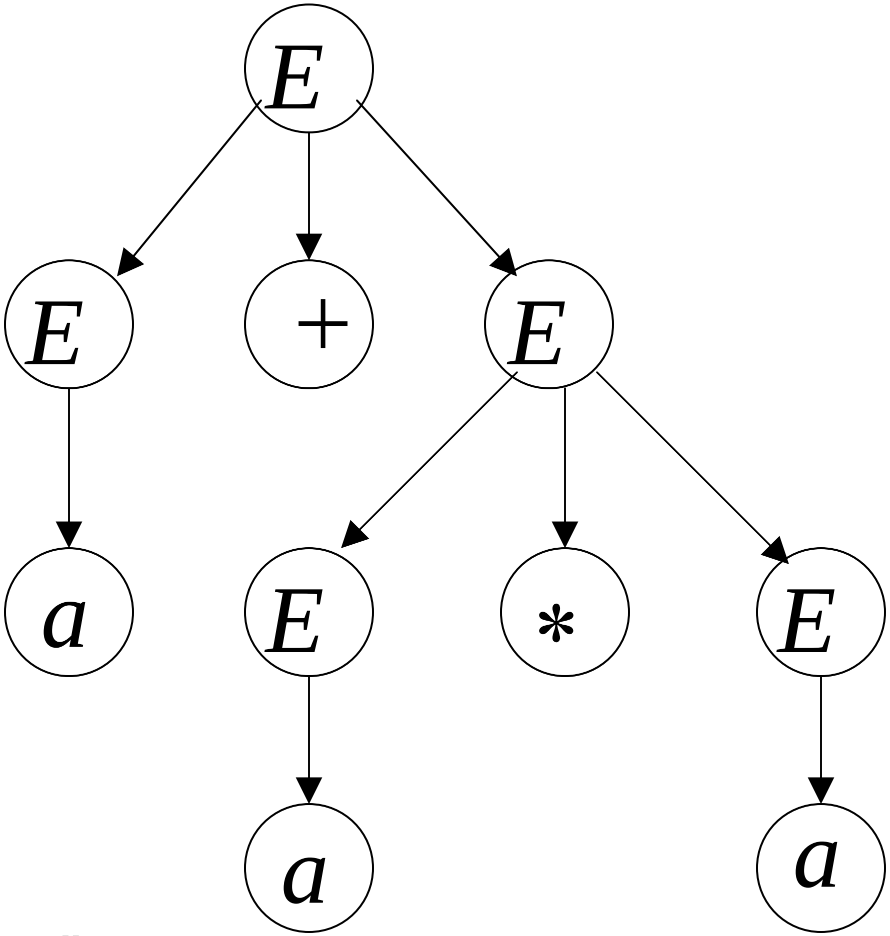
Ambiguity

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

$$w = a + a * a$$

$$E \Rightarrow^{1} E + E \Rightarrow^{4} a + E \Rightarrow^{2} a + E * E$$

$$\Rightarrow^{4} a + a * E \Rightarrow^{4} a + a * a$$



leftmost derivation

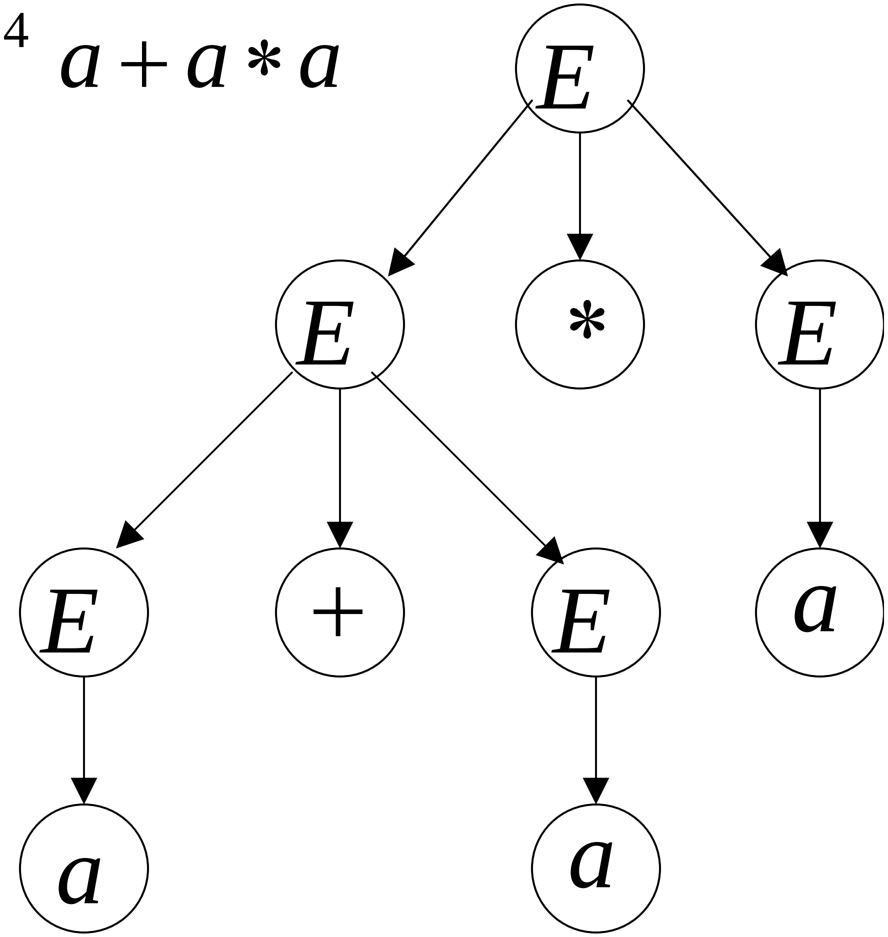
$$E \to E + E \quad | \; E * E \; | \; (E) \; | \; a$$

$$w = a + a * a$$

$$E \Rightarrow^{2} E * E \Rightarrow^{1} E + E * E \Rightarrow^{4} a + E * E$$

$$\Rightarrow^{4} a + a * E \Rightarrow^{4} a + a * a$$

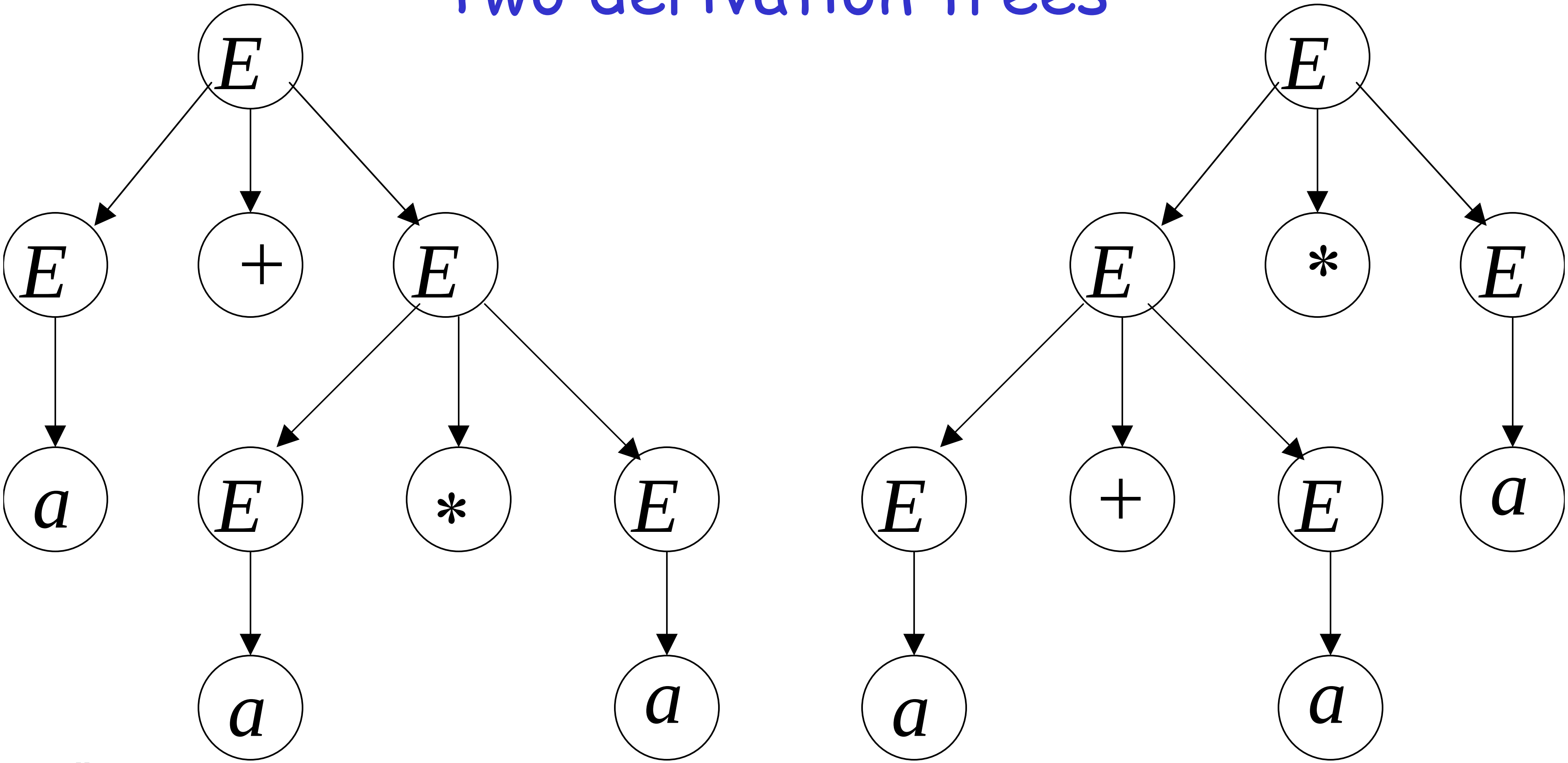
leftmost derivation



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

$$w = a + a * a$$

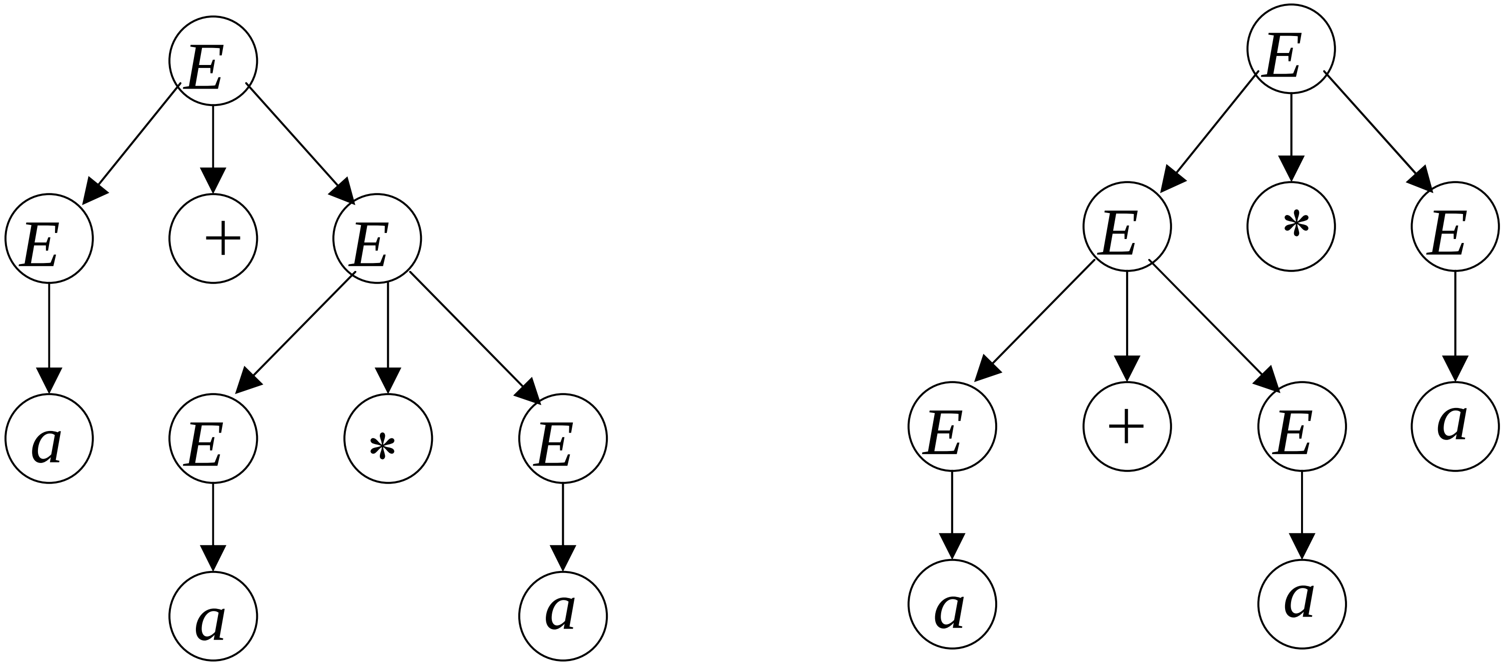
Two derivation trees



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

is **ambiguous:** since there is a string, namely

$w = a + a * a$, which has two derivation trees



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

is ambiguous: since string $a + a * a$

has two leftmost derivations

$$E \Rightarrow^{1} E + E \Rightarrow^{4} a + E \Rightarrow^{2} a + E * E$$

$$\Rightarrow^{4} a + a * E \Rightarrow^{4} a + a * a$$

$$E \Rightarrow^{2} E * E \Rightarrow^{1} E + E * E \Rightarrow^{4} a + E * E$$

$$\Rightarrow^{4} a + a * E \Rightarrow^{4} a + a * a$$

Definition:

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

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

two or more *distinct* derivation trees

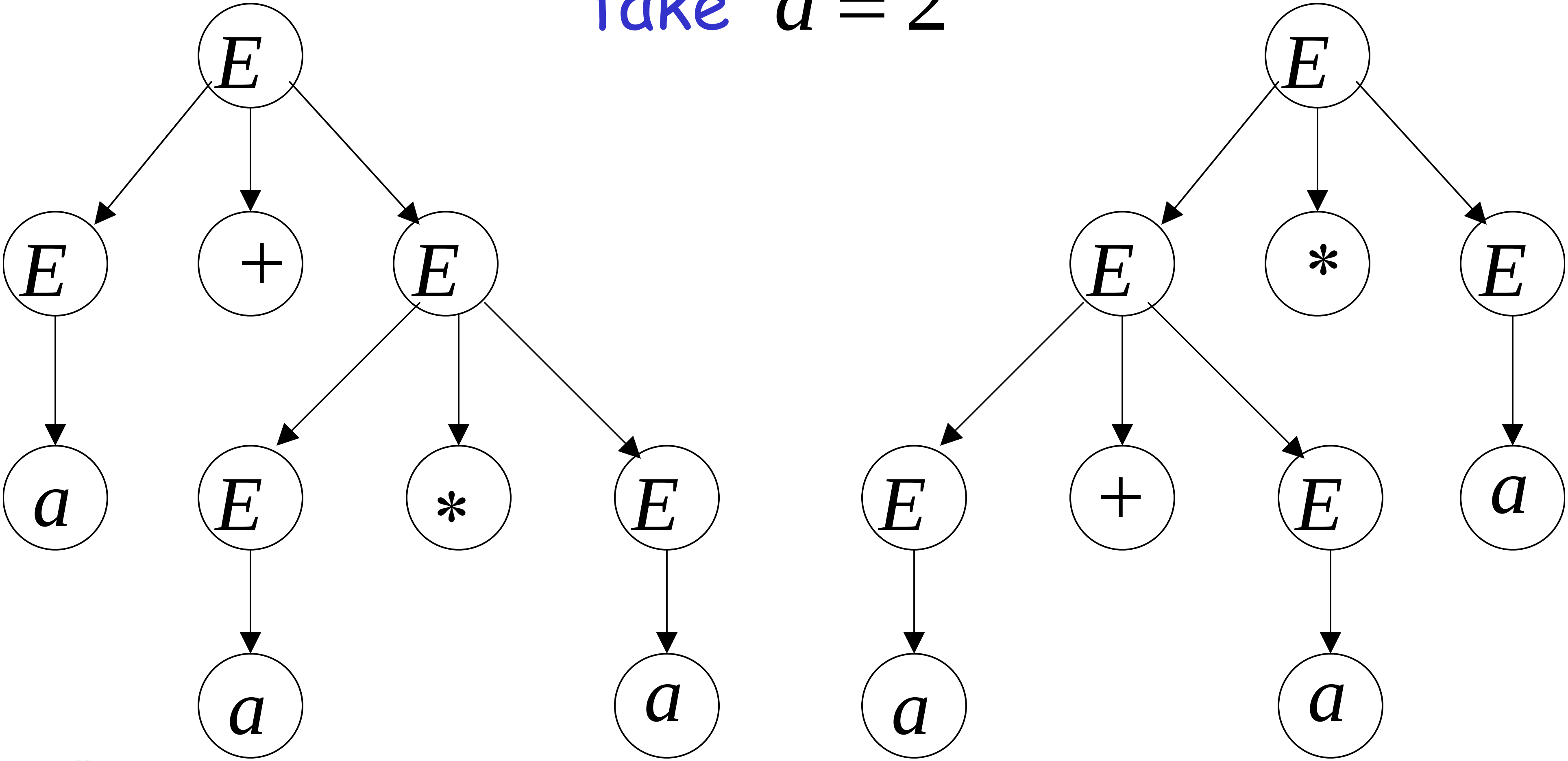
Alternatively we may say:

Ambiguity of a grammar G implies the existence of *two or more* leftmost (or rightmost) derivations

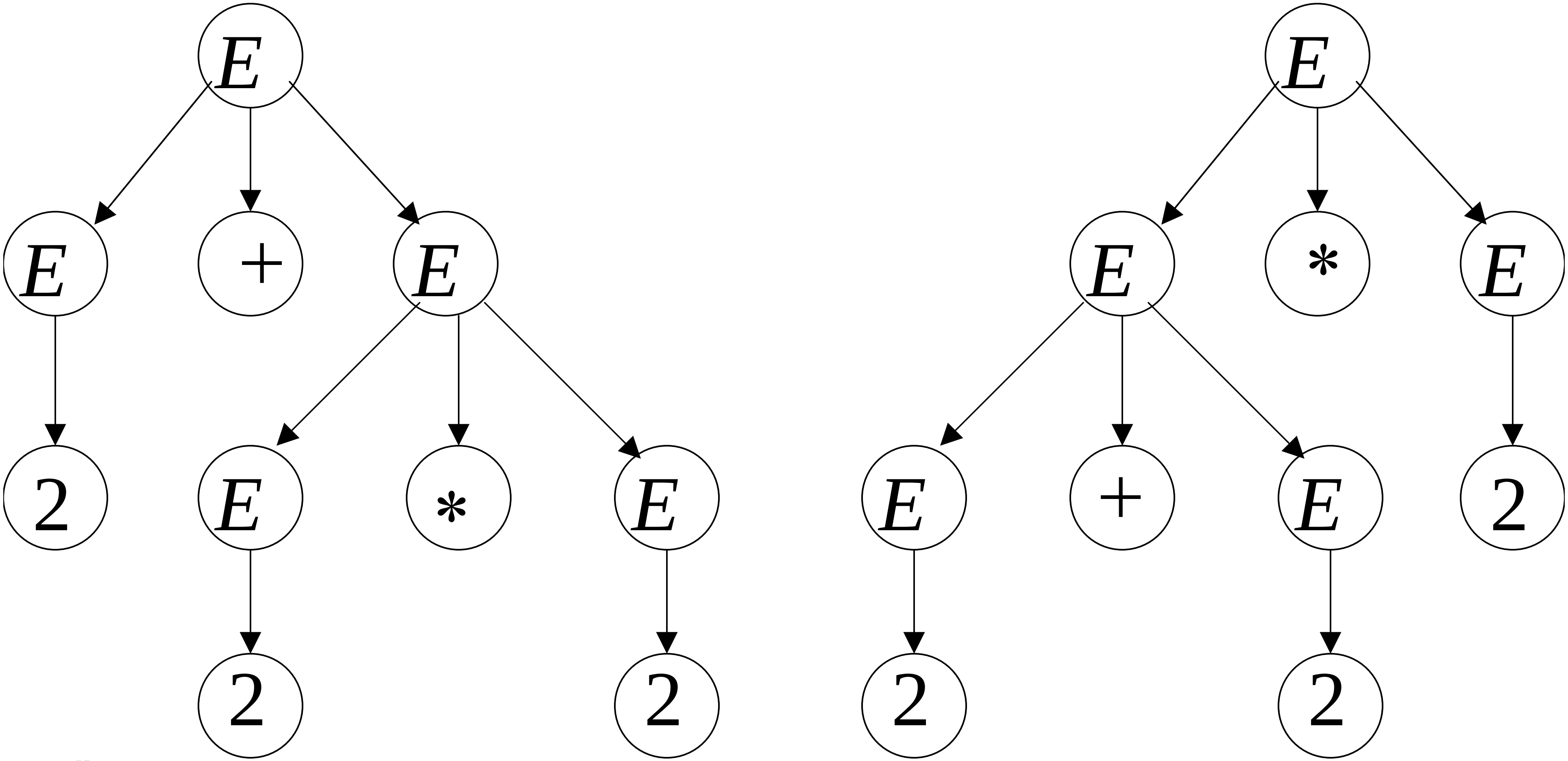
Why do we care about ambiguity?

$$a + a * a$$

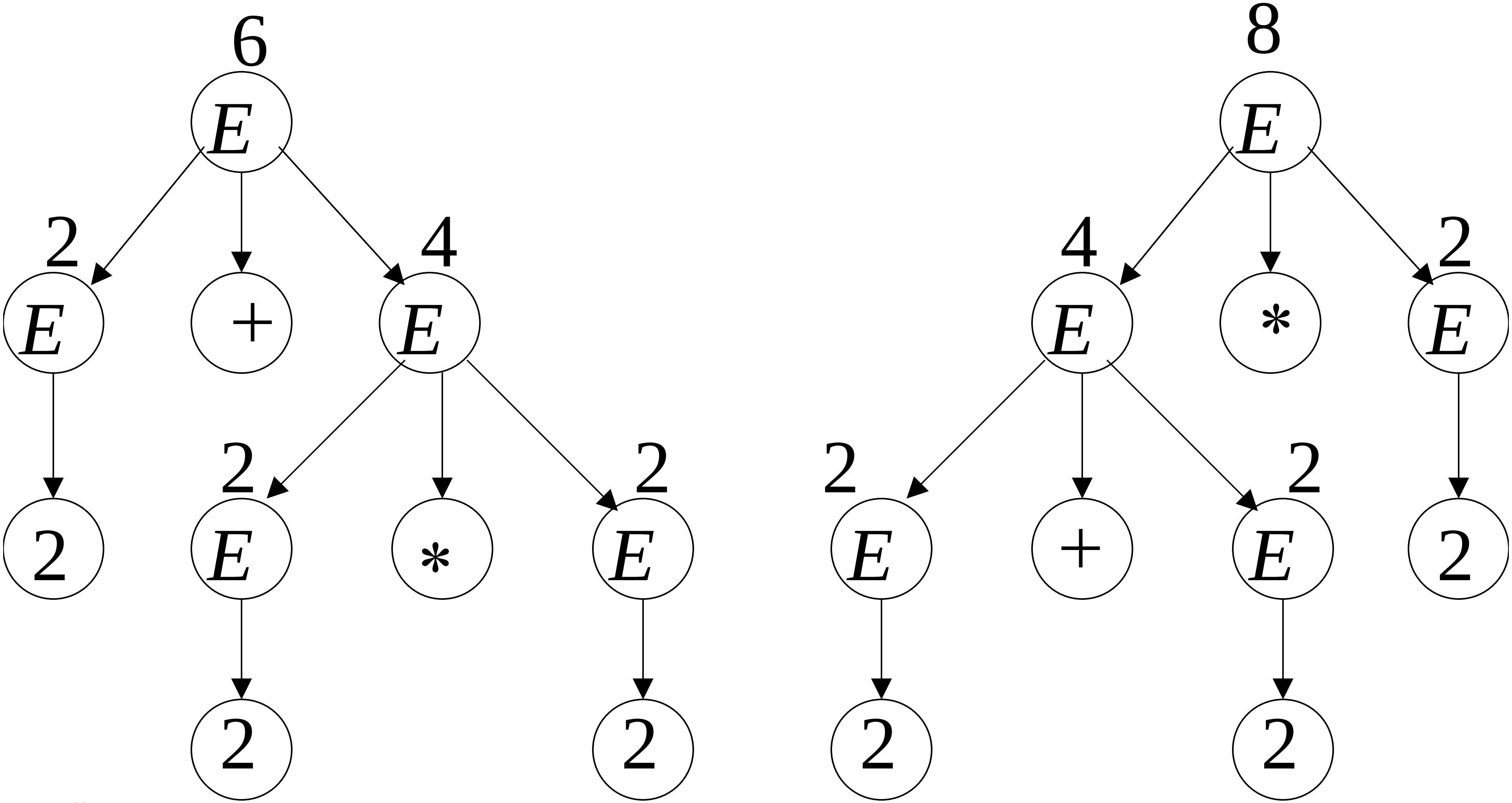
take $a = 2$



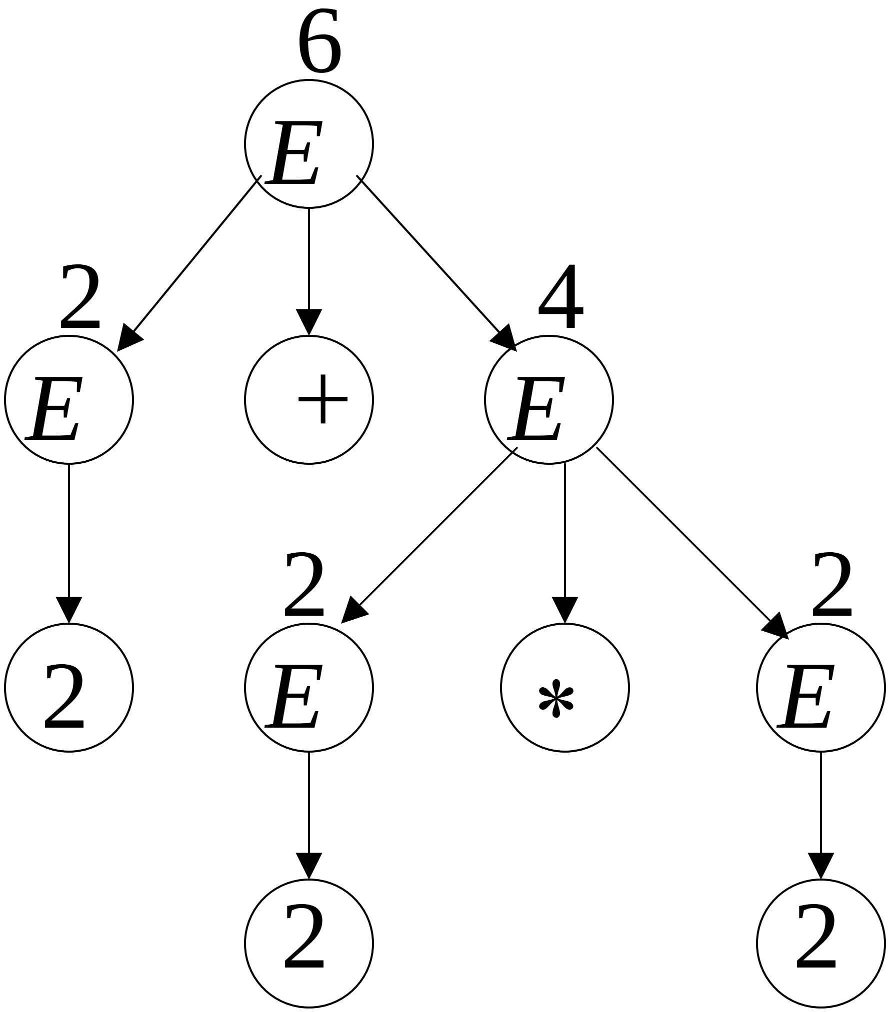
$$2 + 2 * 2$$



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



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



- Ambiguity is **bad** for programming languages

- We want to remove ambiguity

We may be able to fix the ambiguity:

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

New non-ambiguous grammar:

$$E \to E + T$$

$$E \to T$$

$$T \to T * F$$

$$T \to F$$

$$F \to (E)$$

$$F \to a$$

$$E \Rightarrow^1 E+T \Rightarrow^2 T+T \Rightarrow^4 F+T \Rightarrow^6 a+T \Rightarrow^3 a+T*F$$

$$\Rightarrow^4 a+F*F \Rightarrow^6 a+a*F \Rightarrow^6 a+a*a$$

$w = a + a * a$

$E \rightarrow^1 E+T$

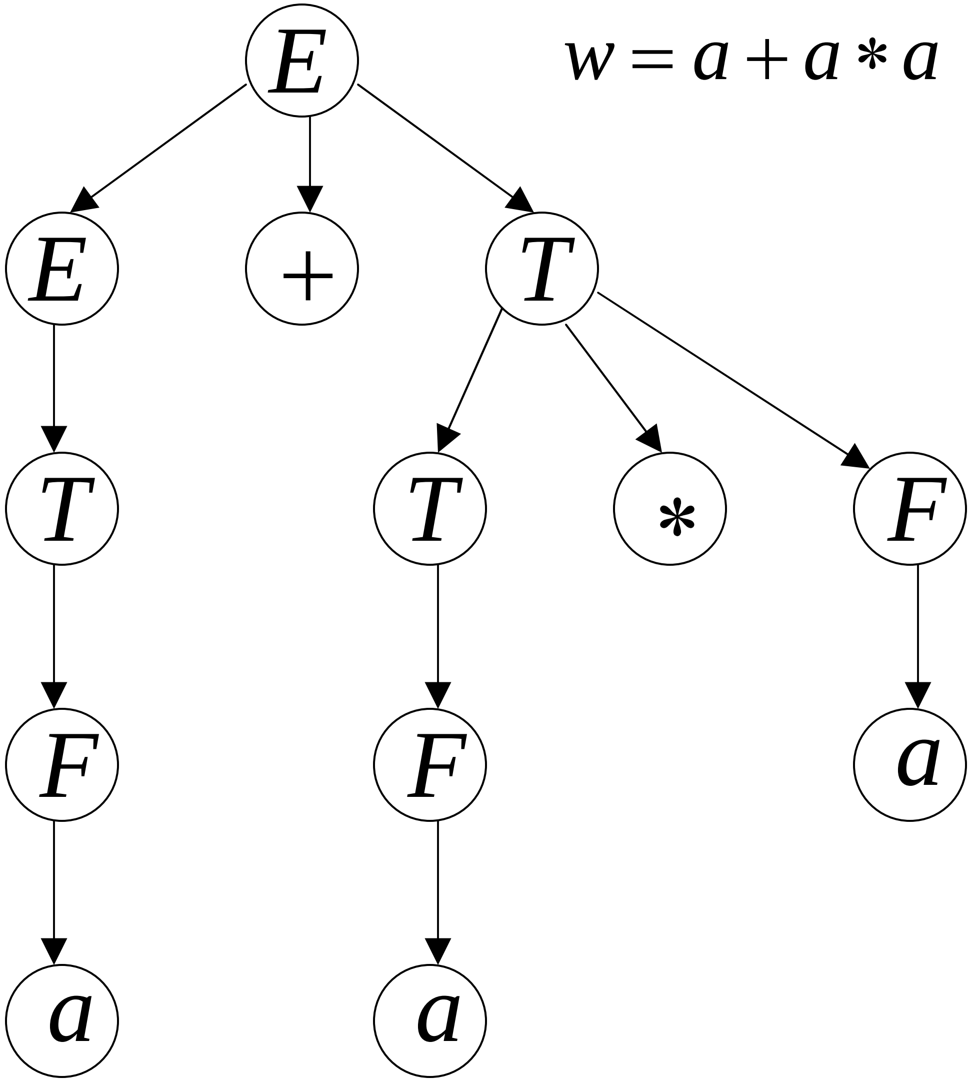
$E \rightarrow^2 T$

$T \rightarrow^3 T*F$

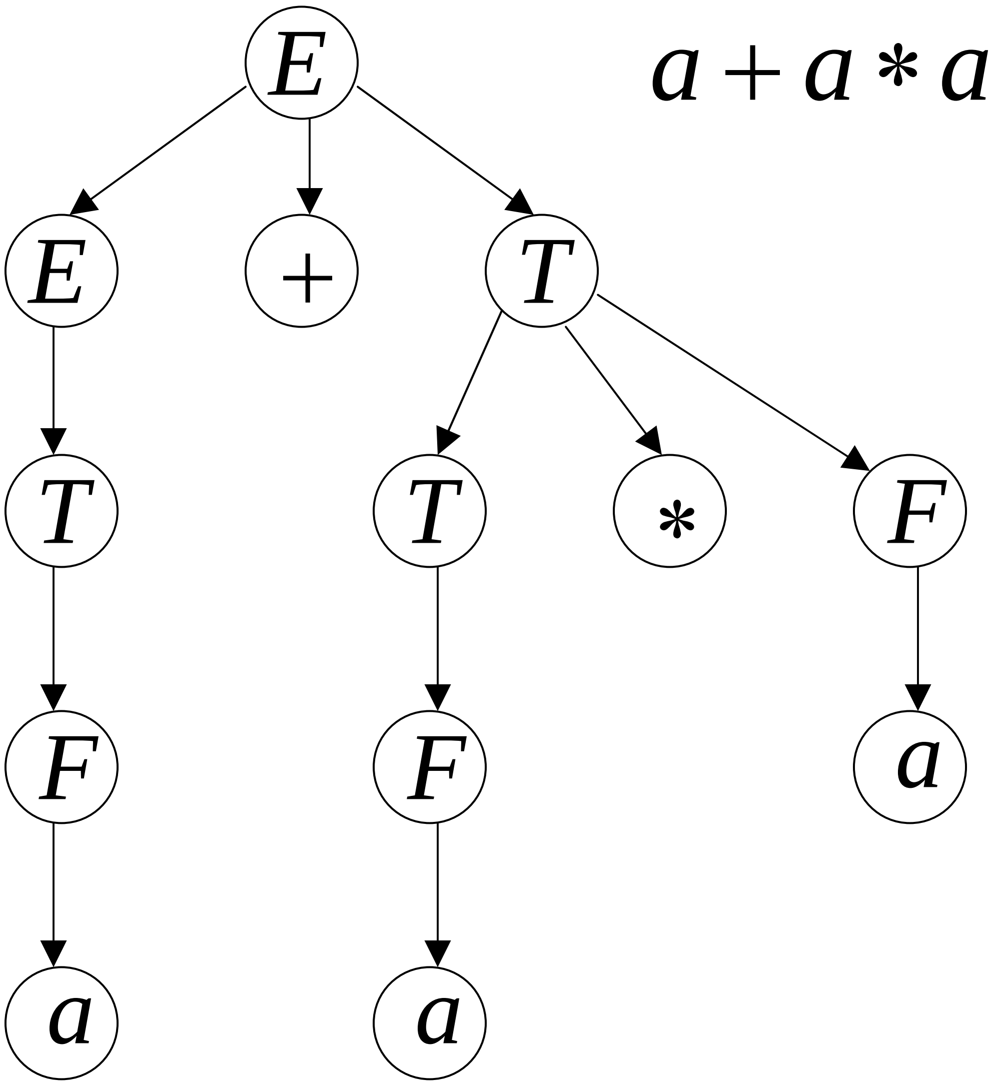
$T \rightarrow^4 F$

$F \rightarrow^5 (E)$

$F \rightarrow^6 a$



Unique derivation tree



$a + a * a$

The grammar G:

$$E \xrightarrow{1} E+T$$

$$E \xrightarrow{2} T$$

$$T \xrightarrow{3} T*F$$

$$T \xrightarrow{4} F$$

$$F \xrightarrow{5} (E)$$

$$F \xrightarrow{6} a$$

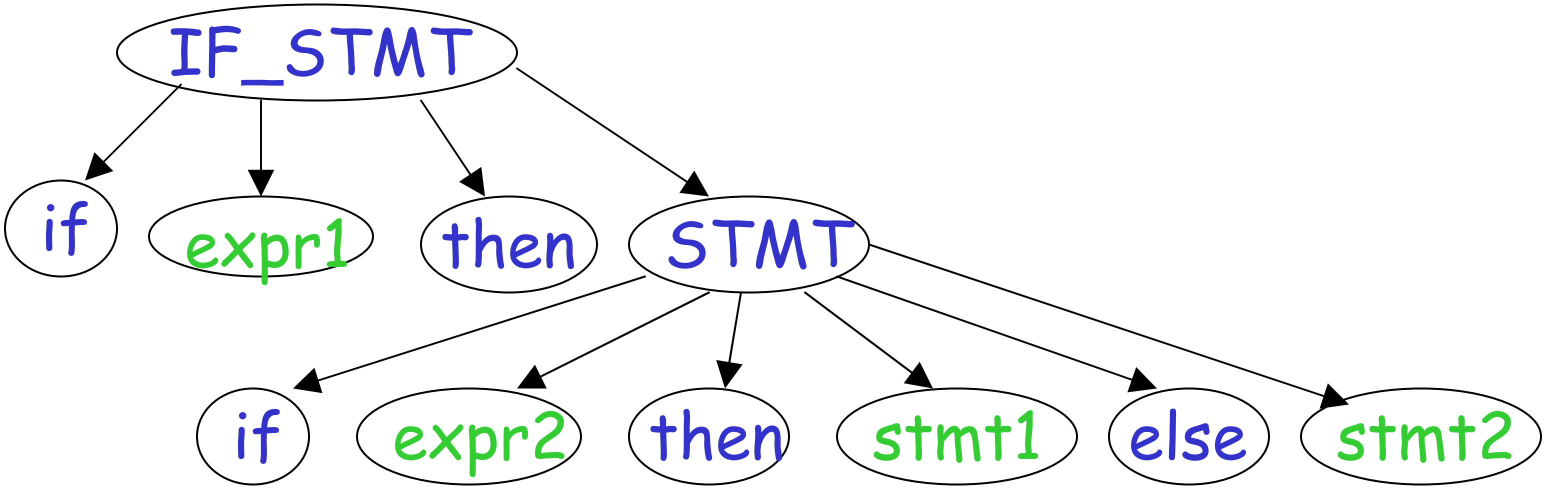
is non-ambiguous:

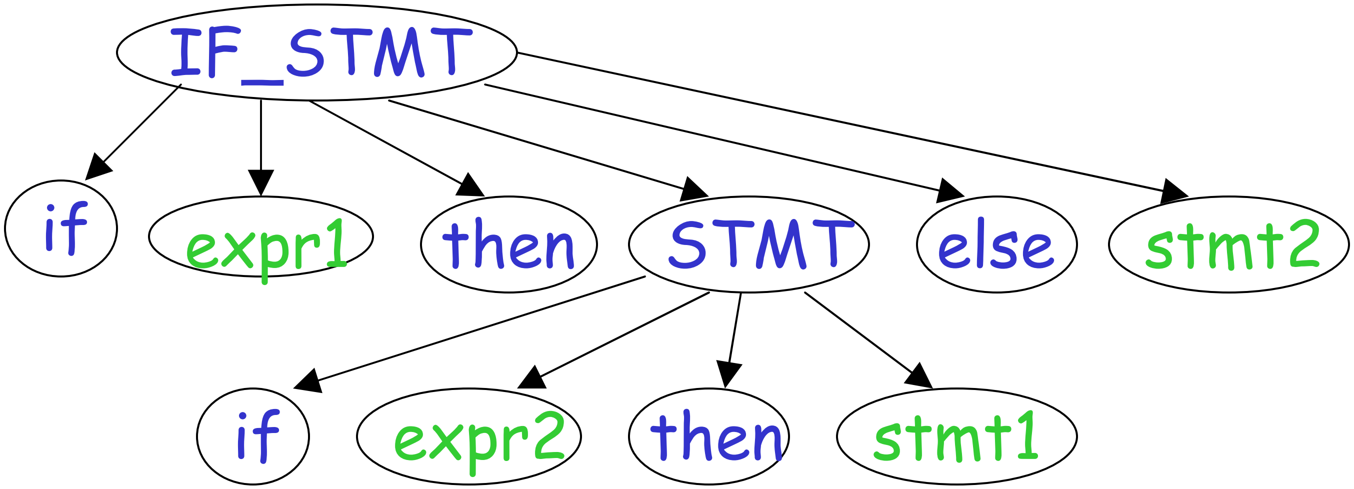
Every string $w \in L(G)$ has a unique derivation tree

Another Ambiguous Grammar

$$\text{IF\_STMT} \rightarrow \text{if EXPR then STMT} \mid \text{if EXPR then STMT else STMT}$$

if expr1 then if expr2 then stmt1 else stmt2





Inherent Ambiguity

Some context-free languages
have only ambiguous grammars

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

$$S \rightarrow S_1 \mid S_2$$

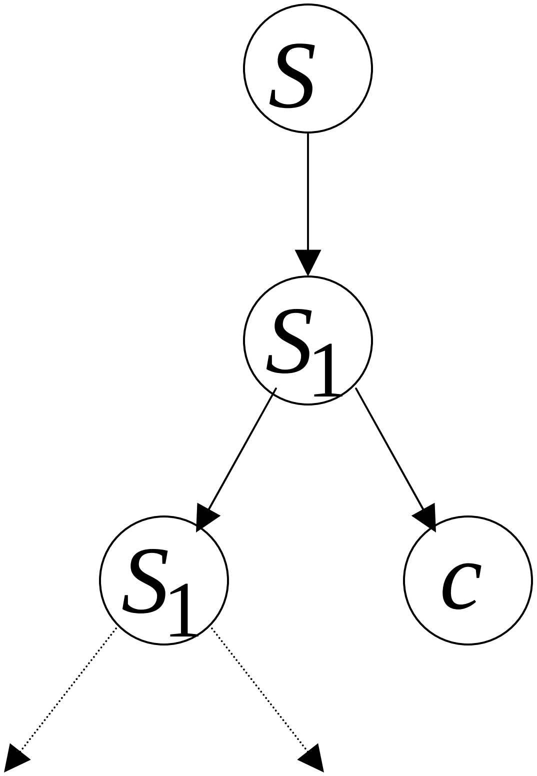
$$S_1 \rightarrow S_1 c \mid A$$

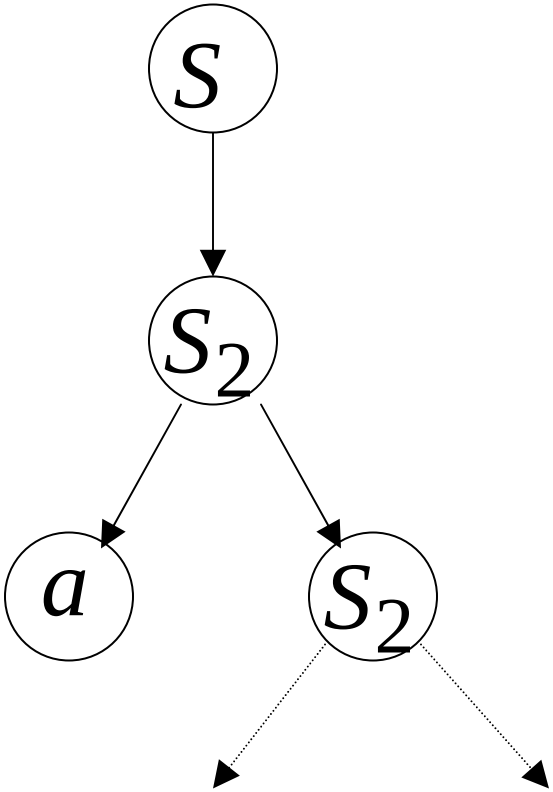
$$A \rightarrow aAb \mid \lambda$$

$$S_2 \rightarrow aS_2 \mid B$$

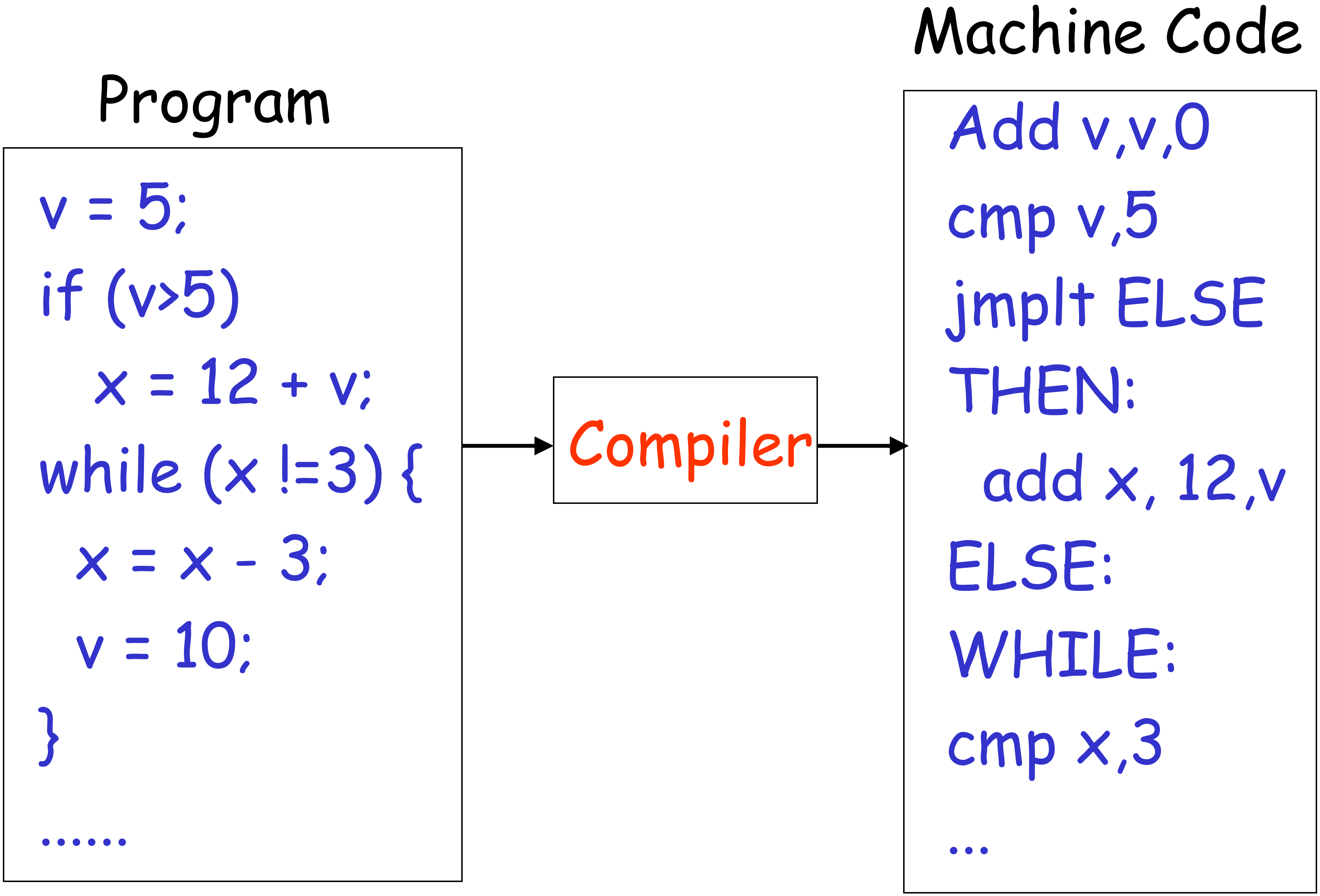
$$B \rightarrow bBc \mid \lambda$$

The string $a^n b^n c^n$ has two derivation trees





Compilers



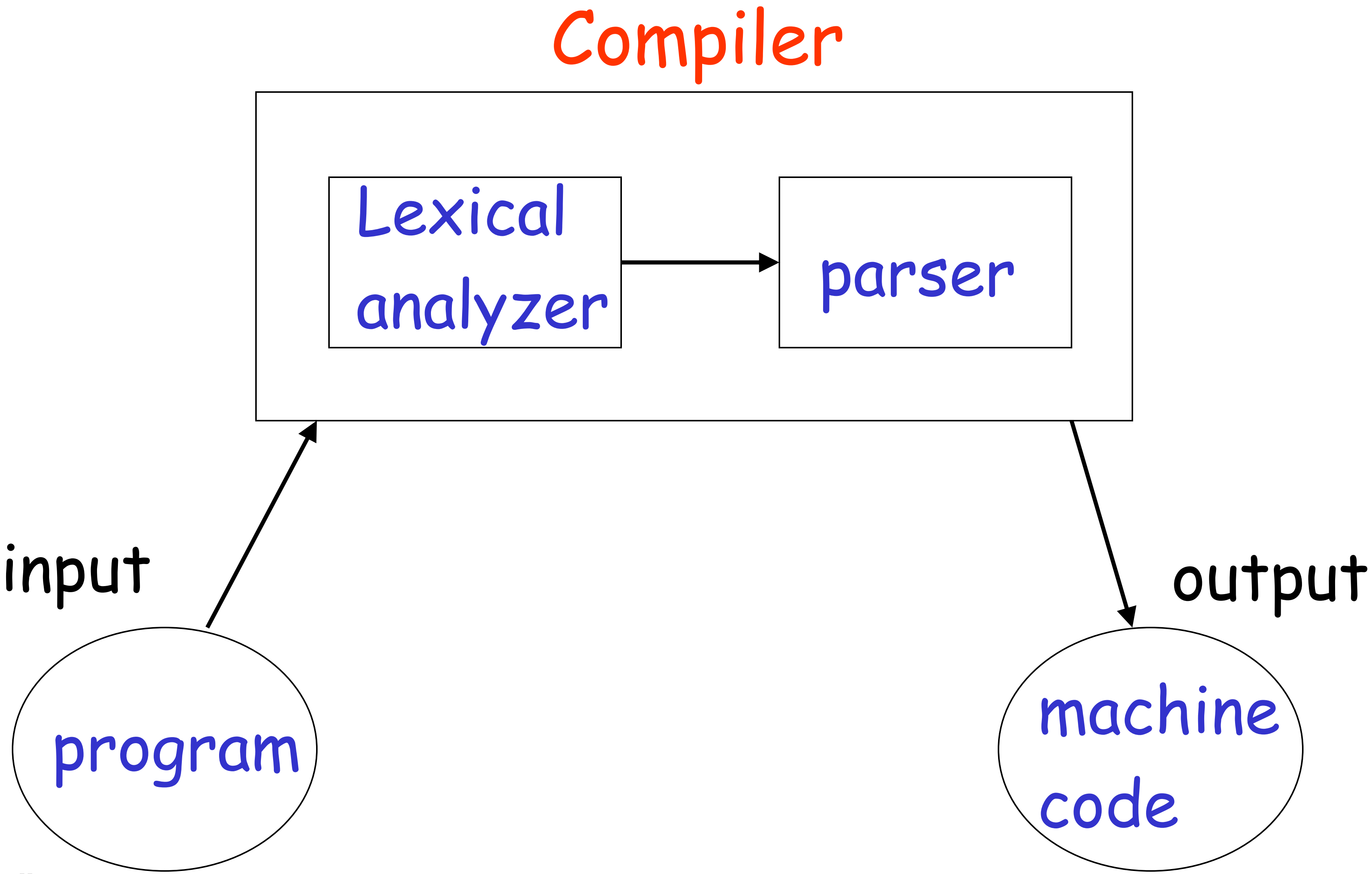
Program

```
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 }

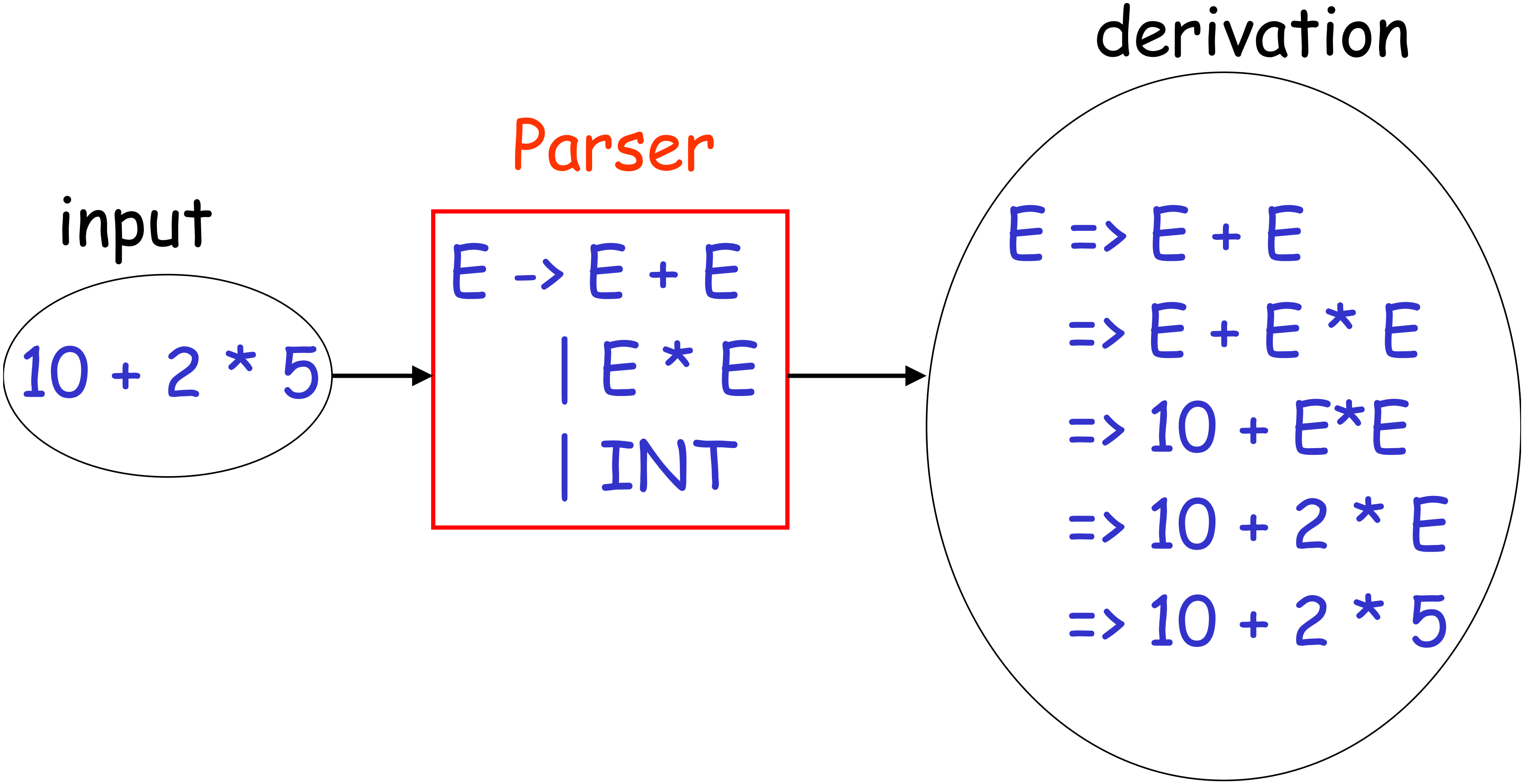
EXPR → EXPR + EXPR | EXPR - EXPR | ID

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

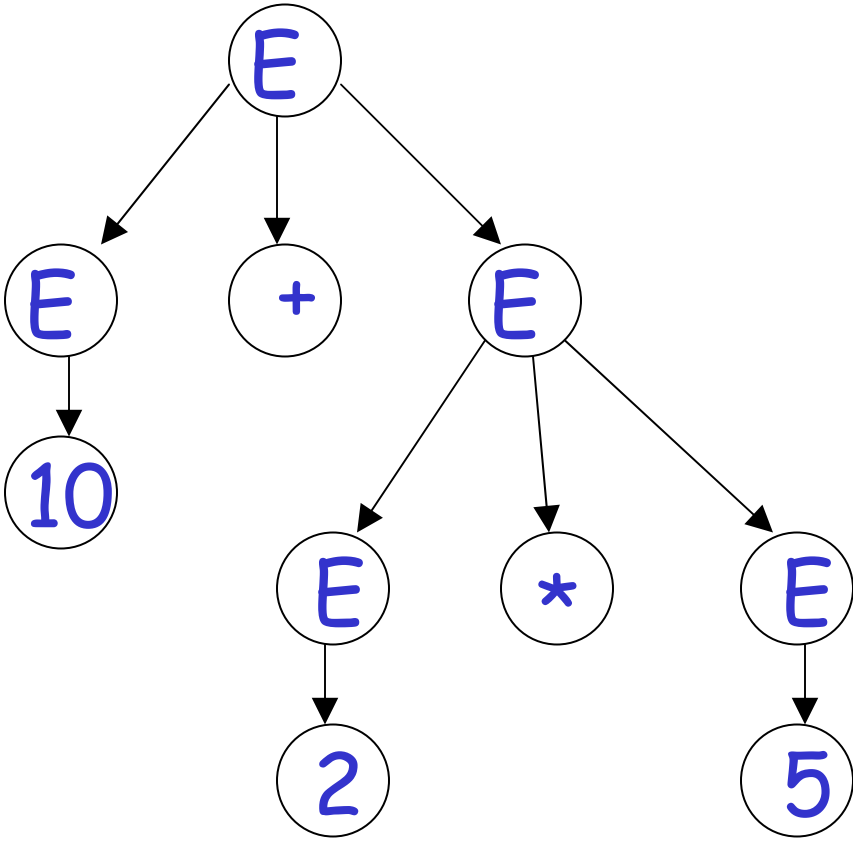
derivation

E => E + E
  => E + E * E
  => 10 + E*E
  => 10 + 2 * E
  => 10 + 2 * 5

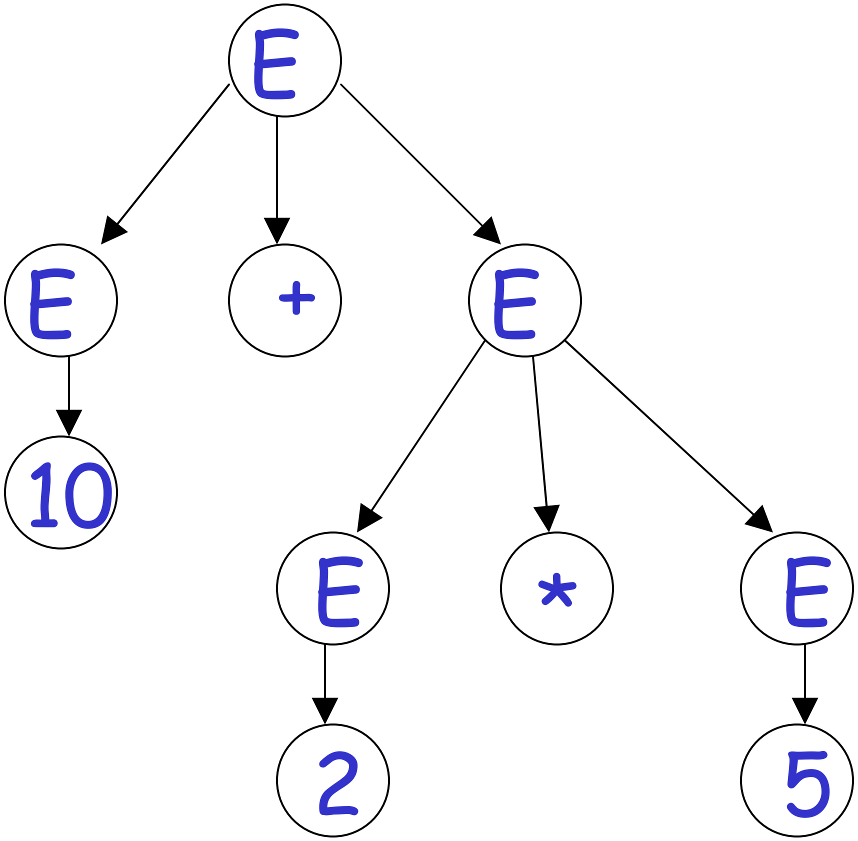
derivation

E => E + E
=> E + E * E
=> 10 + E*E
=> 10 + 2 * E
=> 10 + 2 * 5

derivation tree



derivation tree



machine code

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