2023

Theory of Computation

Kun-Ta Chuang
Department of Computer Science and Information Engineering
National Cheng Kung University

Outline

1 Context-Free Grammars

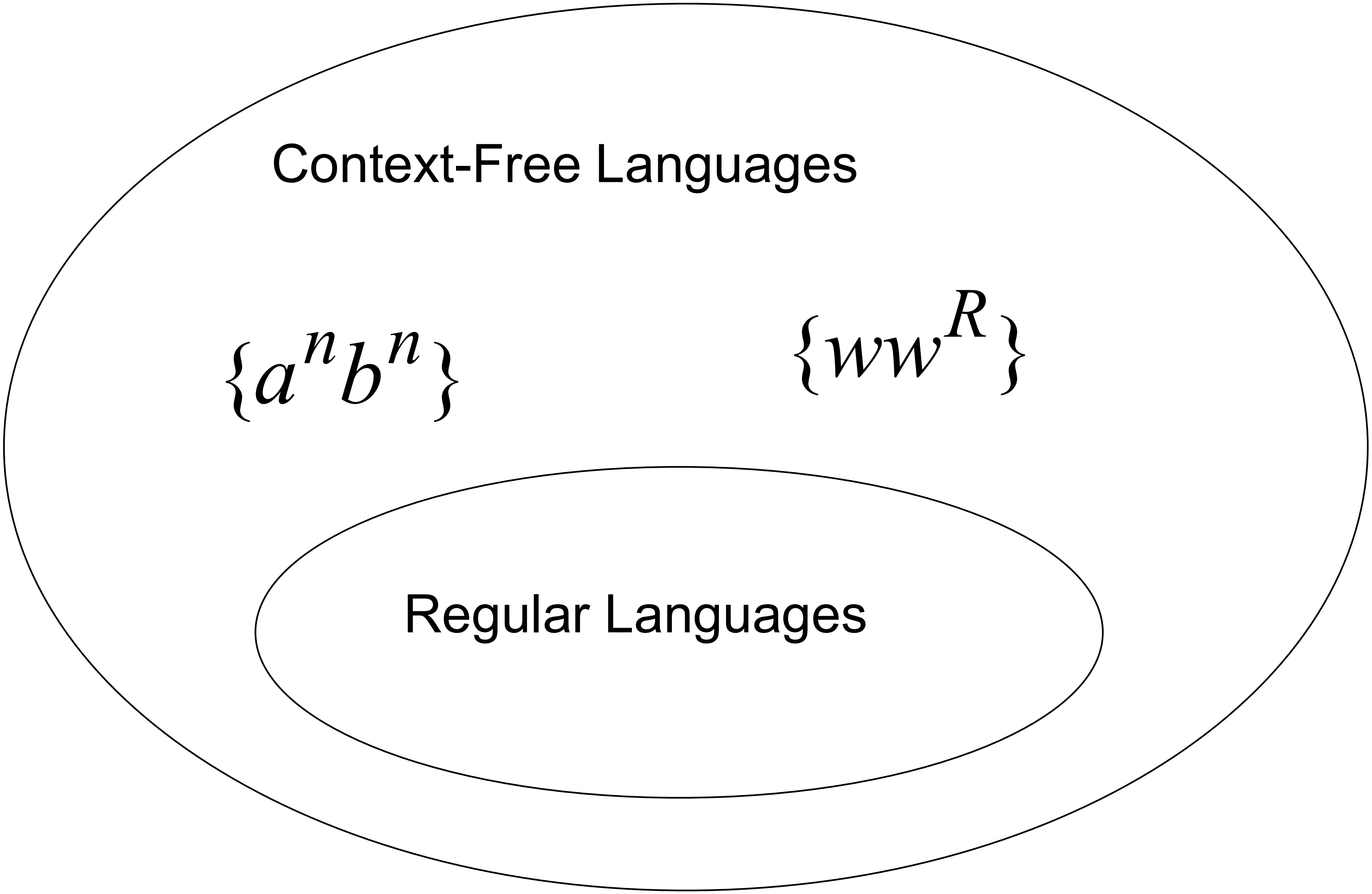
2 Parsing and Ambiguity

3 Context-Free Grammars and Programming Languages

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

Regular Languages

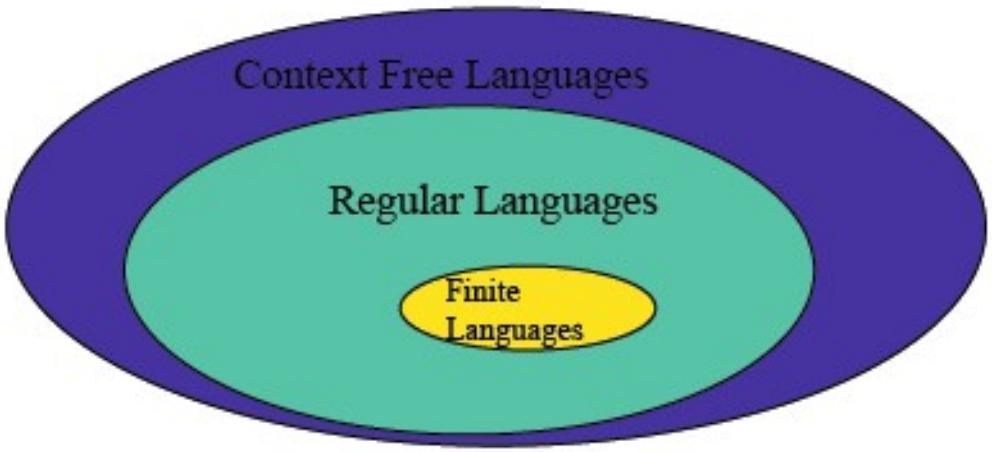
a^*b^* $(a+b)^*$


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

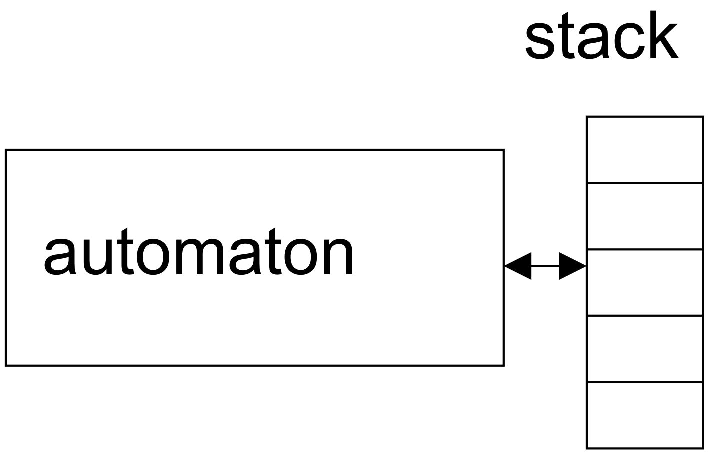
Context-Free Languages

Context-Free Grammars

Pushdown Automata



与所屬關係



Context-Free Grammars

Regular Grammar

$$S \to abS$$

$$S \to a$$

$$S \to Aab$$

$$A \to Aab \mid B$$

$$B \to a$$

Definition 5.1: Context-Free Grammars

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

Variables

Terminal symbols

Start variable

Productions of the form:

Variable $A \rightarrow x$ String of variables and terminals $x \in (V \cup T)^*$

$A \in V$

↱ V和T的任意組合

We say that the grammar is context-free since this substitution can take place regardless of where A is.

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

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

Regular and linear grammars are clearly context-free
But a context-free grammar is not necessarily linear

Definition: Context-Free Languages

A language L is context-free

if and only if

there is a context-free grammar G

with $L = L(G)$

Example

A context-free grammar : G

$$S \to aSb$$

$$S \to \lambda$$

A derivation:

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

Example

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 5.1

A context-free grammar: G

$$S \to aSa$$

$$S \to bSb$$

$$S \to \lambda$$

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:

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

$$S \to aSa$$

$$S \to bSb$$

$$S \to \lambda$$

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

This language is context-free, but it is not regular

Example 5.2

A context-free grammar: G

$$S \to abB$$

$$A \to aaBb$$

$$B \to bbAa$$

$$A \to \lambda$$

A derivation:

$$S \Rightarrow abB \Rightarrow abbbAa \Rightarrow abbbaaBba$$

$$\Rightarrow abbbaabbAaba \Rightarrow abbbaabbaba$$

$$S \to abB$$

$$A \to aaBb$$

$$B \to bbAa$$

$$A \to \lambda$$

$$L(G) = \{ab(bbaa)^n bba(ba)^n : n \geq 0\}$$

The above grammars are not only context-free, but linear.
Regular and linear grammars are context-free,
But a context-free grammar is not necessarily linear.

Example 5.3

The language $L = \{a^n b^m : n \neq m\}$ is context-free

$$S \rightarrow aSb$$
$$S \rightarrow \lambda$$

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

非 linear

$$S \rightarrow AS_1$$
$$S_1 \rightarrow aS_1 b \mid \lambda$$
$$\boxed{A \rightarrow aA \mid a}$$

a 多

$$n > m$$

$$S \rightarrow AS_1 \mid \boxed{S_1 B}$$
$$S_1 \rightarrow aS_1 b \mid \lambda$$
$$A \rightarrow aA \mid a$$
$$\boxed{B \rightarrow bB \mid b}$$

b 多

$$n < m$$

The resulting grammar is context-free but not linear

Example 5.4

A context-free grammar: G

$$S \to aSb$$

$$S \to SS$$

$$S \to \lambda$$

Derivations:

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

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

A context-free grammar: G

$$S \to aSb$$

$$S \to SS$$

$$S \to \lambda$$

More derivations:

$$S \Rightarrow SS \Rightarrow aSbS \Rightarrow abS \Rightarrow abaSb$$
$$\Rightarrow abaaSbb \Rightarrow abaabb$$

$$S \Rightarrow aSb \Rightarrow aSSb \Rightarrow aaSbSb$$
$$\Rightarrow aabSb \Rightarrow aabaSbb \Rightarrow aababb$$

$$S \to aSb$$

$$S \to SS$$

$$S \to \lambda$$

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

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

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

⇒ 任何時點，左 ≥ 右括號

Describes
matched
parentheses:

( )   ( ) ( )   ( ) (( ))   (( ) ( ))

Derivation Order

In CFGs that are not linear, a derivation may involve sentential forms with more than one variables.

1. $S \rightarrow AB$

2. $A \rightarrow \underline{aa}A$

3. $A \rightarrow \lambda$

4. $B \rightarrow Bb$

5. $B \rightarrow \lambda$

$L(G) = \{a^{\underline{2}n}b^m: n \geq 0, m \geq 0\}$

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: 先展左邊

$$S \xRightarrow{1} \boxed{A}B \xRightarrow{2} aa\boxed{A}B \xRightarrow{3} aa\boxed{B} \xRightarrow{4} aa\boxed{B}b \xRightarrow{5} aab$$

Rightmost derivation:

$$S \xRightarrow{1} A\boxed{B} \xRightarrow{4} A\boxed{B}b \xRightarrow{5} \boxed{A}b \xRightarrow{2} aa\boxed{A}b \xRightarrow{3} aab$$

Example 5.5

$$1.\ S \to aAB \qquad 2.\ A \to bBb$$

$$3.\ B \to A \qquad 4.\ B \to \lambda$$

Leftmost derivation:

$$S \overset{1}{\Rightarrow} aAB \overset{2}{\Rightarrow} abBbB \overset{3}{\Rightarrow} abAbB \overset{2}{\Rightarrow} abbBbbB$$

$$\overset{4}{\Rightarrow} abbbbB \overset{4}{\Rightarrow} abbbb$$

Rightmost derivation:

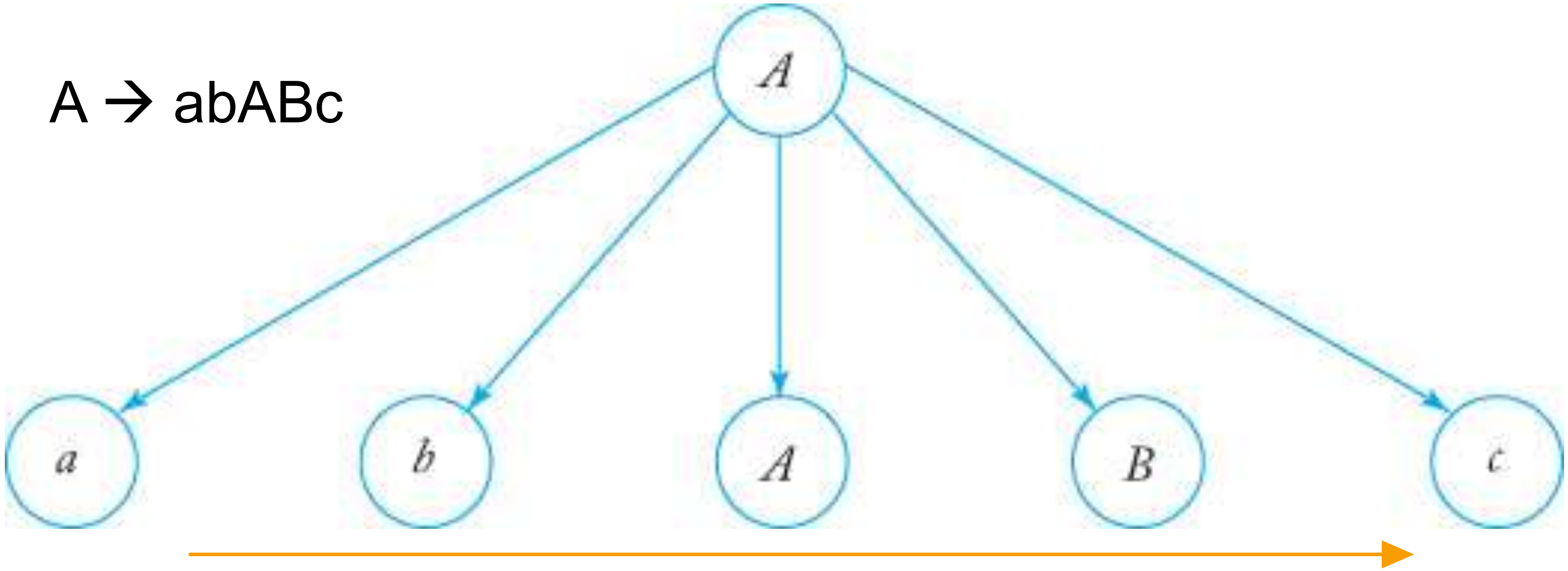
$$S \overset{1}{\Rightarrow} aAB \overset{4}{\Rightarrow} aA \overset{2}{\Rightarrow} abBb \overset{3}{\Rightarrow} abAb$$

$$\overset{2}{\Rightarrow} abbBbb \overset{4}{\Rightarrow} abbbb$$

Derivation (Parse) Trees

An ordered tree in which nodes are labeled with the left sides of productions and in which the children of a node represent its corresponding right sides.

A → abABc



Definition 5.3

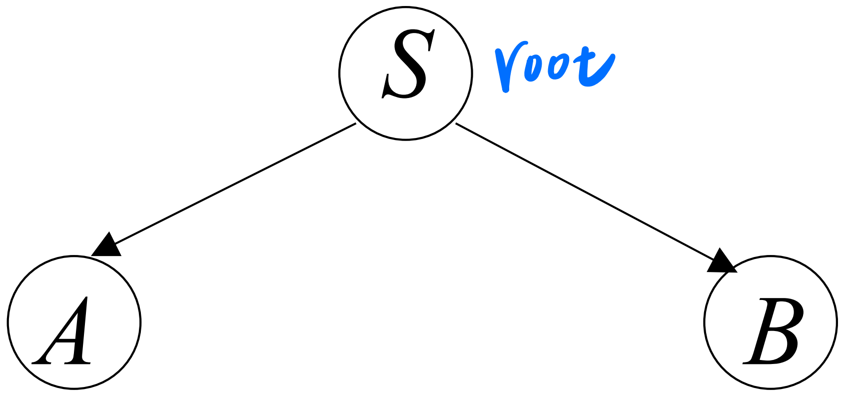
- Let G = (V, T, S, P) be a CFG. An ordered tree is a derivation tree for G iff it has the following properties.
    1. The root is labeled S
    2. Every leaf has a label from T U {λ}
    3. Every internal vertex has a label from V
    4. If a vertex has label A ϵ V, and its children are labeled $a_1, a_2, \ldots, a_n$, then P must contain a production

$$A \rightarrow a_1 a_2 \ldots a_n$$

    5. A leaf labeled λ has no sibling

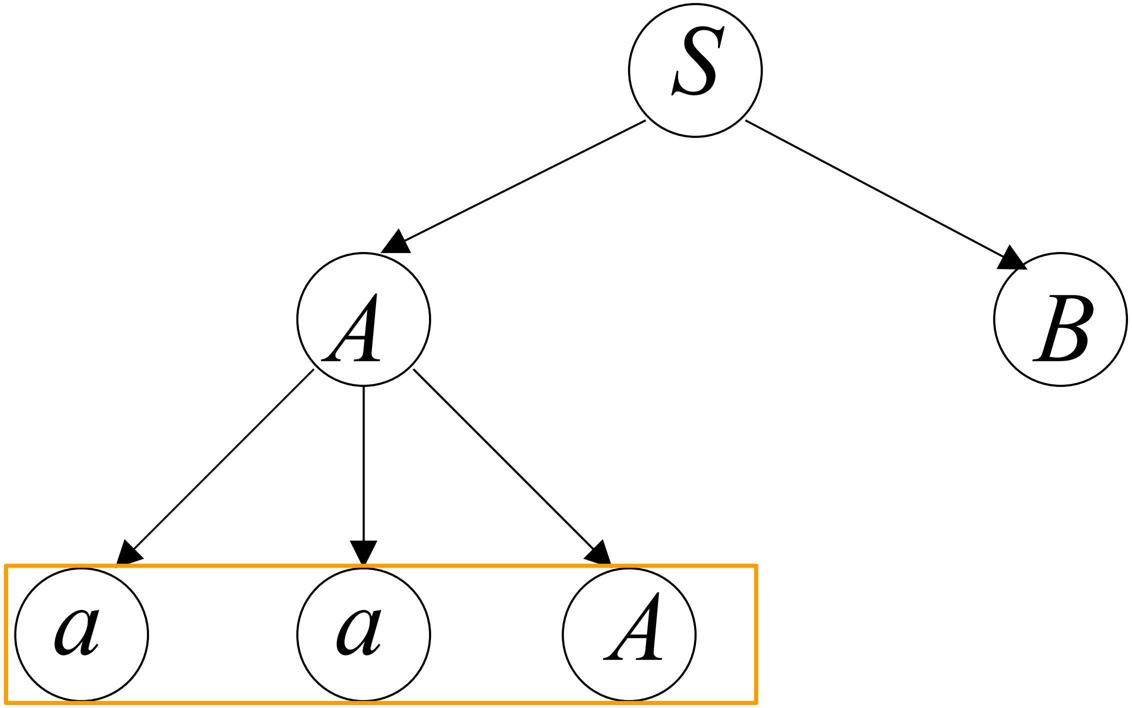
$$S \to AB \qquad A \to aaA \mid \lambda \qquad B \to Bb \mid \lambda$$

$$S \Rightarrow AB$$



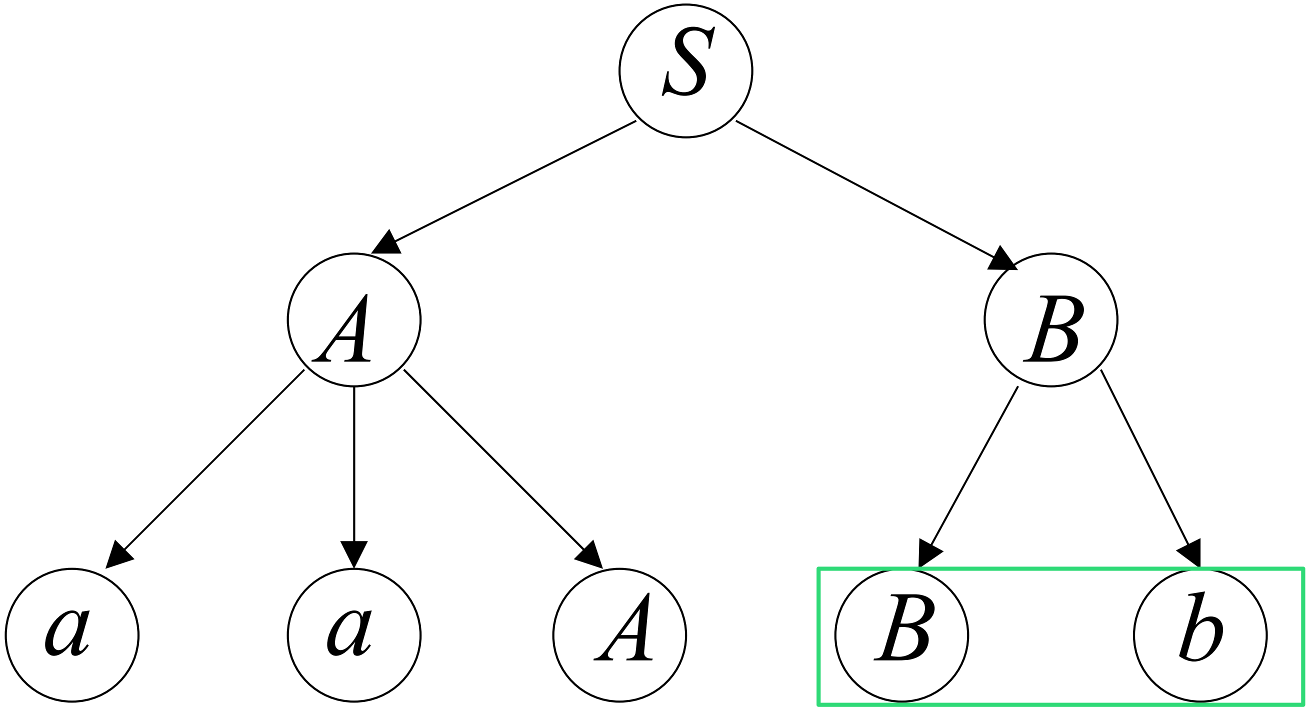
$$S \to AB \qquad A \to \boxed{aaA} \mid \lambda \qquad B \to Bb \mid \lambda$$

$$S \Rightarrow AB \Rightarrow aaAB$$



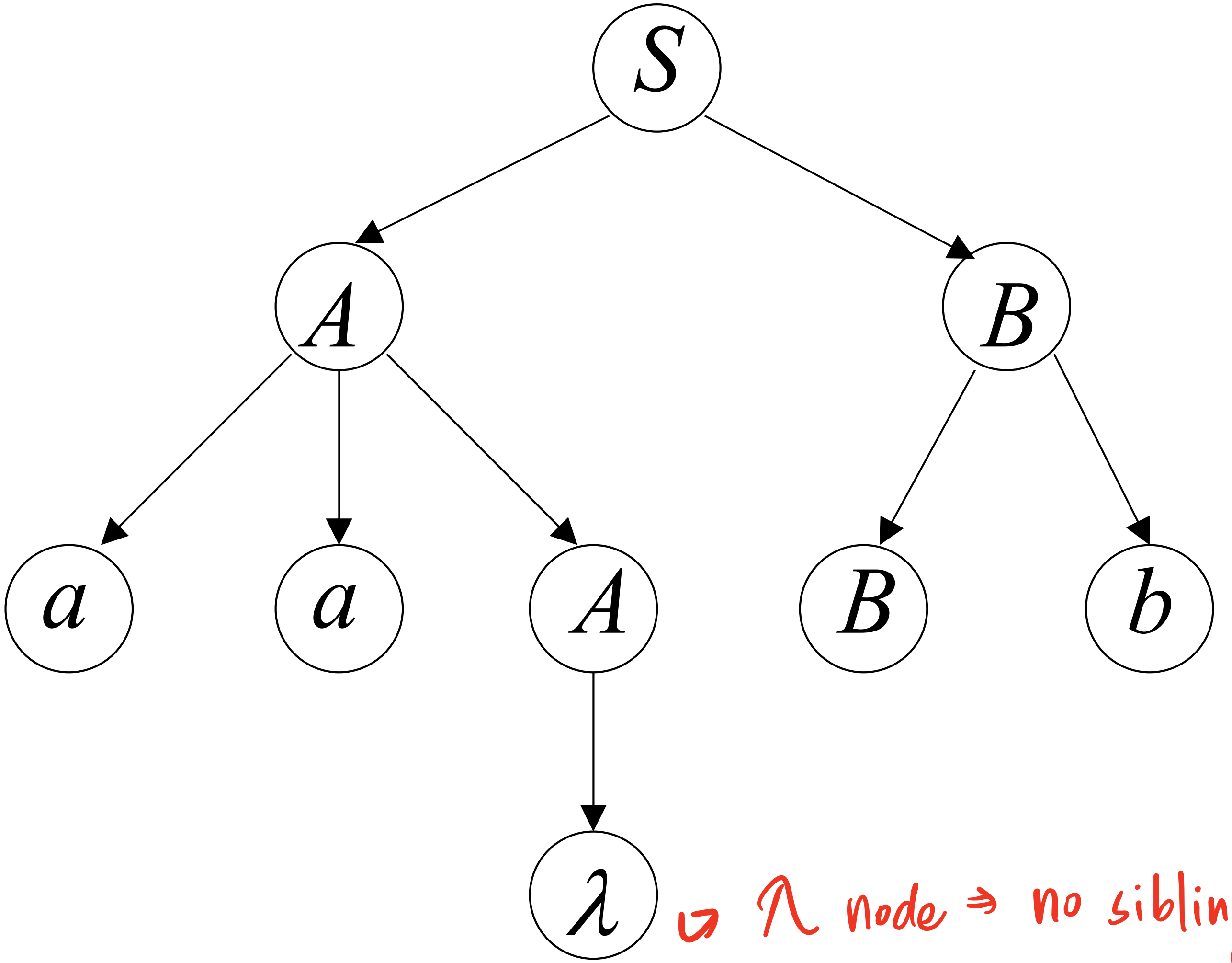
$S \to AB \qquad A \to aaA \mid \lambda \qquad B \to \boxed{Bb} \mid \lambda$

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



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

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

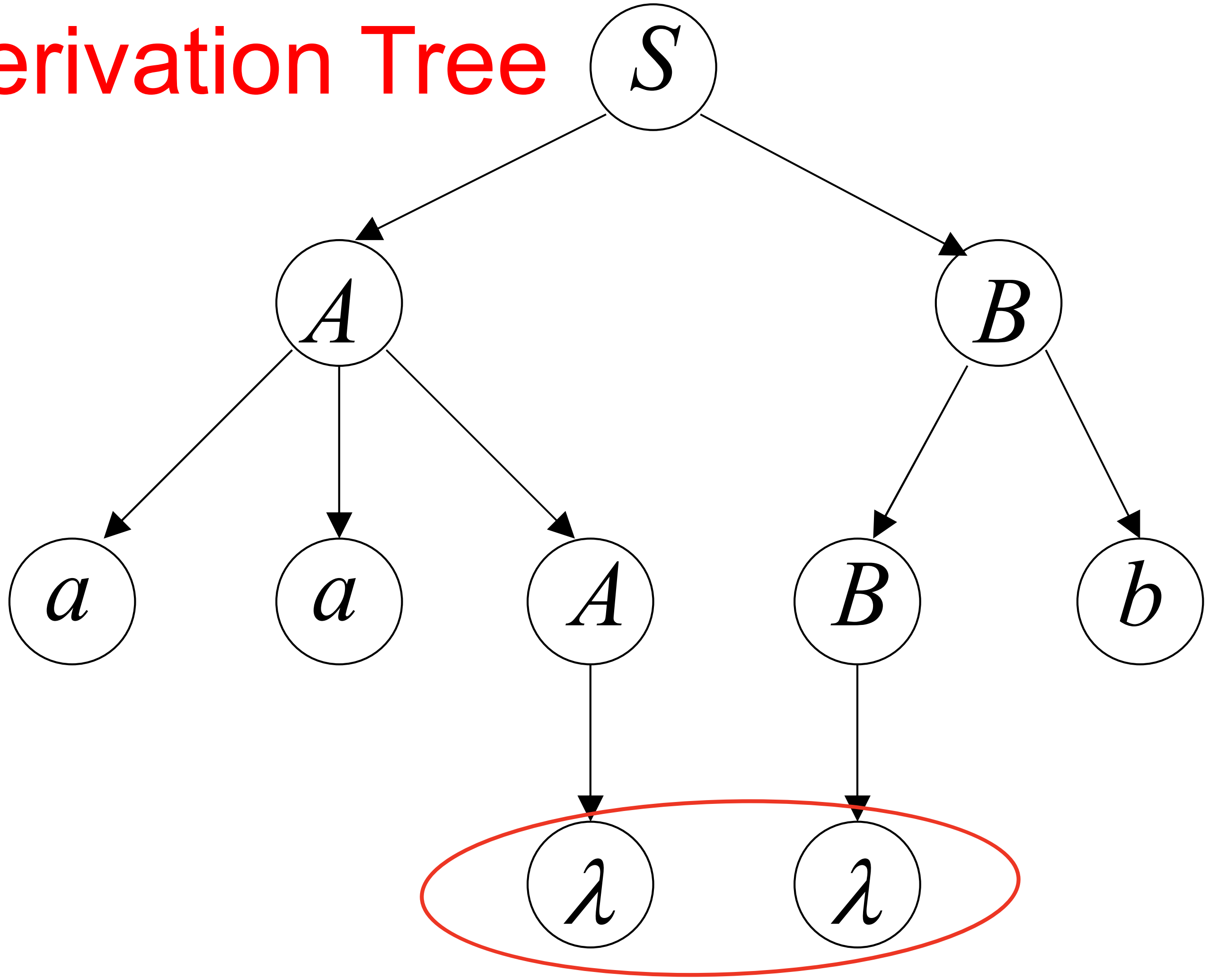


↳ λ node ⇒ no siblings

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

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

Derivation Tree

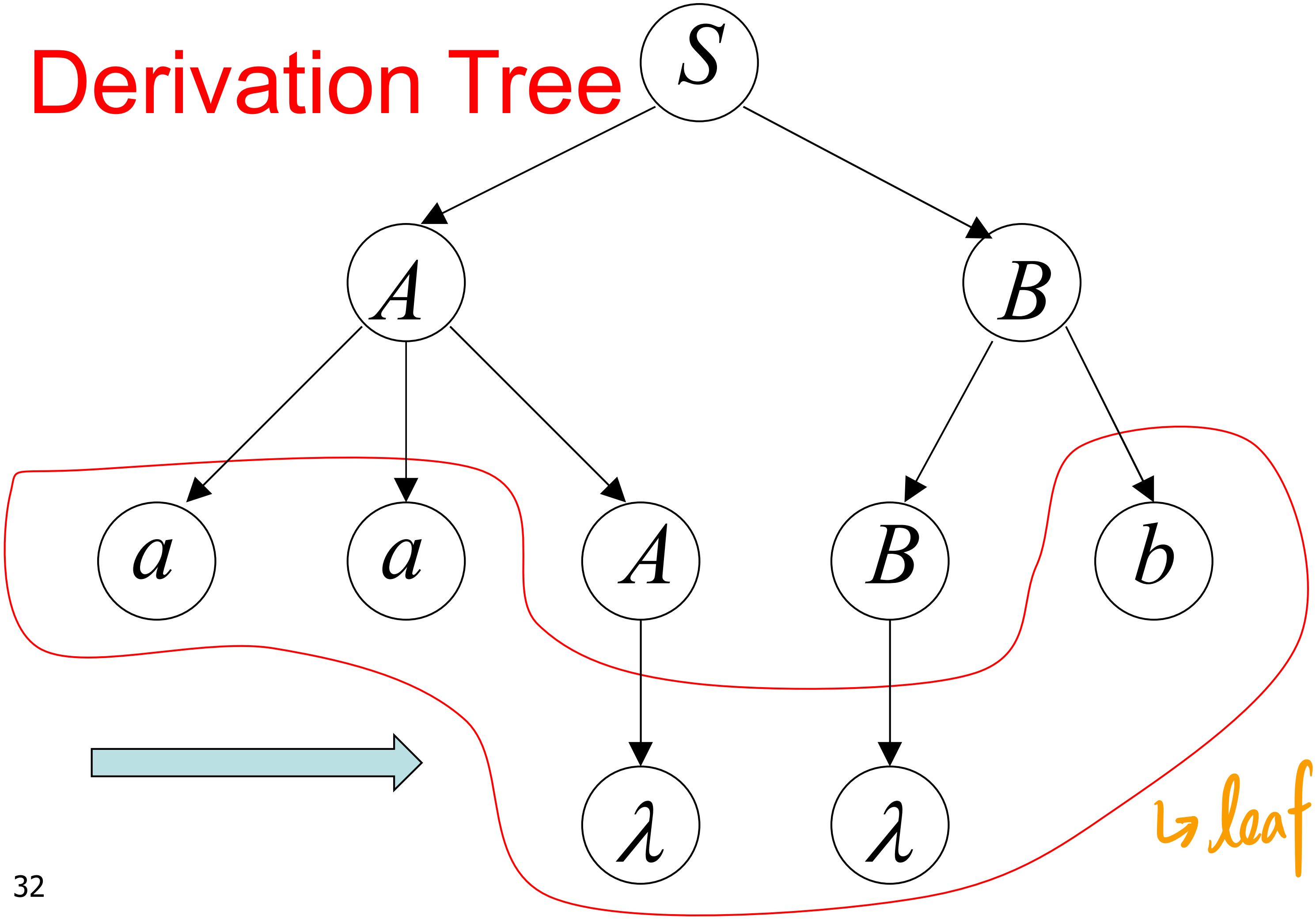


Depth-first search manner from the leftmost unexplored branch

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

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

Derivation Tree



Yield

$aa\lambda\lambda b$

$= aab$

↳ leaf 只能為 terminal or λ

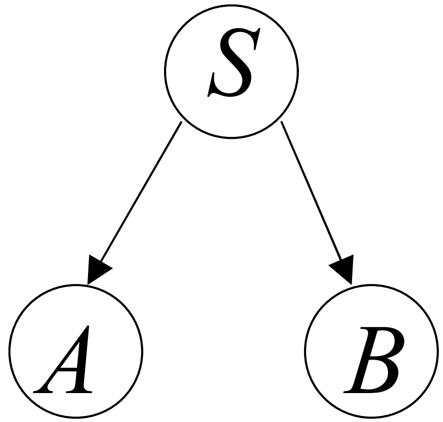
Partial Derivation Trees

$$S \to AB$$

$$A \to aaA \mid \lambda$$

$$B \to Bb \mid \lambda$$

$$S \Rightarrow AB$$

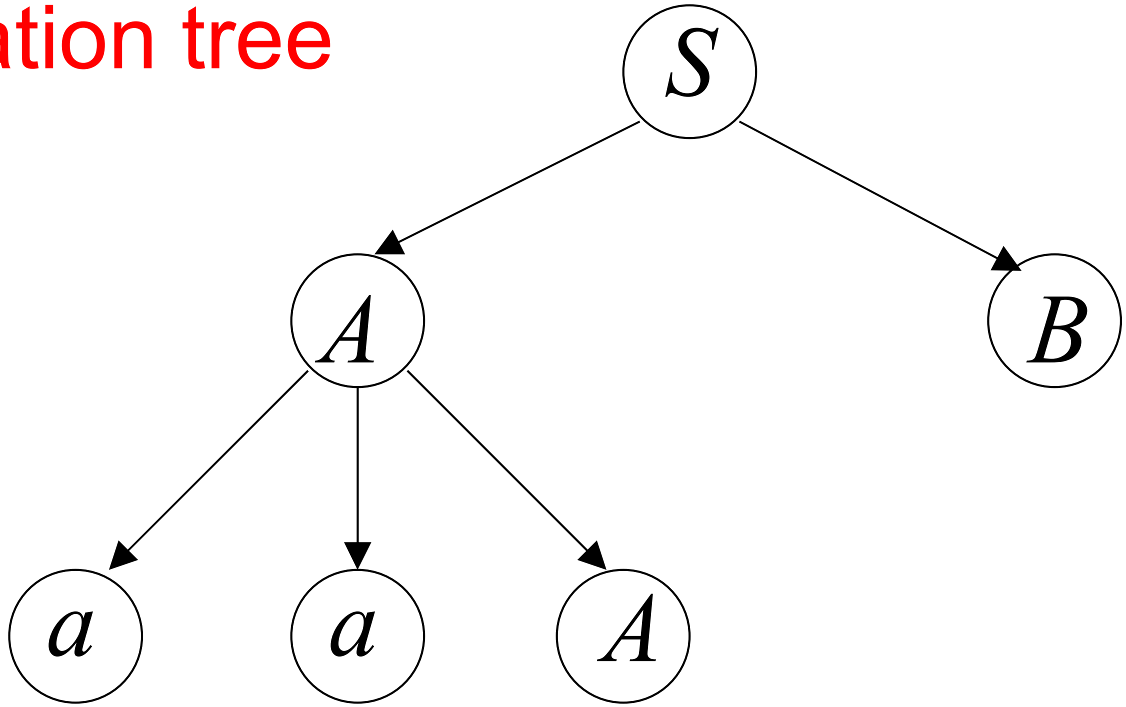


Partial derivation tree

- A tree that has properties 3, 4, and 5.
- 1 does not necessarily hold.
- 2 is replaced by:
  - Every leaf has a label from V U T U {λ}

$$S \Rightarrow AB \Rightarrow aaAB$$

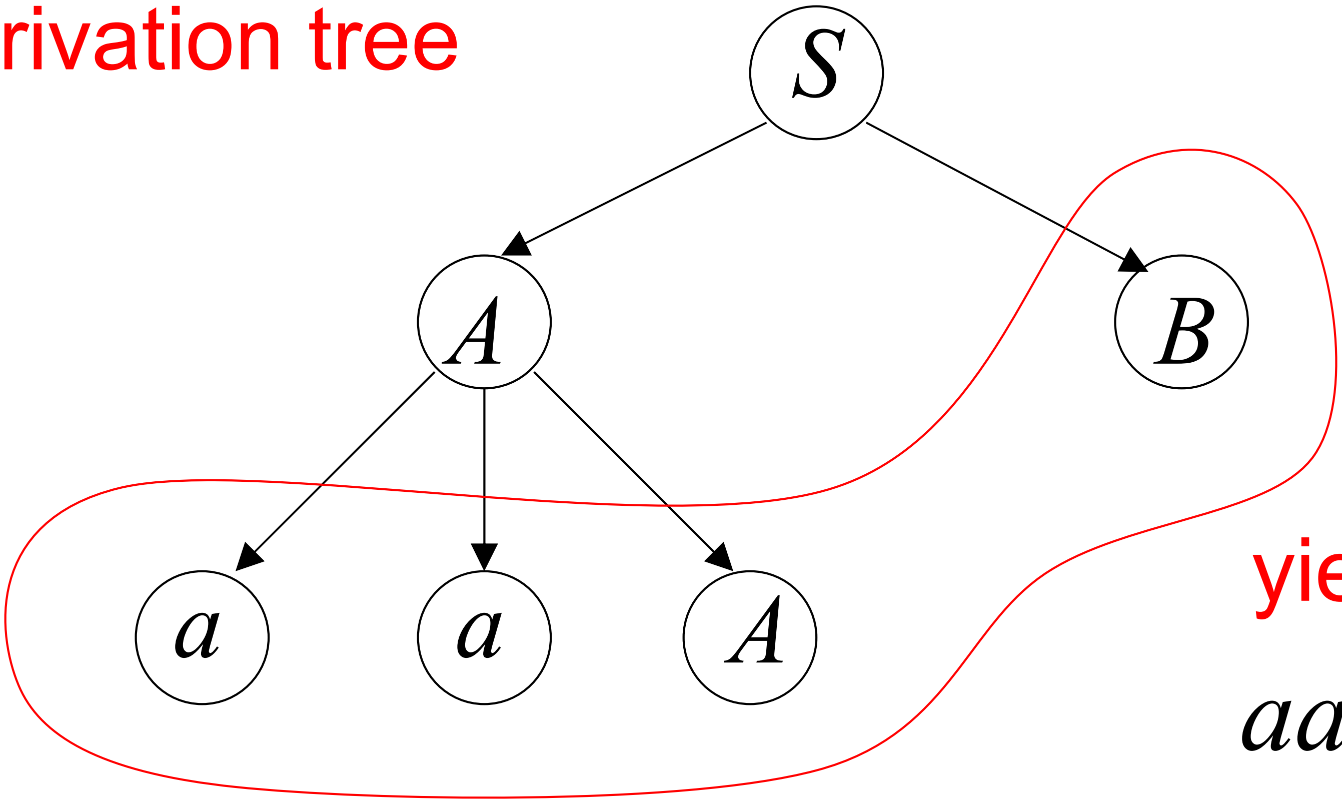
Partial derivation tree



$$S \Rightarrow AB \Rightarrow aaAB$$ sentential form

↳中間階段

Partial derivation tree



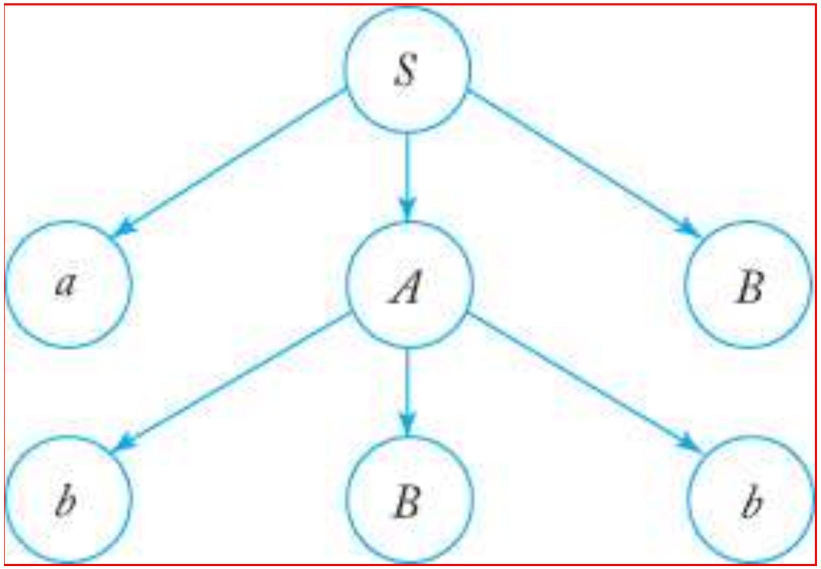
$aaAB$

Example 5.6

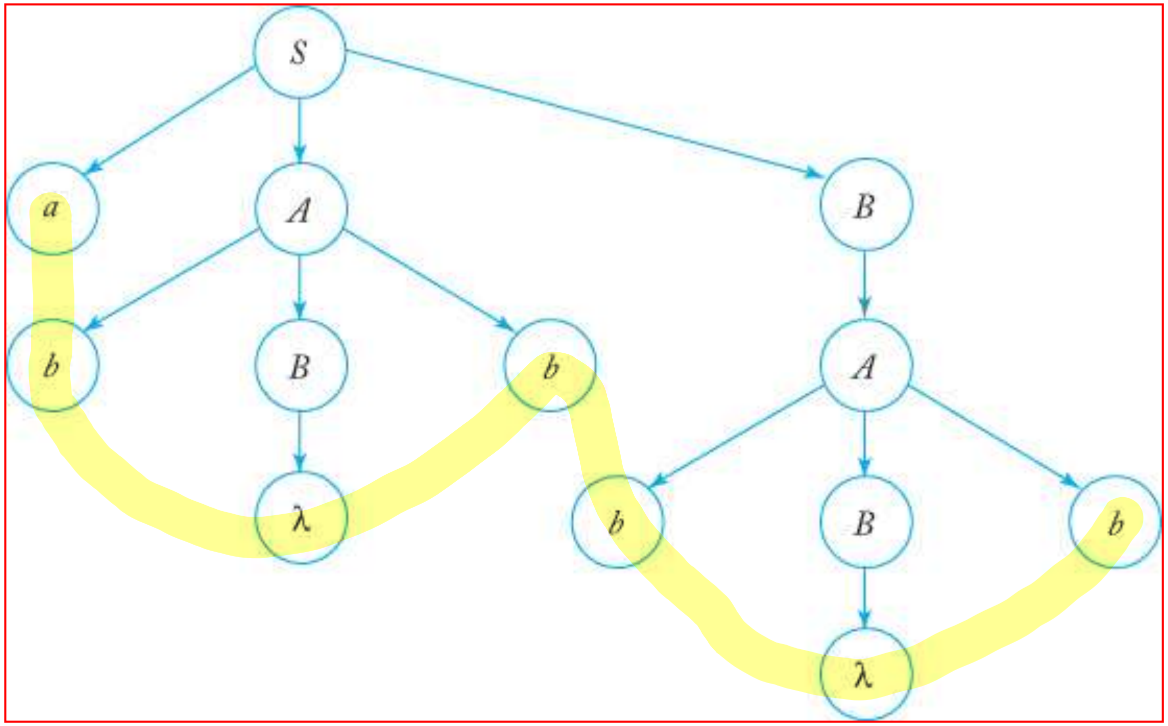
$$S \rightarrow aAB \qquad A \rightarrow bBb \qquad B \rightarrow A \mid \lambda$$

Yield: abBbB is a sentential form of G

abbbb ϵ L(G)



Partial derivation tree



Derivation tree

Theorem 5.1

- Let $G = (V, T, S, P)$ be a CFG. Then for every $w \in L(G)$, there exists a derivation tree of G whose yield is w. Conversely, the yield of any derivation tree is in $L(G)$.
- Also, if t_G is any partial derivation tree for G whose root is labeled S, then the yield of t_G is a sentential form of G.

Sometimes, derivation order doesn't matter

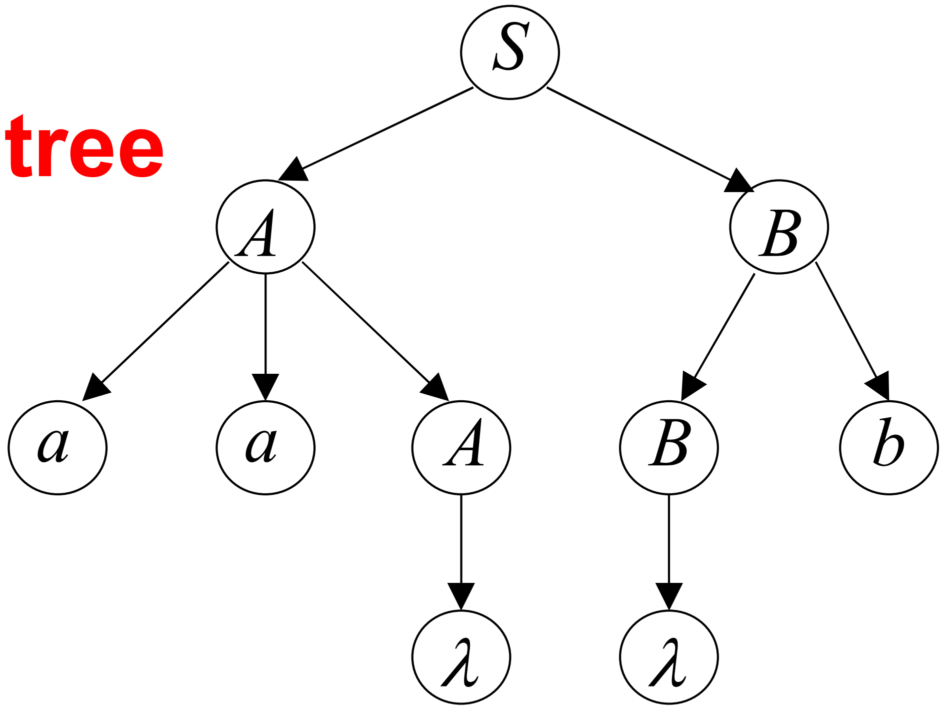
Leftmost:

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

Rightmost:

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

Same derivation tree



Outline

1 Context-Free Grammars

2 Parsing and Ambiguity

3 Context-Free Grammars and Programming Languages

Parsing

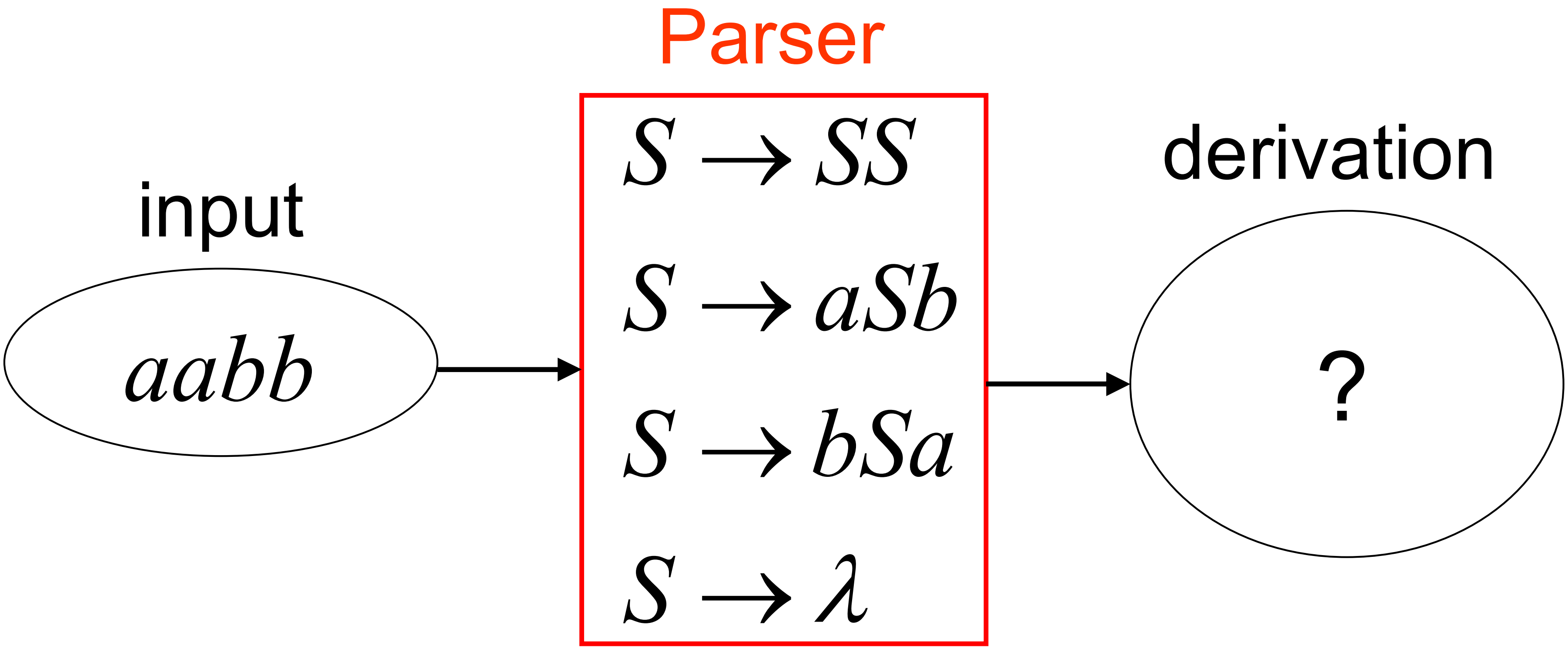
Given a grammar G, we studied the set of strings that can be derived using G, but…

Given a string w of terminals, we want to know whether or not w is in L(G) (membership question) ↳ 判斷string是否 ∈ L(G)

◆Parsing describes finding a sequence of productions by which a w ϵ L(G) is derived.


Parser
input
string
grammar
derivation

Example:



Exhaustive Search Parsing
(Brute Force Parsing)

暴力解 ⇒ 用grammar推

Example 5.7

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

Find derivation of

$$aabb$$

Phase 1:

$$S \Rightarrow SS$$

$$S \Rightarrow aSb$$

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

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

不可能生成 aabb

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

$$S \Rightarrow SS \Rightarrow SSS$$

$$S \Rightarrow SS \Rightarrow aSbS$$

$aabb$

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

Phase 1

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

$$S \Rightarrow aSb$$

$$S \Rightarrow aSb \Rightarrow aSSb$$

$$S \Rightarrow aSb \Rightarrow aaSbb$$

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

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

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

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)

Parser

$$S \to SS$$

$$S \to aSb$$

$$S \to bSa$$

$$S \to \lambda$$

input

$aabb$

derivation

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

Flaws of Exhaustive Search Parsing

- Tediousness (bad for efficiency)
- It is possible that it never terminates for strings not in L(G) ↳ 進入無窮迴圈

w=abb?

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

$$A \to \lambda$$

$$A \to B$$

↳ Variable 取代 variable

↳ 不利於用暴力法

Example 5.8

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

by $S \to SS \mid aSb \mid bSa \mid ab \mid ba$

If so, given any w ϵ $\{a,b\}^+$,
the exhaustive search parsing will always terminate in no more than |w| rounds.
上限

It is trivial because the length of the sentential form grows by at least one symbol in each round

Theorem 5.2

Suppose a CFG does not have any rules of the form

$$A \to \lambda$$

$$A \to B$$

Then the exhaustive search parsing can be made into an algorithm which, for any $w \in \Sigma^*$, either produces a parsing of w or tells us that no parsing is possible

Theorem 5.2

Suppose a CFG does not have any rules of the form

$$A \to \lambda$$

$$A \to B$$

$\because$ Neither the length of a sentential form nor the number of terminals can exceed |w|

$\therefore$ Number of phases for string w can not more than: 2|w|

$$\text{Ex: } S \to SS \mid aSb \mid bSa \mid ab \mid ba$$

For grammar with *P* rules

Phase 1:

we have no more than |P| sentential forms

Phase 2:

we have no more than $|P|^2$ sentential forms

⋮

Phase 2|w|:

we have no more than $|P|^{2|w|}$ sentential forms

ex: $\underbrace{SSSS}_{4層} \longrightarrow \underbrace{aaaa}_{4層}$

Total time needed for string : w

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

phase 1 phase 2 phase 2|w|

$O(P^{2|w|+1})$

Extremely bad!!!

For general context-free grammars:

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

The construction of more efficient parsing methods for CFGs is a complicated matter that belongs to a course on compilers

There exist faster algorithms for specialized grammars

Simple grammar (s-grammar):

$$A \to ax$$

terminal a; string of variables x

Pair (A, a) appears at most once in P

Example 5.9

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

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

不能有2個a開頭的

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: $|w|$

Linear time!!!

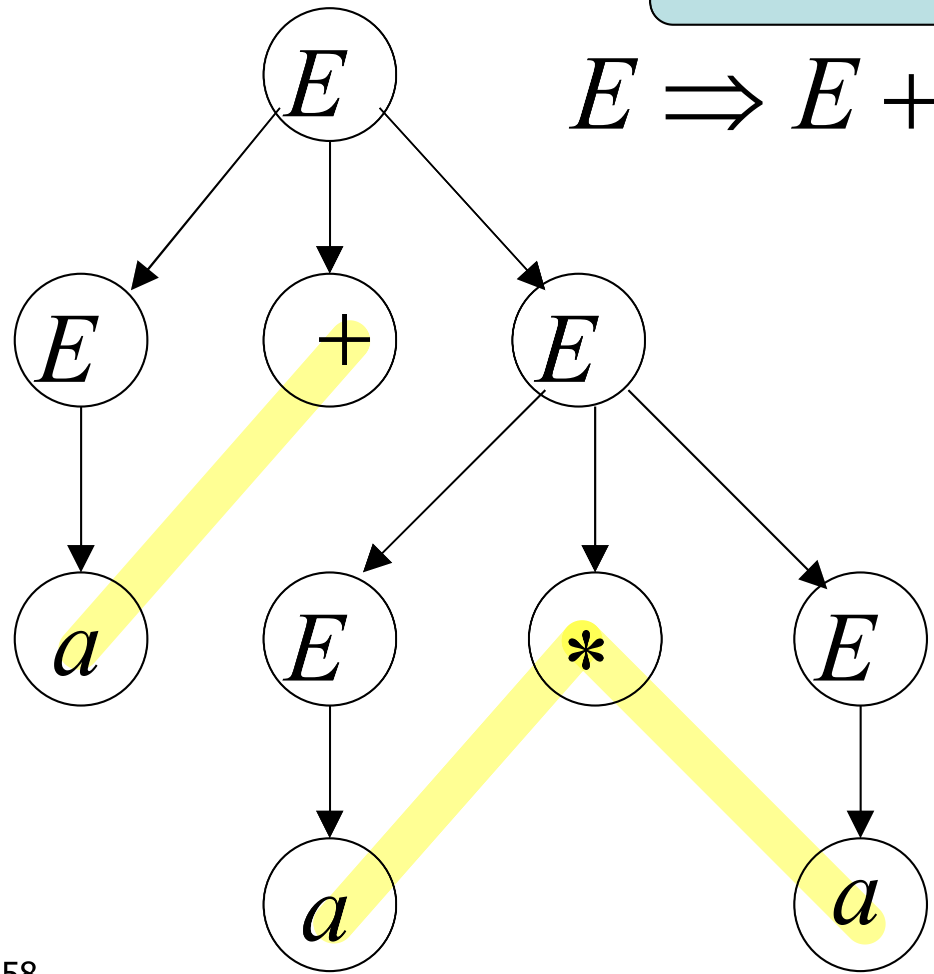
<while_stmt> ::= while <expr><stmt>

Ambiguity

Example 5.11

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

$$\boxed{a + a * a}$$



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

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

leftmost derivation

Example 5.11

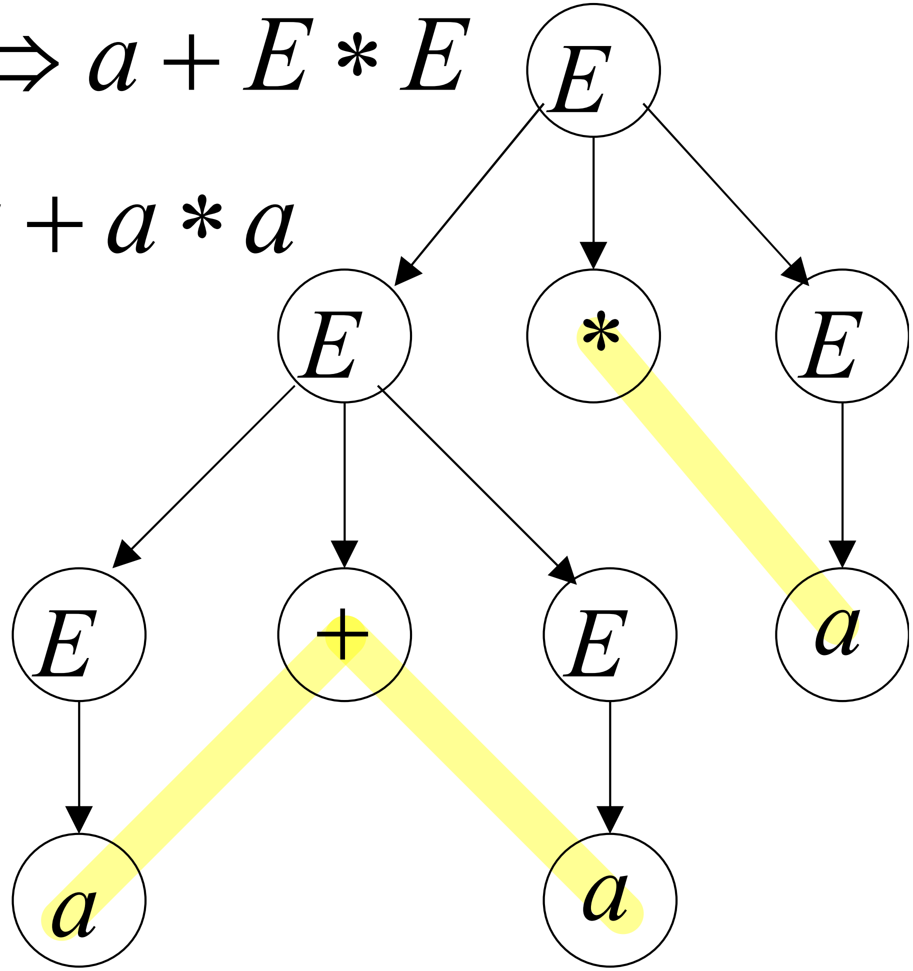
$$E \to E + E \ \mid E * E \mid (E) \mid a$$

$$\boxed{a + a * a}$$

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

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

leftmost derivation

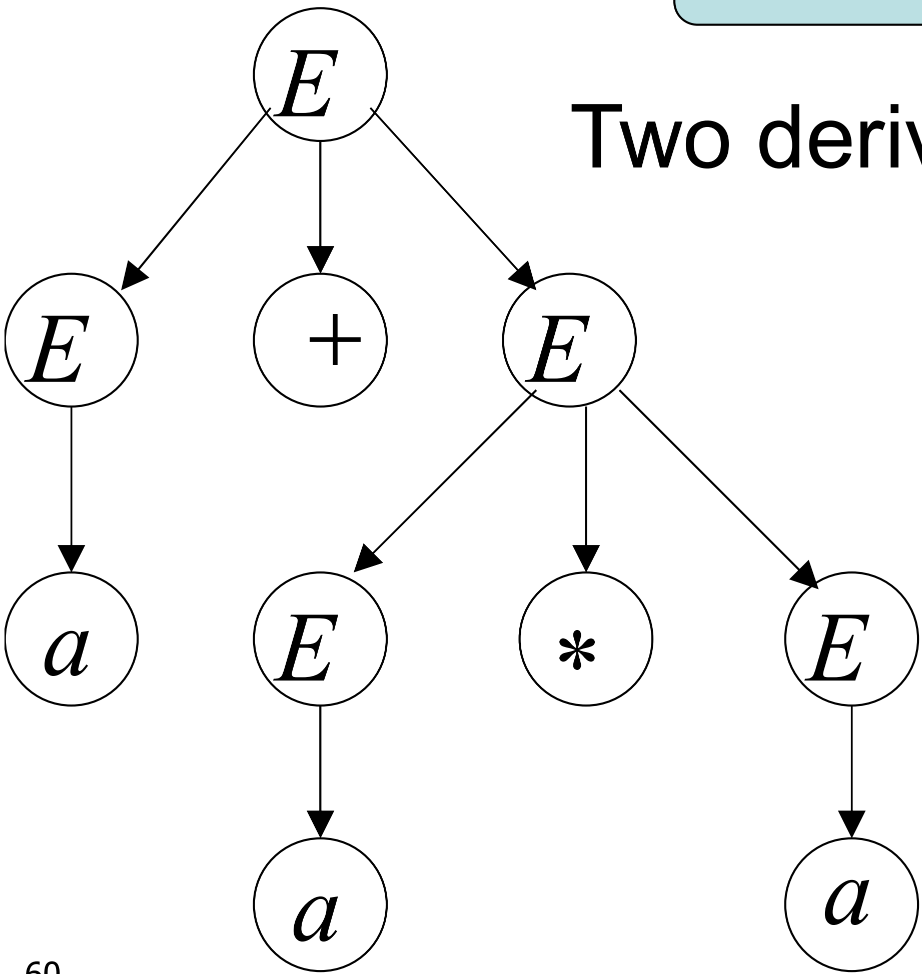


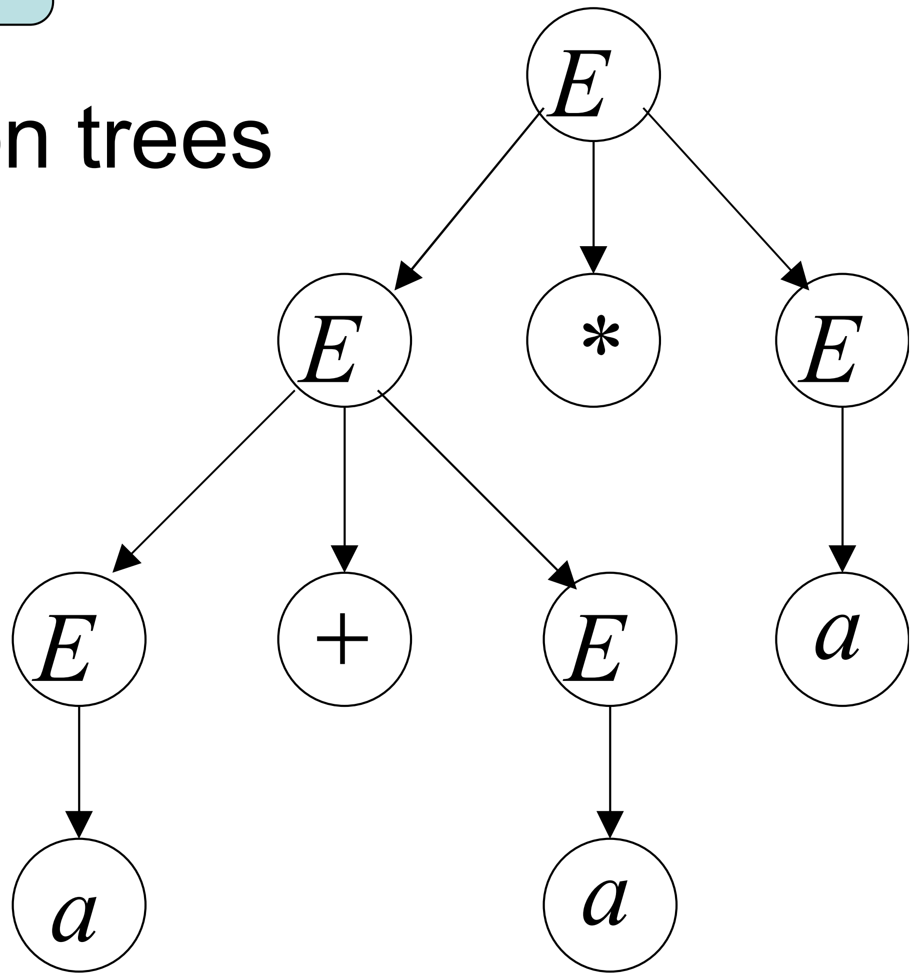
Example 5.11

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

$$a + a * a$$

Two derivation trees



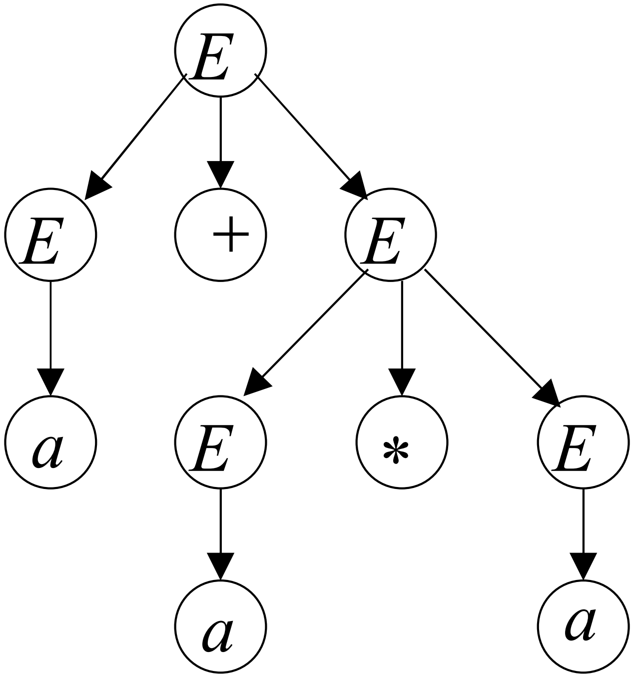


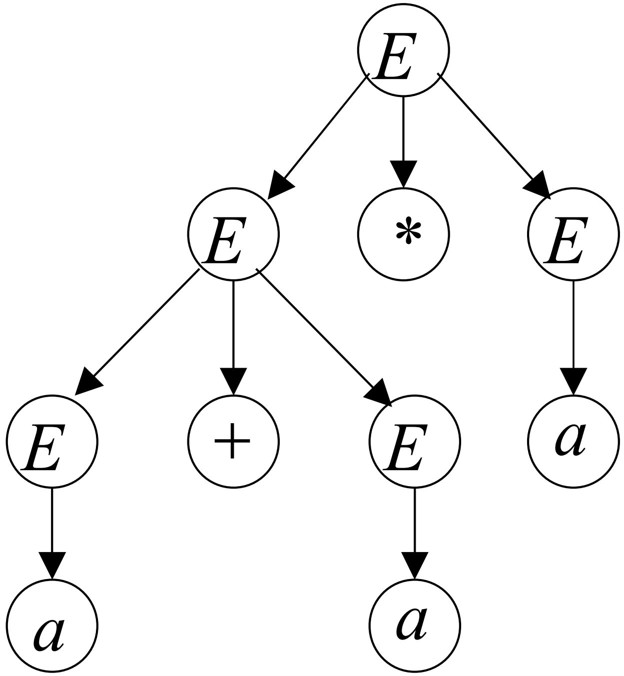
The grammar $E \to E + E \;\; | \;\; E * E \;|\; (E) \;|\; a$

is ambiguous:

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

↳ ambiguous





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 5.5:

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:

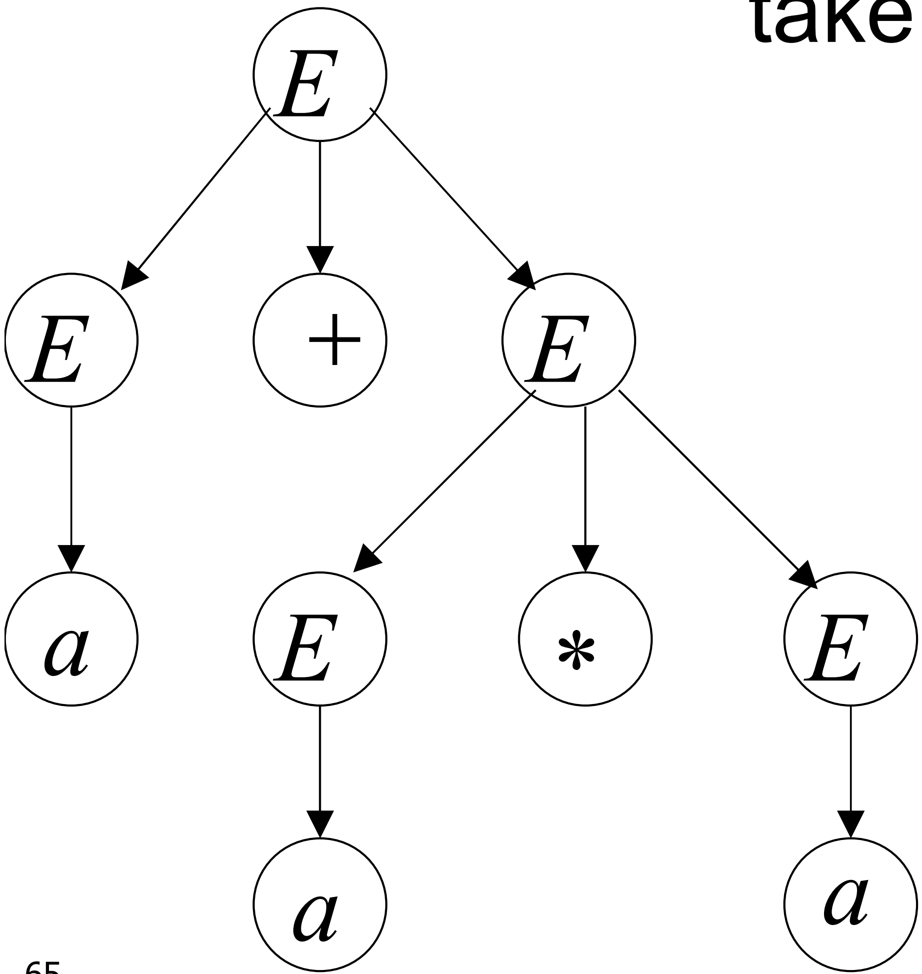
two or more leftmost derivations

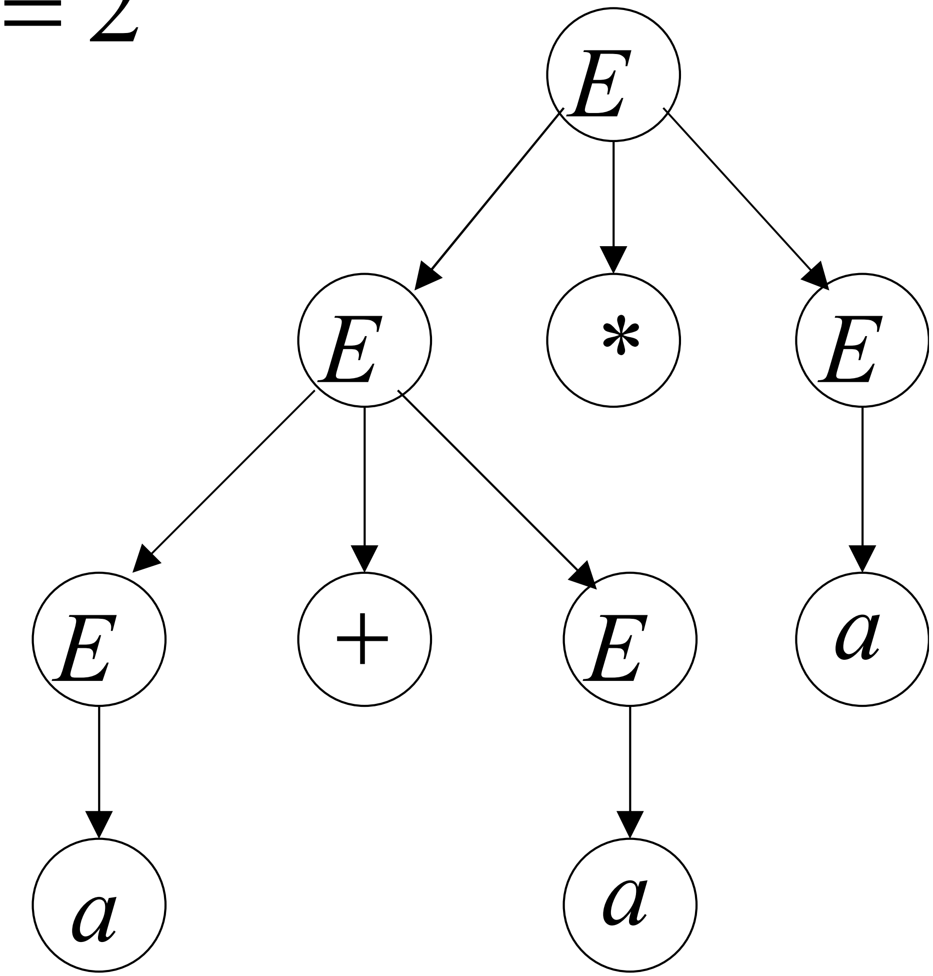
(or rightmost)

Why do we care about ambiguity?

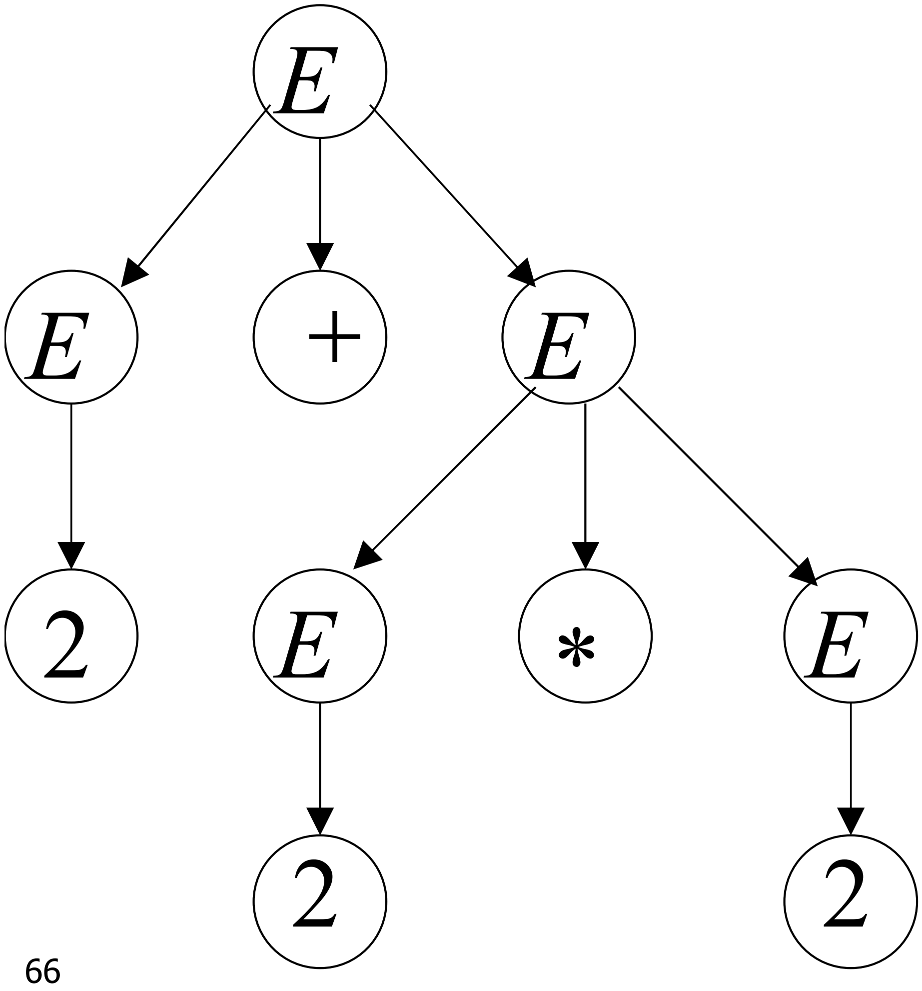
$$a + a * a$$

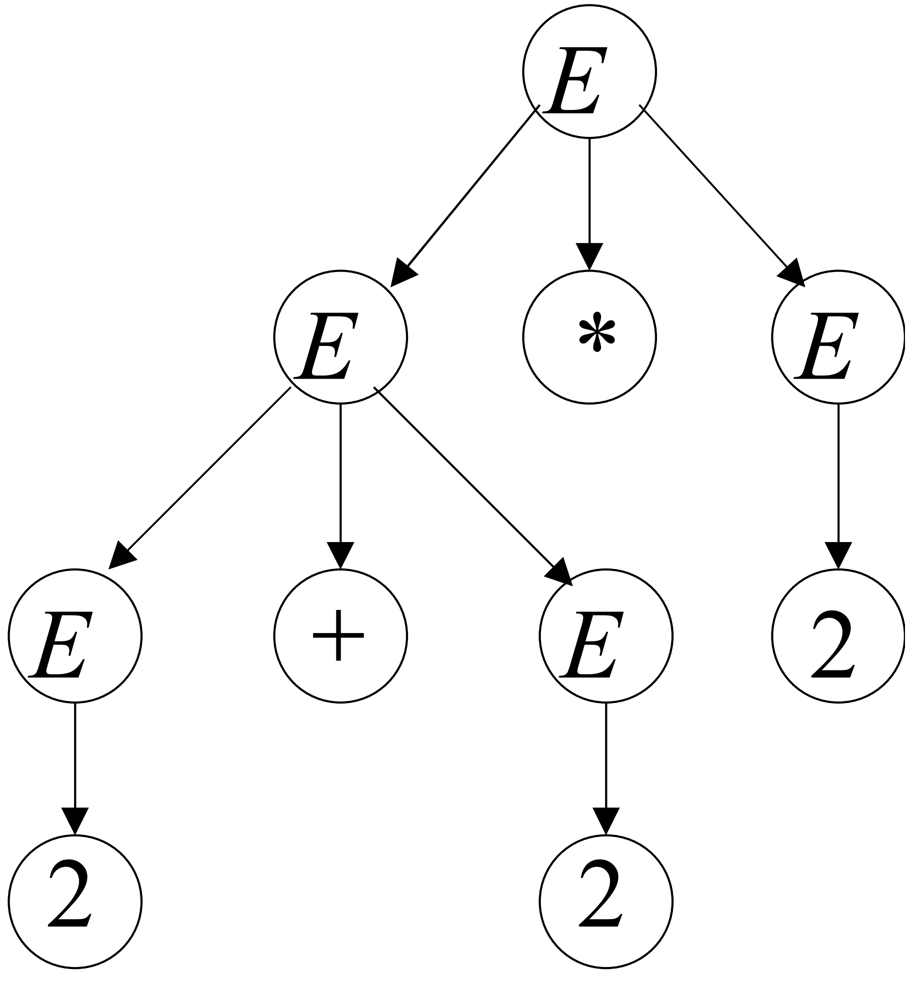
take $a = 2$





$$2 + 2 * 2$$

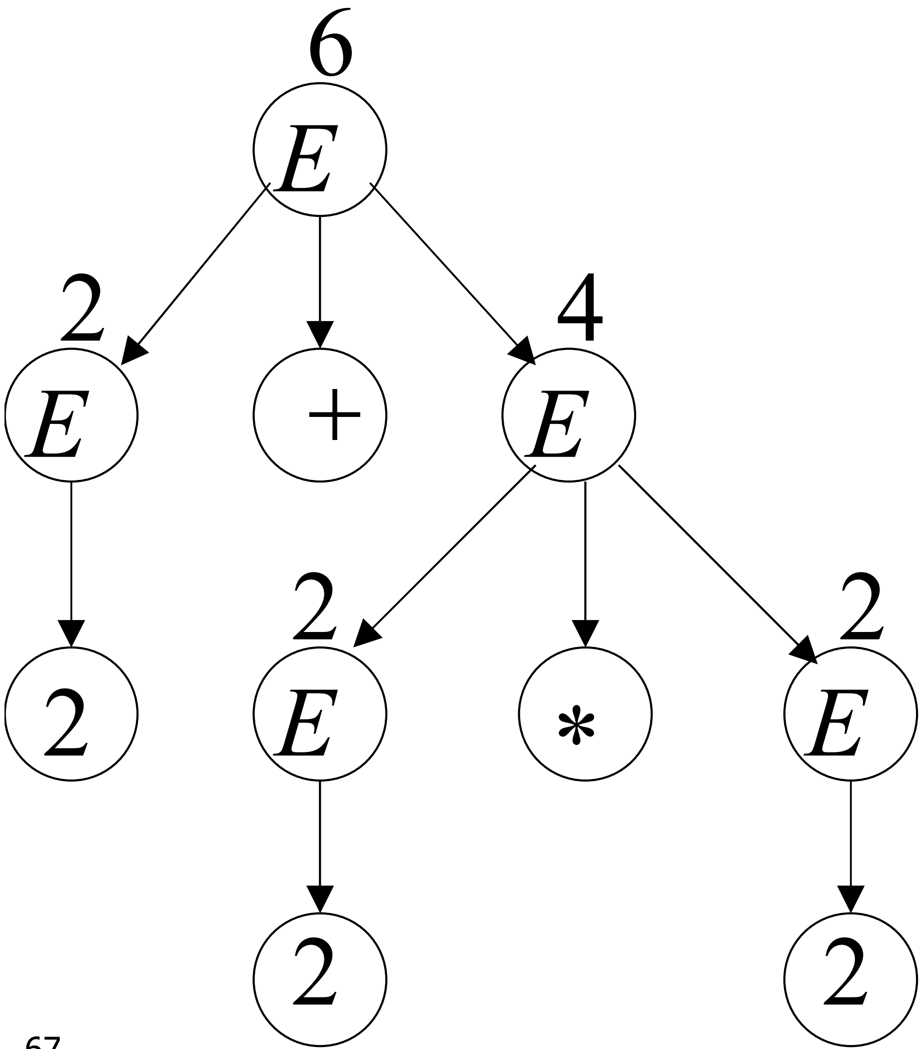


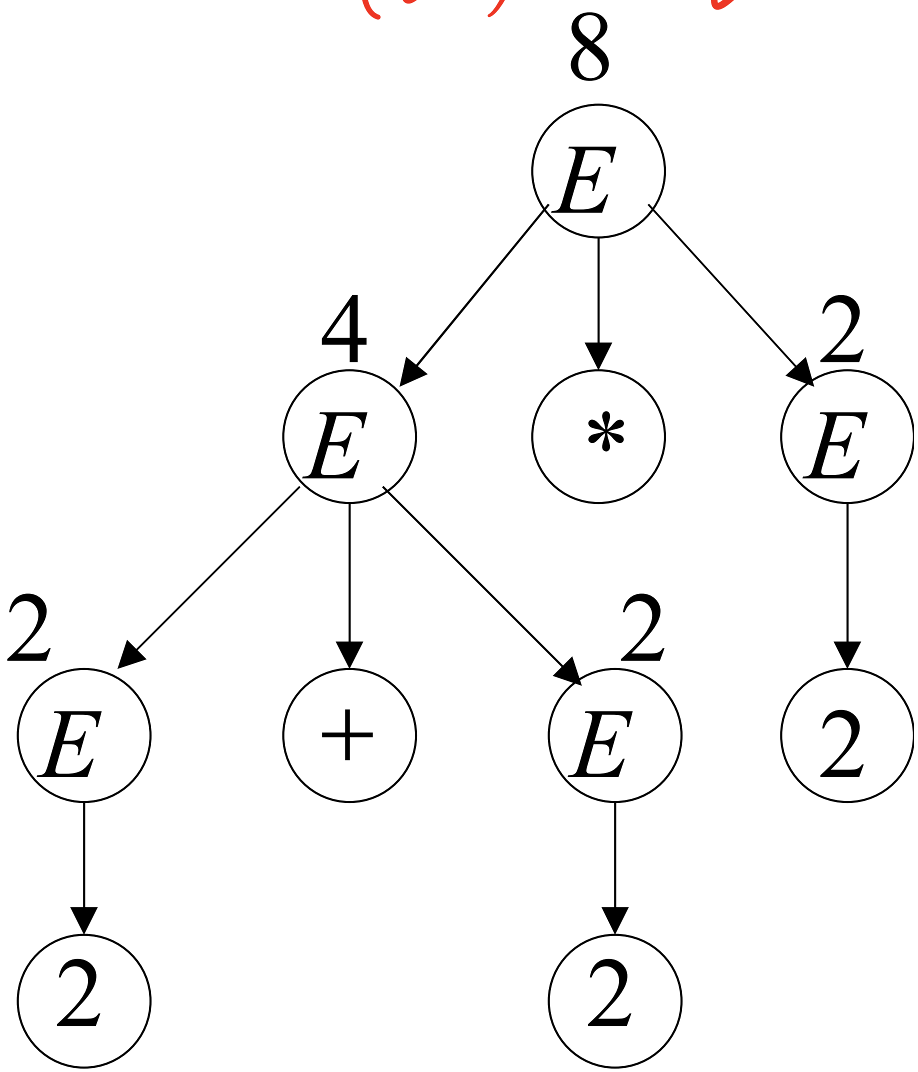


$$2 + 2 * 2 = 6$$

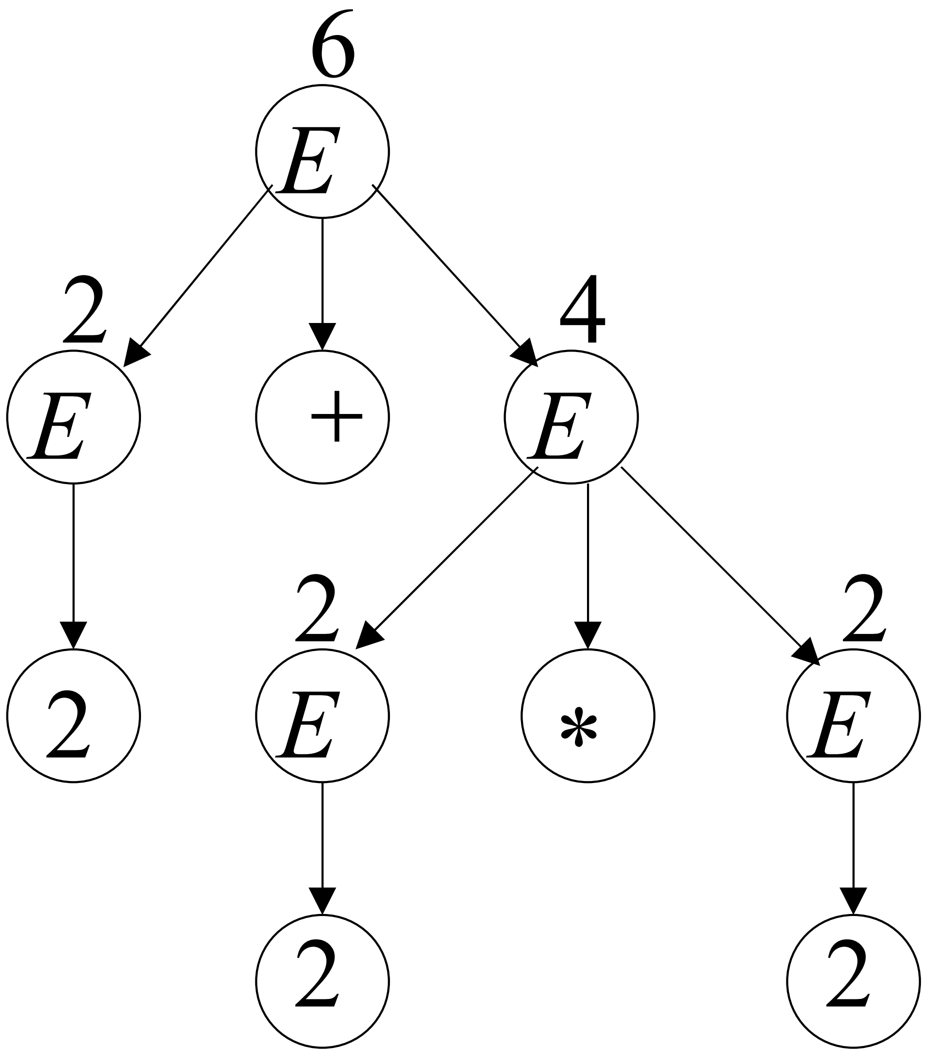
$$2 + 2 * 2 = 8$$

$\hookrightarrow (2+2) \times 2 = 8$





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



- Ambiguity is **bad** for programming languages

- We want to remove ambiguity

We fix the ambiguous grammar:

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

New non-ambiguous grammar:

{E, T, F} ϵ V

定優先權

priority

$$E \rightarrow E + T$$ 後做

$$E \rightarrow T$$

$$T \rightarrow T * F$$

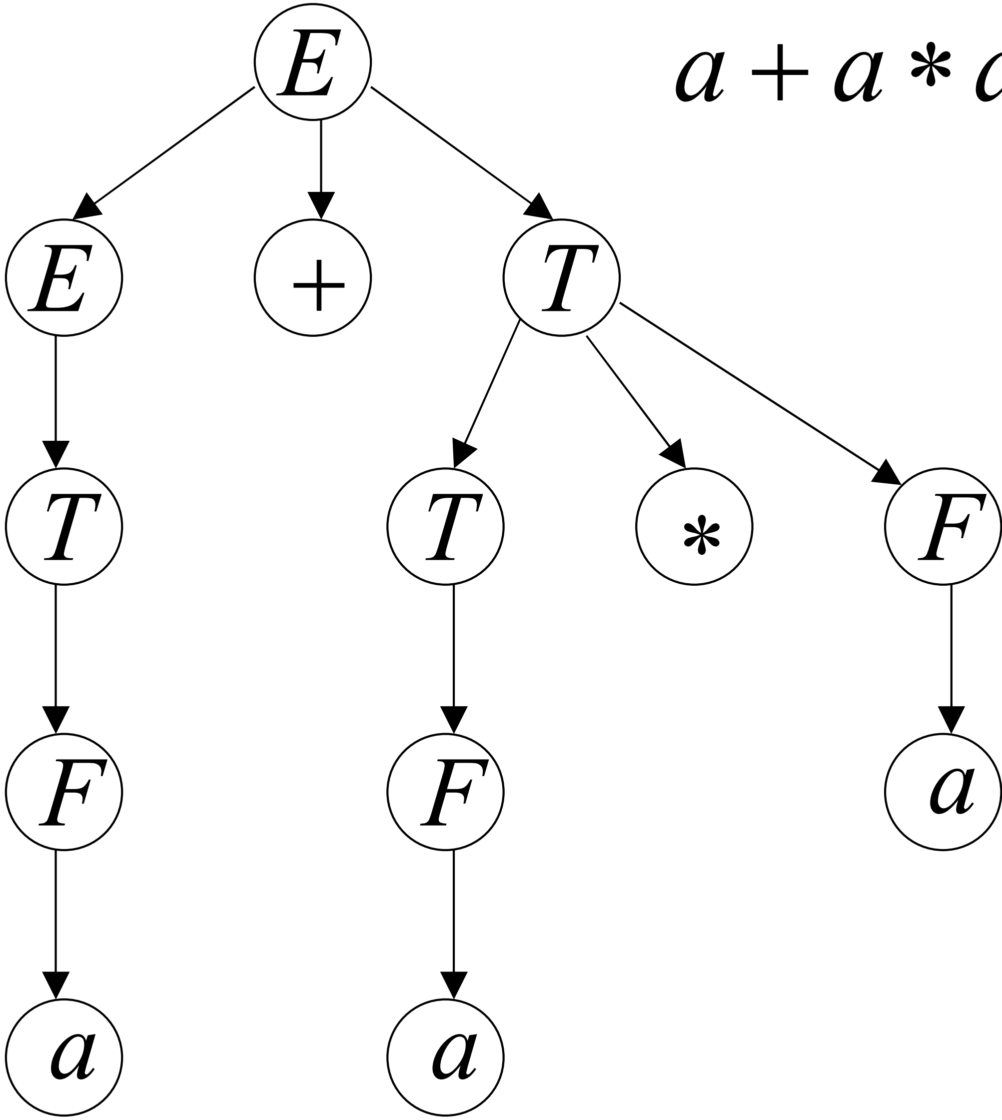
$$T \rightarrow F$$

$$F \rightarrow (E)$$

$$F \rightarrow a$$ 先做

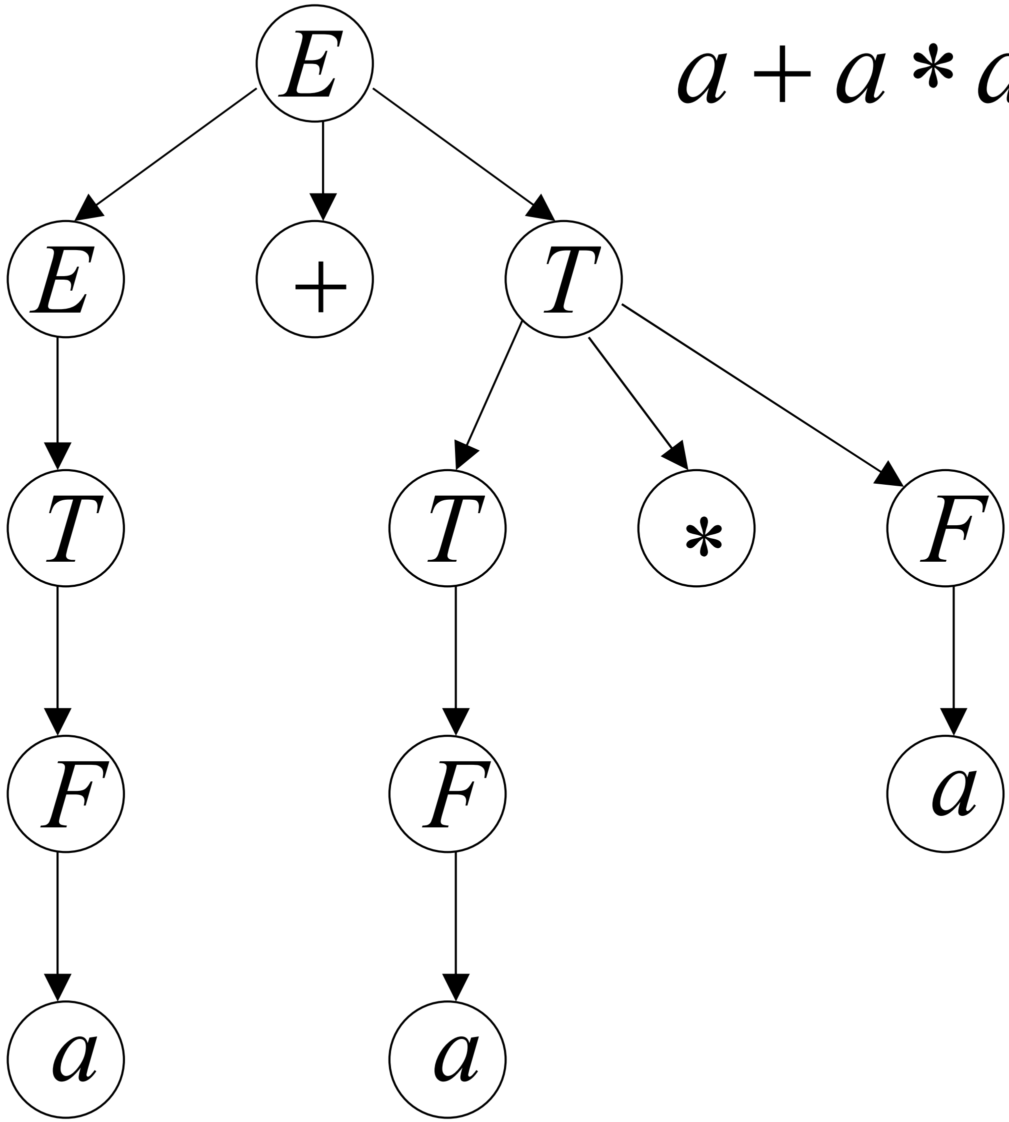
$$E \Rightarrow E + T \Rightarrow T + T \Rightarrow F + T \Rightarrow a + T \Rightarrow a + T * F$$
$$\Rightarrow a + F * F \Rightarrow a + a * F \Rightarrow a + a * a$$

$$a + a * a$$



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

Unique derivation tree



$$a + a * a$$

The grammar: G

$$E \rightarrow E + T$$

$$E \rightarrow T$$

$$T \rightarrow T * F$$

$$T \rightarrow F$$

$$F \rightarrow (E)$$

$$F \rightarrow a$$

is unambiguous: ※只要能畫出2種tree ⇒ ambiguous

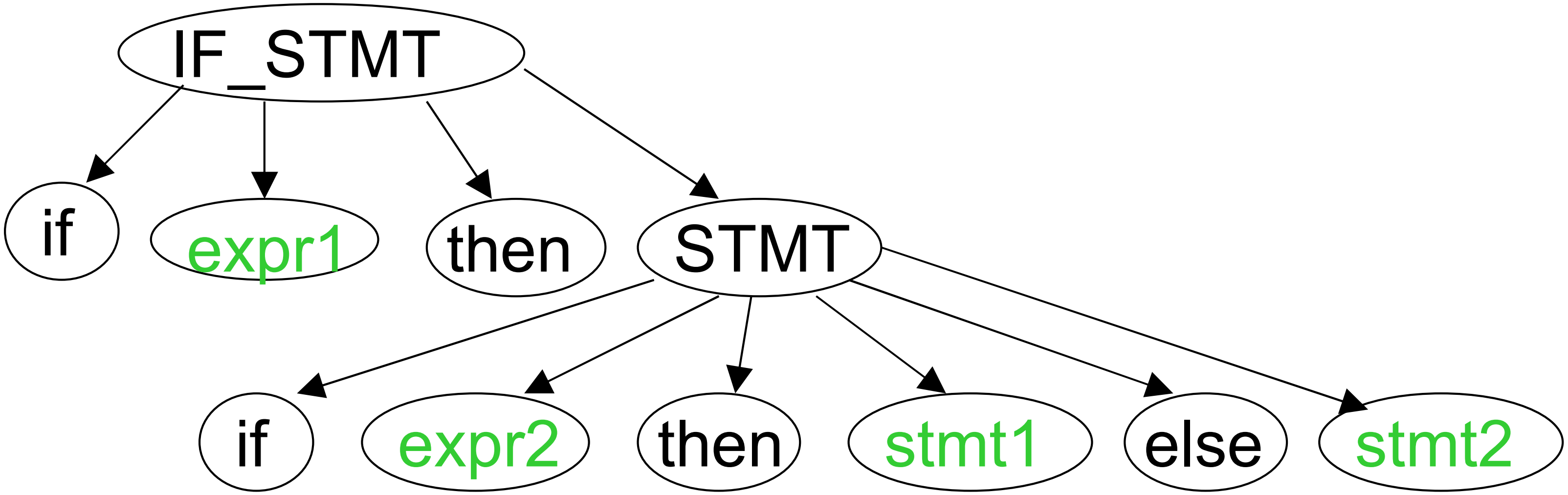
Every string $w \in L(G)$ has a unique derivation tree

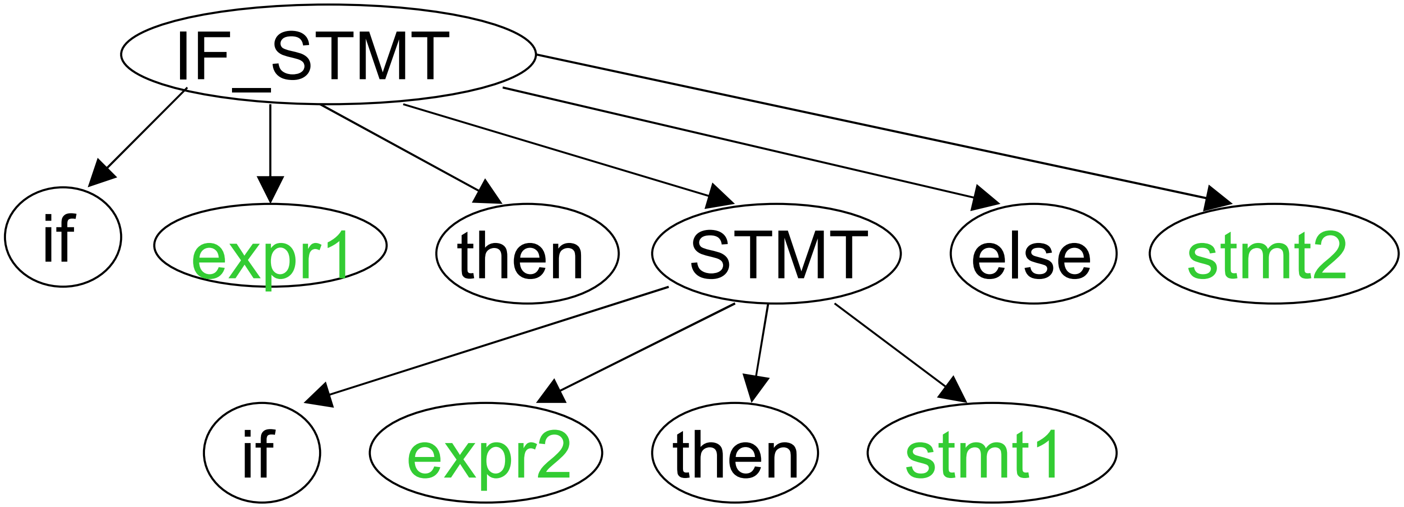
Another Ambiguous Grammar (Dangling Else)

IF_STMT → if EXPR then STMT

| if EXPR then STMT1 else STMT2

If expr1 then if expr2 then stmt1 else stmt2





Inherent Ambiguity

Some context free languages
have only ambiguous grammars

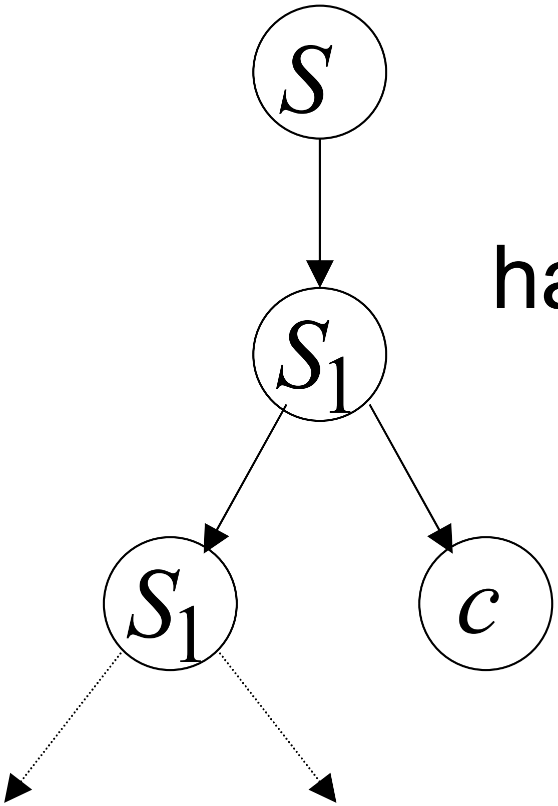
Example: $L = \{a^n b^n c^m\} \cup \{a^n b^m c^m\}$

$$S \rightarrow S_1 \mid S_2 \qquad S_1 \rightarrow S_1 c \mid A \qquad S_2 \rightarrow a S_2 \mid B$$

$$A \rightarrow aAb \mid \lambda \qquad B \rightarrow bBc \mid \lambda$$

$$S \to S_1 \mid S_2 \qquad S_1 \to S_1 c \mid A \qquad S_2 \to a S_2 \mid B$$

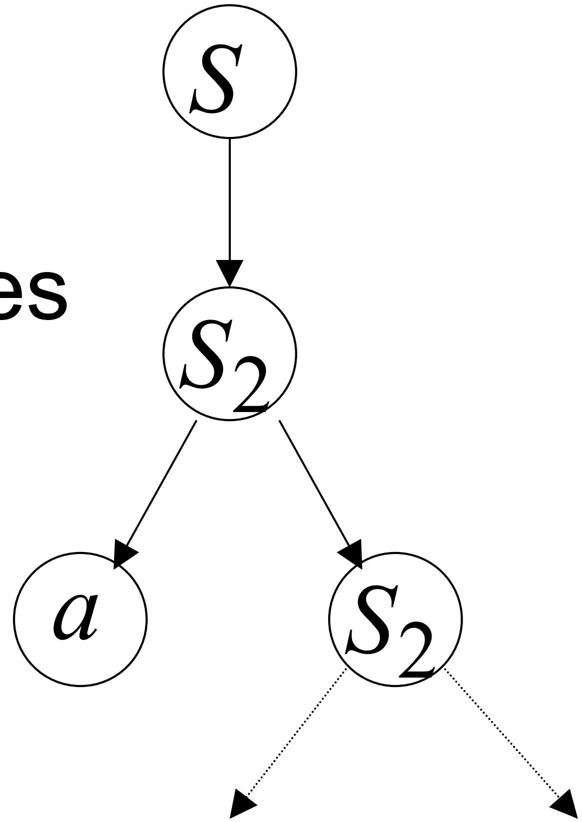
$$A \to aAb \mid \lambda \qquad B \to bBc \mid \lambda$$



$$a^n b^n c^n$$

has two derivation trees

Proof by
Harrison, 1978



Exercise 5.3.6

- Show that the following grammar is ambiguous.

$$S \to AB \mid aaB$$

$$A \to a \mid Aa$$

$$B \to b$$

There are two leftmost derivations for w = aab

$$S \Rightarrow aaB \Rightarrow aab,$$

$$S \Rightarrow AB \Rightarrow AaB \Rightarrow aaB \Rightarrow aab$$

Outline

1 Context-Free Grammars

2 Parsing and Ambiguity

3 Context-Free Grammars and Programming Languages

Machine Code

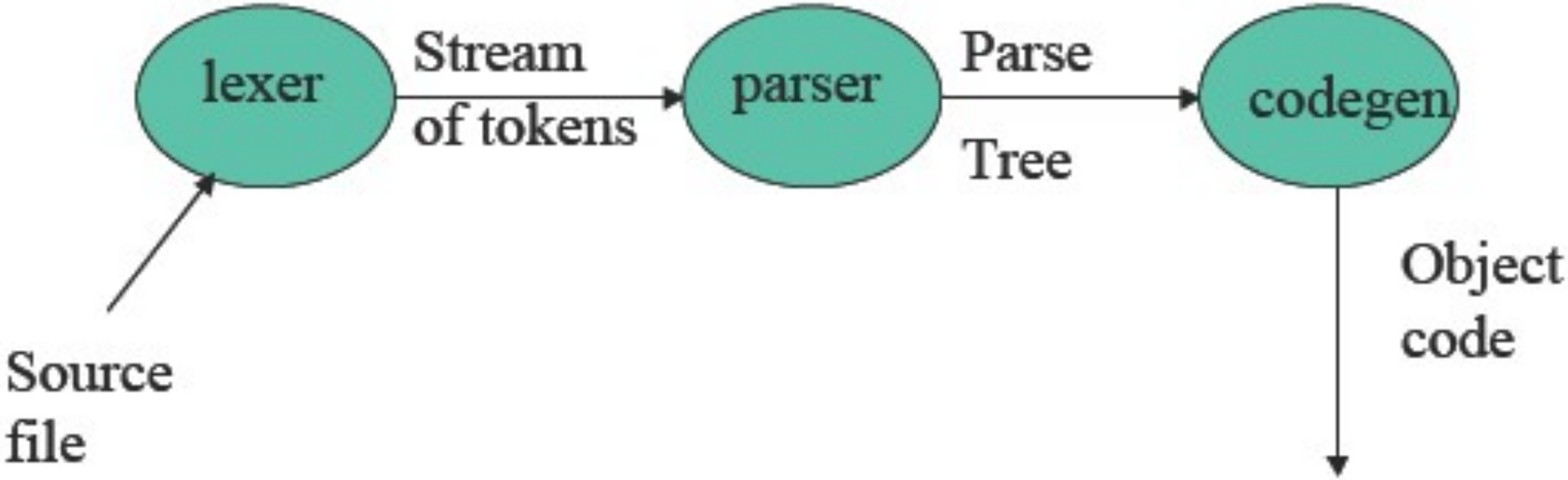
Program

```
v = 5;
if (v>5)
   x = 12 + v;
while (x !=3) {
  x = x - 3;
  v = 10;
}
......
```

Compiler

```
Add v,v,0
cmp v,5
jmplt ELSE
THEN:
  add x, 12,v
ELSE:
WHILE:
cmp x,3
...
```

- How a compiler works



Ken Thompson

Regular expressions in UNIX / grep / vi

Eric E. Schmidt

Mike Lesk

lex

Stephen C Johnson

yacc

A parser knows the grammar of the programming language

Parser

PROGRAM → STMT_LIST

STMT_LIST → STMT; STMT_LIST | STMT;

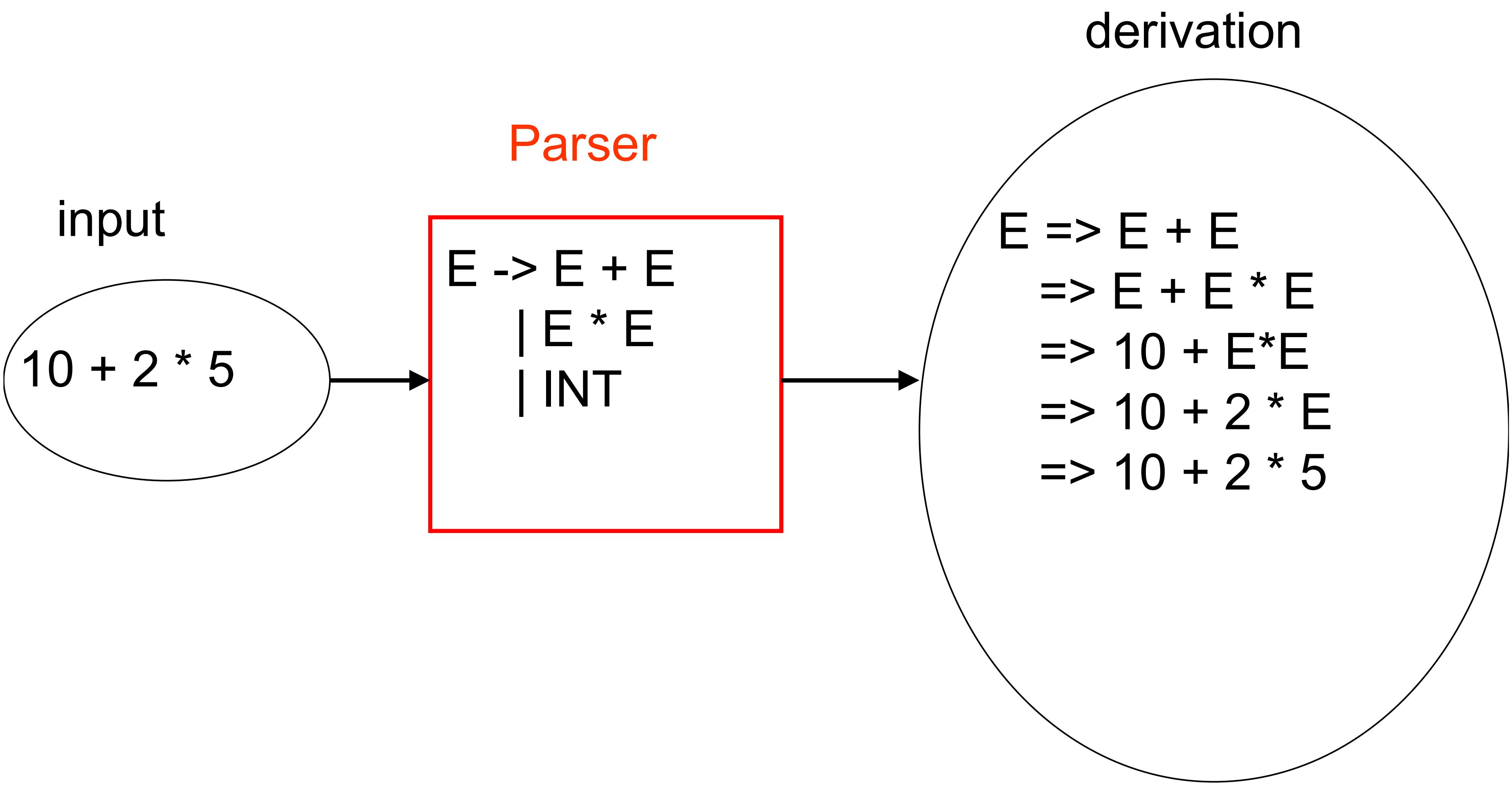
STMT → EXPR | IF_STMT | WHILE_STMT
| { STMT_LIST }

EXPR → EXPR + EXPR | EXPR - EXPR | ID

IF_STMT → if (EXPR) then STMT
| if (EXPR) then STMT else STMT

WHILE_STMT → while (EXPR) do STMT

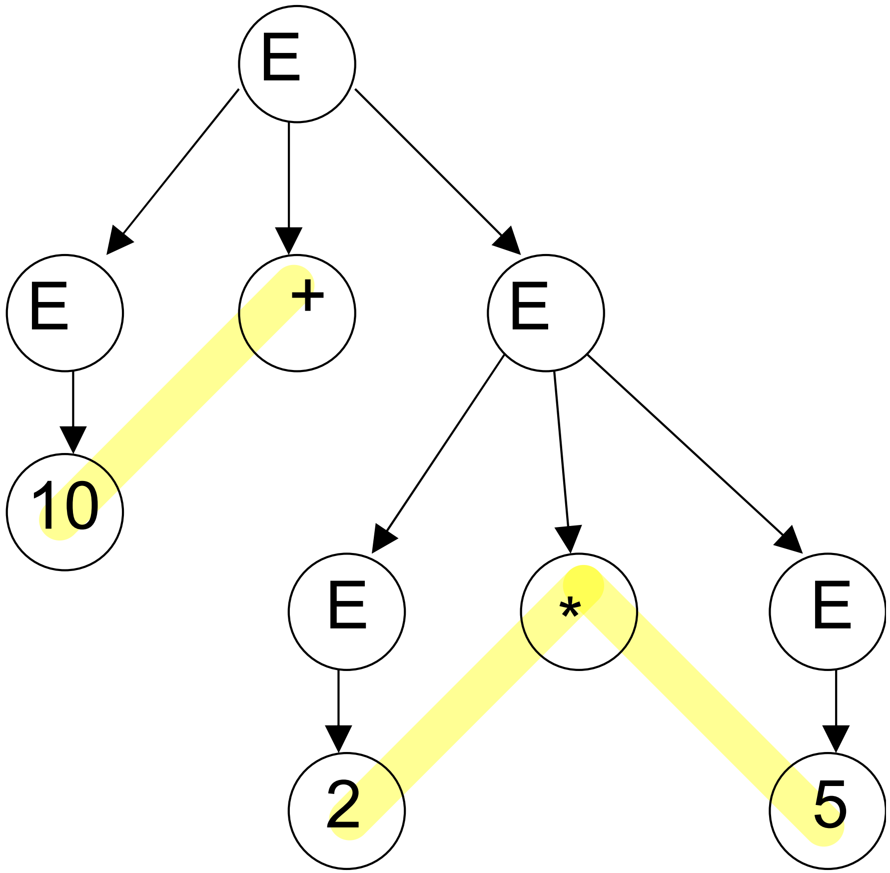
The parser finds the derivation of a particular input



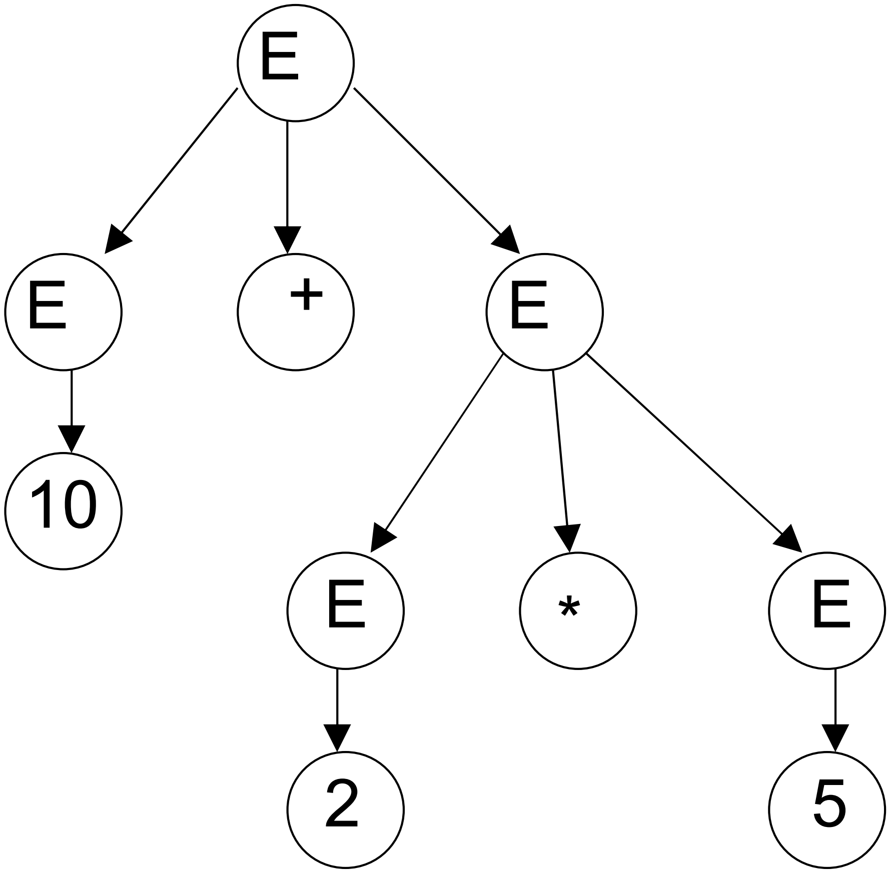
derivation

$$
\begin{aligned}
E &\Rightarrow E + E \\
&\Rightarrow E + E * E \\
&\Rightarrow 10 + E*E \\
&\Rightarrow 10 + 2 * E \\
&\Rightarrow 10 + 2 * 5
\end{aligned}
$$

derivation tree



derivation tree



machine code

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

Programming Language and Grammar

- Backus-Naur Form (BNF)
  - Common used grammar for programming languages
  - Ex:
    <expr> ::= <term> | <expr> + <term>
    <term> ::= <factor> | <term> * <factor>
- LL and LR grammar: parse in linear time

Questions?