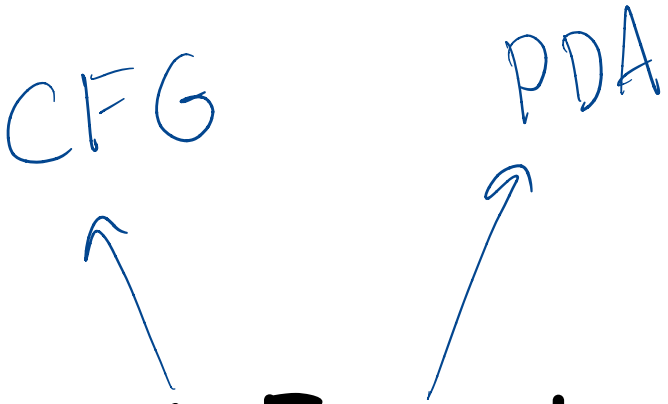
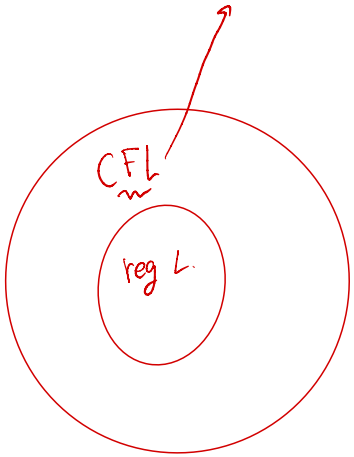
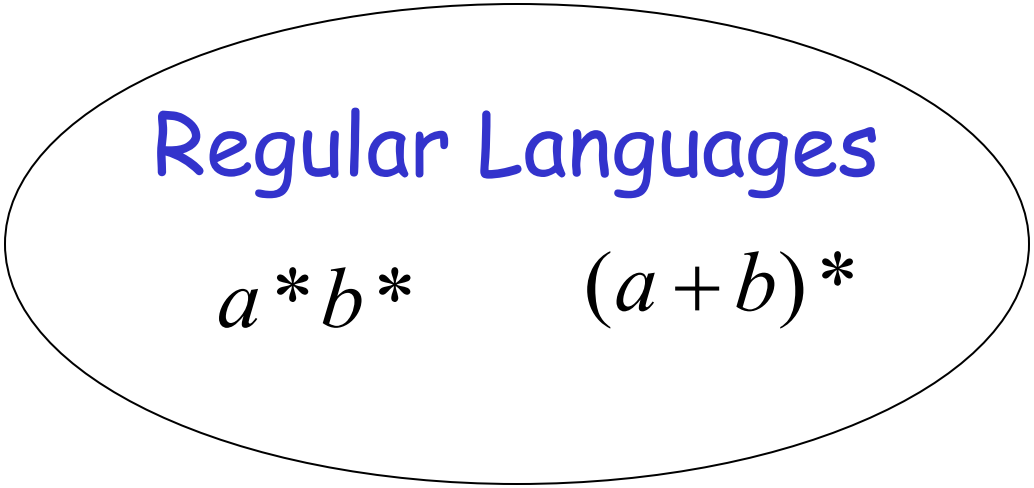
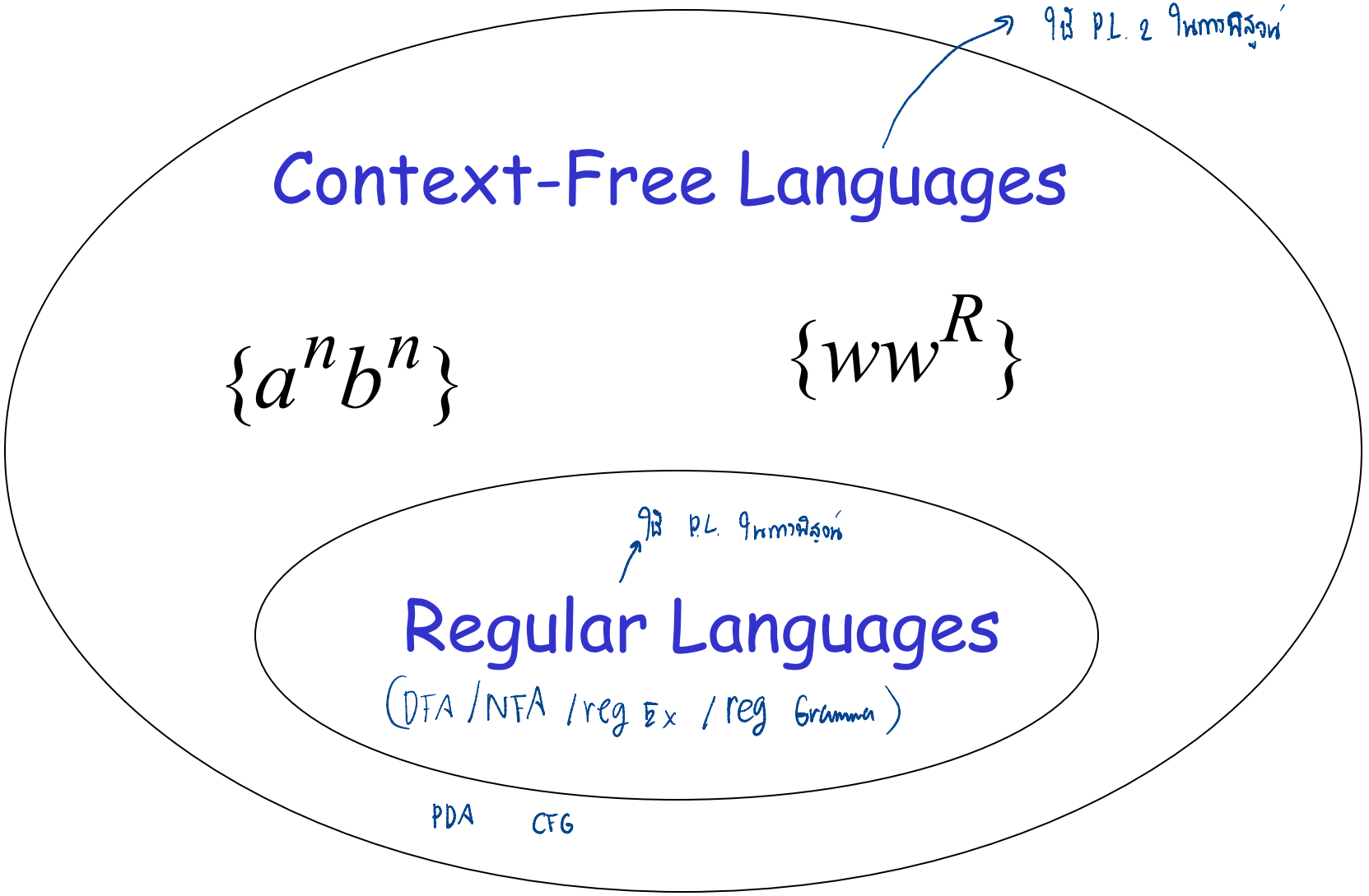


Context-Free Languages

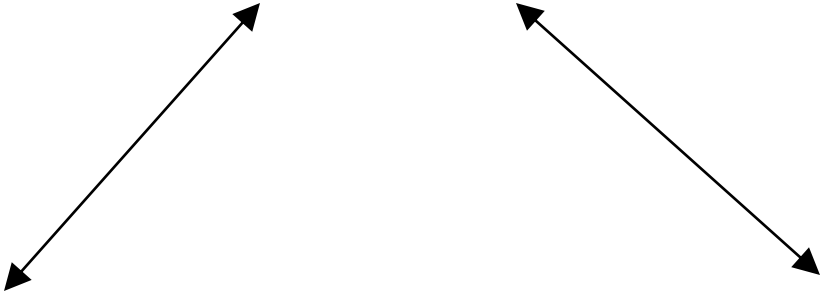


$\{a^n b^n : n \geq 0\}$ $\{ww^R\}$




ใช้ P.L. 2 พิสูจน์
Context-Free Languages
$\{a^n b^n\}$
$\{ww^R\}$
ใช้ P.L. พิสูจน์
Regular Languages
(DFA / NFA / reg Ex / reg Grammar)
PDA
CFG

Context-Free Languages



Context-Free Grammars

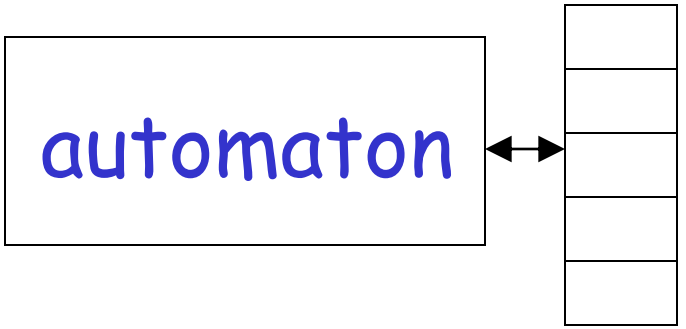
Pushdown Automata

RL

reg gram

Finite Automata (DFA/NFA)

stack



Context-Free Grammars

Example

Gramma 1

ไม่ใช่ Reg gram
เพราะ: S ขนาดต้องรู้ซ้าย-ขวา

A context-free grammar G:

$$S \to aSb$$

$$S \to \lambda$$

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

A derivation:

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

อีกวิธี

A context-free grammar G:

$$S \to aSb$$

$$S \to \lambda$$

Another derivation:

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

$$S \rightarrow aSb$$

$$S \rightarrow \lambda$$

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

(()()) ไม่ของใช้

ถ้า a = (
b =)

ได้แบบนี้

Describes parentheses: ((( )))

Example

Gramma 2.

A context-free grammar G:

$$S \to aSa$$

$$S \to bSb$$

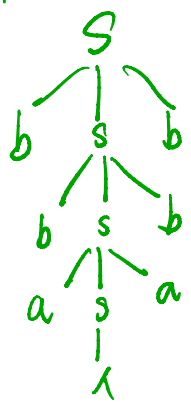
$$S \to \lambda$$

bba

V

abb

V A



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:

อีก 1 str

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

$$S \rightarrow aSa$$

$$S \rightarrow bSb$$

$$S \rightarrow \lambda$$

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

ต้อง define ว่า w เป็นอะไร

Grammaที่ 3

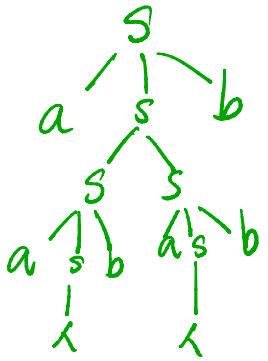
Example

A context-free grammar G:

$$S \to aSb$$

$$S \to SS$$

$$S \to \lambda$$



A derivation:

aabaabb

S→λ str ที่สั้นที่สุด

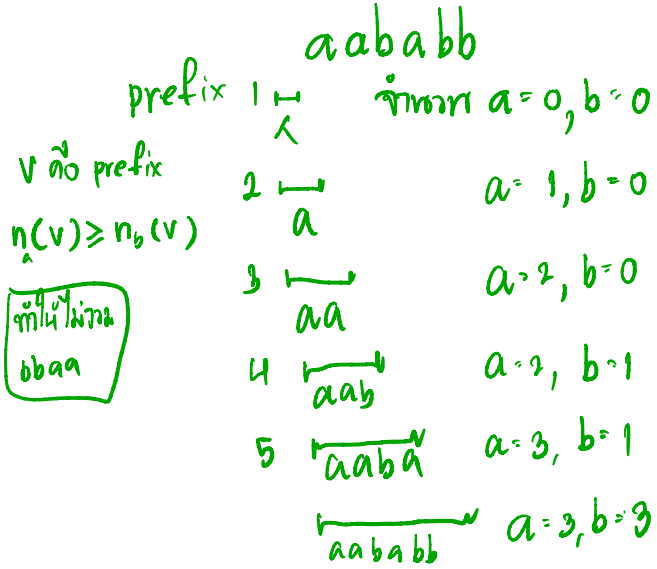
$$S \Rightarrow SS \Rightarrow aSbS \Rightarrow abS \Rightarrow ab$$

A context-free grammar G:

$$S \to aSb$$

$$S \to SS$$

$$S \to \lambda$$



A derivation:

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

จำนวน
a = b

$$S \rightarrow aSb$$

$$S \rightarrow SS$$

$$S \rightarrow \lambda$$

bbaa X

เรียงไงก็ไม่ผ่าน

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

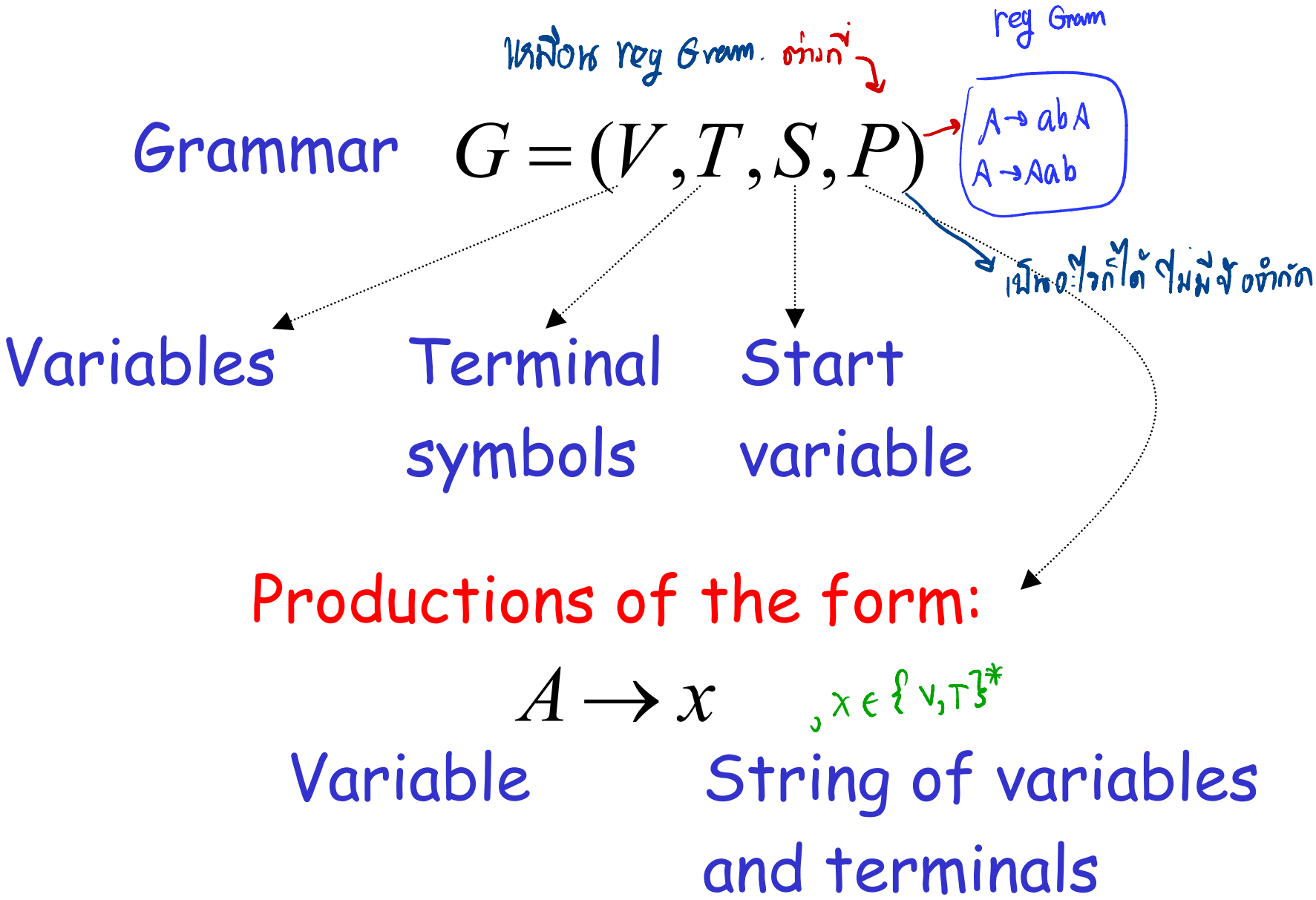
a : (

b :)

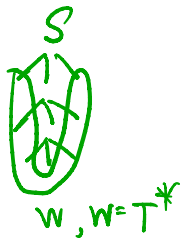
Gram ไม่ควรมี)) ((

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

Definition: Context-Free Grammars



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



นิยาม

ทำกี่ครั้งก็ได้ตั้งแต่ 0

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

Definition: Context-Free Languages

A language L is context-free

if and only if

there is a context-free grammar G

with $L = L(G)$

มี 5 production

Derivation Order

Left most

Right most

1. $S \rightarrow AB$

2. $A \rightarrow aaA$

3. $A \rightarrow \lambda$

4. $B \rightarrow Bb$

5. $B \rightarrow \lambda$

Leftmost derivation:

มองจากทางซ้ายมือ หาตัวไหน = generate ตัวนั้นก่อน (เลือก product ตัวไหนก็ได้)

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

ทำ str เตรียมก้าวออกมาได้

Start

A = λ

B = λ

Rightmost derivation:

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

$$S \to aAB$$ ทำตามเข้าใจเอง

$$A \to bBb$$

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

Leftmost derivation:

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

$$\Rightarrow abbbbB \Rightarrow abbbb$$

Rightmost derivation:

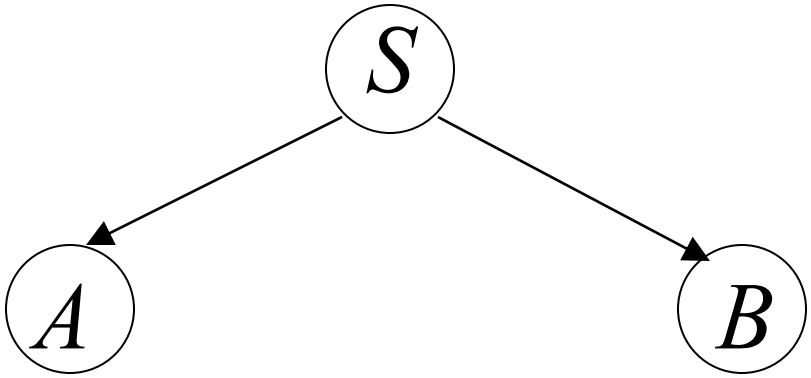
$$S \Rightarrow aAB \Rightarrow aA \Rightarrow abBb \Rightarrow abAb$$

$$\Rightarrow abbBbb \Rightarrow abbbb$$

Derivation Trees

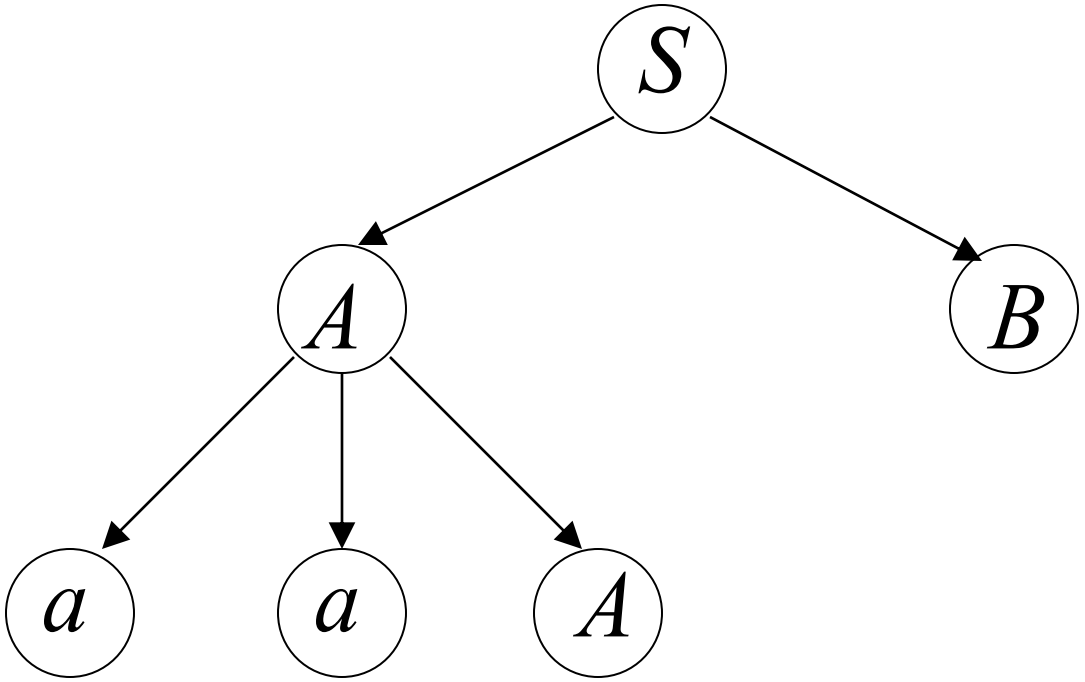
$S \to AB$ $A \to aaA \mid \lambda$ $B \to Bb \mid \lambda$

$S \Rightarrow AB$



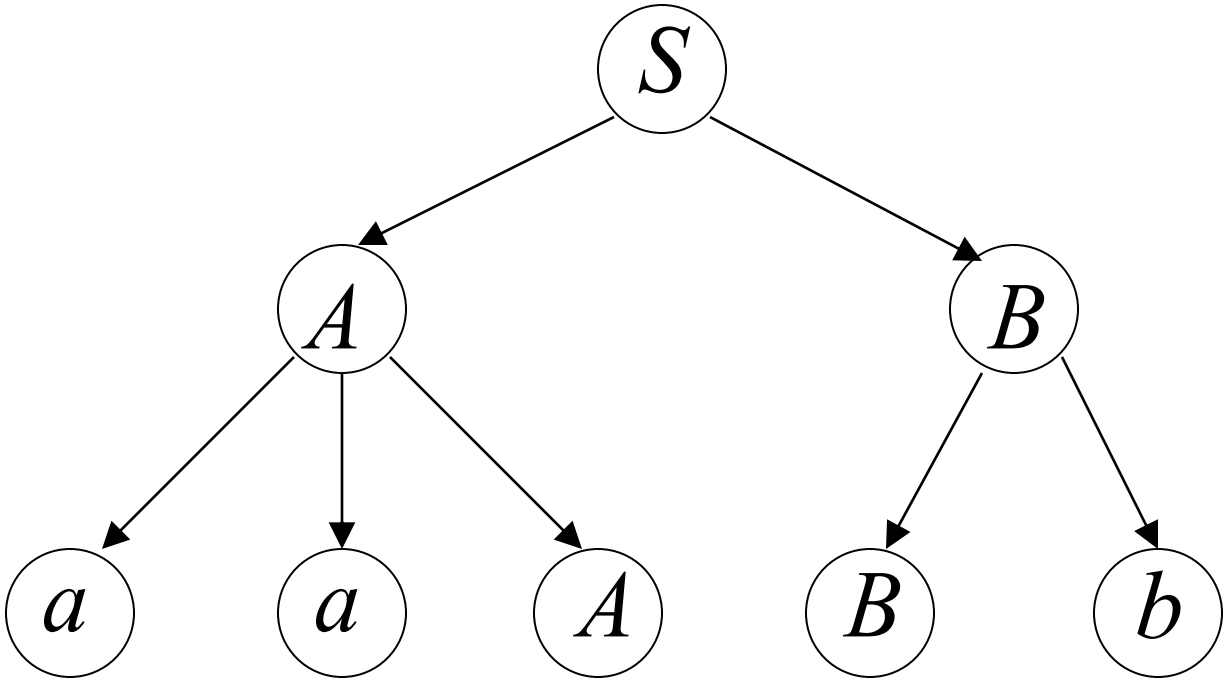
$S \to AB$ $A \to aaA \mid \lambda$ $B \to Bb \mid \lambda$

$$S \Rightarrow AB \Rightarrow aaAB$$



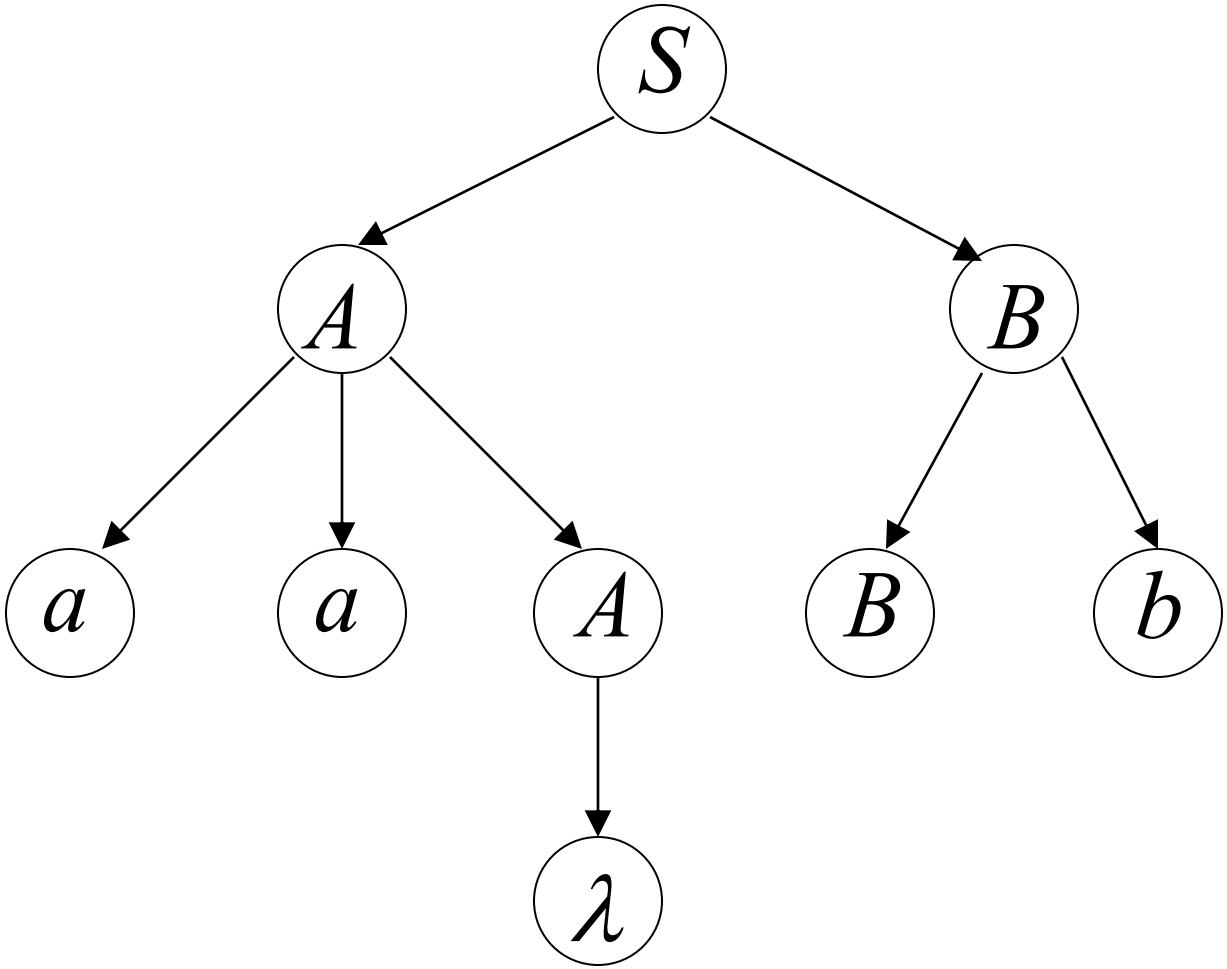
$$S \to AB \qquad A \to aaA \mid \lambda \qquad B \to Bb \mid \lambda$$

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



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

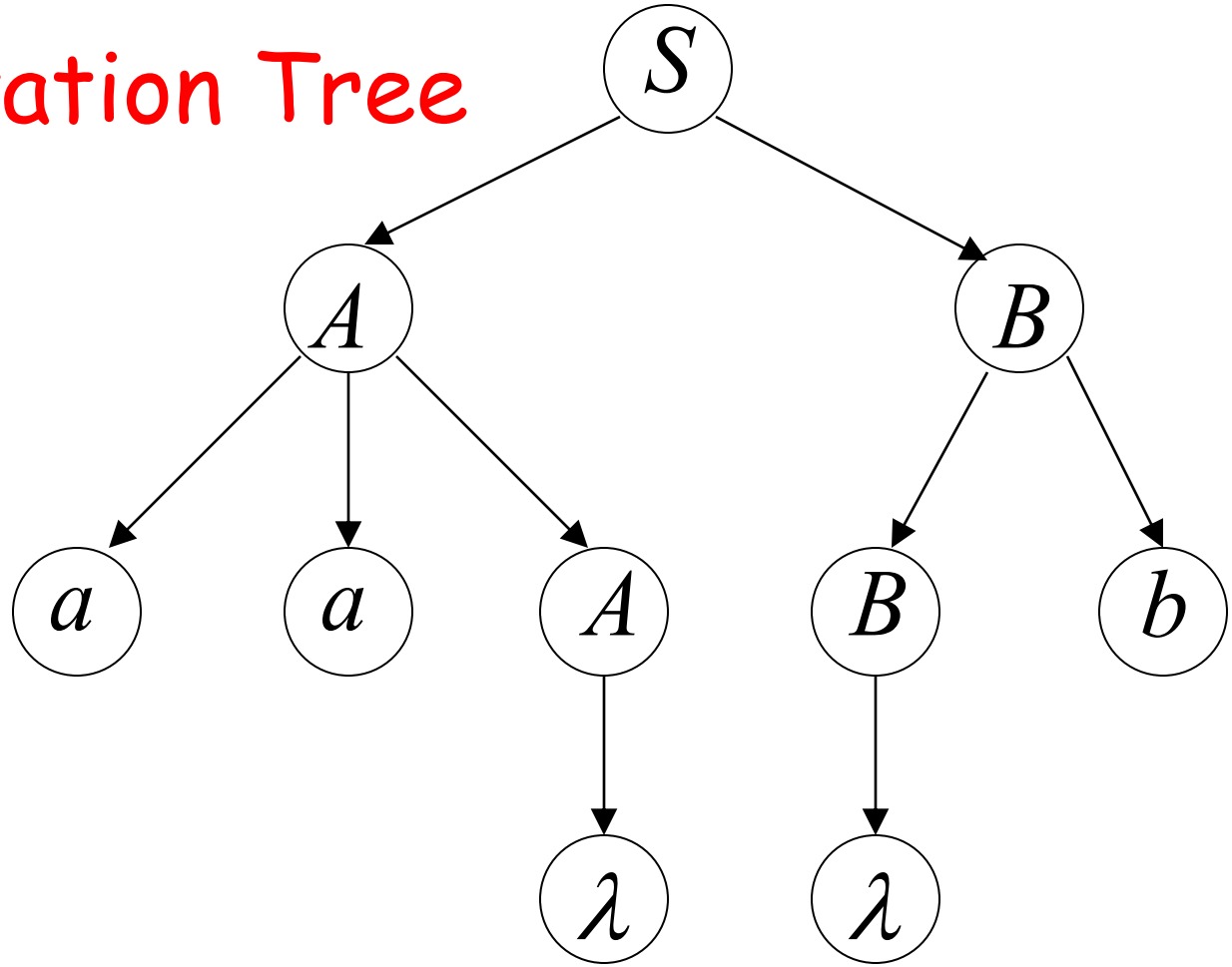
$$S \Rightarrow AB \Rightarrow aaAB \Rightarrow aaABb \Rightarrow aaBb$$



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

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

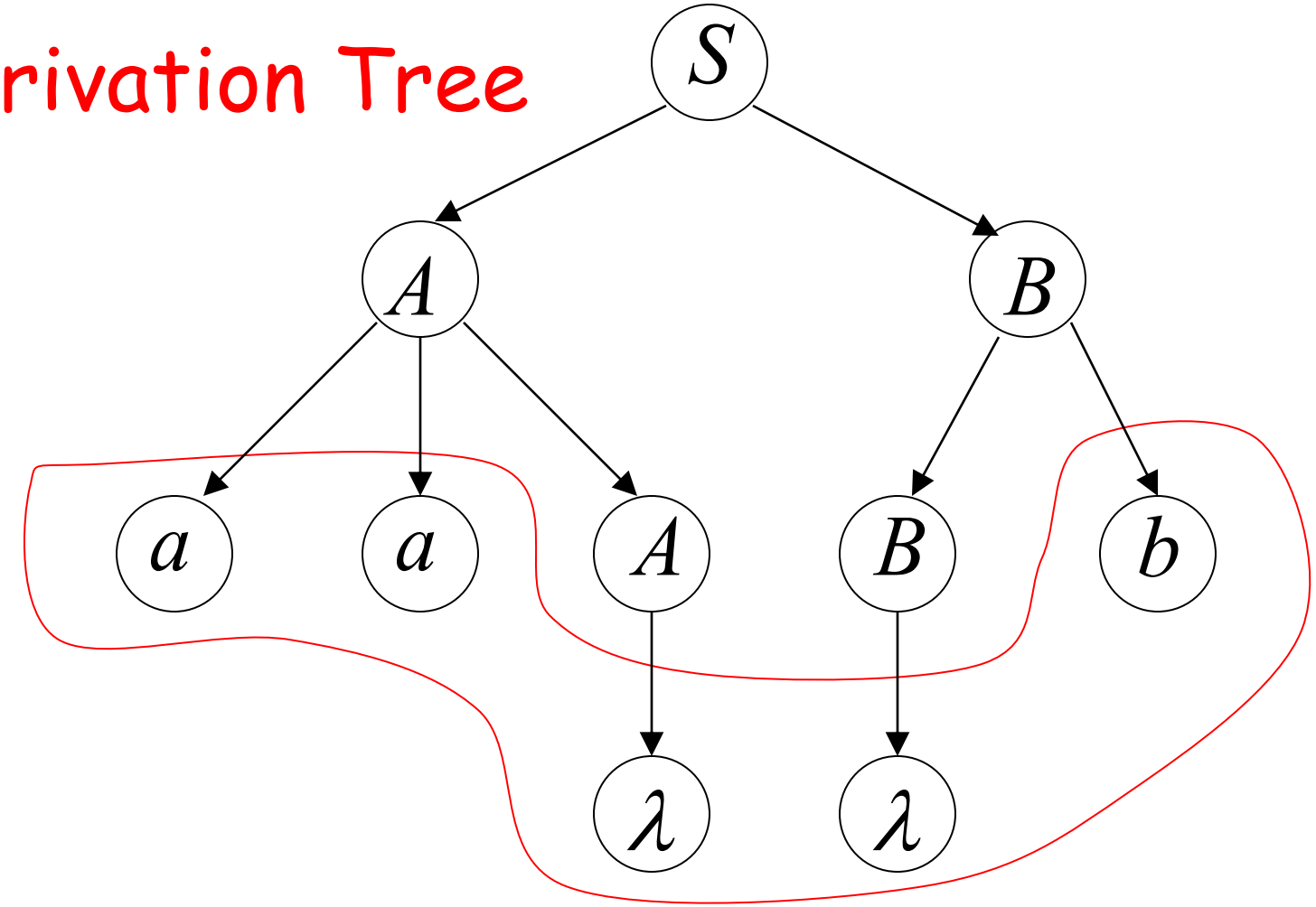
Derivation Tree



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

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

Derivation Tree



yield

$aa\lambda\lambda b$

$= aab$

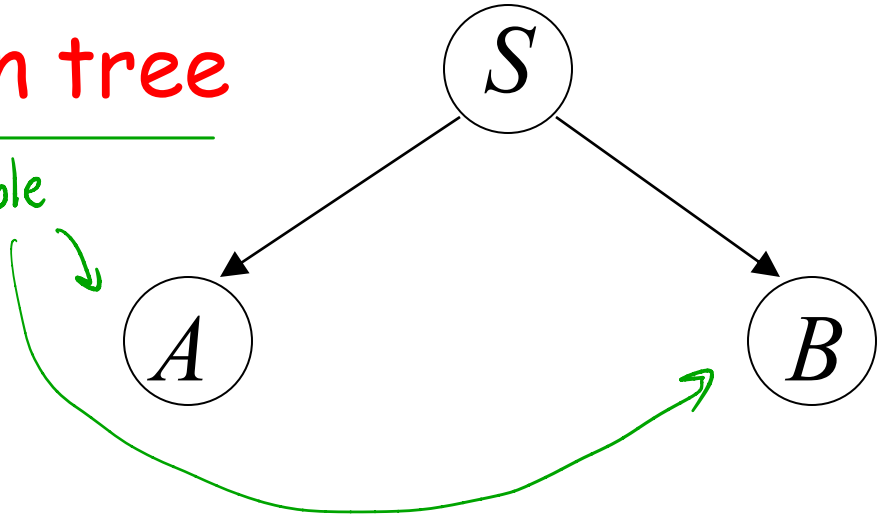
Partial Derivation Trees

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

$S \Rightarrow AB$

Partial derivation tree

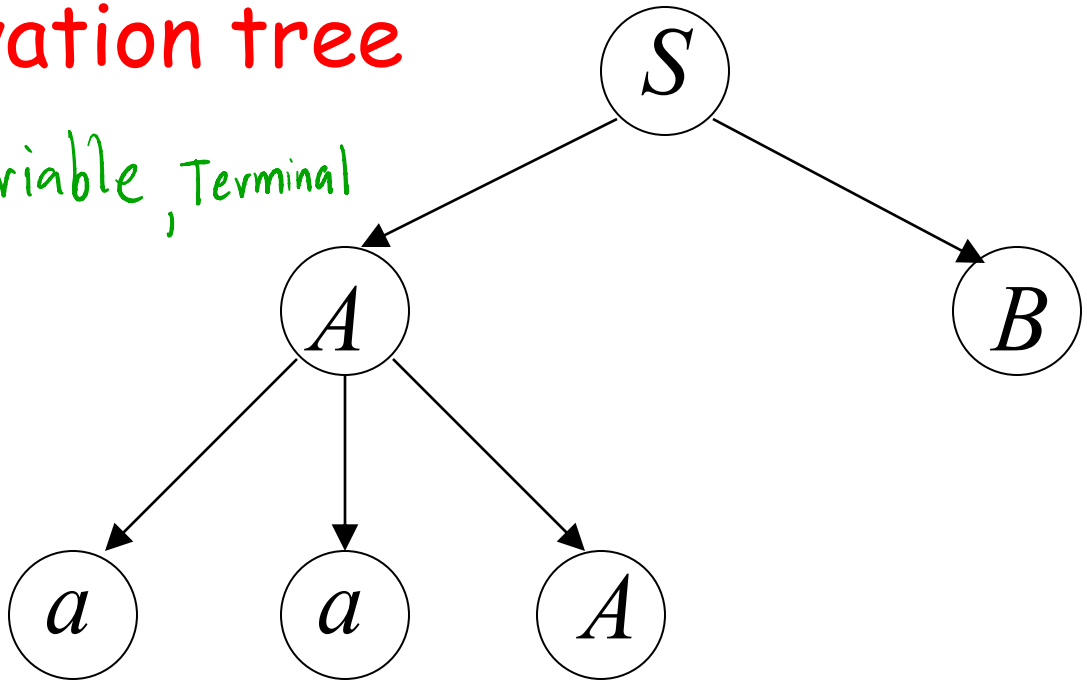
ยัง ทำไม่เสร็จ ที่ลูก ยังมี variable



$$S \Rightarrow AB \Rightarrow aaAB$$

Partial derivation tree

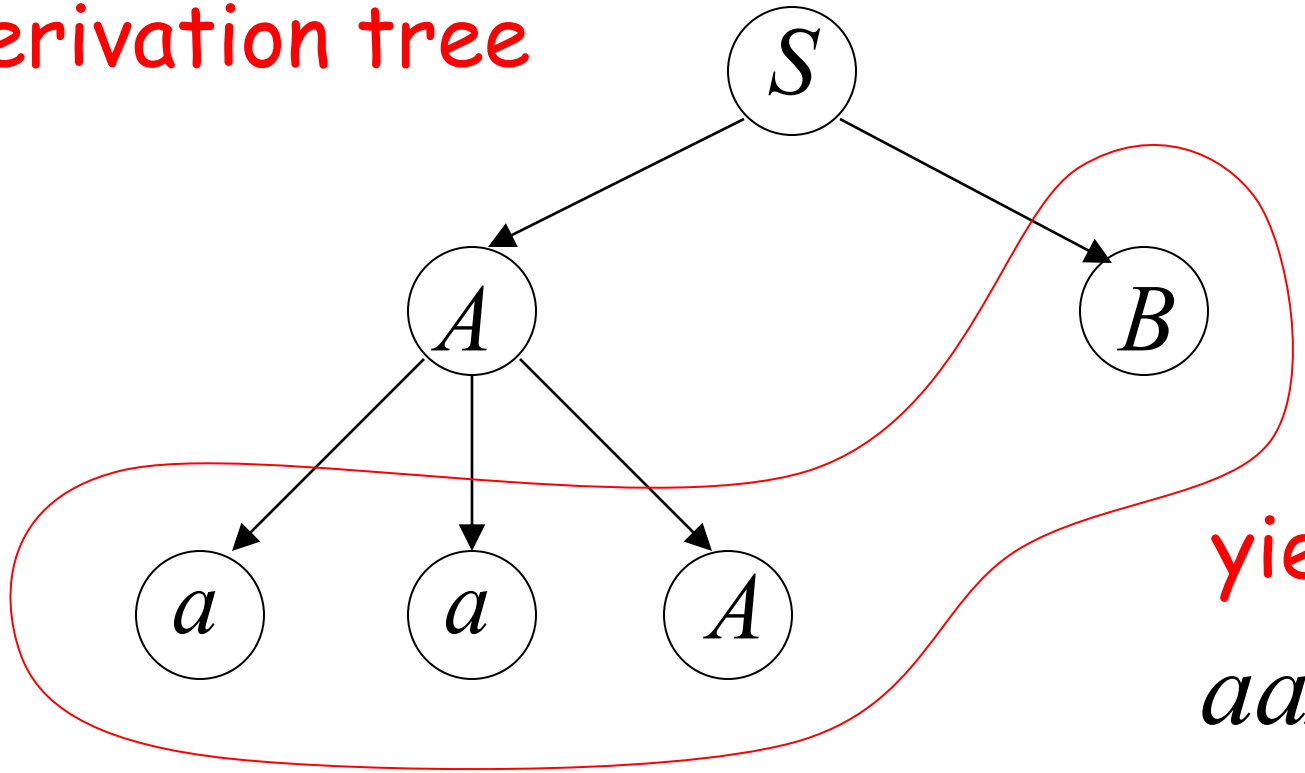
ที่ผสมทั้ง variable, Terminal



$$S \Rightarrow AB \Rightarrow aaAB$$

sentential form

Partial derivation tree



yield

$aaAB$

Sometimes, derivation order doesn't matter

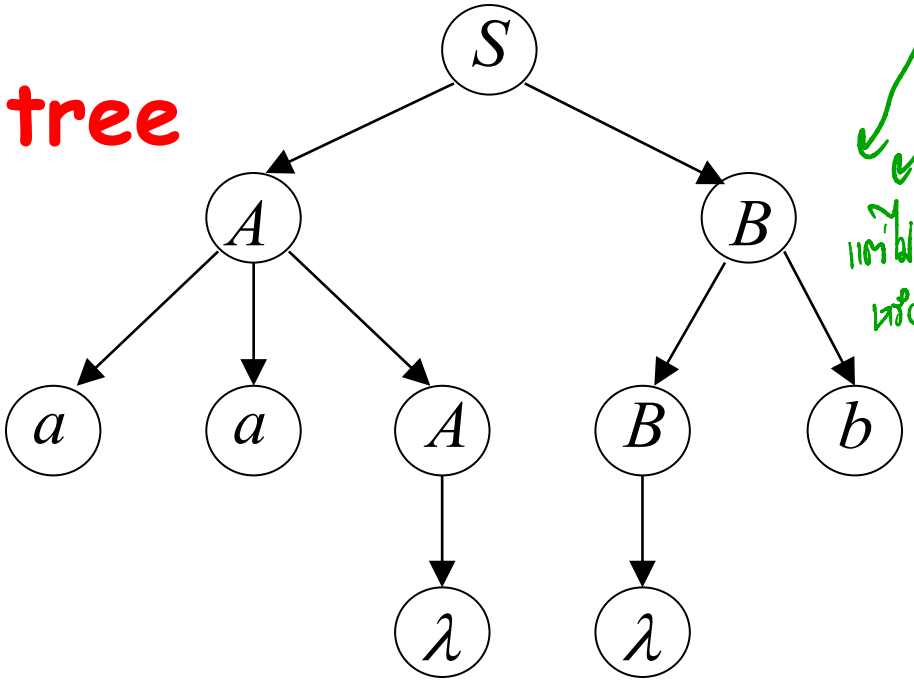
Leftmost:

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

Rightmost:

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

Same derivation tree



เขียนแบบนี้ได้
แต่ไม่รู้ว่า left most
หรือ Right most

แปลว่า กำกวม

Ambiguity

แปลว่า bi face

2 หน้า

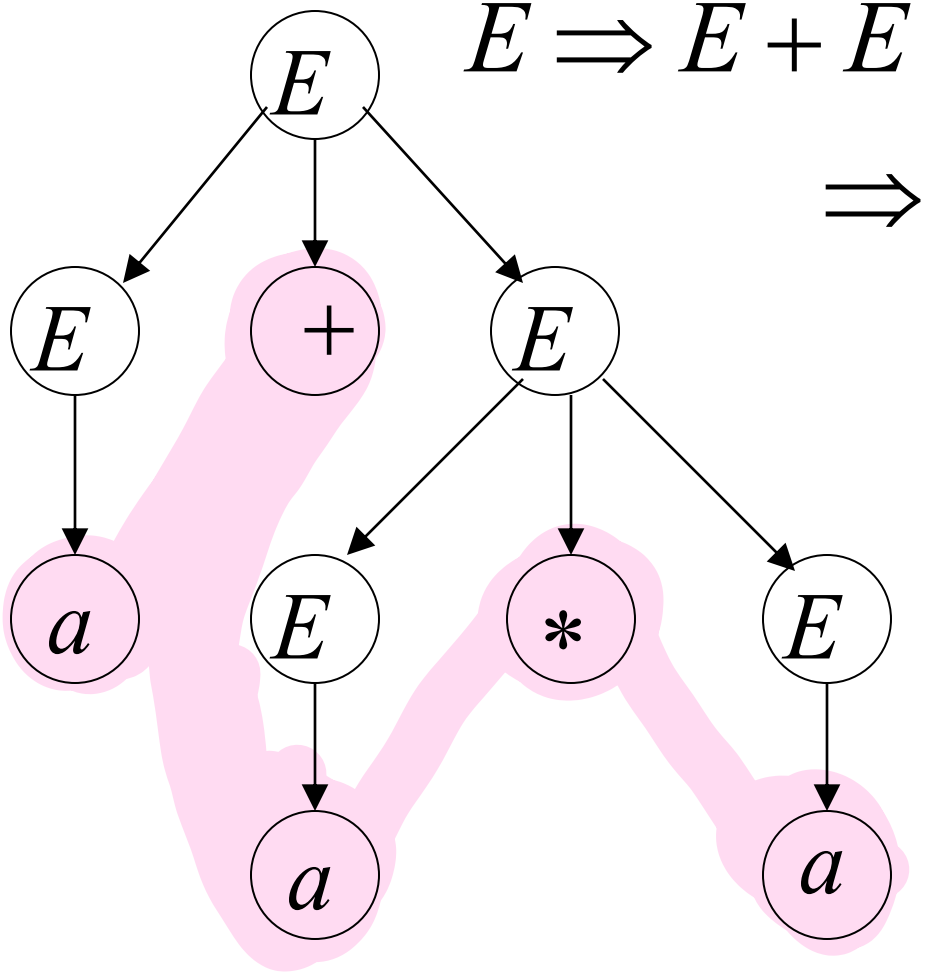
Str เดียวกัน มี 2 tree ที่แตกต่างกัน

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

$a+a*a$

Tree

ทำ + non



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

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

leftmost derivation

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

ทำ * ก่อน

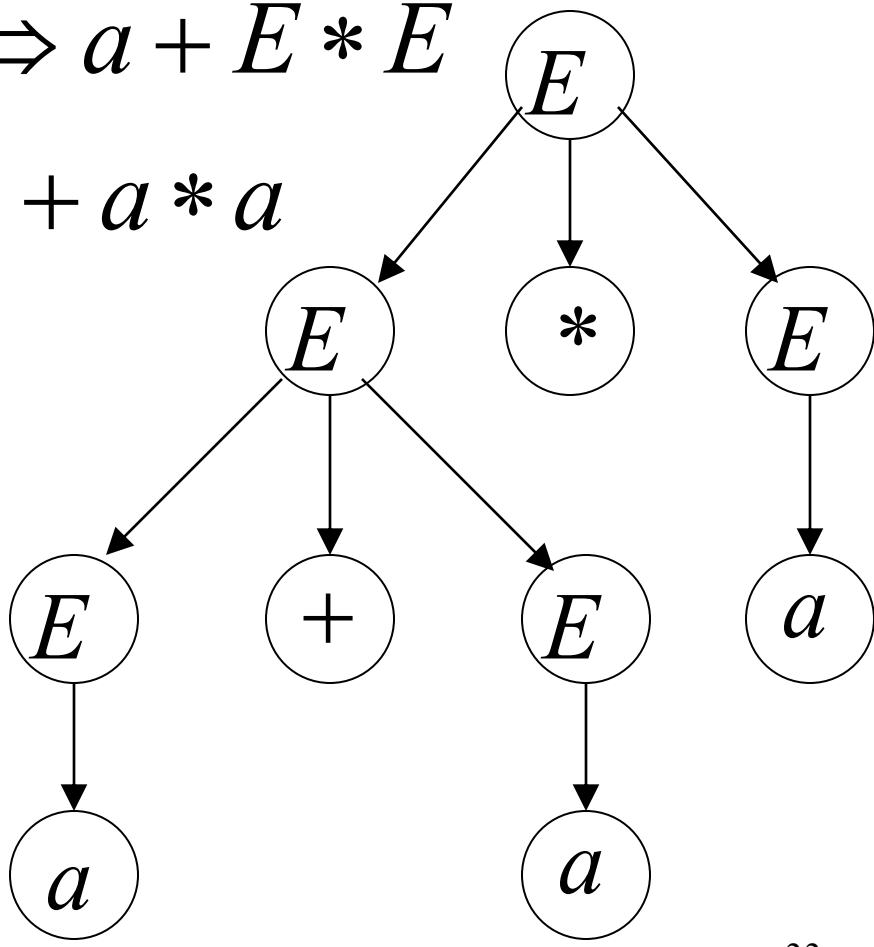
$$a + a * a$$

Tree

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

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

leftmost derivation

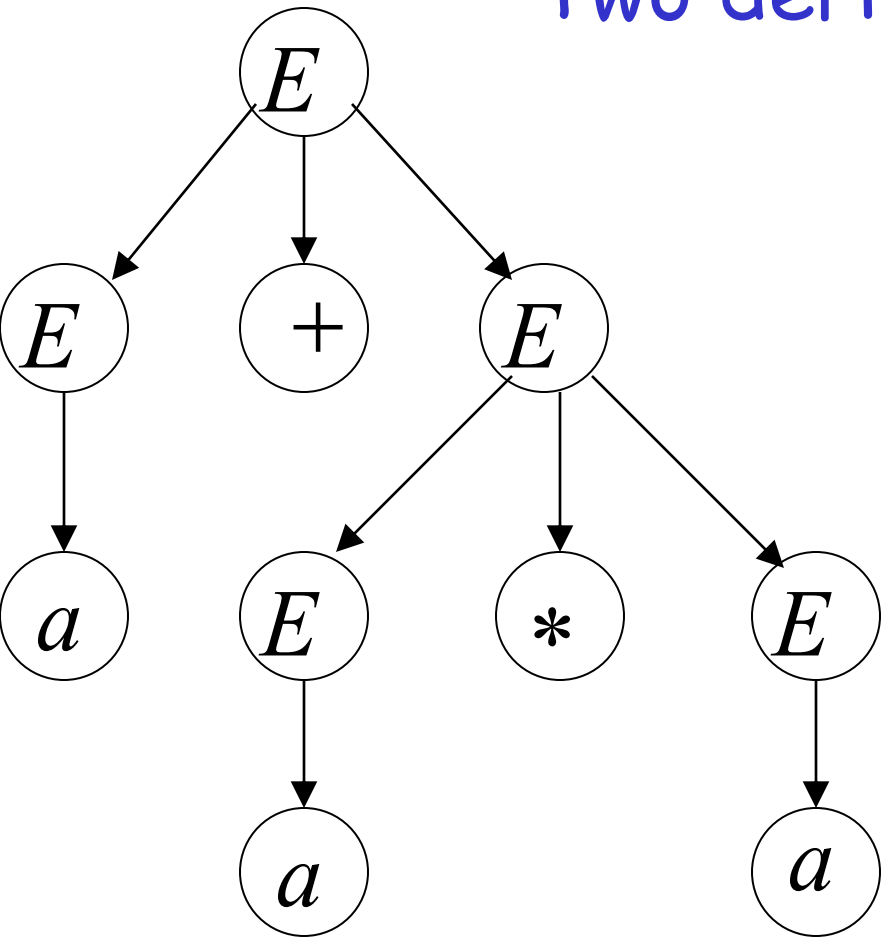


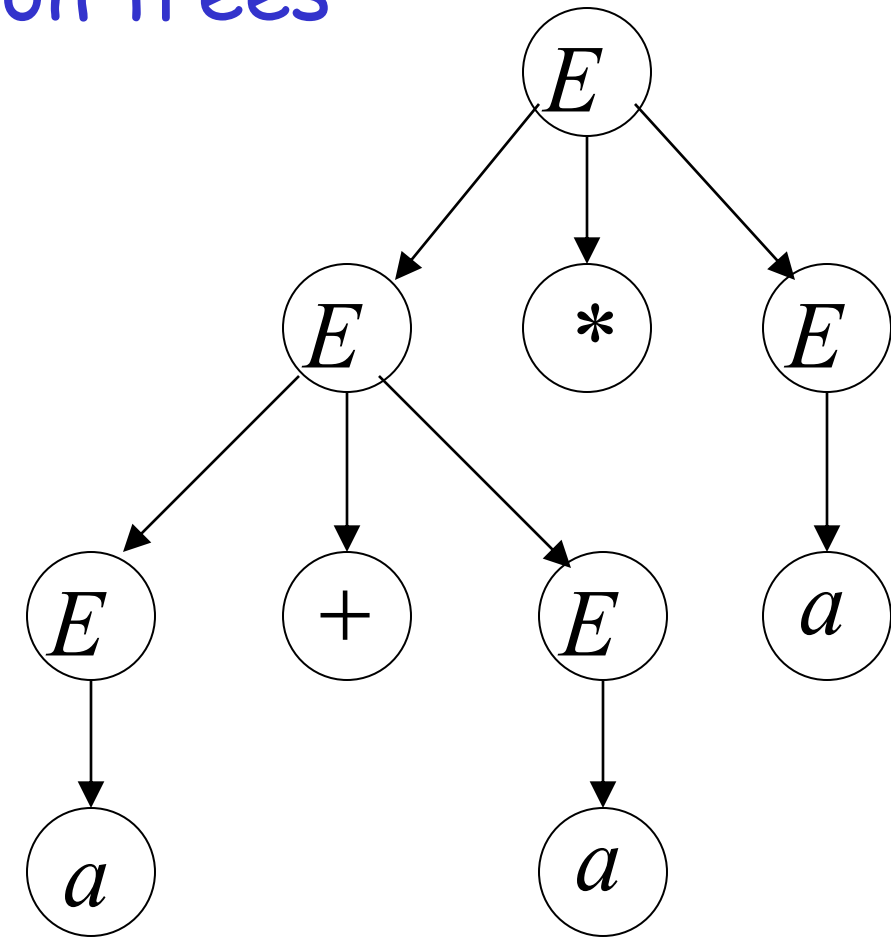
$$E \rightarrow E + E \mid E * E \mid (E) \mid a$$

$$a + a * a$$

str เดียวกัน มี 2 tree ที่แตกต่างกัน

Two derivation trees





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

is ambiguous:

อธิบายสั้นๆ

พิสูจน์ว่า Gramma เป็น Gramma ที่กำกวม

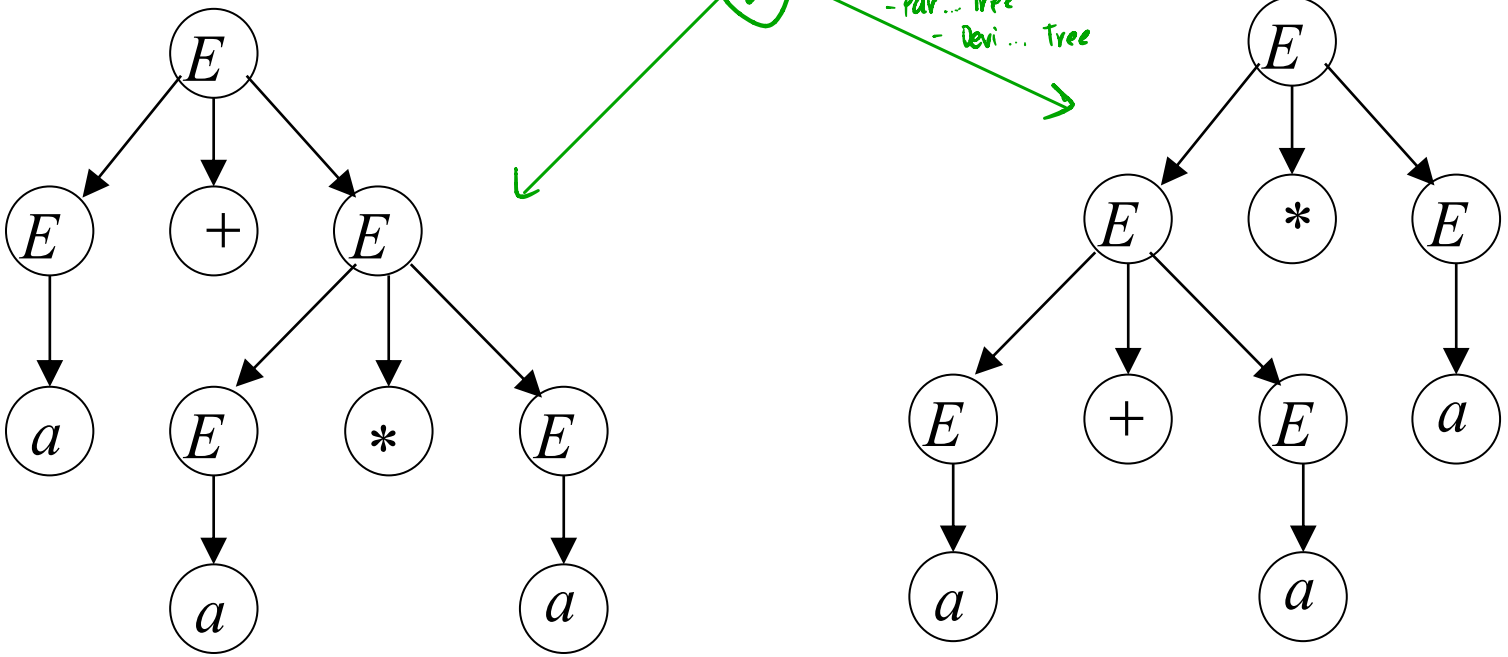
string $a + a * a$ has two derivation trees

(1) ยกตัวอย่าง 1 str

หา Gramma Tree ที่แตกต่างกัน

(2) 2 Tree ขึ้นไป

เป็นได้ทั้ง
- Par... Tree
- Deri... Tree



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

is ambiguous:

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

(2) ทำ Derivation แบบนี้ได้ แต่ "สวนทับแตกต่าง" Derivation Order < left / Right

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

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

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

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

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:

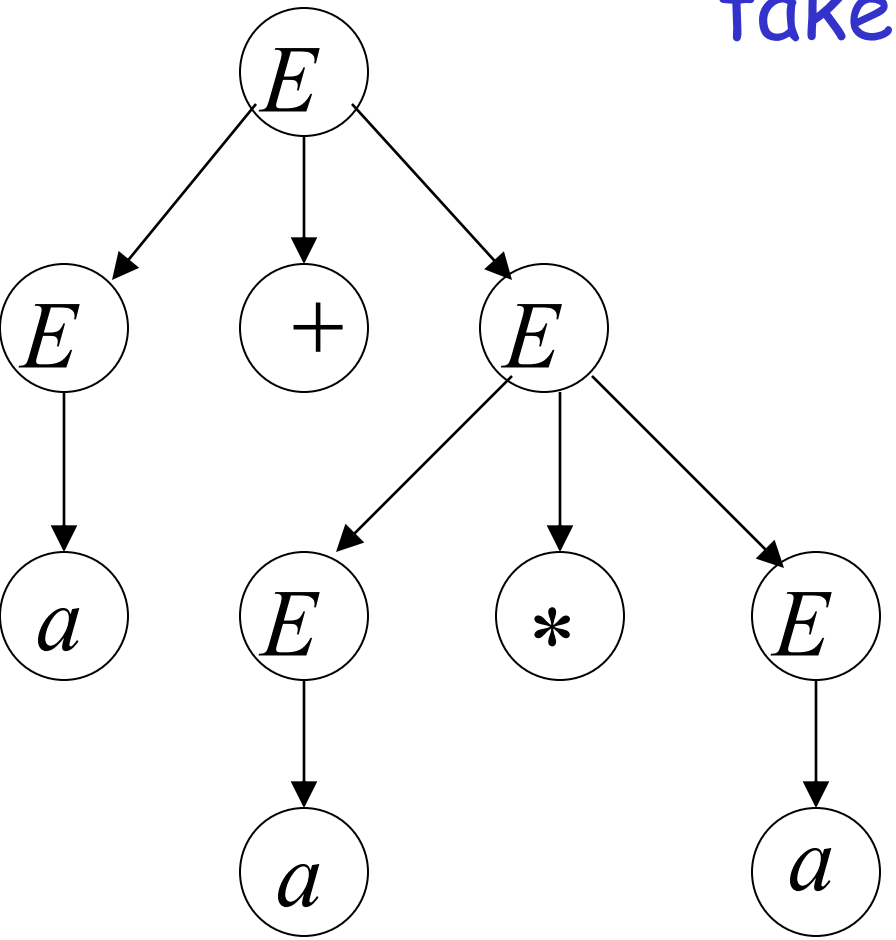
two or more leftmost derivations

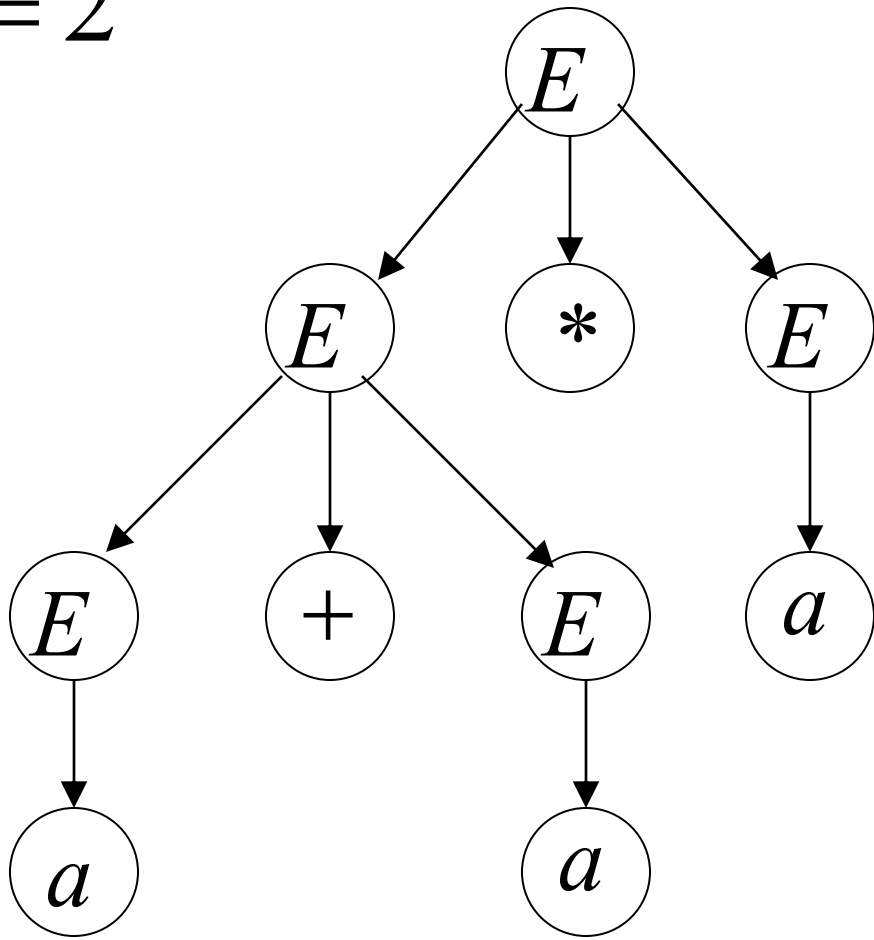
(or rightmost)

Why do we care about ambiguity?

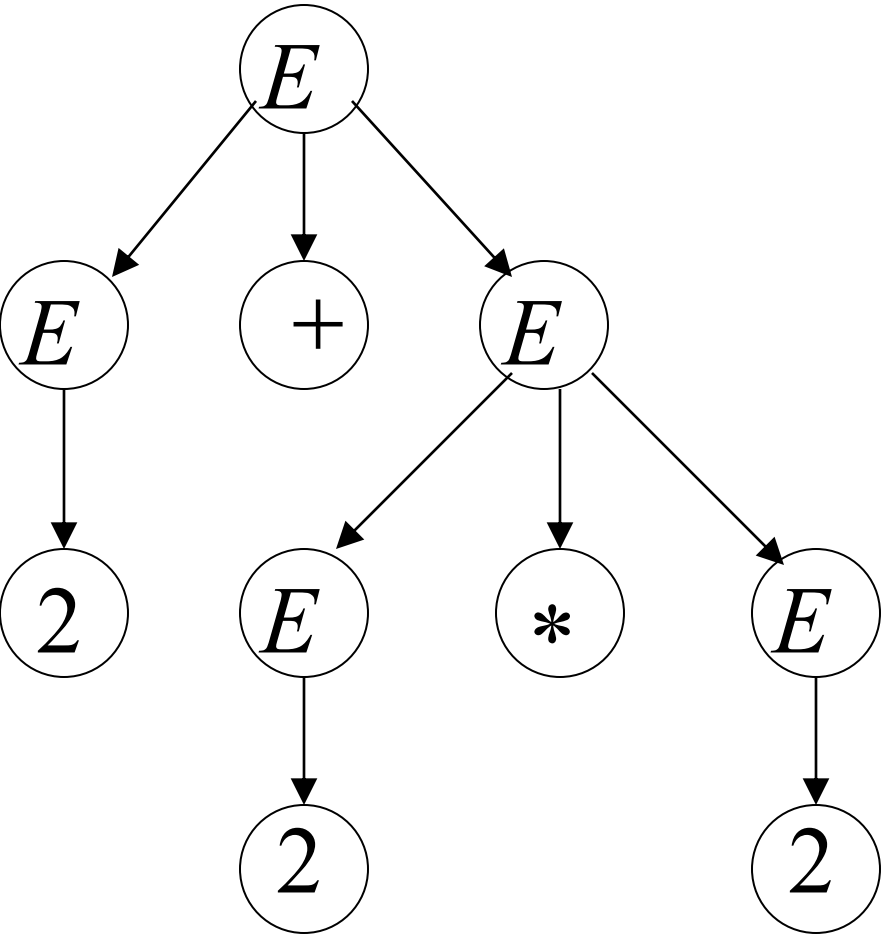
$$a + a * a$$

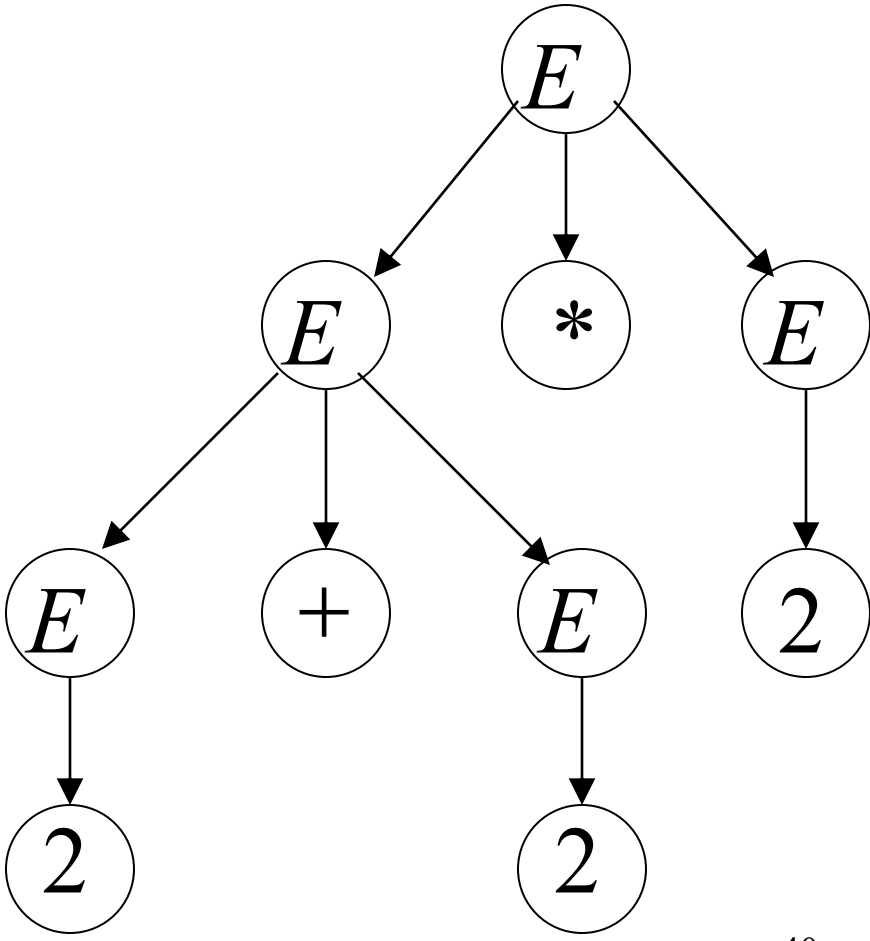
take $a = 2$



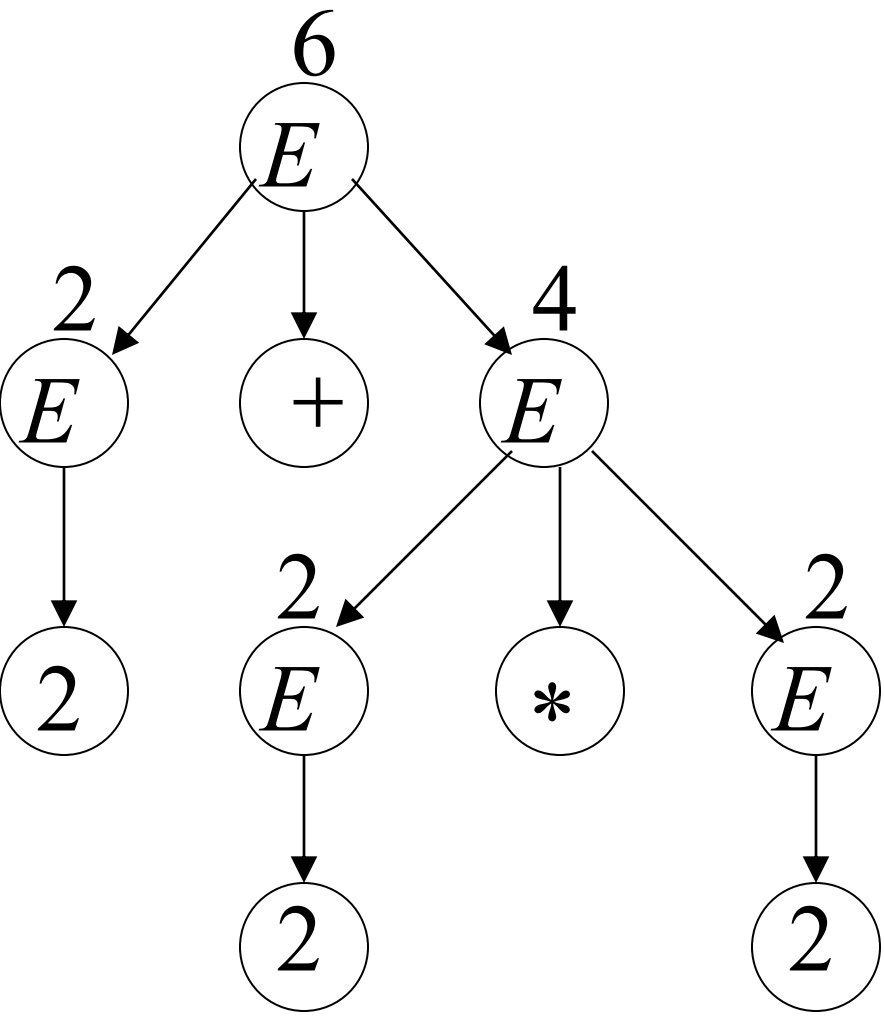


$$2 + 2 * 2$$

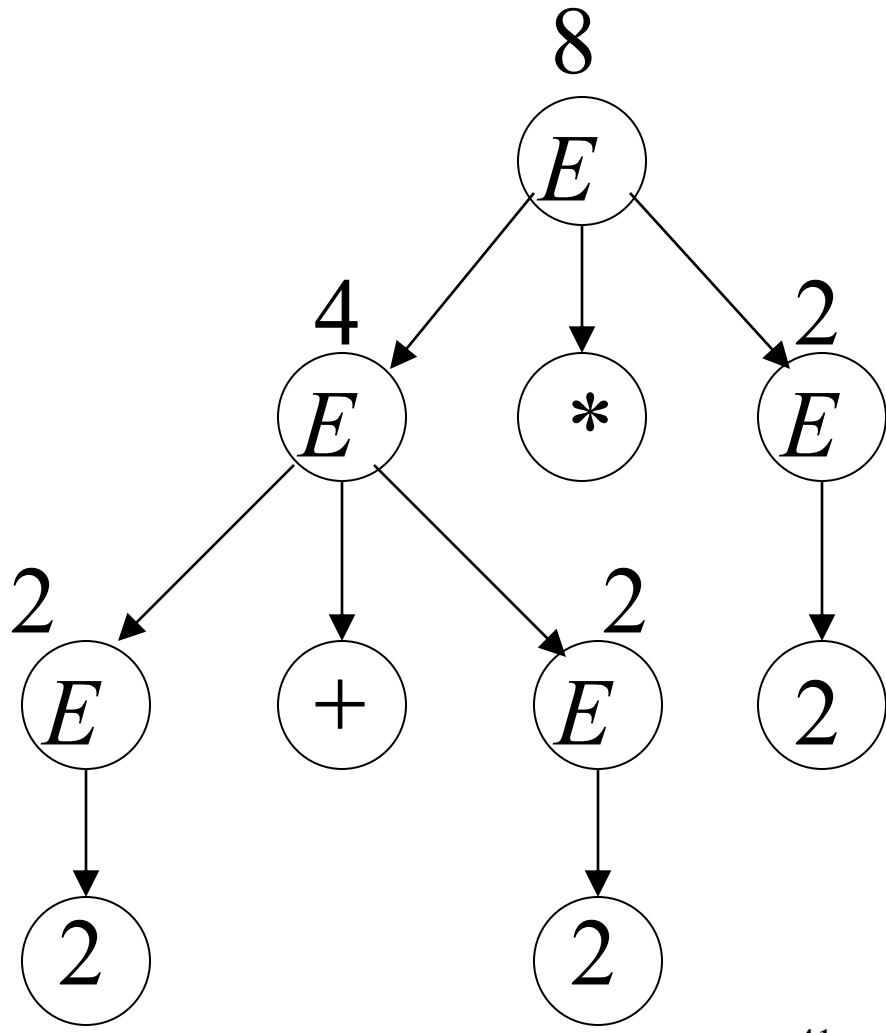




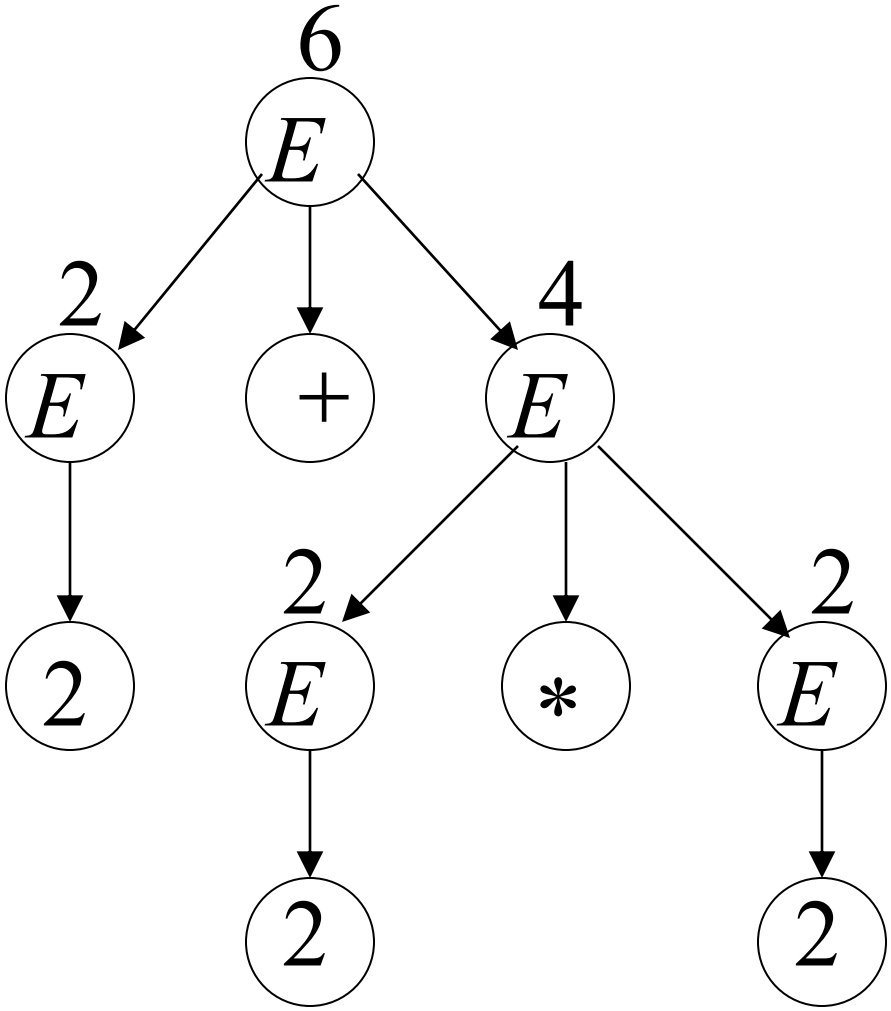
$2+2*2=6$



$2+2*2=8$



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



- Ambiguity is **bad** for programming languages

  จำเป็น ต้องกำจัด

- We want to remove ambiguity

We fix the ambiguous grammar:

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

New non-ambiguous grammar:

$$E \to E + T$$

$$E \to T$$

$$T \to T * F$$

$$T \to F$$

$$F \to (E)$$

$$F \to a$$

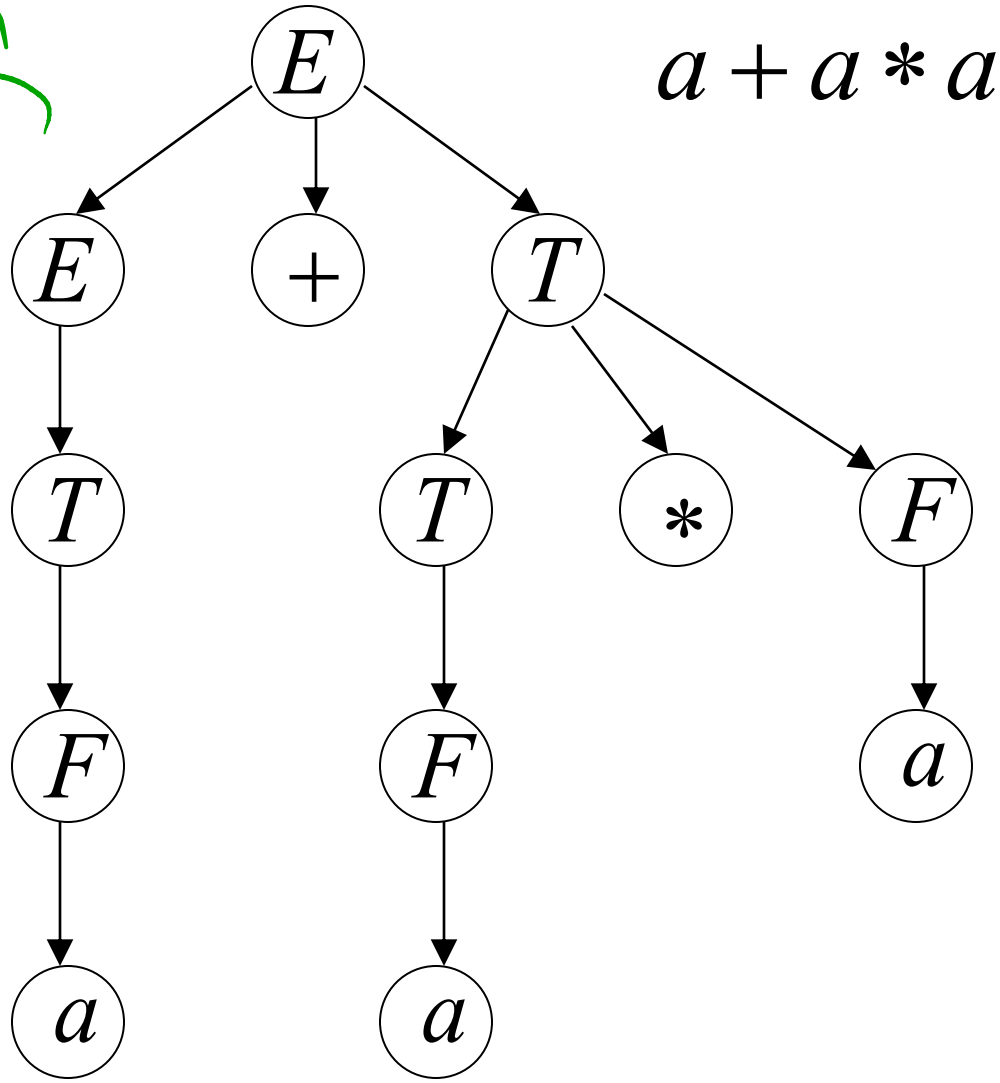
ไทยเป็นไม่สอน

am → non-am

$$E \Rightarrow E+T \Rightarrow T+T \Rightarrow F+T \Rightarrow a+T \Rightarrow a+T*F$$
$$\Rightarrow a+F*F \Rightarrow a+a*F \Rightarrow \underline{a+a*a}$$

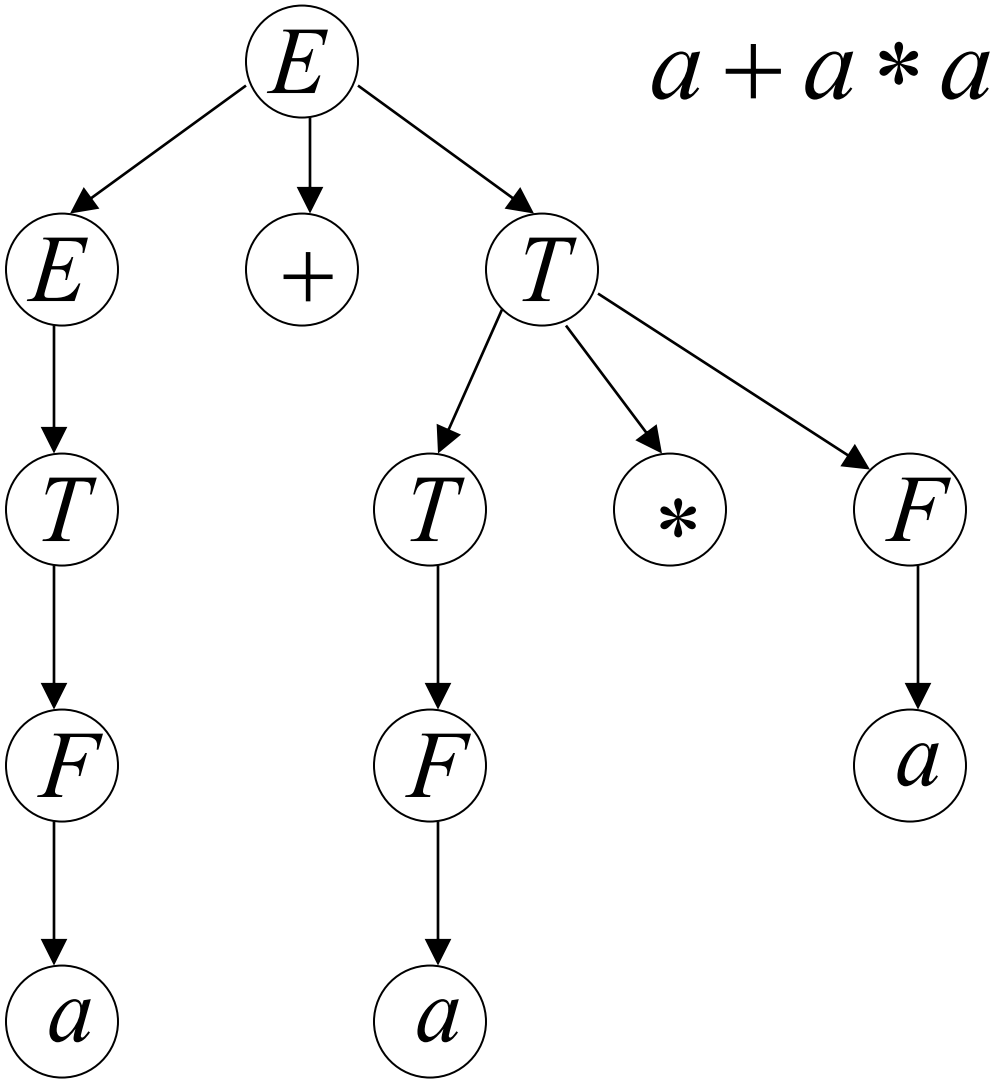
สังเกต

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



$a+a*a$

Unique derivation tree



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

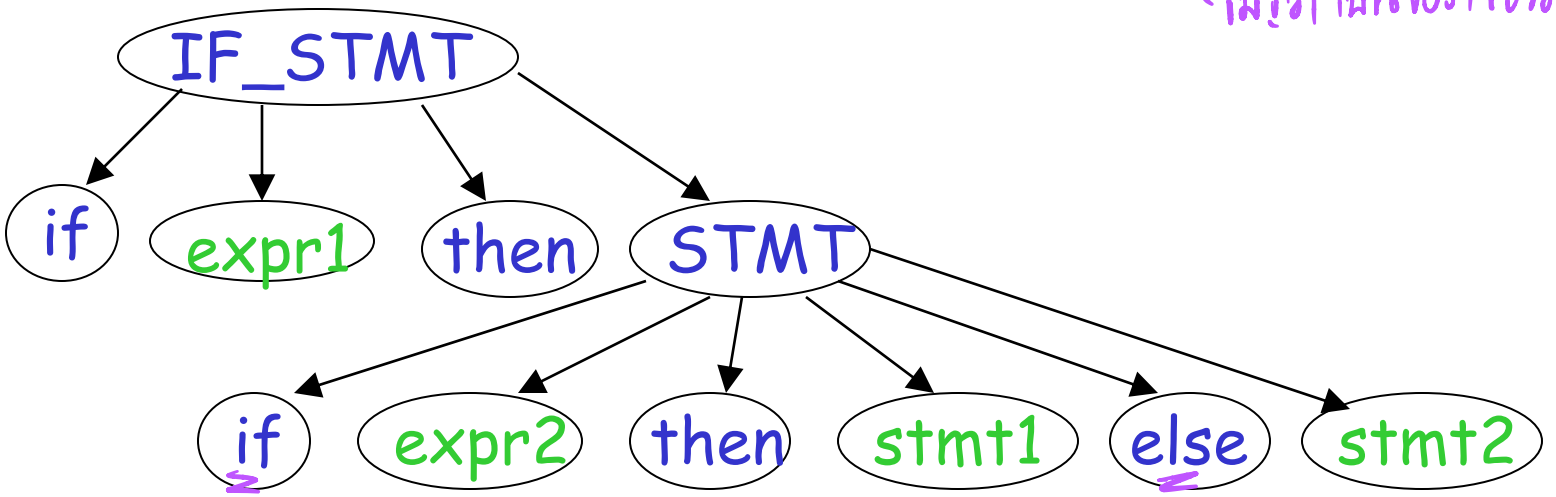
Another Ambiguous Grammar

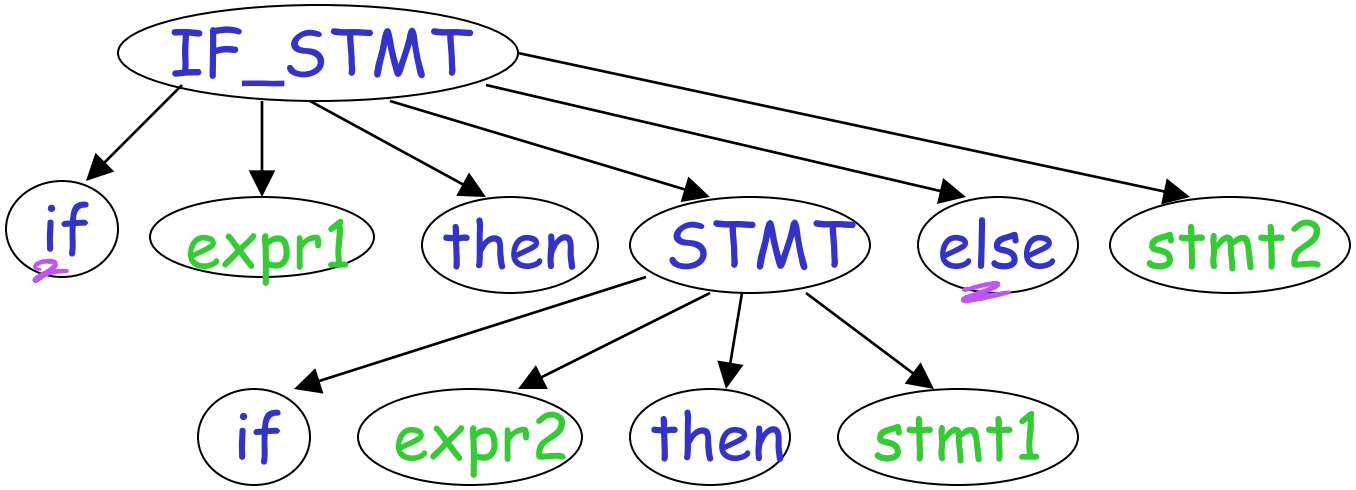
IF_STMT → if EXPR then STMT

| if EXPR then STMT else STMT

If expr1 then if expr2 then stmt1 else stmt2

ดูไม่รู้ว่า เป็นของ if ตัวไหน





Inherent Ambiguity

บาง grammar Am สร้างแต่เกิด สร้าง non-Am ไม่ได้

Some context free languages have only ambiguous grammars

Ambiguous

$a^n b^n c^n$

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

ถ้า n=m m=n

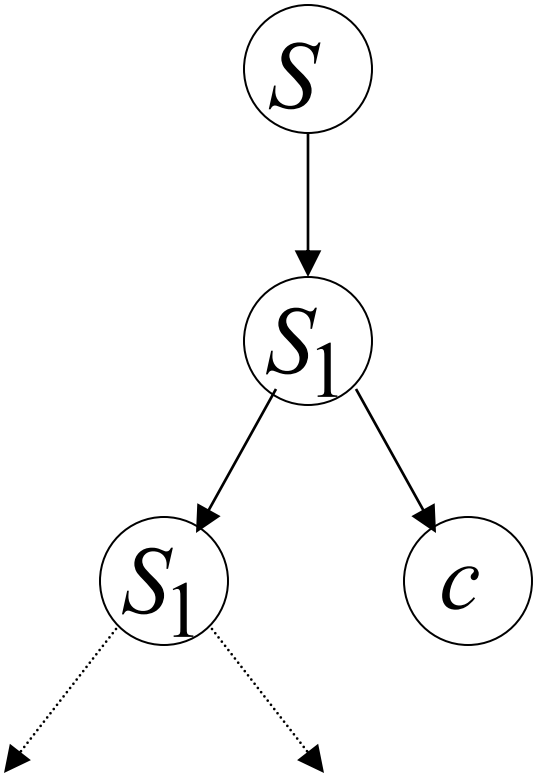
$$S \to S_1 \mid S_2$$

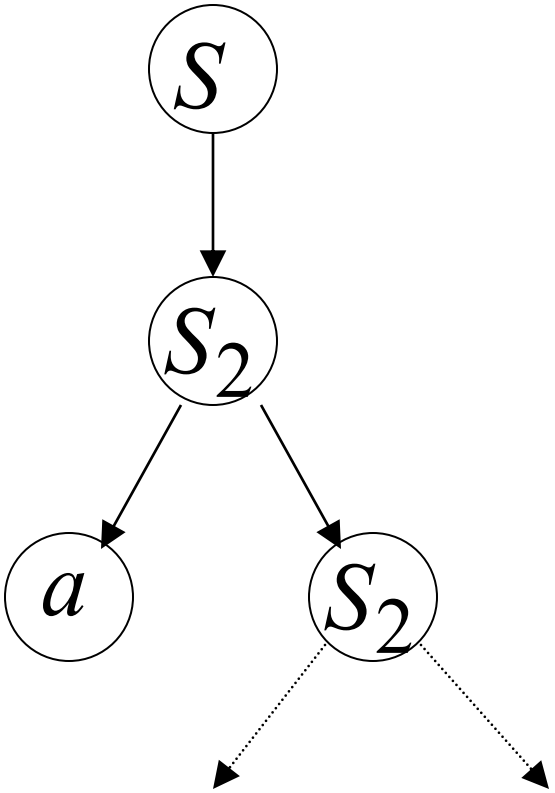
$$S_1 \to S_1 c \mid A \qquad A \to aAb \mid \lambda$$

$$S_2 \to aS_2 \mid B \qquad B \to bBc \mid \lambda$$

The string $a^n b^n c^n$ → สร้างได้ 2 Tree

has two derivation trees



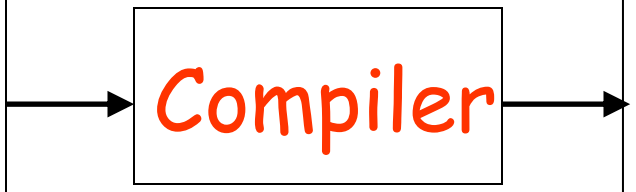


Compilers

Program

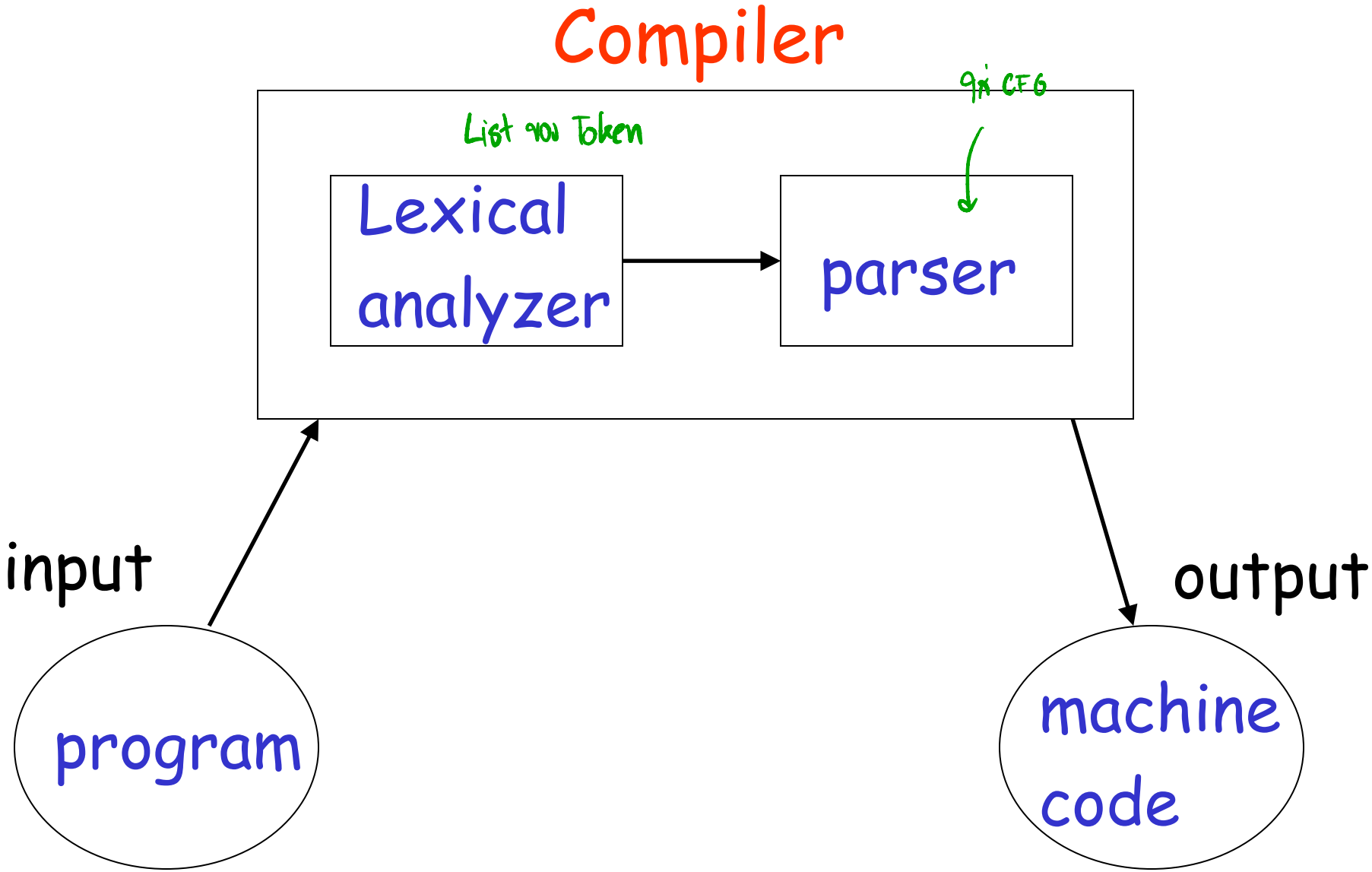
source

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



Machine Code

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


Compiler
List ของ Token
ฐาน CFG
Lexical analyzer
parser
input
output
program
machine code

สนใจ

A parser knows the grammar of the programming language

Parser

start var.

G. ของ ภาษา C

PROGRAM → STMT_LIST

STMT_LIST → STMT; STMT_LIST | STMT;

STMT → EXPR | IF_STMT | WHILE_STMT

| { STMT_LIST }

compound Statement

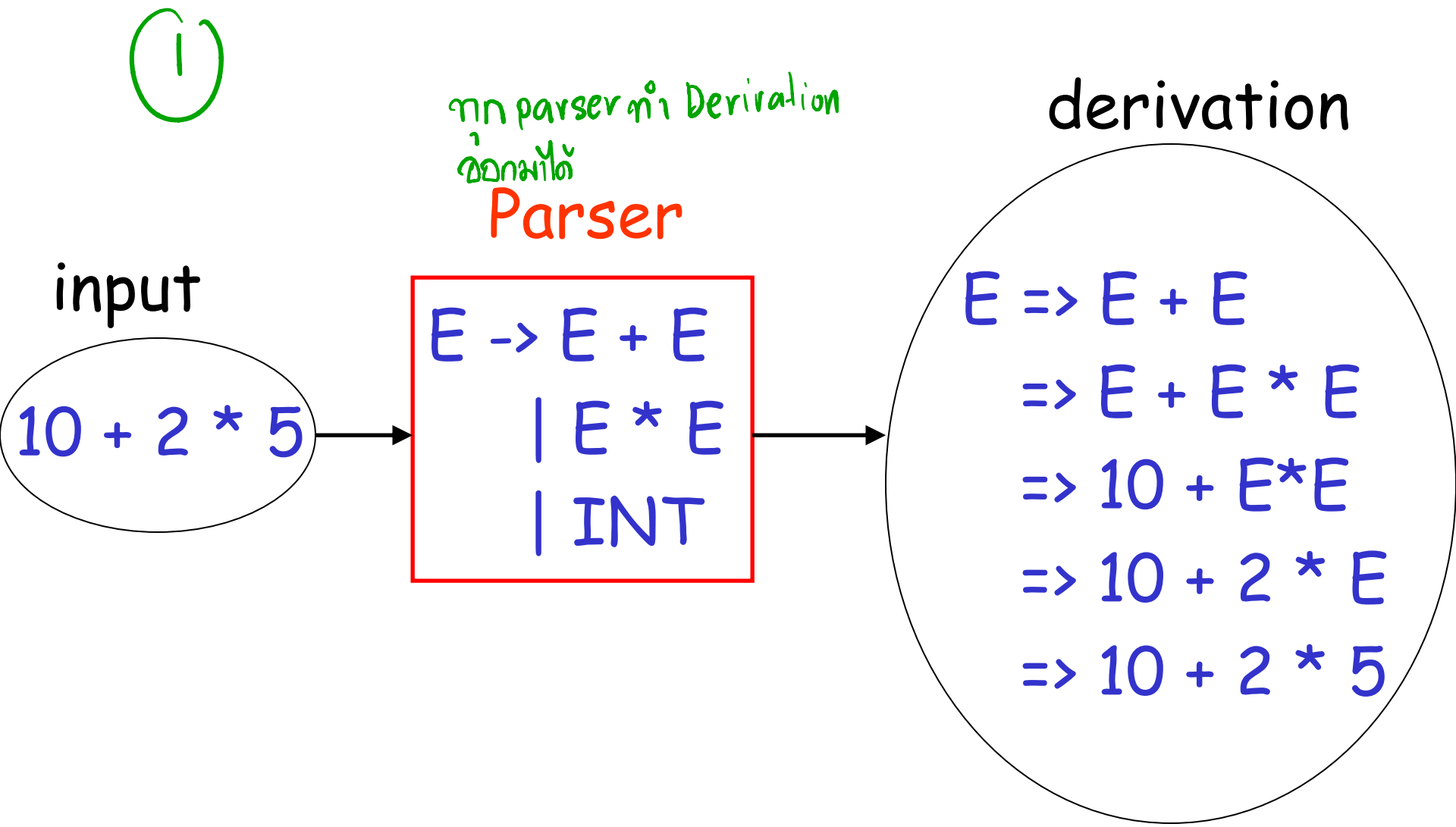
EXPR → EXPR + EXPR | EXPR - EXPR | ID

IF_STMT → if (EXPR) then STMT แต่ตัวอย่าง

| if (EXPR) then STMT else STMT

WHILE_STMT → while (EXPR) do STMT

The parser finds the derivation of a particular input

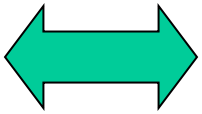


(2)

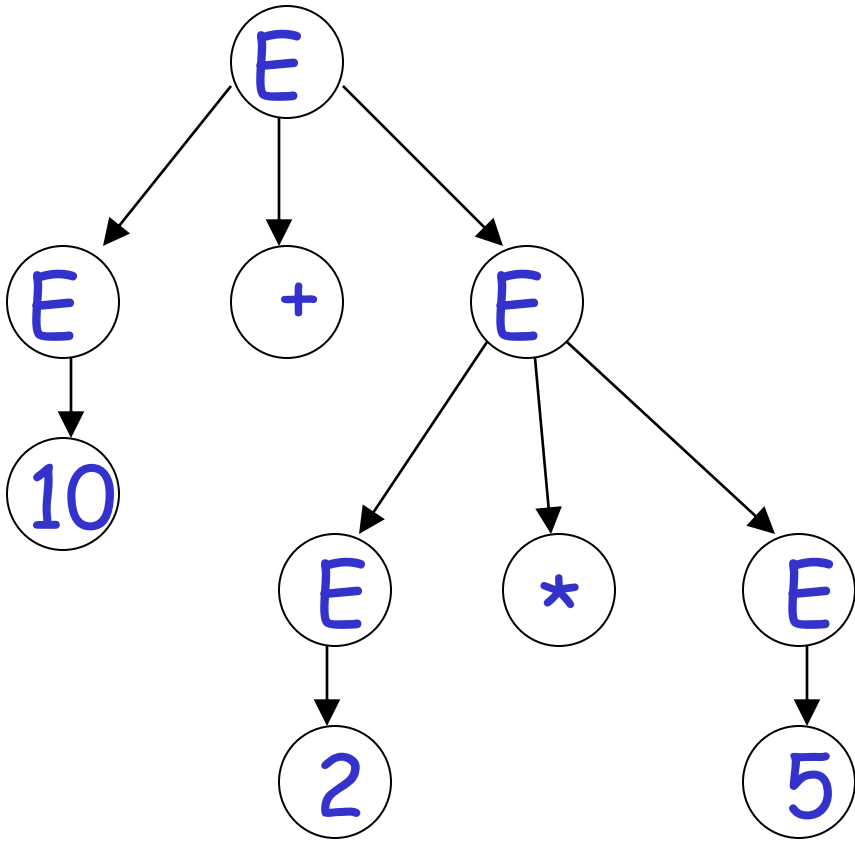
derivation

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

ได้จาก source

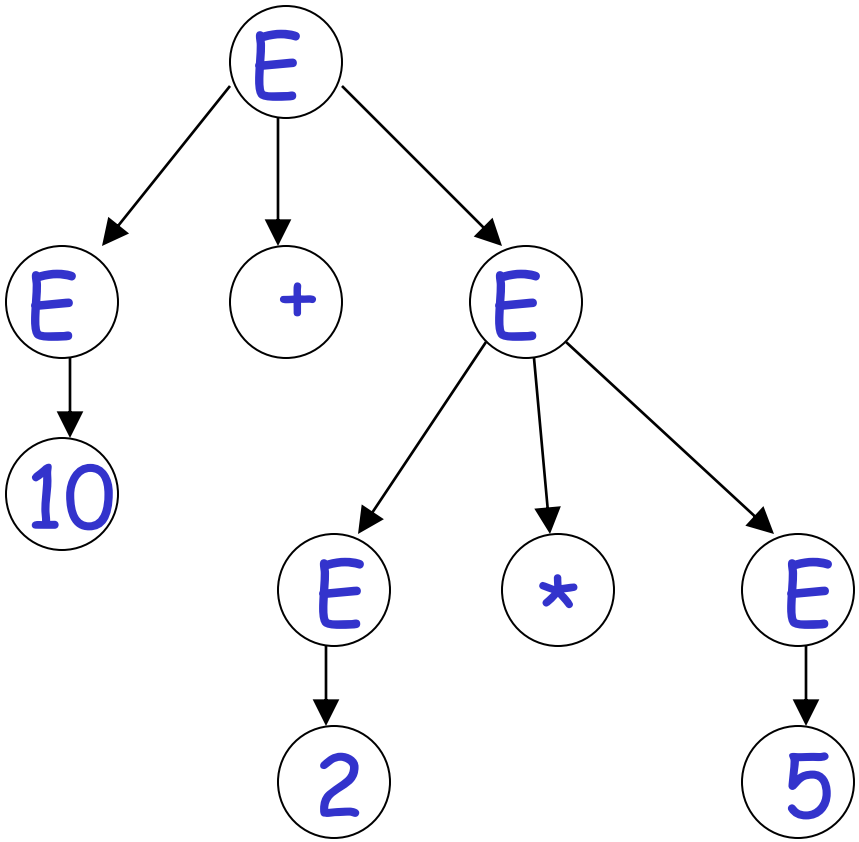


derivation tree



3

derivation tree



ถูกสร้าง มาจาก D. Tree

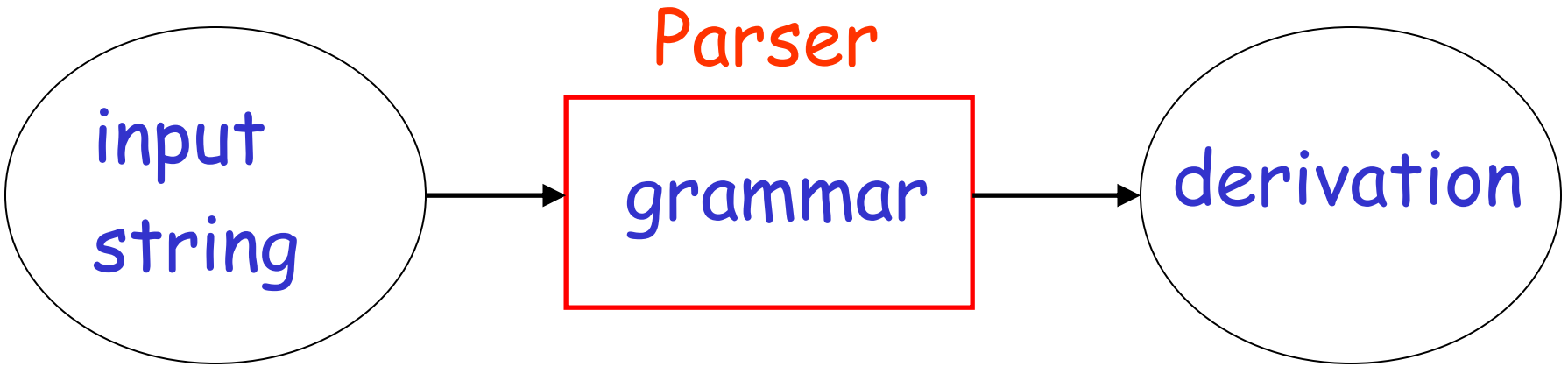
machine code

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

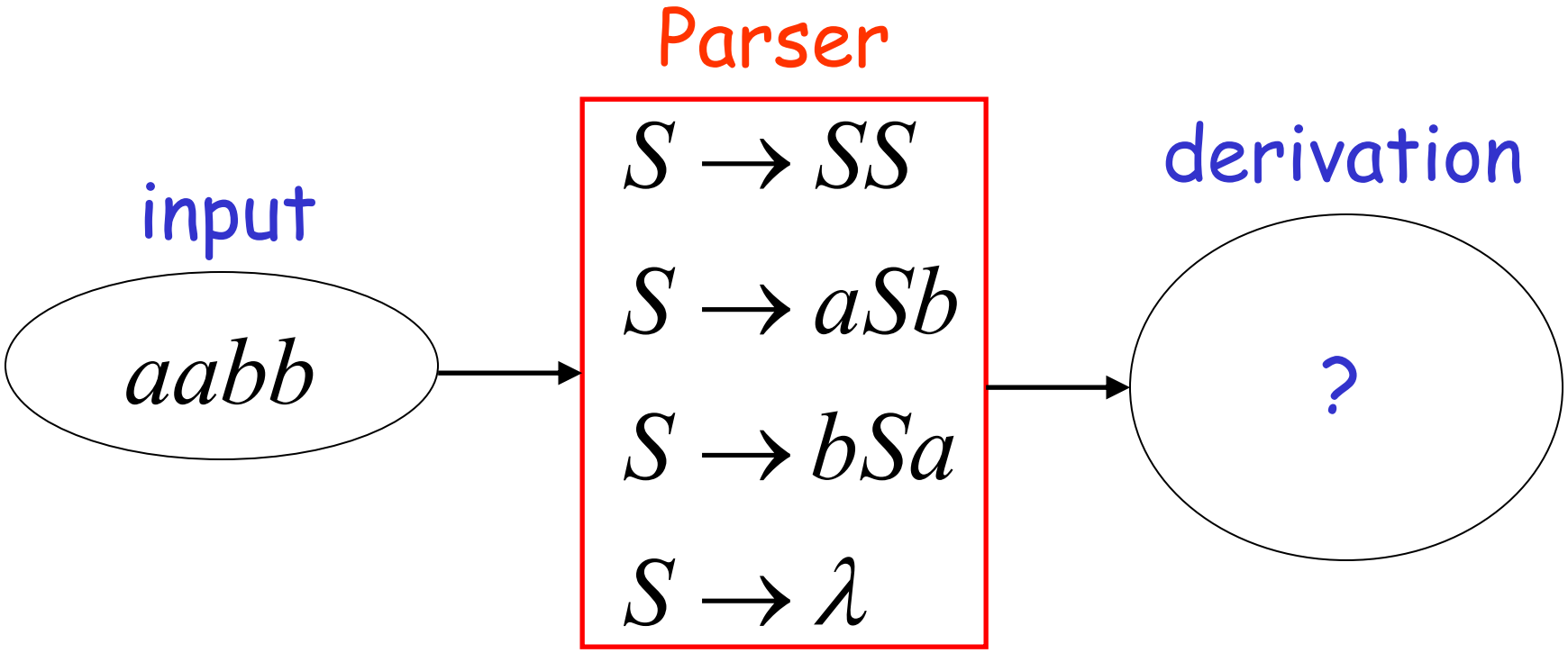
สนใจ กระบวนการ

"Parsing"

คือ กระบวนการ parse str


Parser
input string
grammar
derivation

Example:


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

Exhaustive Search

Search ทุกวิธีที่เป็นไปได้

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

Phase 1:

$$S \Rightarrow SS$$

$$S \Rightarrow aSb$$

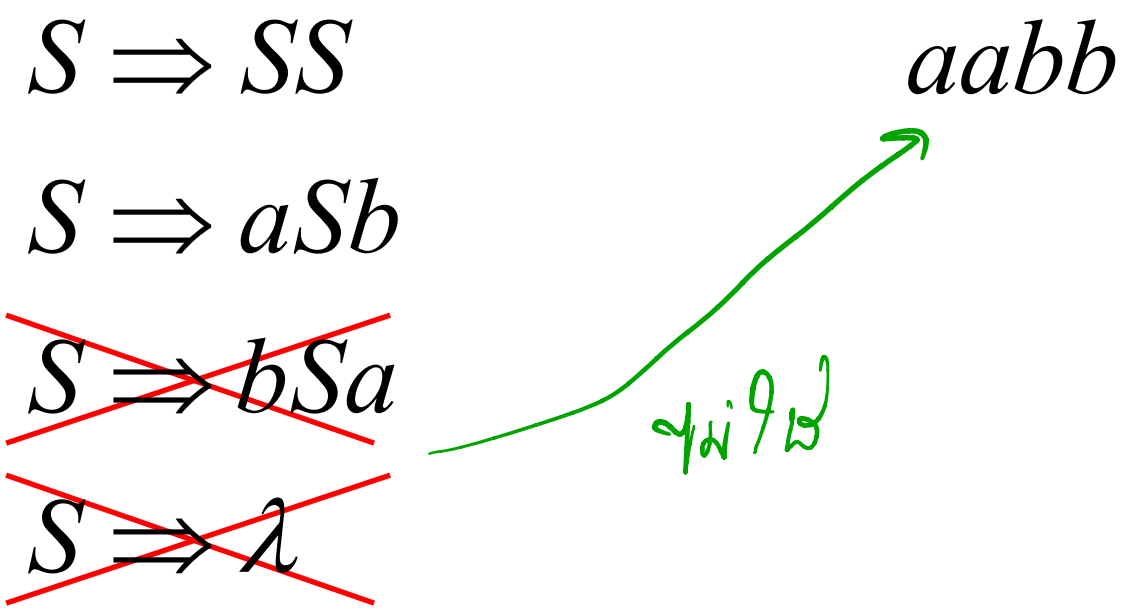
$$S \Rightarrow bSa$$

$$S \Rightarrow \lambda$$

Find derivation of

$$aabb$$

All possible derivations of length 1


$S \Rightarrow SS$
$S \Rightarrow aSb$
$S \Rightarrow bSa$
$S \Rightarrow \lambda$
$aabb$

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

ทางเลือก 1 ได้ 4 กรณี

$S \Rightarrow \underline{S}S \Rightarrow SSS$

$S \Rightarrow \underline{S}S \Rightarrow aSbS$ $aabb$

~~$S \Rightarrow \underline{S}S \Rightarrow bSaS$~~

Phase 1

$S \Rightarrow SS$

$S \Rightarrow \underline{S}S \Rightarrow S$

$S \Rightarrow aSb$

$S \Rightarrow a\underline{S}b \Rightarrow aSSb$

ทางเลือก 2 ได้ 4 กรณี

$S \Rightarrow a\underline{S}b \Rightarrow aaSbb$

~~$S \Rightarrow a\underline{S}b \Rightarrow abSab$~~

~~$S \Rightarrow a\underline{S}b \Rightarrow ab$~~

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

Phase 2

$S \Rightarrow SS \Rightarrow SSS$ ← ผกพ

$S \Rightarrow SS \Rightarrow aSbS$ ← ผ กพ

$S \Rightarrow SS \Rightarrow S$ ← ผ กพ

$aabb$

$S \Rightarrow aSb \Rightarrow aSSb$ ← ผ กพ

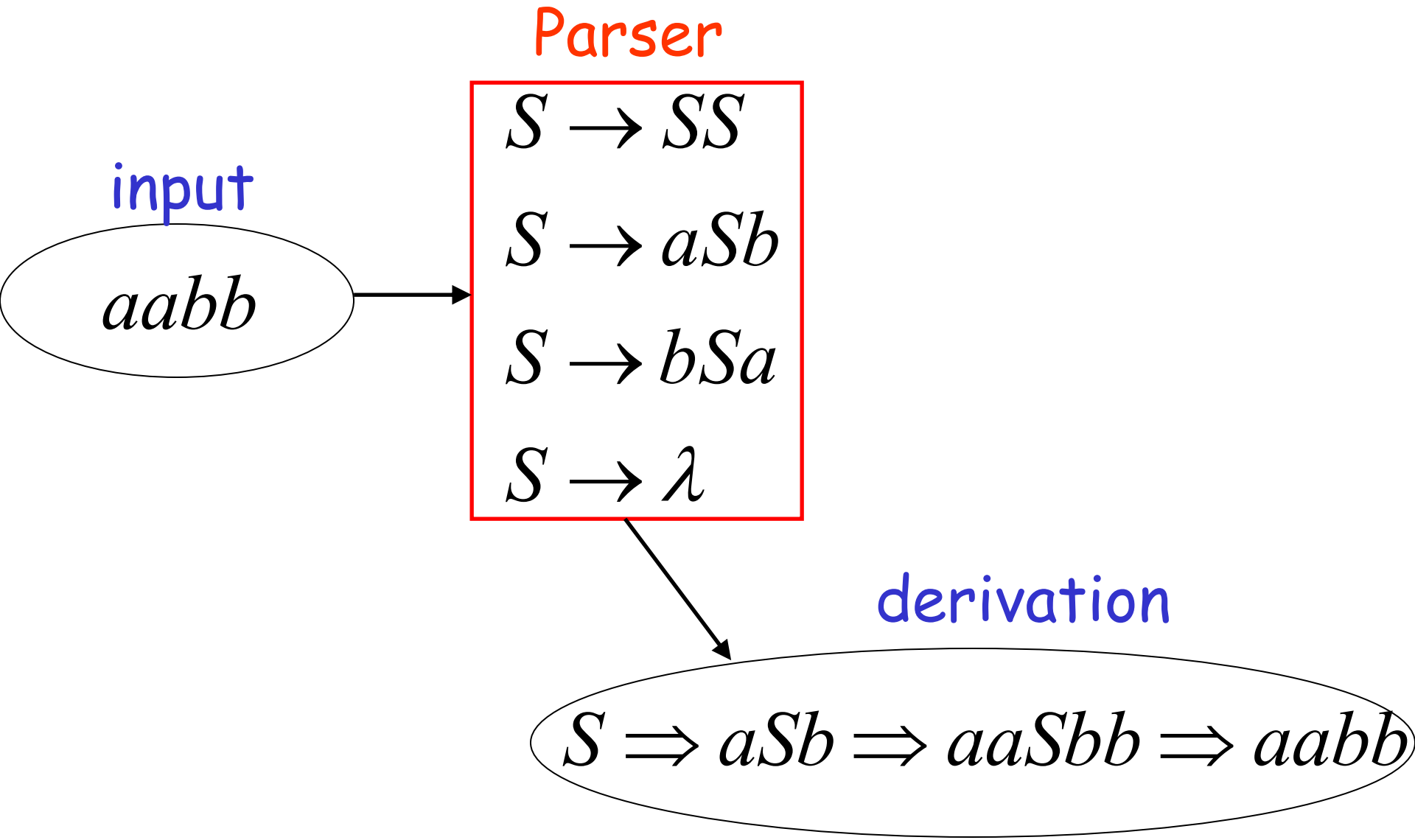
$S \Rightarrow aSb \Rightarrow aaSbb$ ← 3 กพ กพที่ ผ

ทา ตอที่ละที่

Phase 3

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

Final result of exhaustive search (top-down parsing)



Time complexity of exhaustive search

Suppose there are no productions of the form

$$A \to \lambda$$

$$A \to B$$

จำนวนขั้น

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

(ขั้นตอนทำ อย่างนี้ซ้ำ)

$S \xrightarrow{1} A \xrightarrow{2} a$ → จำนวน phase

$n(a) = 1$

For grammar with k rules

↳ production

Time for phase 1: k

k possible derivations

ตัดทางเลือกที่ไม่ได้

Time for phase 2: k^2

k^2 possible derivations

phase 3 k^3

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

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

Total time needed for string w:

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

= เวลาที่เราใช้
" เยอะมาก "

phase 1

phase 2

phase 2|w|

Extremely bad!!!

There exist faster algorithms for specialized grammars

input a ได้, input b ไม่ได้ ต้องออกเลย

S-grammar:

$$A \rightarrow ax$$

เรียกว่า E. search

symbol

string of variables

Pair (A, a) appears once

S-grammar example:

ไม่ใช่ทุกภาษาที่เขียนเป็น S-grammar ได้

$$S \to aS$$

$$S \to bSS$$

$$S \to c$$

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)