BLM2502 Theory of Computation

Spring 2015

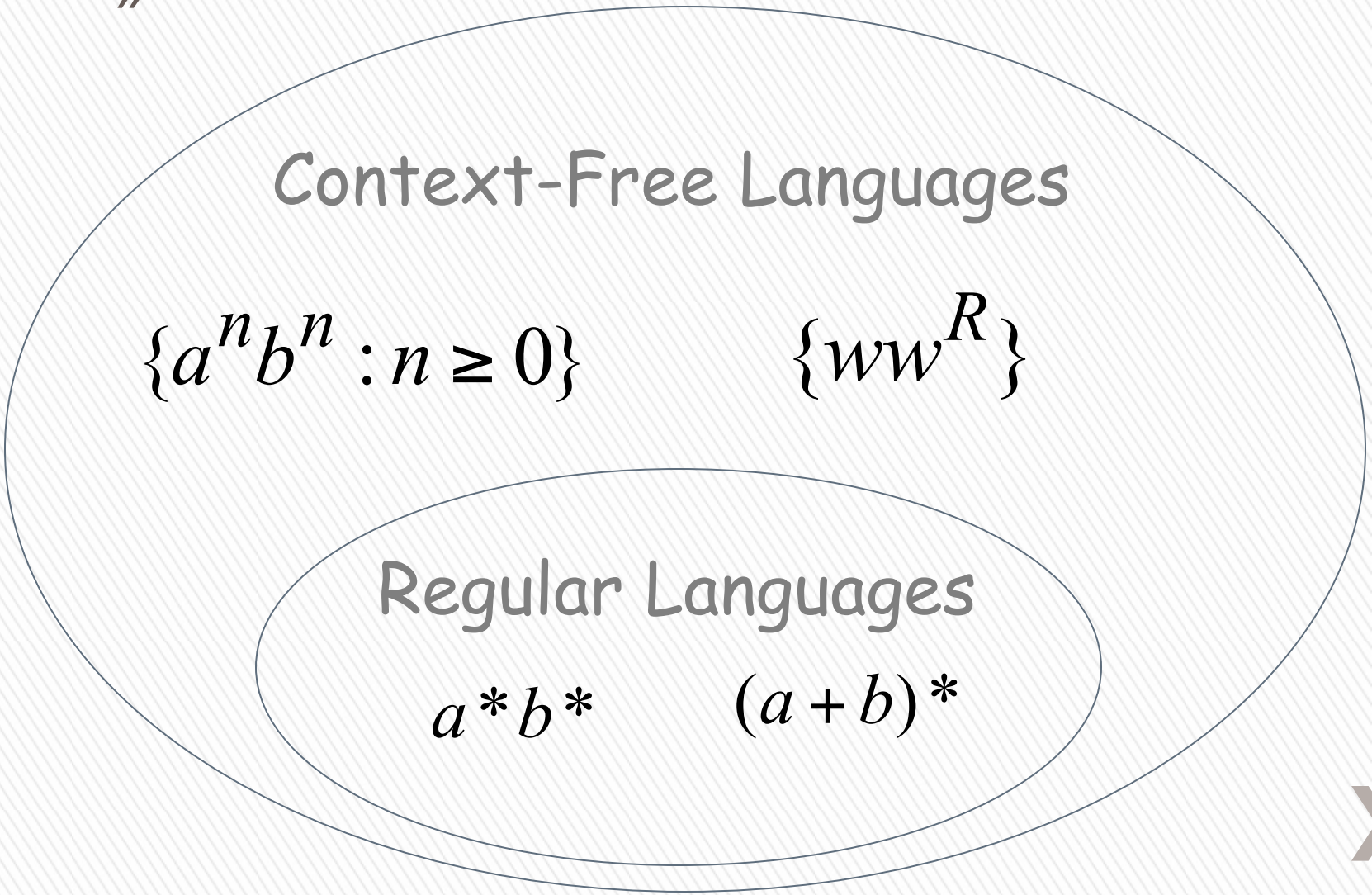
BLM2502 Theory of Computation

- **Course Outline**

Week	Content
1	Introduction to Course
2	Computability Theory, Complexity Theory, Automata Theory, Set Theory, Relations, Proofs, Pigeonhole Principle
3	Regular Expressions
4	Finite Automata
5	Deterministic and Nondeterministic Finite Automata
6	Epsilon Transition, Equivalence of Automata
7	Pumping Theorem
8	April 10 - 14 week is the first midterm week
9	**Context Free Grammars**
10	**Parse Tree, Ambiguity,**
11	Pumping Theorem
12	Turing Machines, Recognition and Computation, Church-Turing Hypothesis
13	Turing Machines, Recognition and Computation, Church-Turing Hypothesis
14	May 22 – 27 week is the second midterm week
15	Review
16	Final Exam date will be announced

Context-Free Languages

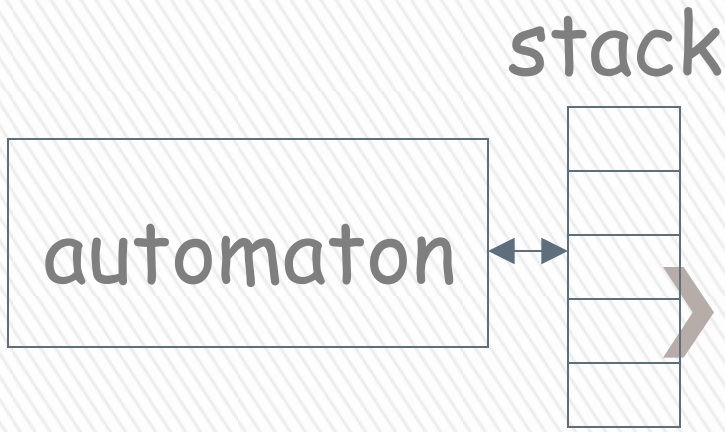
»


Context-Free Languages
$\{a^n b^n : n \geq 0\}$
$\{ww^R\}$
Regular Languages
$a*b*$
$(a+b)*$

Context-Free Languages

Context-Free Grammars

Pushdown Automata



Context-Free Grammars

Grammars

- Grammars express languages
- Example: the English language grammar

$$\langle sentence \rangle \rightarrow \langle noun\_phrase \rangle \ \langle predicate \rangle$$

$$\langle noun\_phrase \rangle \rightarrow \langle article \rangle \ \langle noun \rangle$$

$$\langle predicate \rangle \rightarrow \langle verb \rangle$$

$$\langle article \rangle \rightarrow a$$

$$\langle article \rangle \rightarrow the$$

$$\langle noun \rangle \rightarrow cat$$

$$\langle noun \rangle \rightarrow dog$$

$$\langle verb \rangle \rightarrow runs$$

$$\langle verb \rangle \rightarrow sleeps$$

» Derivation of string "the dog walks" :

$$\begin{aligned}
\langle sentence \rangle &\Rightarrow \langle noun_phrase \rangle \ \langle predicate \rangle \\
&\Rightarrow \langle noun_phrase \rangle \ \langle verb \rangle \\
&\Rightarrow \langle article \rangle \ \langle noun \rangle \ \langle verb \rangle \\
&\Rightarrow the \ \langle noun \rangle \ \langle verb \rangle \\
&\Rightarrow the \ dog \ \langle verb \rangle \\
&\Rightarrow the \ dog \ sleeps
\end{aligned}$$

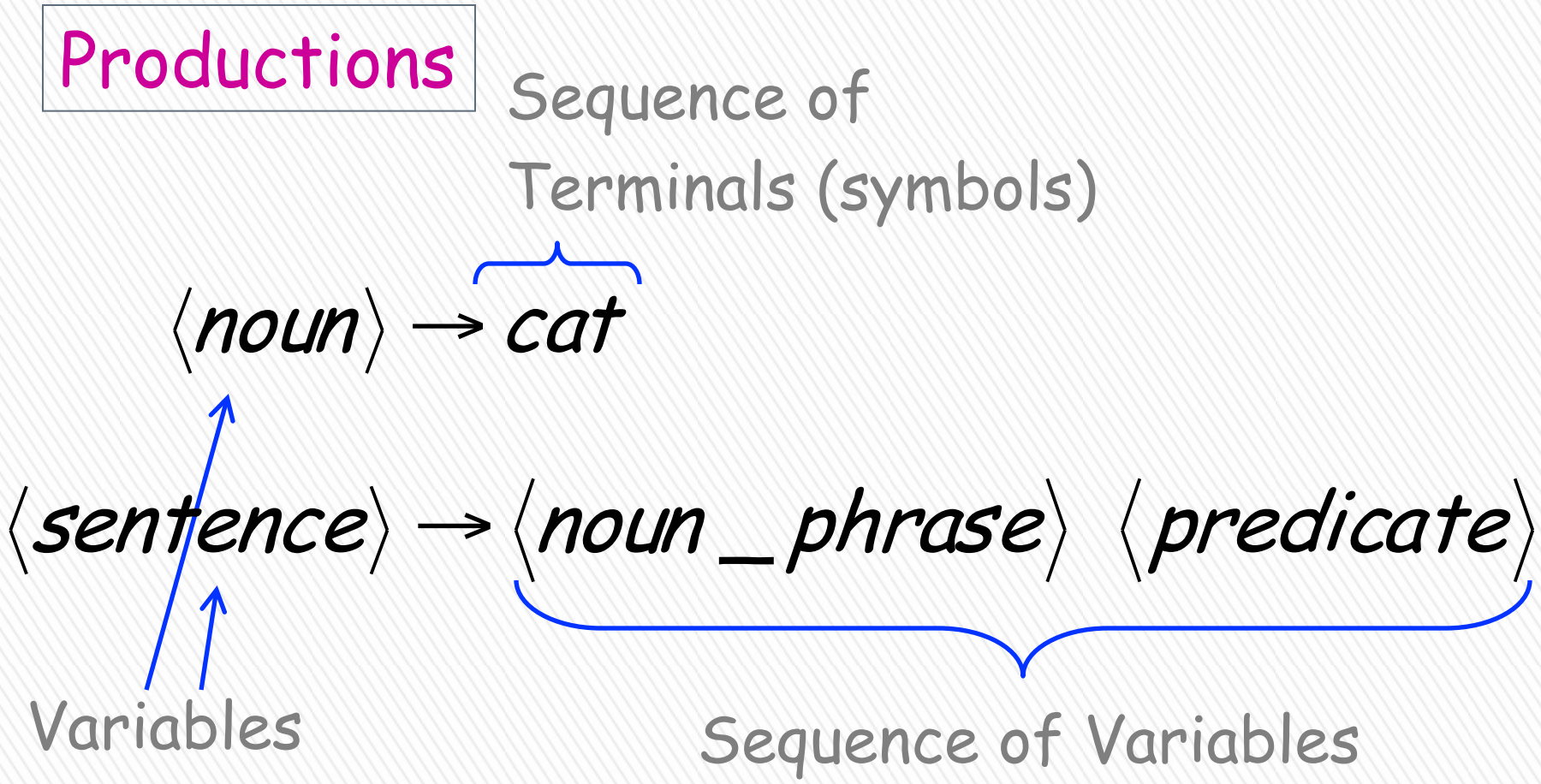
» Derivation of string "a cat runs" :

$$\begin{aligned}
\langle sentence \rangle &\Rightarrow \langle noun_phrase \rangle \ \langle predicate \rangle \\
&\Rightarrow \langle noun_phrase \rangle \ \langle verb \rangle \\
&\Rightarrow \langle article \rangle \ \langle noun \rangle \ \langle verb \rangle \\
&\Rightarrow a \ \langle noun \rangle \ \langle verb \rangle \\
&\Rightarrow a \ \ cat \ \langle verb \rangle \\
&\Rightarrow a \ \ cat \ \ runs
\end{aligned}$$

» Language of the grammar:

L = { “a cat runs”,
“a cat sleeps”,
“the cat runs”,
“the cat sleeps”,
“a dog runs”,
“a dog sleeps”,
“the dog runs”,
“the dog sleeps” }

Productions



Another Example

Sequence of terminals and variables

Grammar:

$$S \rightarrow aSb$$

$$S \rightarrow \varepsilon$$

Variable

The right side may be ε

» Grammar:

$$S \rightarrow aSb$$

$$S \rightarrow \varepsilon$$

» Derivation of string ab:

$$S \Rightarrow aSb \Rightarrow ab$$

$S \rightarrow aSb$ $\qquad$ $S \rightarrow \varepsilon$

» Grammar:

$$S \rightarrow aSb$$

$$S \rightarrow \varepsilon$$

» Derivation of string :

$$aabb$$

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

$$S \rightarrow aSb$$

$$S \rightarrow \varepsilon$$

Grammar: $S \rightarrow aSb$

$S \rightarrow \varepsilon$

Other derivations:

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

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

$$\Rightarrow aaaaSbbbb \Rightarrow aaaabbbb$$

Grammar:

$$S \rightarrow aSb$$

$$S \rightarrow \varepsilon$$

Language of the grammar:

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

A Convenient Notation

» We write: $S \overset{*}{\Rightarrow} aaabbb$

for one or more derivation steps

» Instead of:

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

In general we write:

$$w_1 \overset{*}{\Rightarrow} w_n$$

If:

$$w_1 \Rightarrow w_2 \Rightarrow w_3 \Rightarrow \cdots \Rightarrow w_n$$

in zero or more derivation steps

Trivially:

$$w \overset{*}{\Rightarrow} w$$

Example Grammar

$$S \to aSb$$

$$S \to \varepsilon$$

Possible Derivations

$$S \overset{*}{\Rightarrow} \varepsilon$$

$$S \overset{*}{\Rightarrow} ab$$

$$S \overset{*}{\Rightarrow} aaabbb$$

$$S \overset{*}{\Rightarrow} aaSbb \overset{*}{\Rightarrow} aaaaaSbbbbb$$

Another convenient notation:

$$S \to aSb$$
$$S \to \varepsilon$$

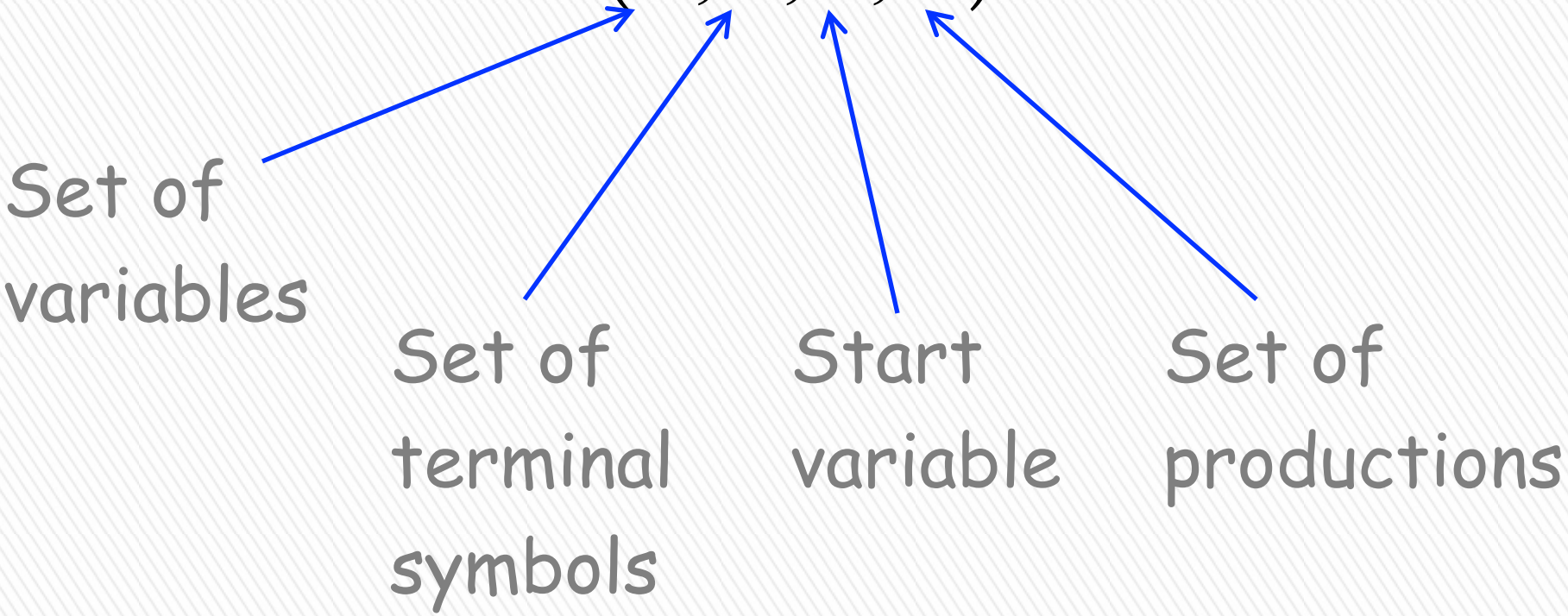
$$S \to aSb \mid \varepsilon$$

$$\langle article \rangle \to a$$
$$\langle article \rangle \to the$$

$$\langle article \rangle \to a \mid the$$

Formal Definition

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



Context-Free Grammar: $G = (V, T, S, P)$

All productions in P are of the form

$$A \rightarrow s$$

Variable

String of variables and terminals

Example for Context-Free Grammar

$$S \rightarrow aSb \mid \varepsilon$$

productions

$$P = \{S \rightarrow aSb,\ \ S \rightarrow \varepsilon\}$$

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

$V = \{S\}$

variables

$T = \{a, b\}$

terminals

start variable

Language of a Grammar:

» For a grammar G with start variable S

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

String of terminals or ε

Example:

context-free grammar G : $S \to aSb \mid \varepsilon$

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

Since, there is derivation

$$S \overset{*}{\Rightarrow} a^n b^n \quad \text{for any } n \geq 0$$

Context-Free Language:

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

Example:

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

is a context-free language

since context-free grammar G :

$$S \rightarrow aSb \mid \varepsilon$$

generates $L(G) = L$

Another Example

Context-free grammar G :

$$S \to aSa \mid bSb \mid \varepsilon$$

Example derivations:

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

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

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

Palindromes of even length

Another Example

Context-free grammar G:

$$S \rightarrow aSb \mid SS \mid \varepsilon$$

Example derivations:

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

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

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

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

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

Describes matched parentheses:

() ((( ))) (( )) $a = ($, b =)

Derivation Trees

Derivation Order

Consider the following example grammar with 5 productions:

1. $S \rightarrow AB$
2. $A \rightarrow aaA$
3. $A \rightarrow \varepsilon$
4. $B \rightarrow Bb$
5. $B \rightarrow \varepsilon$

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

3. $A \rightarrow \varepsilon$ 5. $B \rightarrow \varepsilon$

Leftmost derivation order of string *aab*:

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

At each step, we substitute the leftmost variable

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

3. $A \to \varepsilon$ 5. $B \to \varepsilon$

Rightmost derivation order of string *aab*:

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

At each step, we substitute the rightmost variable

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

3. $A \to \varepsilon$ 5. $B \to \varepsilon$

Leftmost derivation of *aab*:

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

Rightmost derivation of *aab* :

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

Consider the same example grammar:

$$S \to AB \qquad A \to aaA \mid \varepsilon \quad B \to Bb \mid \varepsilon$$

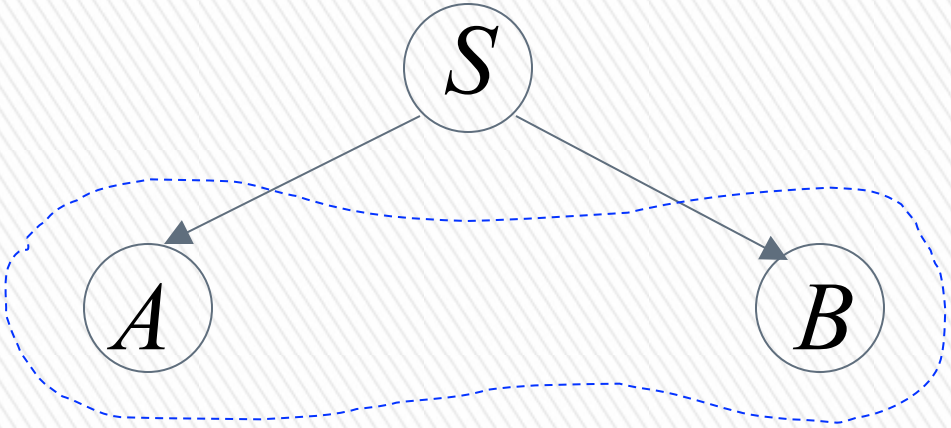
And a derivation of *aab*:

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

Derivation Tree

$$S \rightarrow AB \qquad A \rightarrow aaA \mid \varepsilon \quad B \rightarrow Bb \mid \varepsilon$$

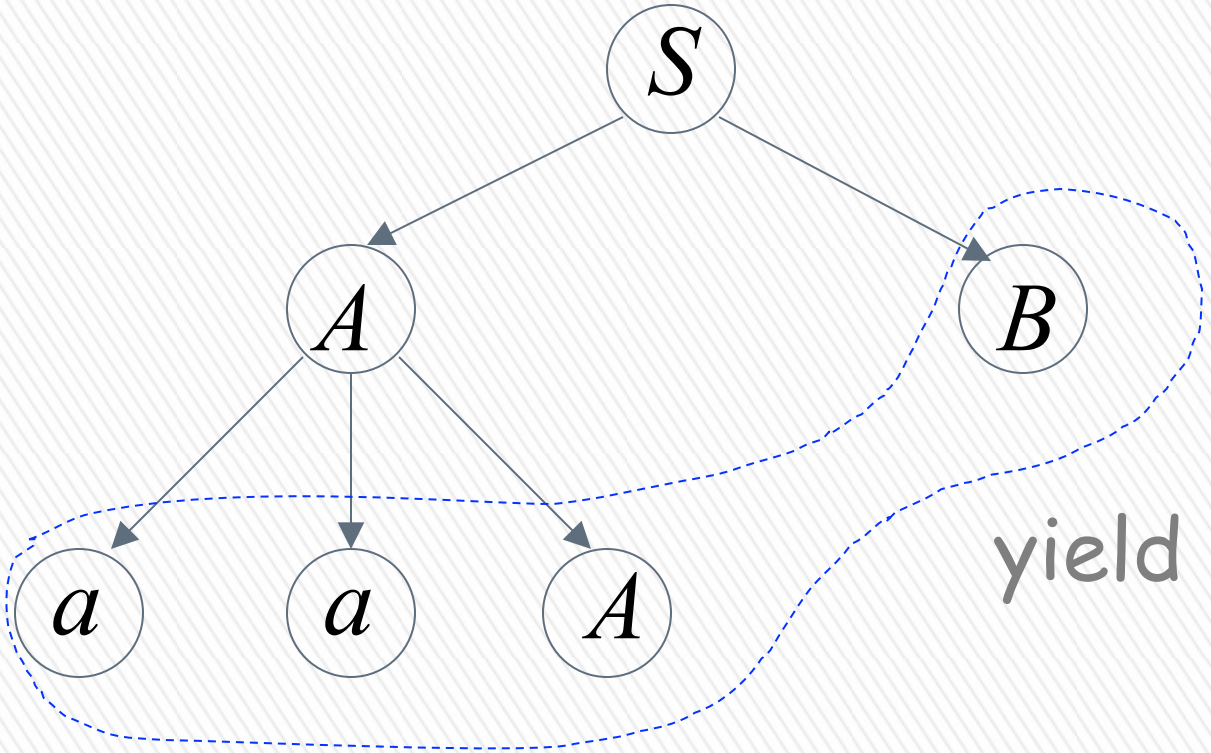
$$S \Rightarrow AB$$



yield AB

$$S \rightarrow AB \qquad A \rightarrow aaA \mid \varepsilon \quad B \rightarrow Bb \mid \varepsilon$$

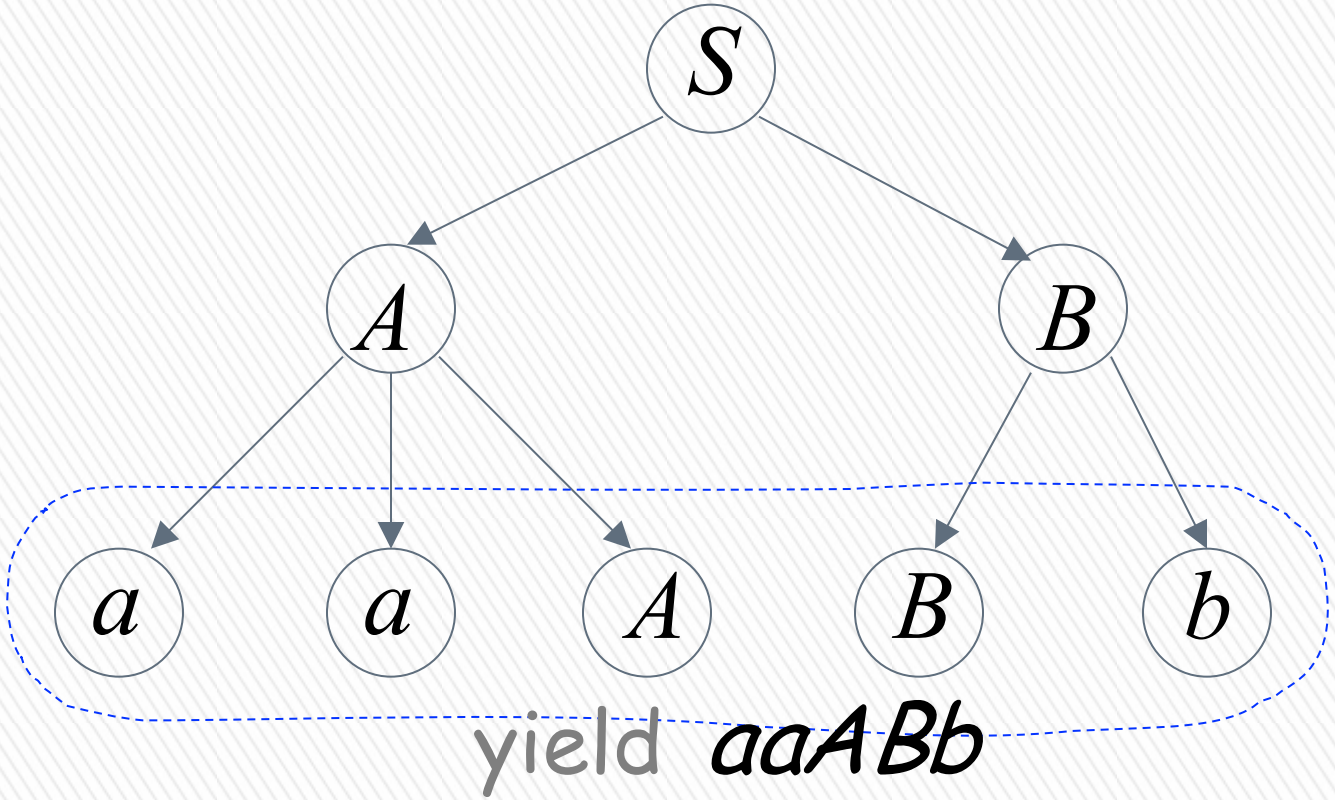
$$S \Rightarrow AB \Rightarrow aaAB$$



yield *aaAB*

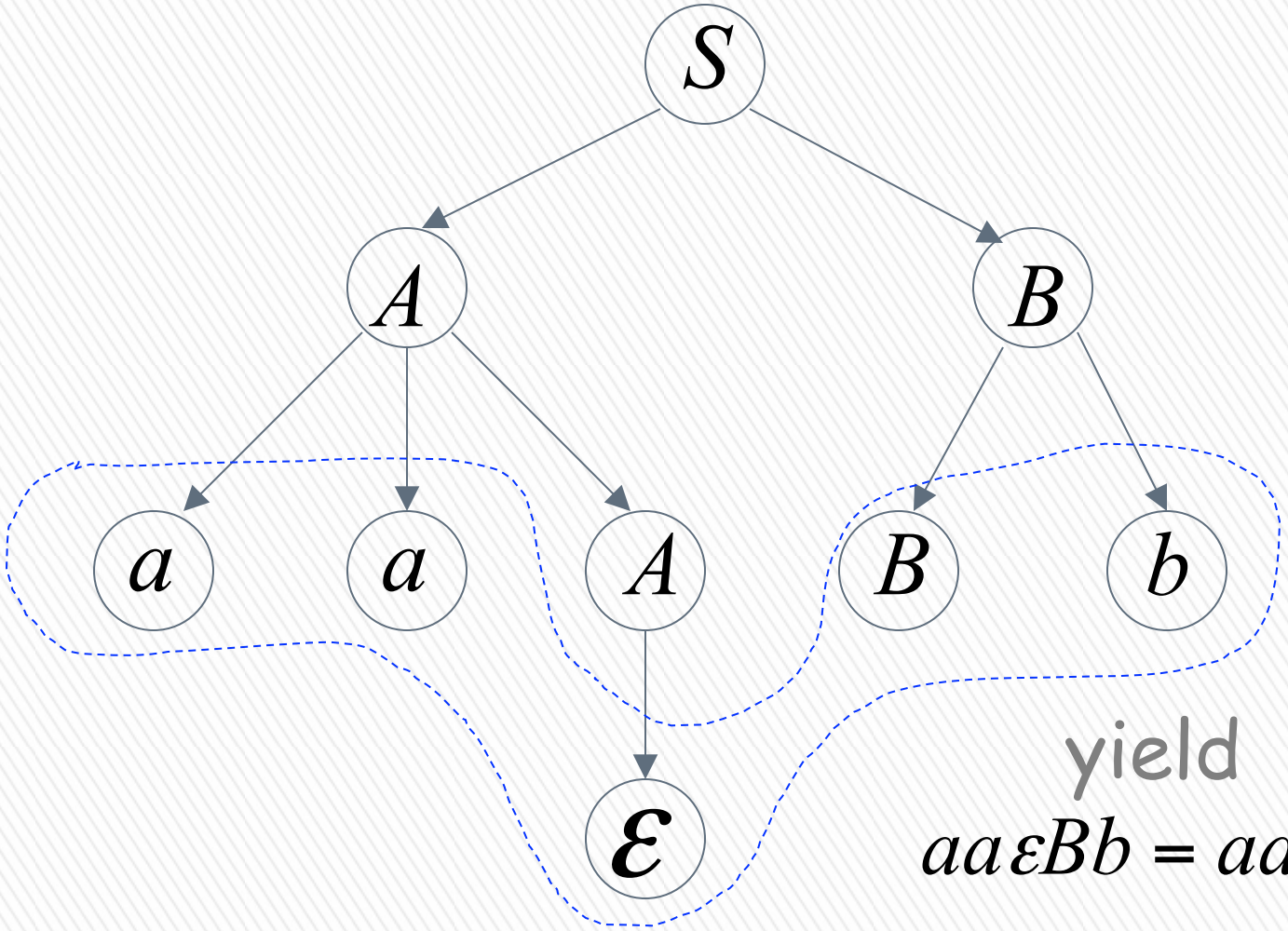
$$S \rightarrow AB \qquad A \rightarrow aaA \mid \varepsilon \quad B \rightarrow Bb \mid \varepsilon$$

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



$$S \rightarrow AB \qquad A \rightarrow aaA \mid \varepsilon \quad B \rightarrow Bb \mid \varepsilon$$

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



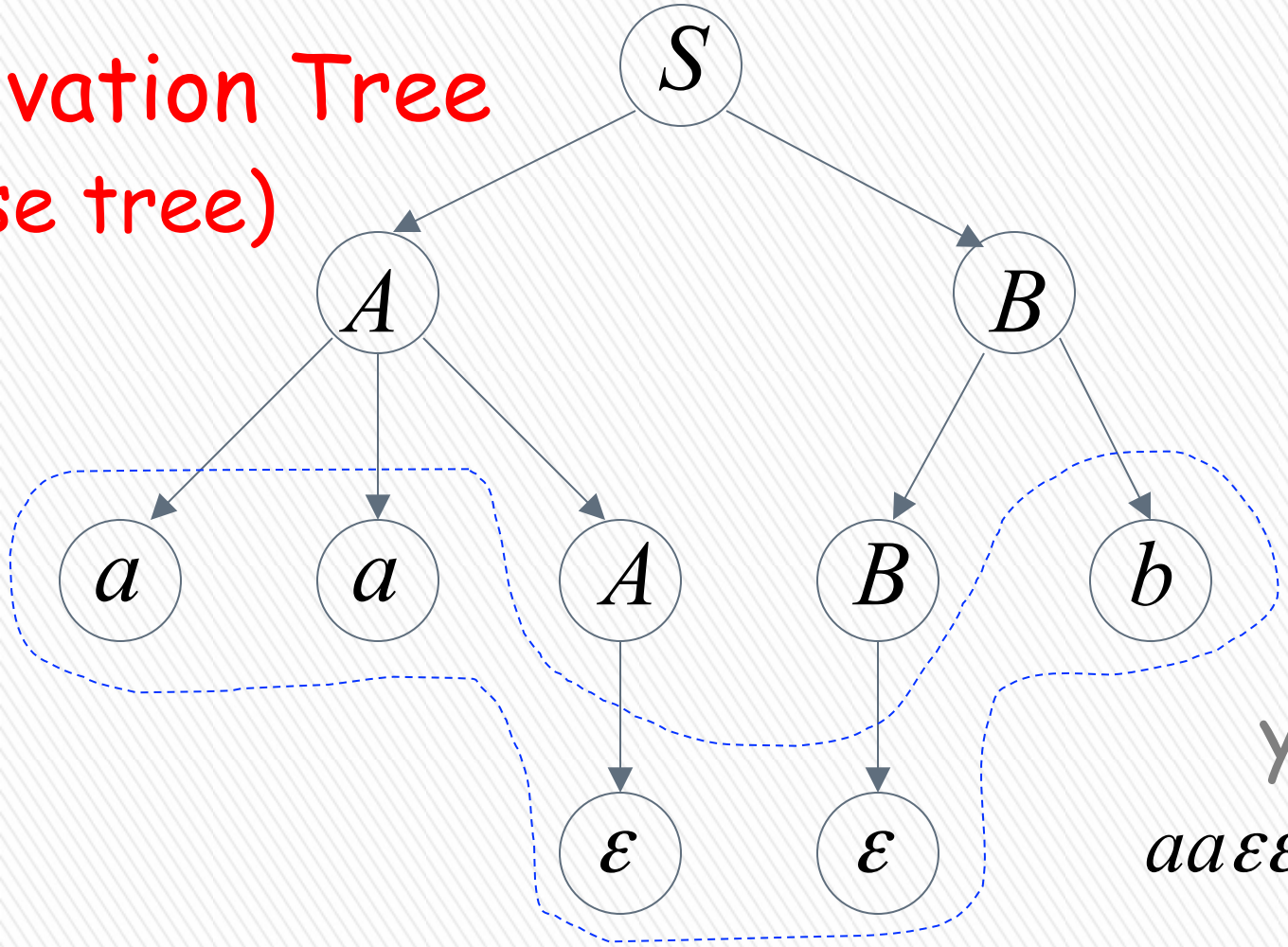
yield

$aa\varepsilon Bb = aaBb$

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

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

Derivation Tree (parse tree)



yield

$aa\varepsilon\varepsilon b = aab$

Sometimes, derivation order doesn't matter

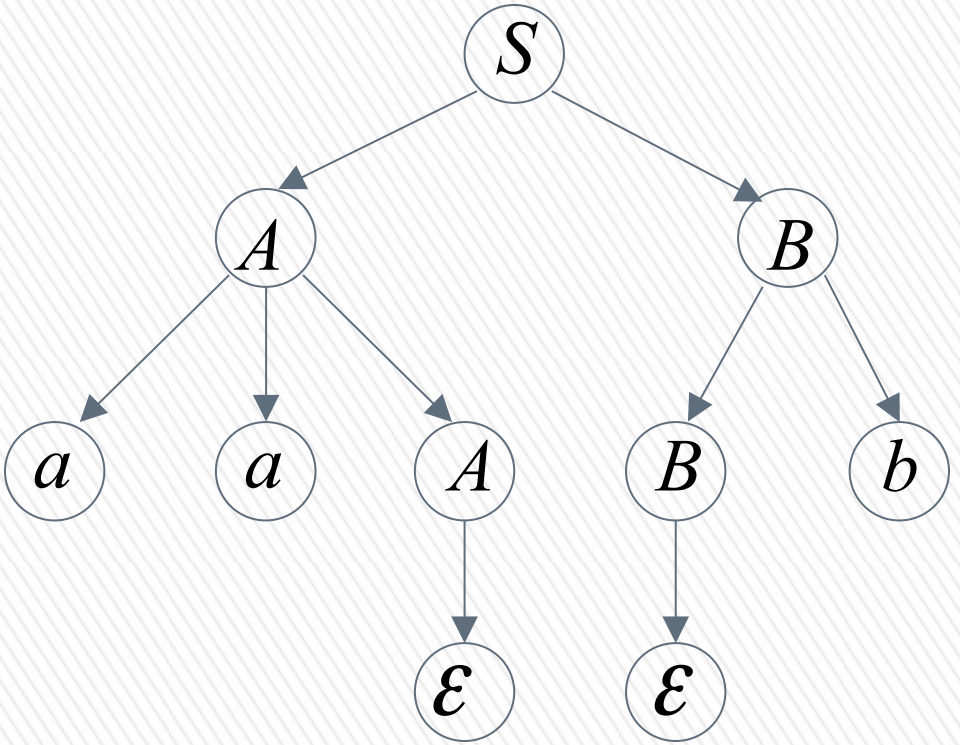
Leftmost derivation:

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

Rightmost derivation:

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

Give same derivation tree



Ambiguity

Grammar for mathematical expressions

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

Example strings:

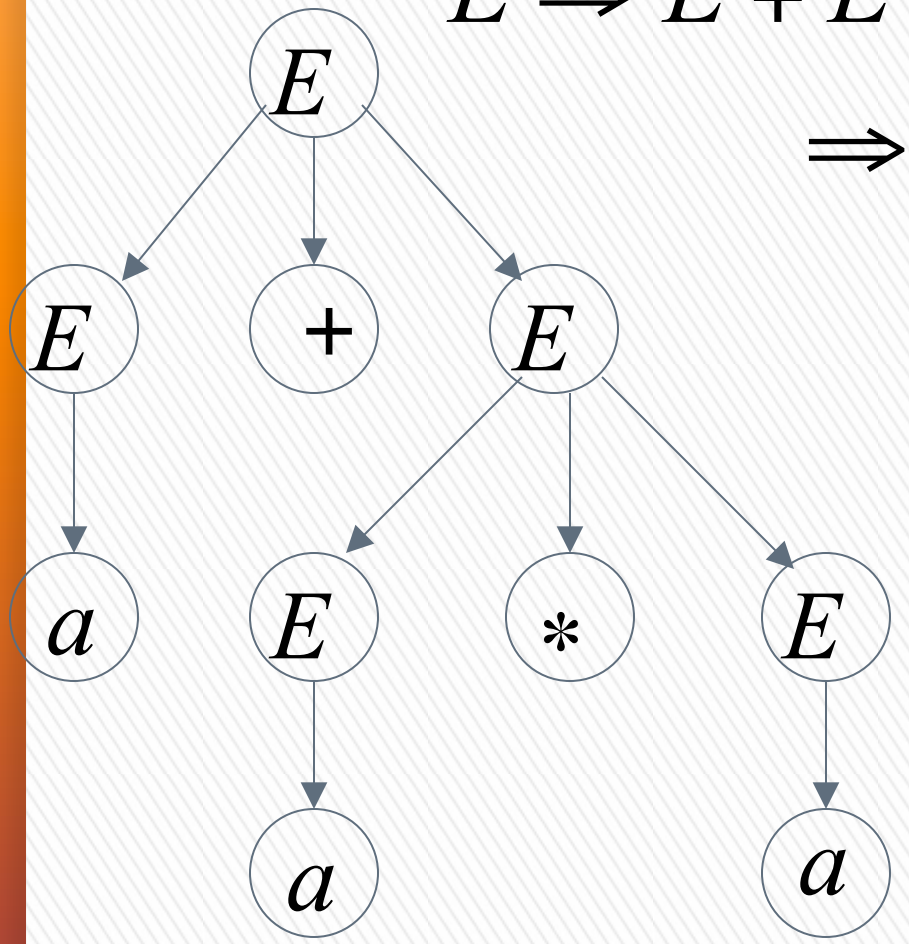
$$(a + a) * a + (a + a * (a + a))$$

Denotes any number

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

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

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



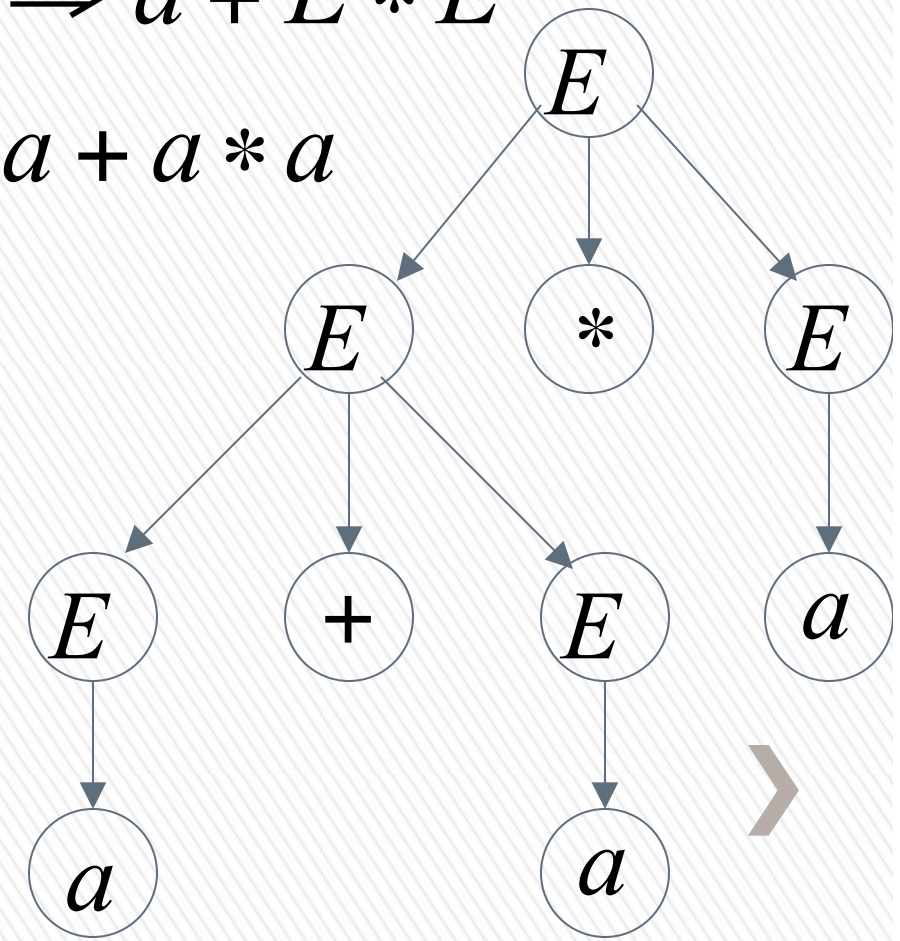
A leftmost derivation for $a + a * a$

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

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

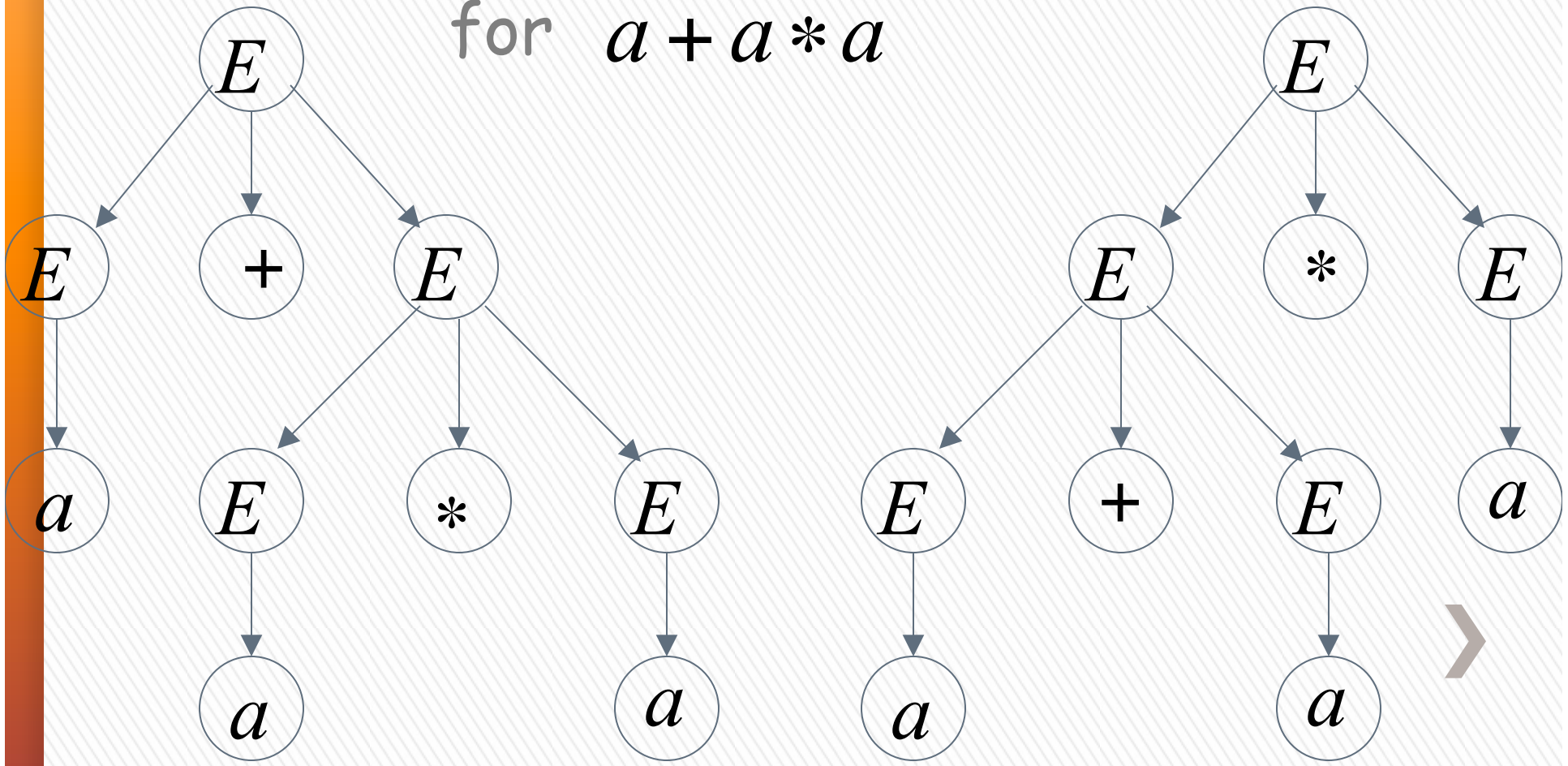
$$\Rightarrow a + a * E \Rightarrow a + a * a$$

Another leftmost derivation for $a + a * a$



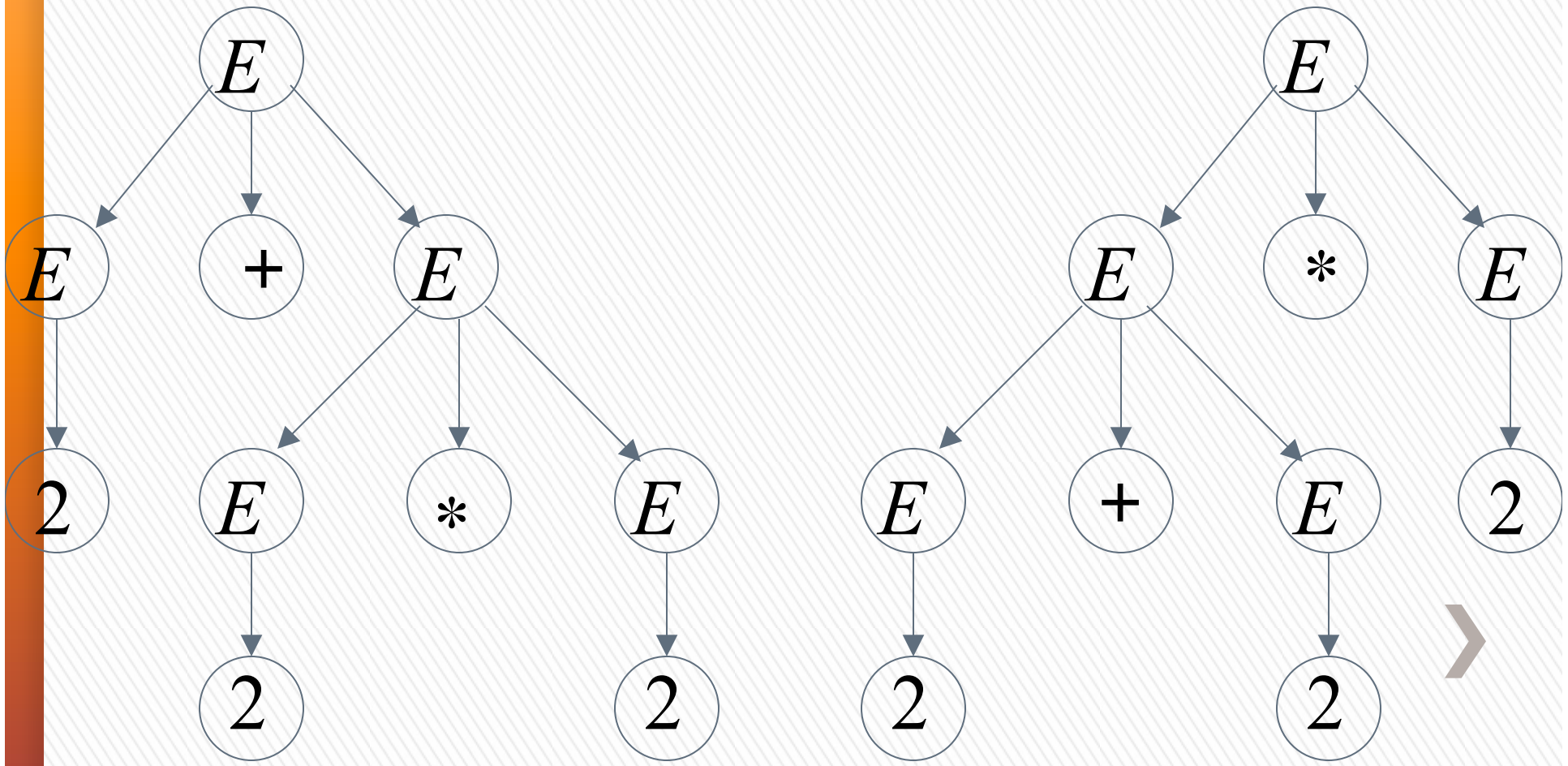
$$E \rightarrow E + E \mid E * E \mid (E) \mid a$$

Two derivation trees for $a + a * a$



take $a = 2$

$$a + a * a = 2 + 2 * 2$$



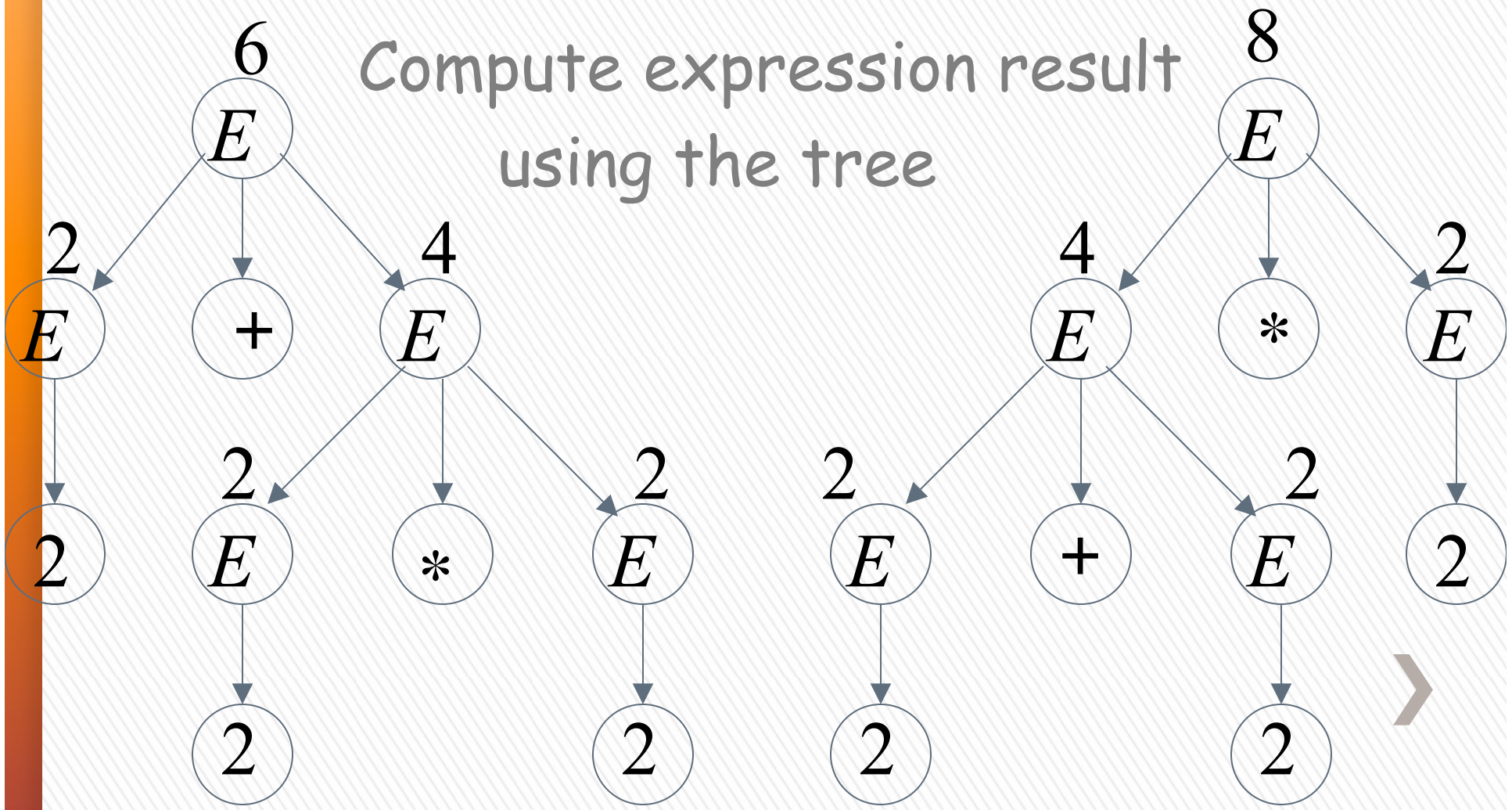
Good Tree

$2+2*2=6$

Bad Tree

$2+2*2=8$

Compute expression result using the tree



Two different derivation trees
may cause problems in applications which use the derivation trees:

- Evaluating expressions

- In general, in compilers for programming languages

Ambiguous Grammar:

A context-free grammar G is ambiguous if there is a string $w \in L(G)$ which has:

two different derivation trees

or

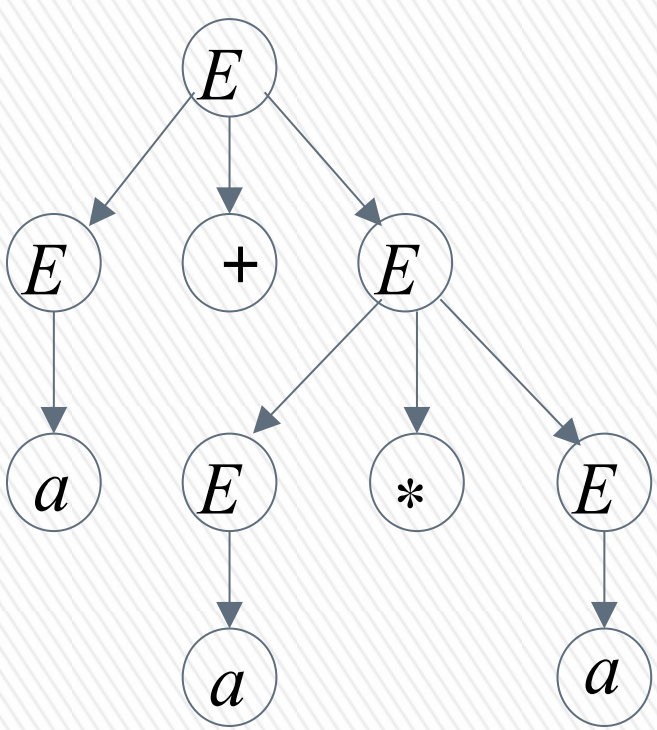
two leftmost derivations

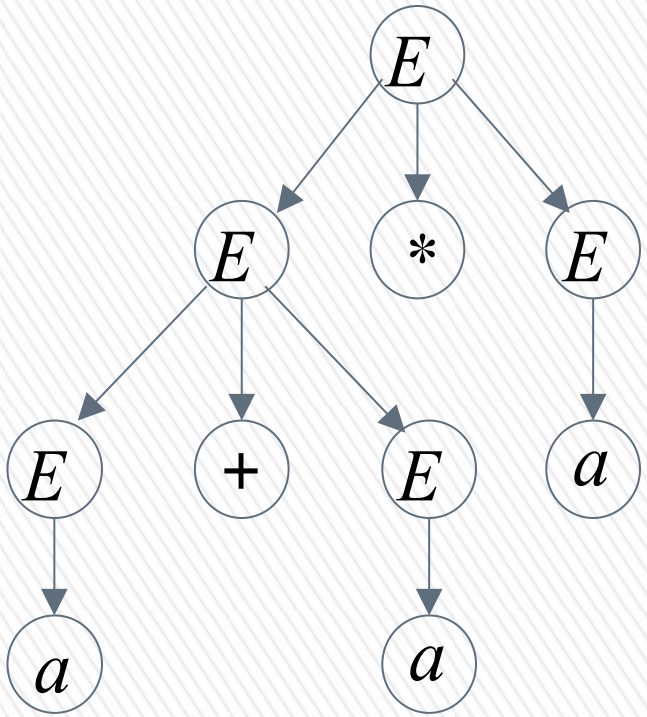
(Two different derivation trees give two different leftmost derivations and vice-versa)

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

this grammar is ambiguous since

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





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

this grammar is ambiguous also because

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

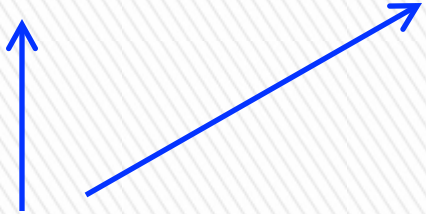
$$E \Rightarrow E + E \Rightarrow a + E \Rightarrow a + E * E$$
$$\Rightarrow a + a * E \Rightarrow a + a * a$$

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

Another ambiguous grammar:

IF_STMT → if EXPR then STMT

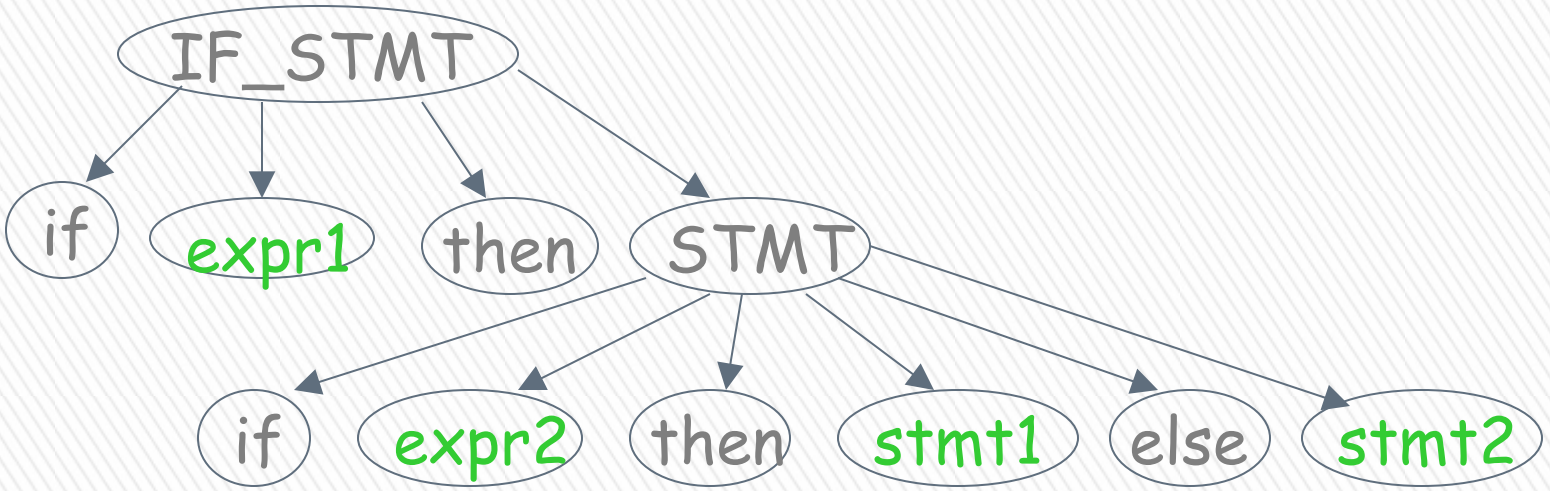
| if EXPR then STMT else STMT



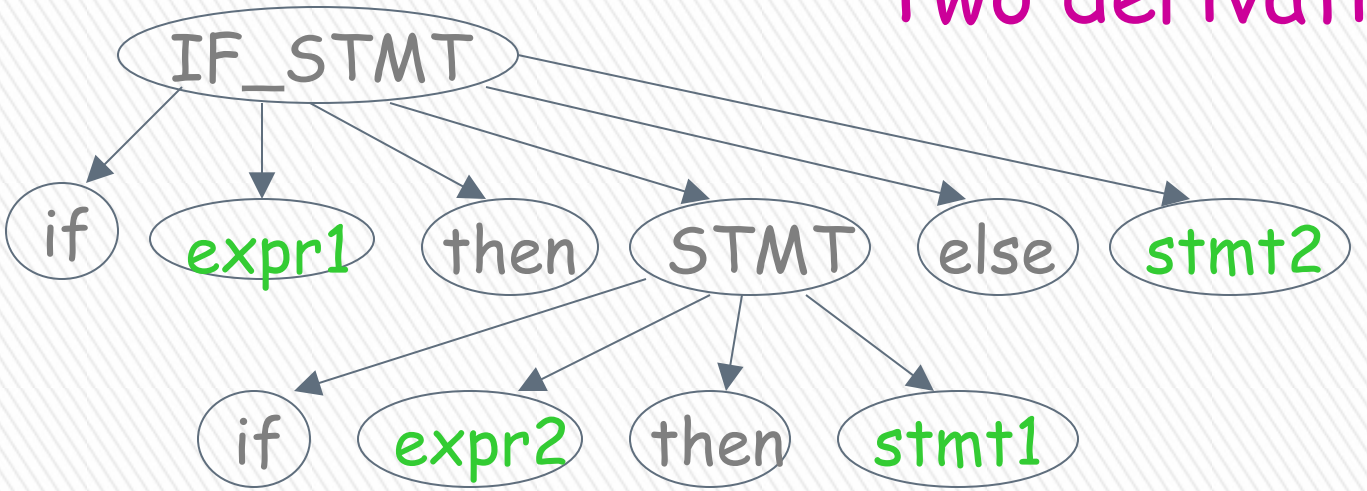
Variables Terminals

Very common piece of grammar in programming languages

If expr1 then if expr2 then stmt1 else stmt2



Two derivation trees



In general, ambiguity is bad and we want to remove it

Sometimes it is possible to find a non-ambiguous grammar for a language

But, in general we cannot do so

A successful example:

Ambiguous Grammar

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

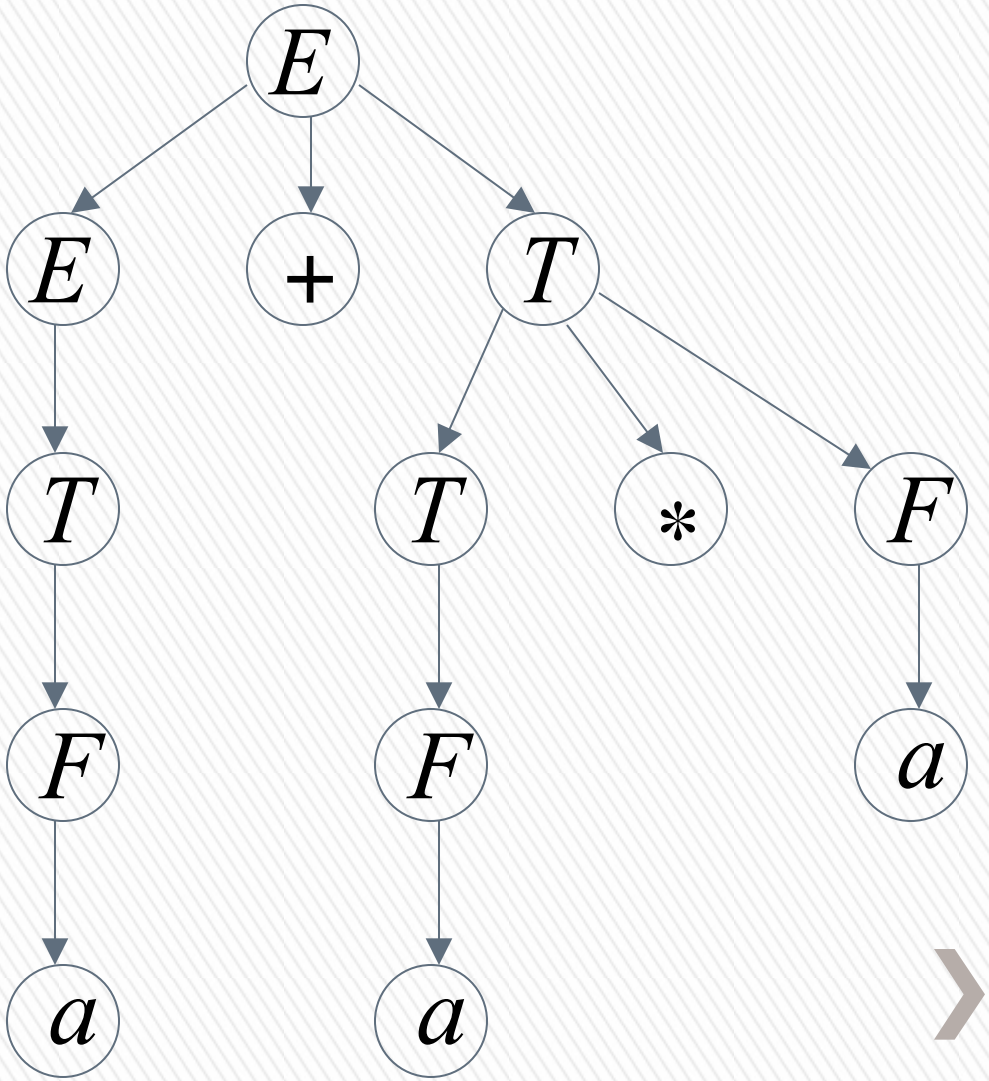
Equivalent Non-Ambiguous Grammar

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

generates the same language

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

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



Unique derivation tree for $a+a*a$

An un-successful example:

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

$$n, m \geq 0$$

L is inherently ambiguous:

every grammar that generates this language is ambiguous

Example (ambiguous) grammar for L:

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

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

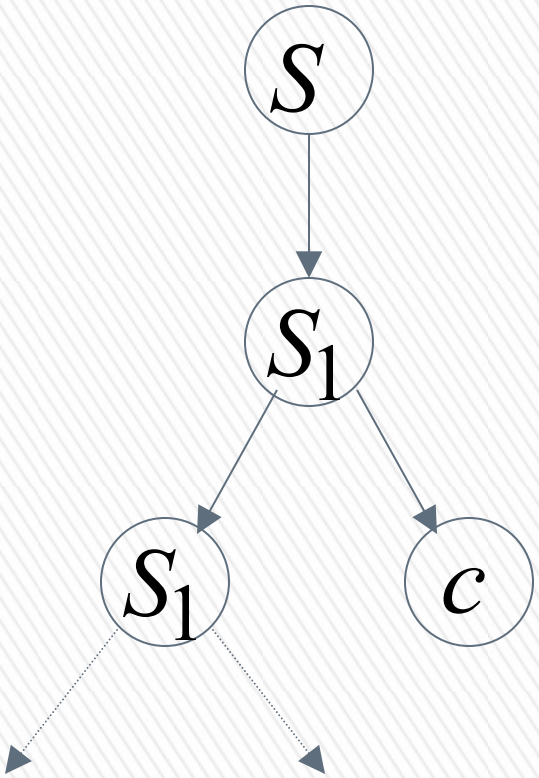
$$A \to aAb \mid \lambda \qquad B \to bBc \mid \lambda$$

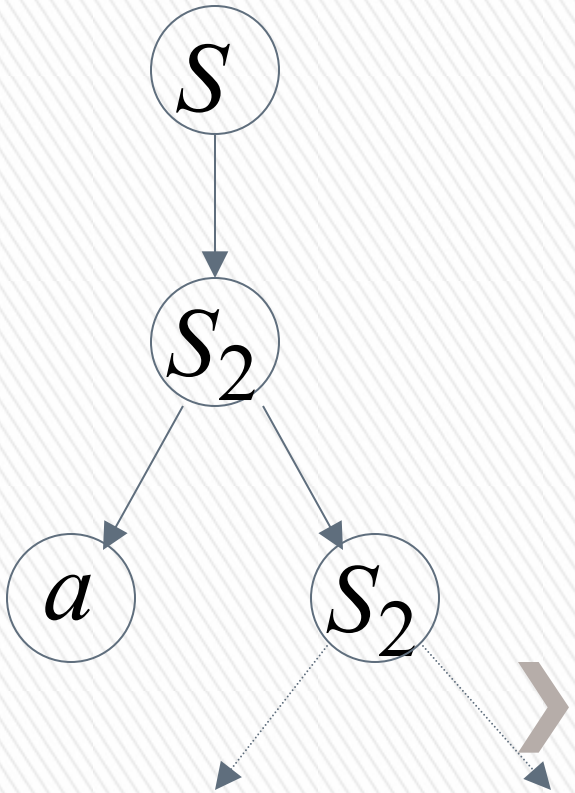
The string $a^n b^n c^n \in L$

has always two different derivation trees

(for any grammar)

For example





BLM2502 Theory of Computation