COM2003

Automata, Logic and Computation

Kirill Bogdanov

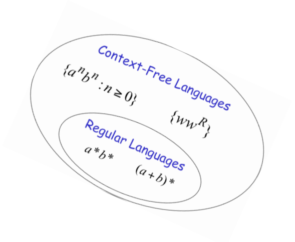
Slides originally written by Lucia Specia
based on those by M.S. Moorthy
who relied on slides by Prof Costas Busch

Week 5 – lecture 10am Friday
and no tutorial.

Languages

“this is a slide”

- When you think what this means, I visualise a transparency with some text and graphics



- When you think of it as text, it is where you can replace all letters ‘i’ with Q, so it becomes

“thQs Qs a slQ de”

- Here we are talking of the difference between syntax and semantics.

Regular expressions 1/2

- Syntax: composed from primitives using operations, such as a*b+c (here * and + are in magenta colour)
- Semantics: what those operation symbols actually mean.
- In terms of sets of strings, * means the same as * defined in week 2,

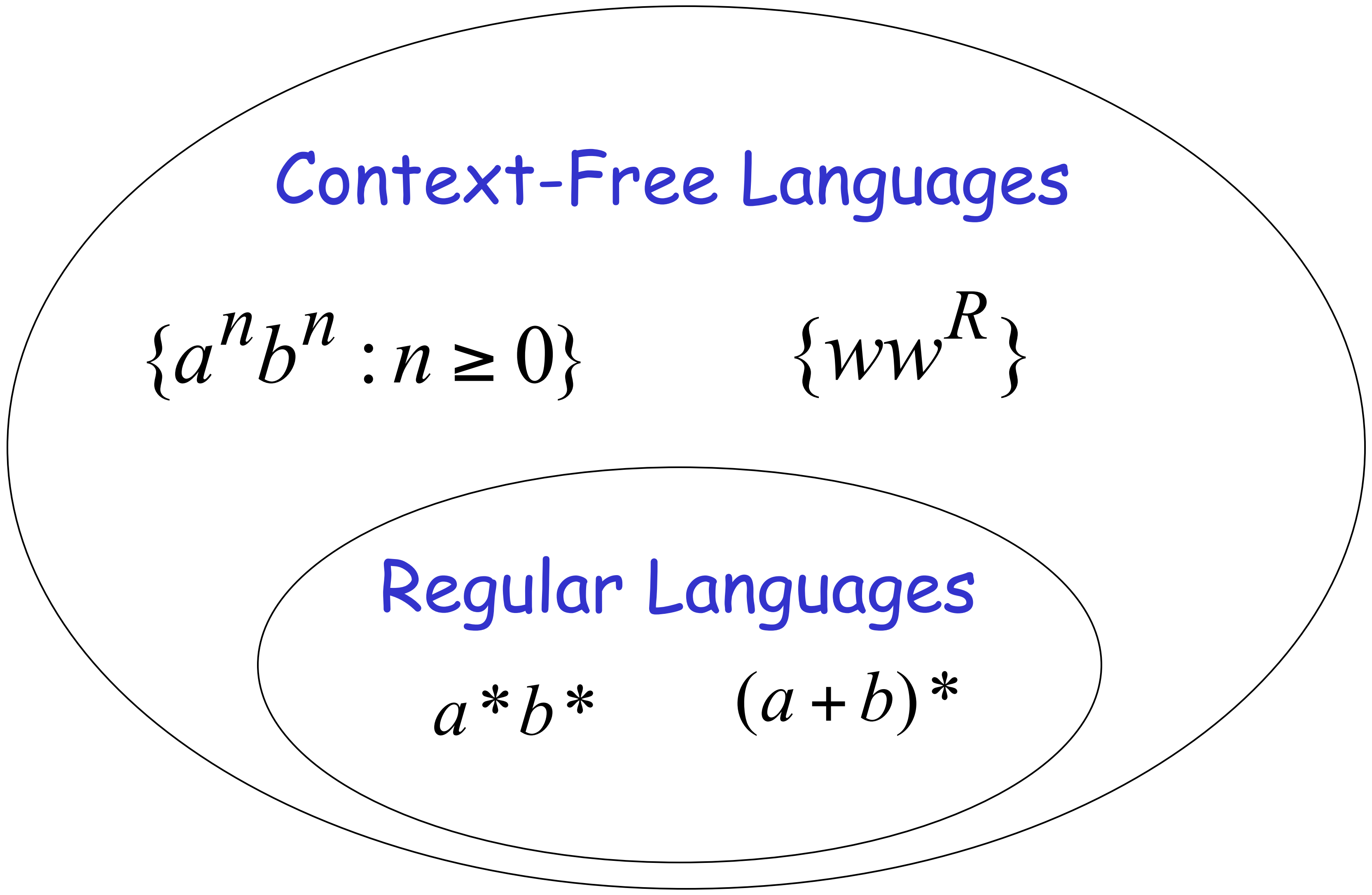
$$\{\varepsilon\} \cup A \cup AA \ldots$$

- These stars look confusingly the same, one is part of syntax, another describes an operation on sets of strings.

Regular expressions 2/2

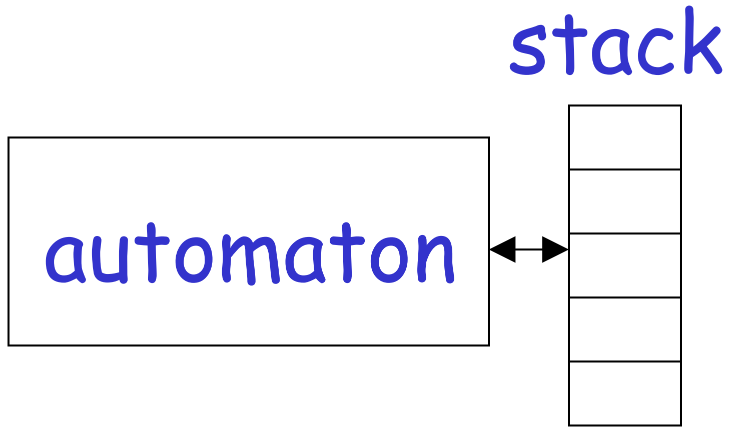
- Syntax: composed from primitives using operations, such as
  a*b+c (here * and + are in magenta colour)
- Semantics: what those operation symbols actually mean.
- + (in magenta colour) means set union, thus A+B means $A \cup B$
  Same for concatenation.
- Operations can be interpreted as NFAs too – see constructs to build an NFA from regular expressions.

Context-Free Languages


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

Context-Free Languages

$$\begin{tikzcd}
 & \text{Context-Free Languages} \arrow[dl, leftrightarrow] \arrow[dr, leftrightarrow] & \\
\text{Context-Free Grammars} & & \text{Pushdown Automata}
\end{tikzcd}$$



Importance of Context-Free Languages

- Designers of compilers and interpreters for programming languages start by writing a grammar for the language:
  - Symbols that are accepted (lexical analyser = RE)
  - How these symbols can be **combined** to execute something
- A parser is a component that then extracts the meaning of a program before compilation/ interpretation
  - Tools to generate parsers from grammars

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”:

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

Similar idea to “recognizing string”

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 this 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" }

Language of this grammar:

L = { "a cat runs",

Often grammars are more complex than this (and infinite)

→listing all strings in the language is not possible. E.g.:

- English
- Python…

"the dog sleeps" }

Productions

$$\langle noun \rangle \rightarrow \overbrace{cat}^{\text{Sequence of Terminals (symbols)}}$$

$$\langle sentence \rangle \rightarrow \underbrace{\langle noun\_phrase \rangle \ \langle predicate \rangle}_{\text{Sequence of Variables (Non-Terminals)}}$$

Variables

Another Example

Sequence of
terminals and variables

Grammar: $S \to \overbrace{aSb}$

$S \to \varepsilon$

Variable

The right side
may be ε

Grammar:

$$S \to a\overset{\text{Recursion}}{S}b$$

$$S \to \varepsilon$$

Derivation of string ab:

$$S \underset{S \to aSb}{\Longrightarrow} aSb \underset{S \to \varepsilon}{\Longrightarrow} ab$$

Grammar: $S \to aSb$

$S \to \varepsilon$

Derivation of string $aabb$:

$$\begin{tikzcd}
S & \Rightarrow & aSb & \Rightarrow & aaSbb & \Rightarrow & aabb \\
 & & S \to aSb \arrow[ul] \arrow[ur] & & & S \to \varepsilon \arrow[u] &
\end{tikzcd}$$

Grammar:

$$S \rightarrow aSb$$

$$S \rightarrow \varepsilon$$

Other derivations:

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

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

$$\Rightarrow aaaaSbbbb \Rightarrow aaaabbbb$$

What is the language recognized by this grammar?

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 zero or more derivation steps

Instead of:

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

In general we write: $w_1 \overset{*}{\Longrightarrow} w_n$

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

in zero or more derivation steps

Trivially: $w \overset{*}{\Longrightarrow} w$

Example Grammar

$$S \rightarrow aSb$$

$$S \rightarrow \varepsilon$$

Possible Derivations

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

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

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

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

Another convenient notation:

$$\left.\begin{array}{l} S \to aSb \\ S \to \varepsilon \end{array}\right. \Longrightarrow S \to aSb \mid \varepsilon$$

$$\left.\begin{array}{l} \langle article \rangle \to a \\ \langle article \rangle \to the \end{array}\right. \Longrightarrow \langle article \rangle \to a \mid the$$

Formal Definition

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

Set of variables
a.k.a "non-terminal"

Set of terminal symbols
"words" in the language

Start variable

Set of productions
"rules" of the language

Formal Definition

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

Set of variables
a.k.a "non-terminal"

Set of terminal symbols
"words" in the language

Start variable

Set of productions
"rules" of the language

Don't confuse T with Σ

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

All productions in P are of the form

$$A \to s$$

Variable

Can ONLY be a variable
and
Can ONLY be ONE variable

String of variables and terminals

Example of Context-Free Grammar

$$\boxed{S \to aSb \mid \varepsilon}$$

productions

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

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

$$V = \{S\}$$

variables

$$T = \{a, b\}$$

terminals

start variable

Language of a Grammar:

For a grammar G with start variable S

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

String of terminals or ε

Example:

Context-free grammar $G : S \rightarrow aSb \mid \varepsilon$

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

Since, there is derivation

$S \overset{*}{\Longrightarrow} a^n b^n$ 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:

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

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

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

Palindromes of even length

Another Example

Context-free grammar G:

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

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 $=)$

Exercise

Write context-free grammar for:

$$L = \{0^n 1^n \mid n \geq 0\} \cup \{1^n 0^n \mid n \geq 0\}$$

Break problem down in two parts

Exercise

Write context-free grammar for:

$$L = \{0^n 1^n \mid n \geq 0\} \cup \{1^n 0^n \mid n \geq 0\}$$

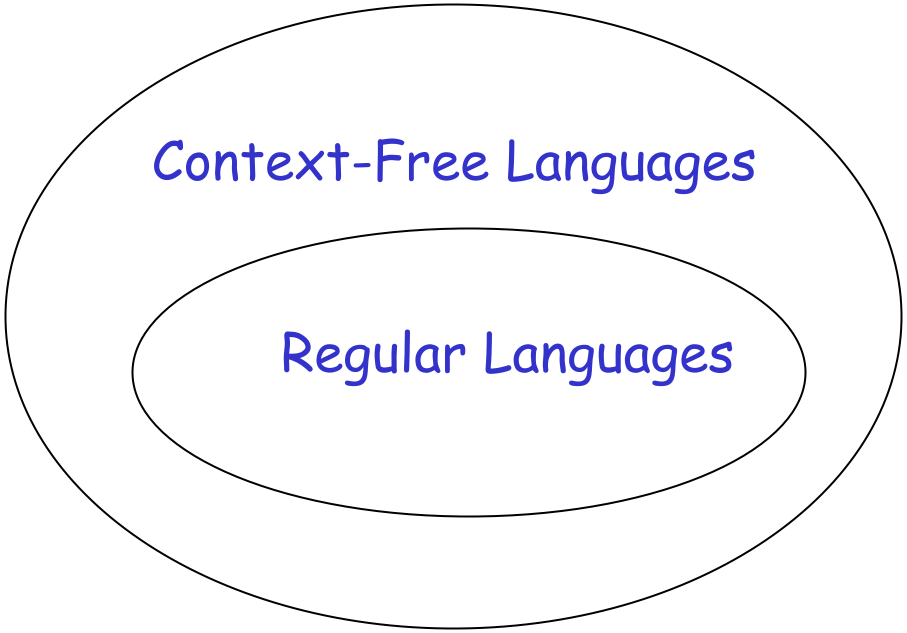
Break problem down in two parts

$$S_1 \rightarrow 0S_1 1 \mid \varepsilon \qquad S_2 \rightarrow 1S_2 0 \mid \varepsilon$$

$$\Downarrow$$

$$\begin{aligned} S &\rightarrow S_1 \mid S_2 \\ S_1 &\rightarrow 0S_1 1 \mid \varepsilon \\ S_2 &\rightarrow 1S_2 0 \mid \varepsilon \end{aligned}$$

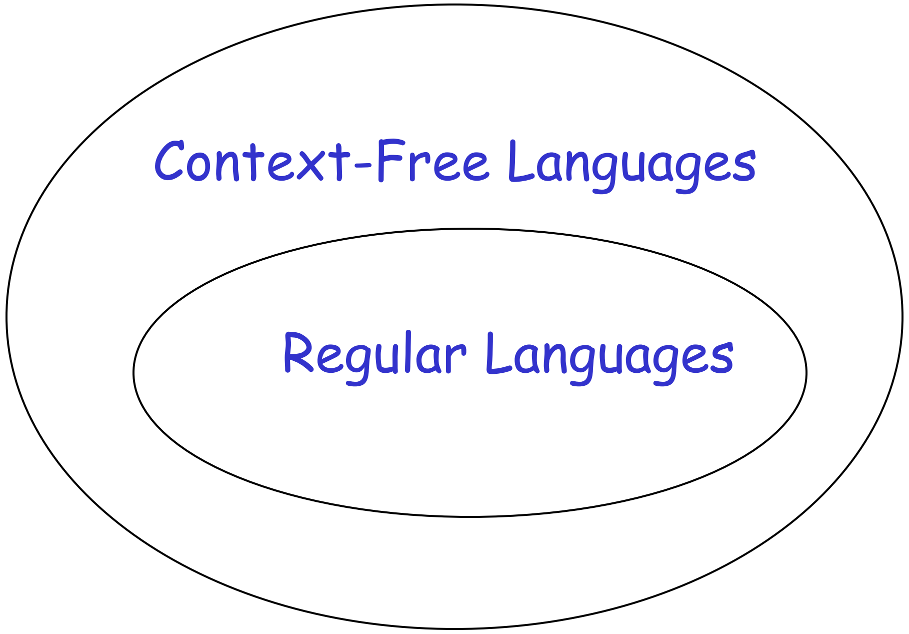
Context-free grammar and regular languages



Can write grammar for DFA of any language regular:

- Make variable R_i for each state q_i
- Make rule $R_i \to aR_j$ as $\delta(q_i, a) = q_j$
- Add rule $R_i \to \varepsilon$ if q_i is accepting state
- Make R_0 the start variable of the grammar where q_0 is the start state

Context-free grammar and regular languages



Can write grammar for DFA of any language regular:

- Make variable R_i for each state q_i
- Make rule $R_i \rightarrow aR_j$ as $\delta(q_i, a) = q_j$
- Add rule $R_i \rightarrow \varepsilon$ if q_i is accepting state
- Make R_0 the start variable of the grammar where q_0 is the start state

Derivation Order
and
Derivation Trees

Derivation Order

Example grammar with 5 productions:

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

$$\boxed{\begin{array}{lll} 1.\ S \to AB & 2.\ A \to aaA & 4.\ B \to Bb \\ & 3.\ A \to \varepsilon & 5.\ B \to \varepsilon \end{array}}$$

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

$$\boxed{\begin{array}{lll} 1.\ S \rightarrow AB & 2.\ A \rightarrow aaA & 4.\ B \rightarrow Bb \\ & 3.\ A \rightarrow \varepsilon & 5.\ B \rightarrow \varepsilon \end{array}}$$

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

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* :

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

Derivation Trees

Consider the same example grammar:

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

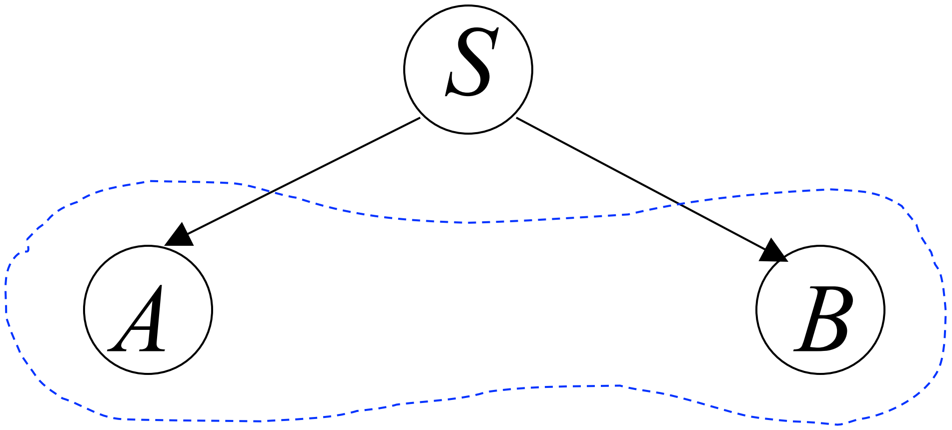
And a derivation of *aab*:

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

Can write this derivation as a **tree**

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

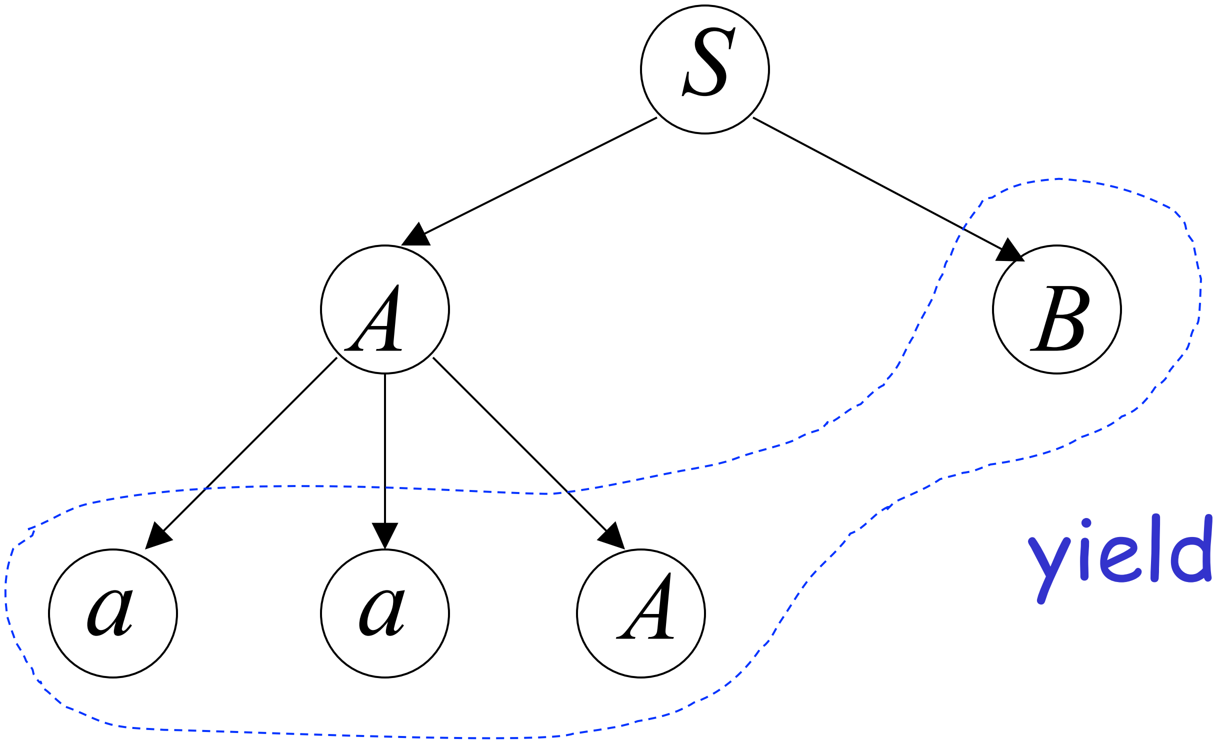
$$S \Rightarrow AB$$



yield AB

$$\boxed{S \rightarrow AB \qquad A \rightarrow aaA \mid \varepsilon \qquad B \rightarrow Bb \mid \varepsilon}$$

$$S \Rightarrow AB \Rightarrow aaAB$$

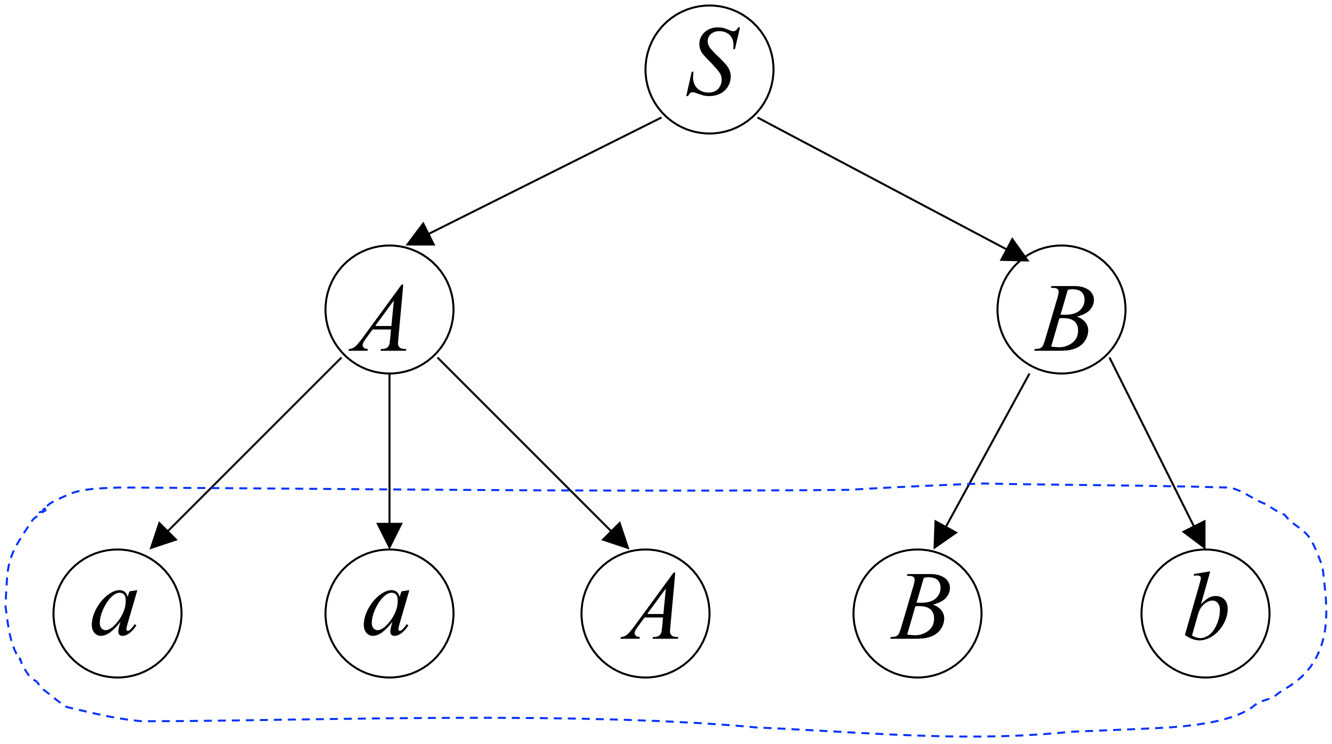


yield $aaAB$

A tree describes how an string is derived.

$$\boxed{S \rightarrow AB \qquad A \rightarrow aaA \mid \varepsilon \qquad B \rightarrow Bb \mid \varepsilon}$$

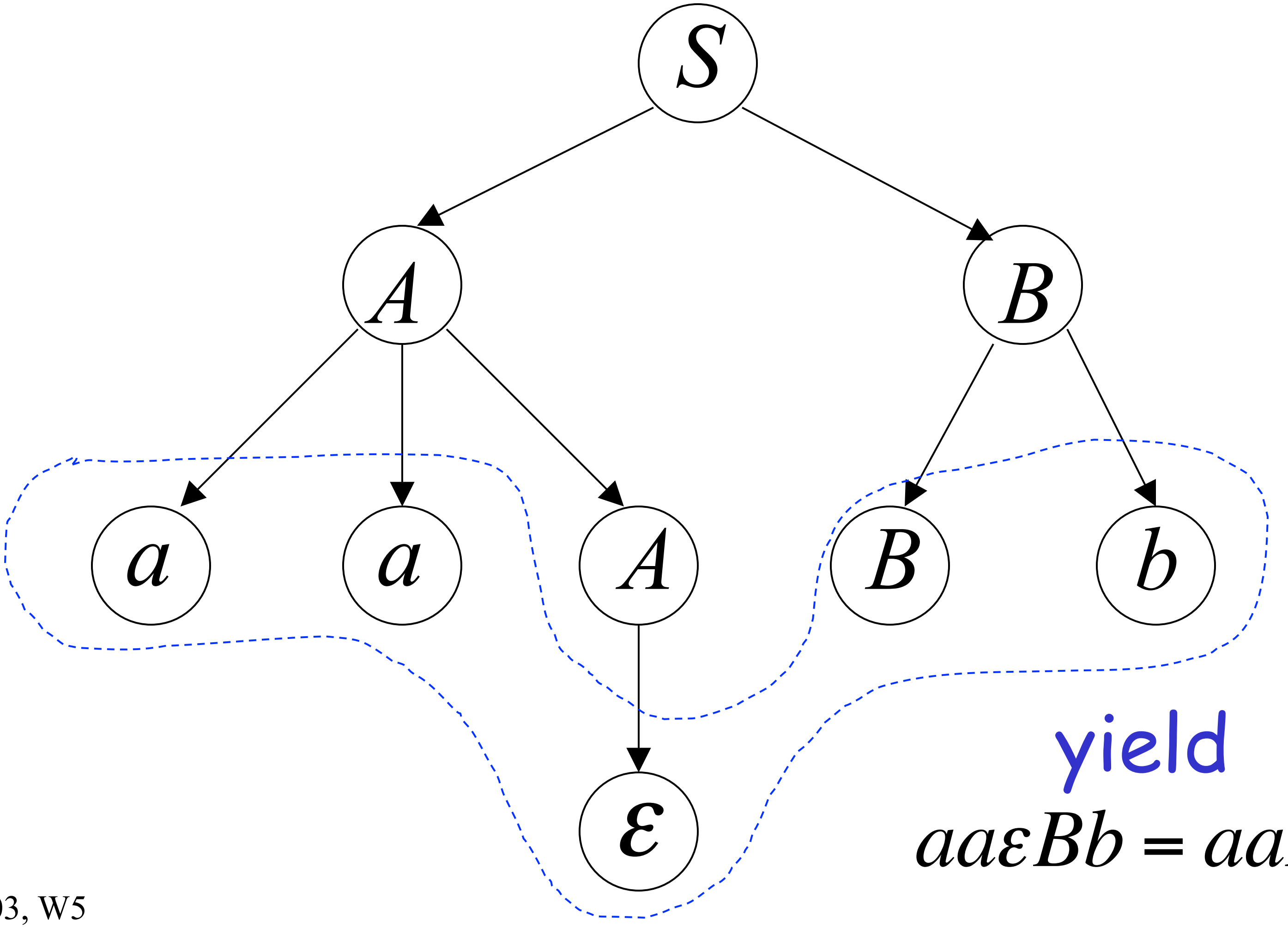
$$S \Rightarrow AB \Rightarrow aaAB \Rightarrow aaABb$$



yield *aaABb*

$$\boxed{S \rightarrow AB \qquad A \rightarrow aaA \mid \varepsilon \qquad B \rightarrow Bb \mid \varepsilon}$$

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



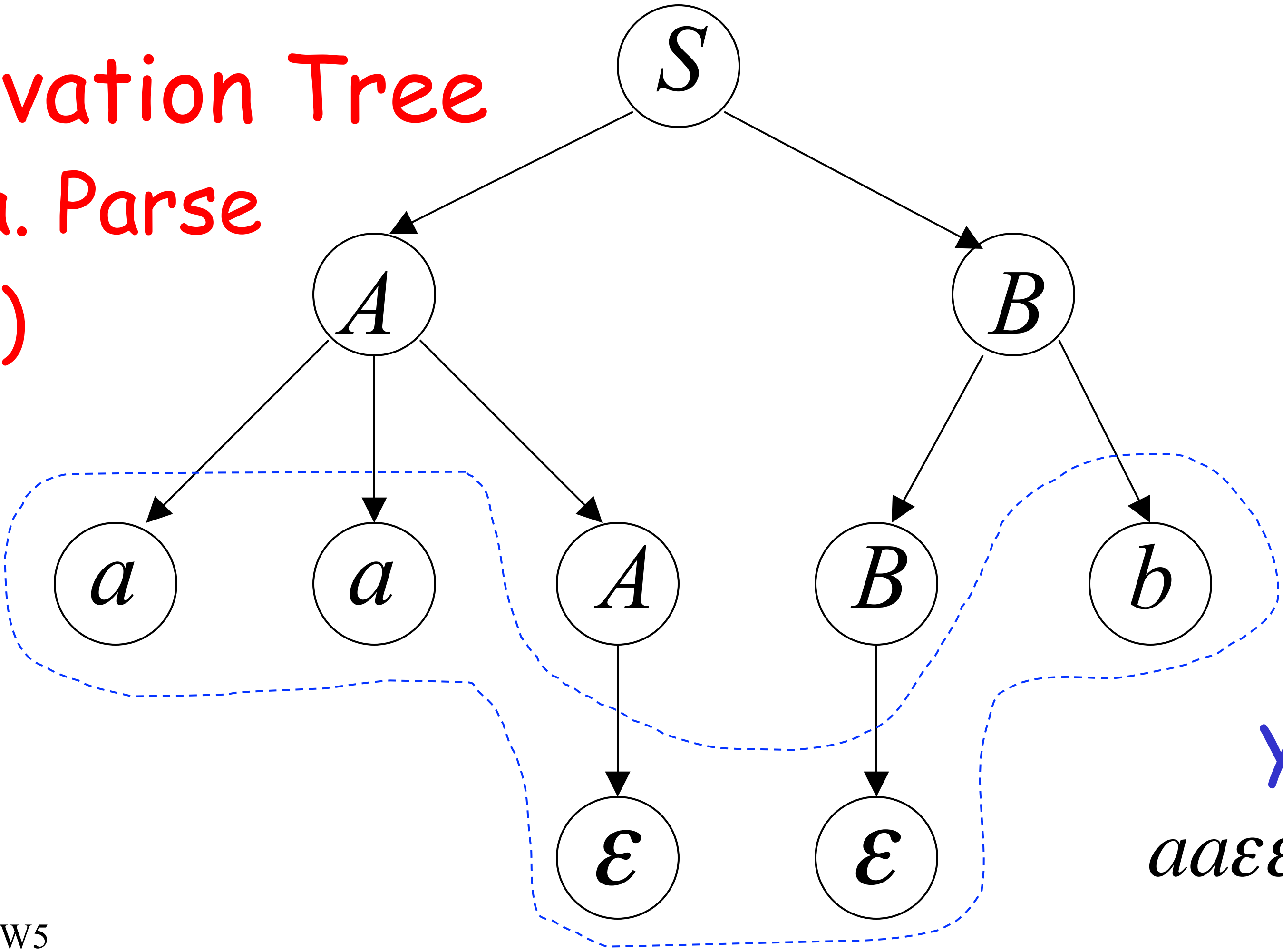
yield

$aa\varepsilon Bb = aaBb$

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

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

Derivation Tree (a.k.a. Parse Tree)



yield

$aa\varepsilon\varepsilon b = aab$

Sometimes, derivation order doesn't matter

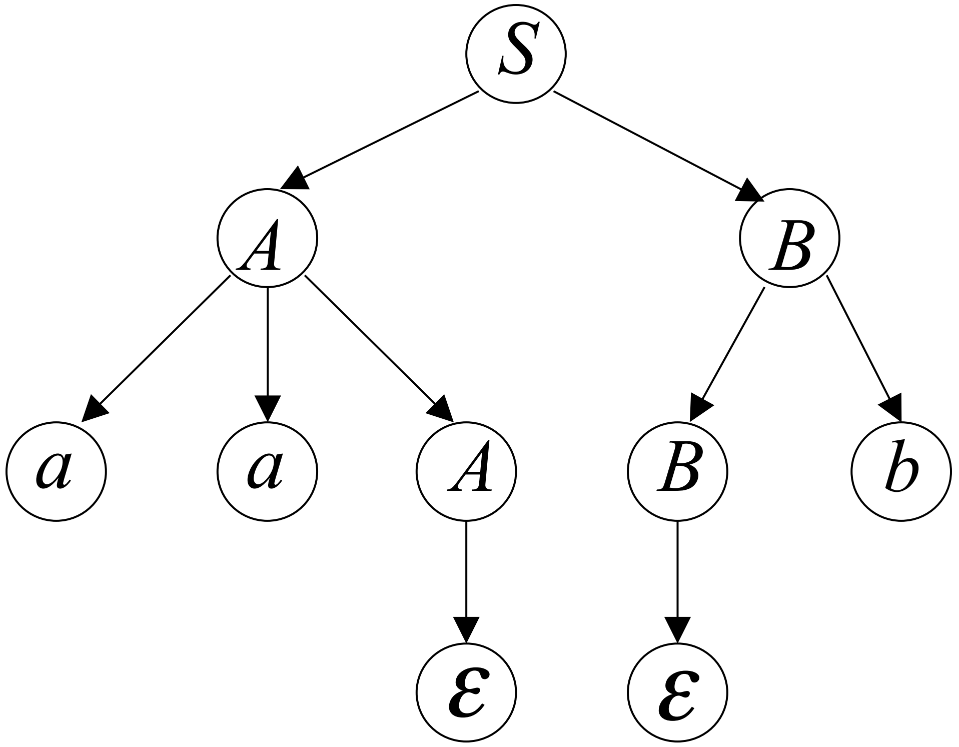
Leftmost derivation:

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

Rightmost derivation:

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

Give same derivation tree



Ambiguity

Grammar for mathematical expressions

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

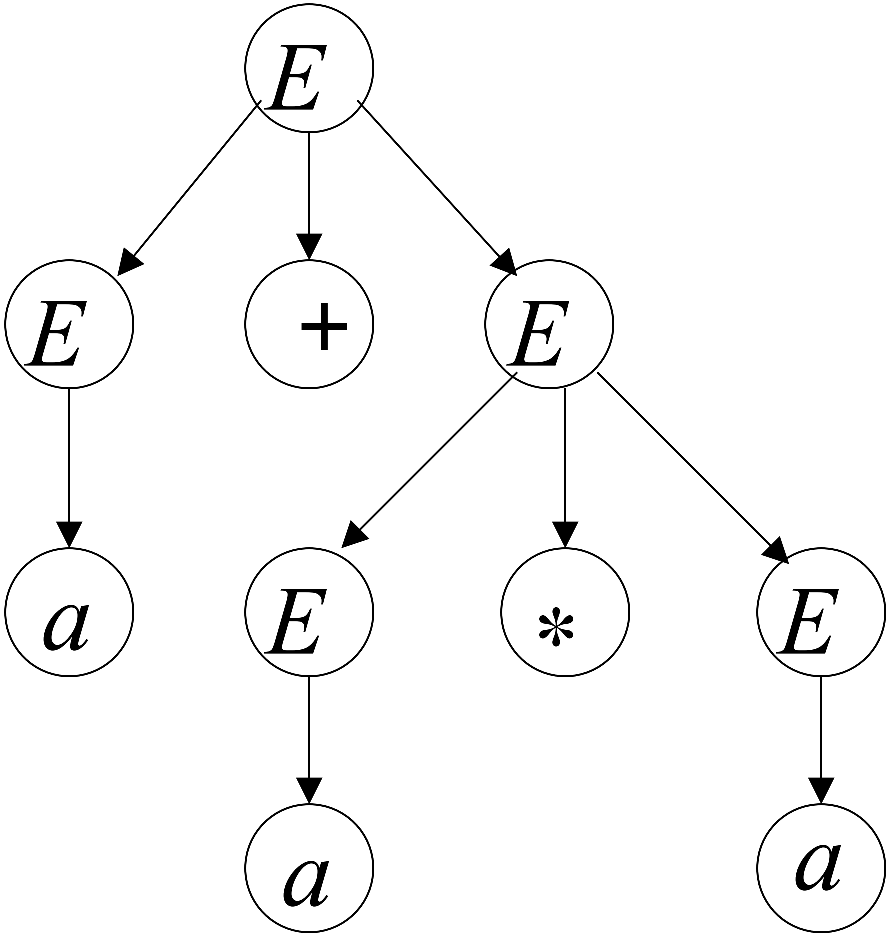
Example string:

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

Denotes any number in {0,1,2...}

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

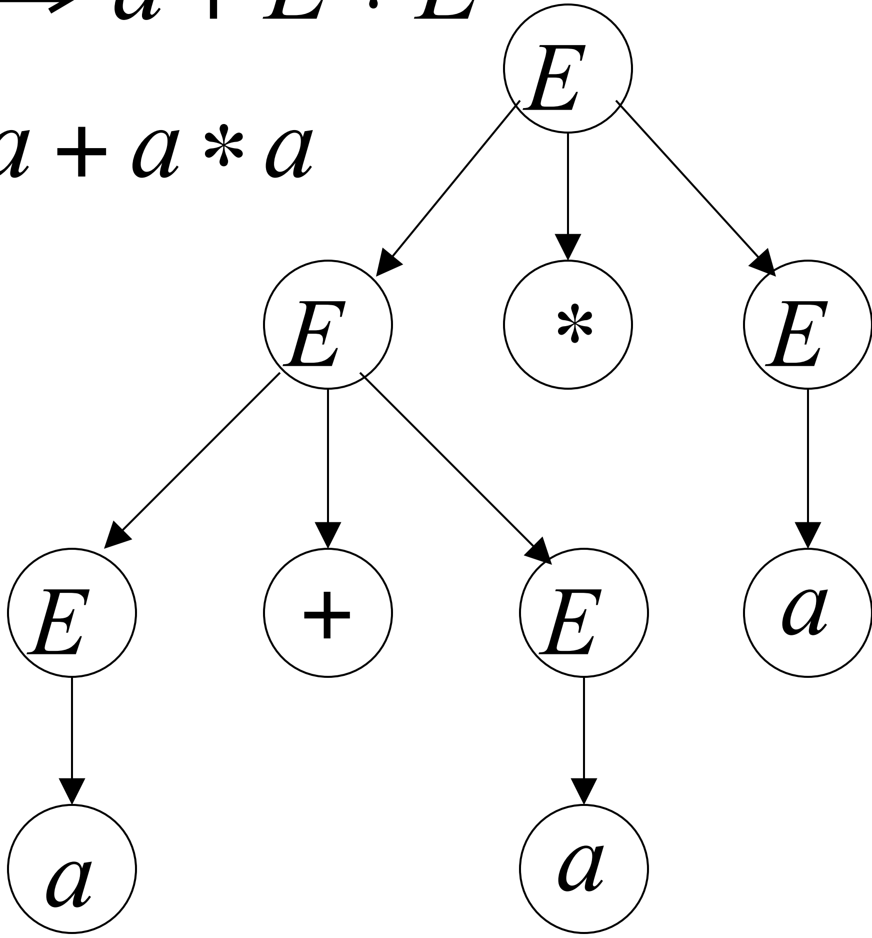
$$E \Rightarrow E + E \Rightarrow a + E \Rightarrow a + E * E$$
$$\Rightarrow a + a * E \Rightarrow a + a * a$$



A leftmost derivation for $a + a * a$

$$\boxed{E \rightarrow E + E \mid E * E \mid (E) \mid a}$$

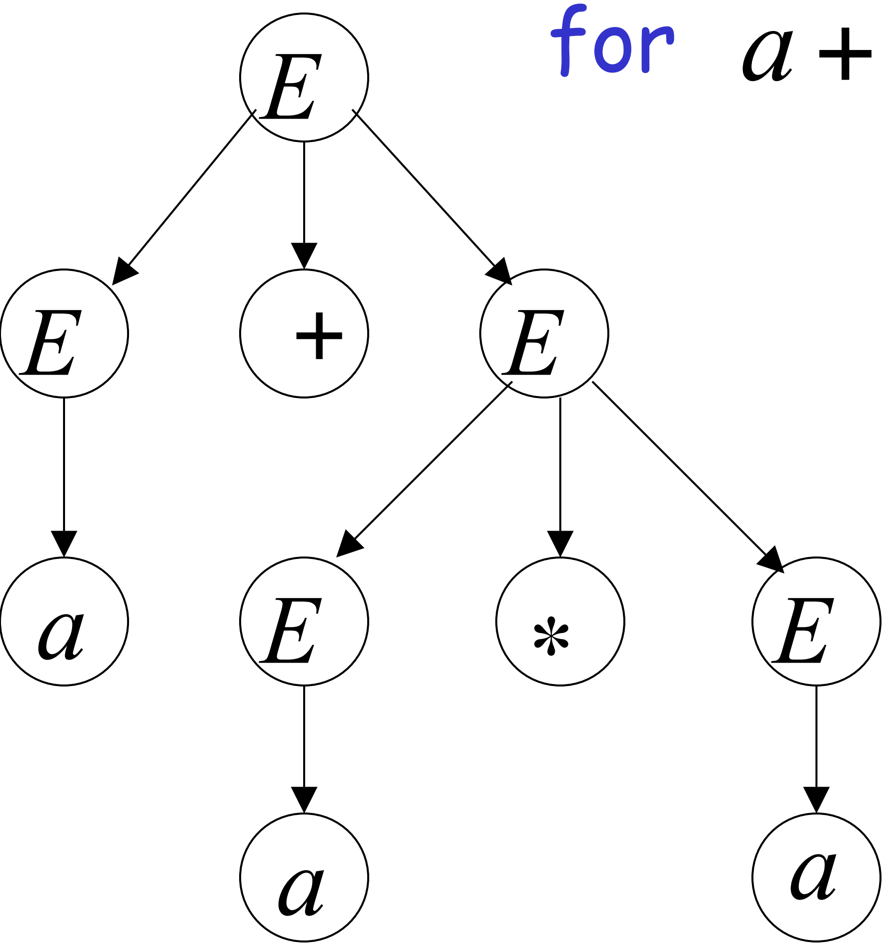
$$E \Rightarrow E * E \Rightarrow E + E * E \Rightarrow a + E * E$$
$$\Rightarrow a + a * E \Rightarrow a + a * a$$

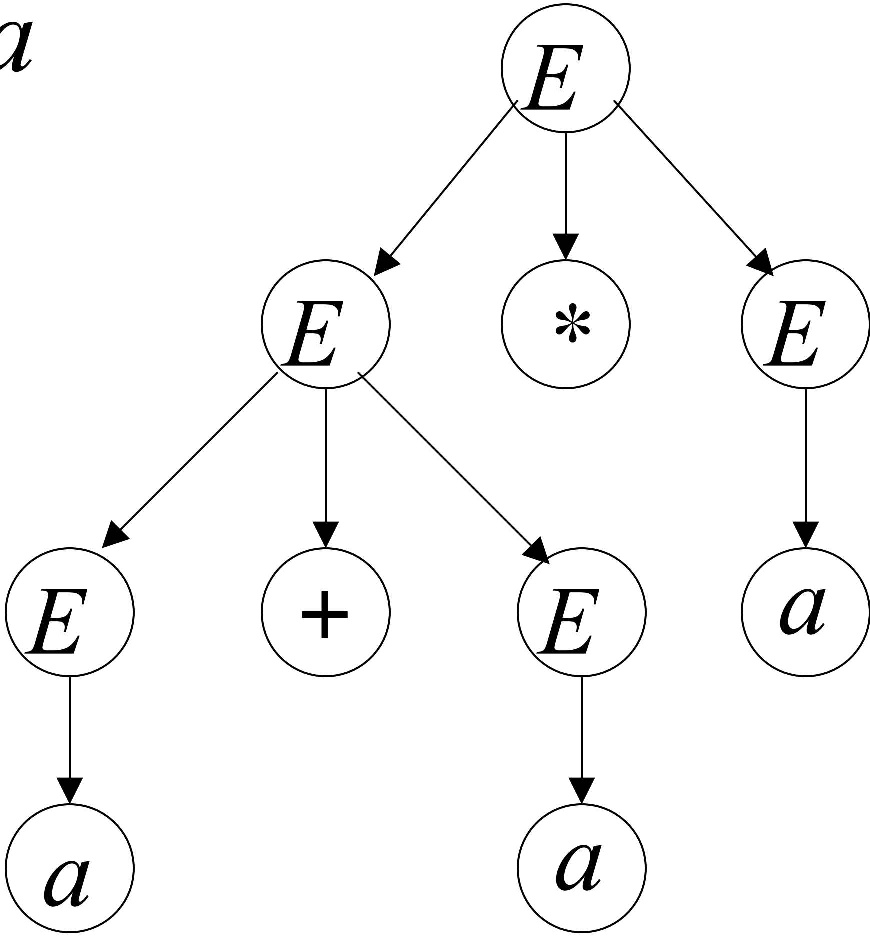


Another leftmost derivation for $a + a * a$

$$\boxed{E \rightarrow E + E \;\mid\; E * E \;\mid\; (E) \;\mid\; a}$$

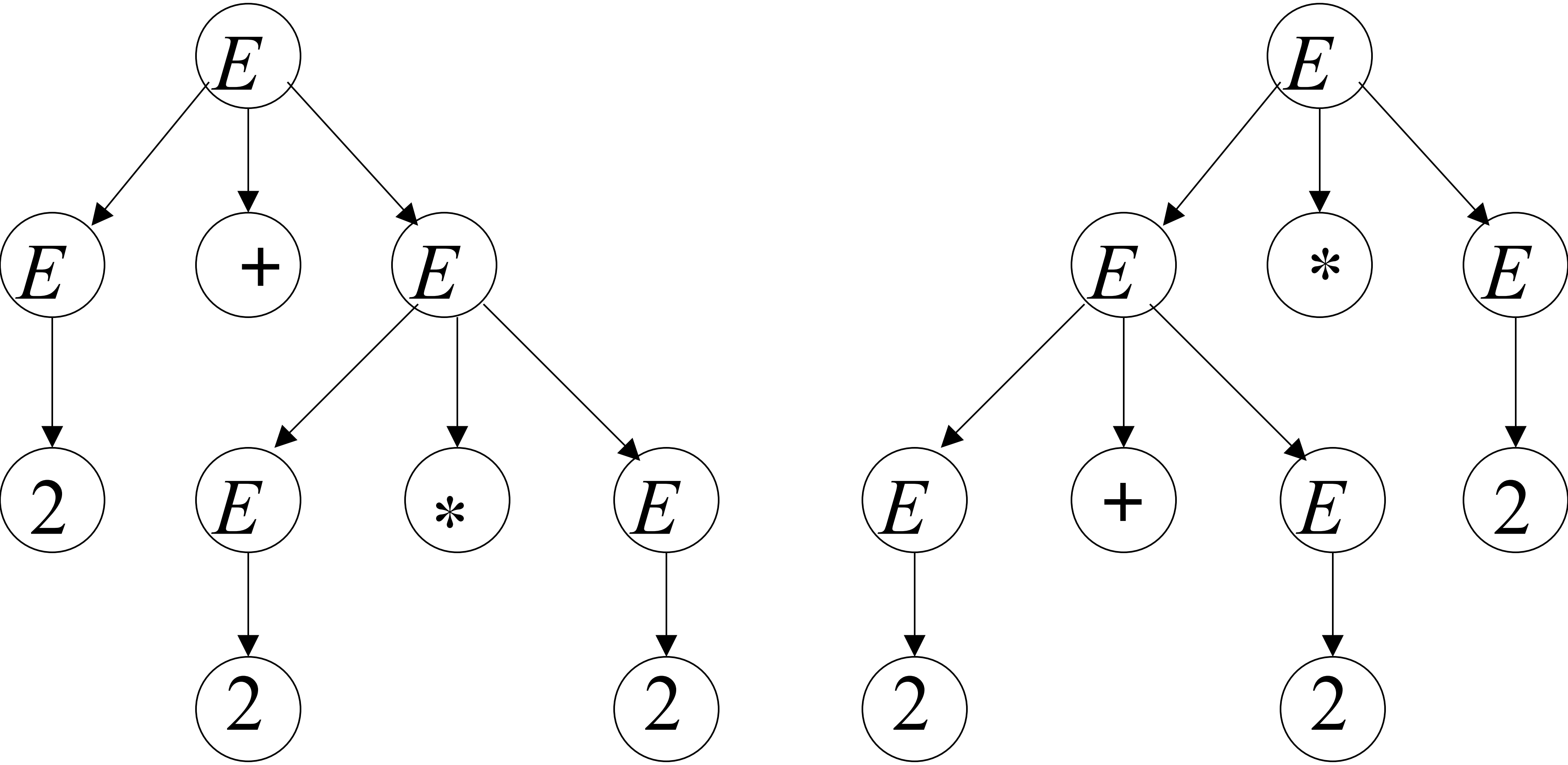
Two derivation trees for $a + a * a$





take $a = 2$

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



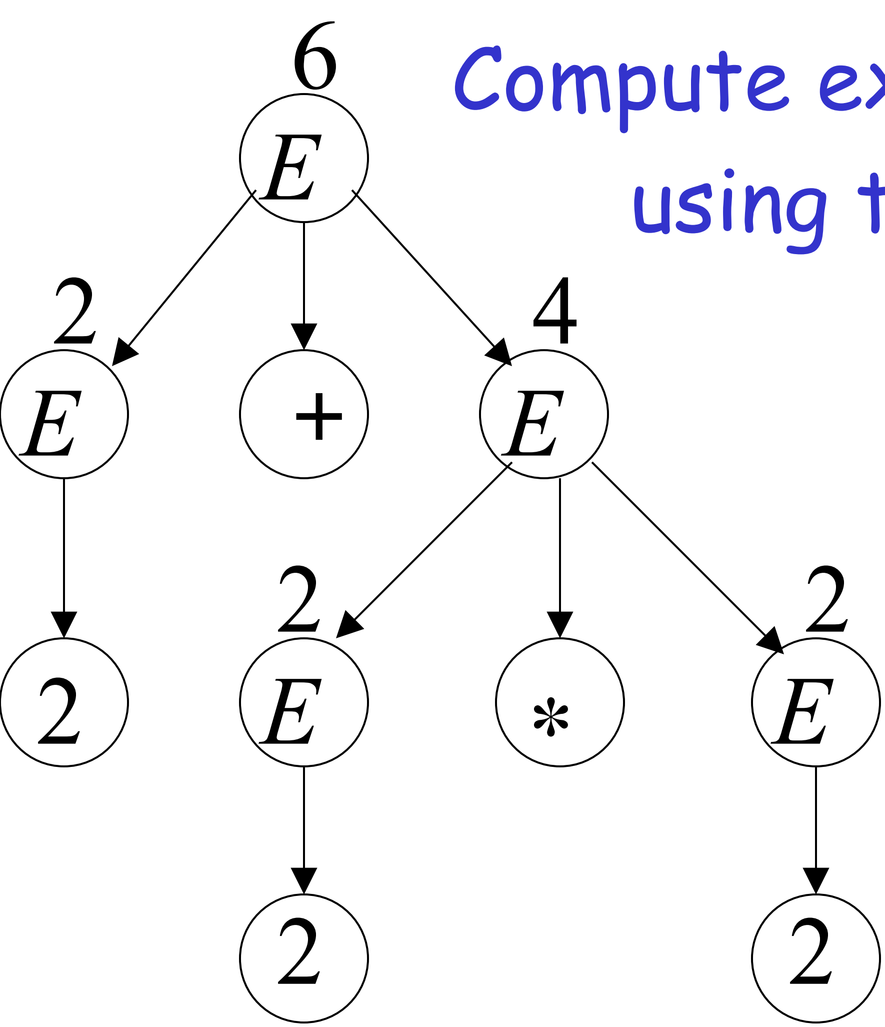
"Good" Tree

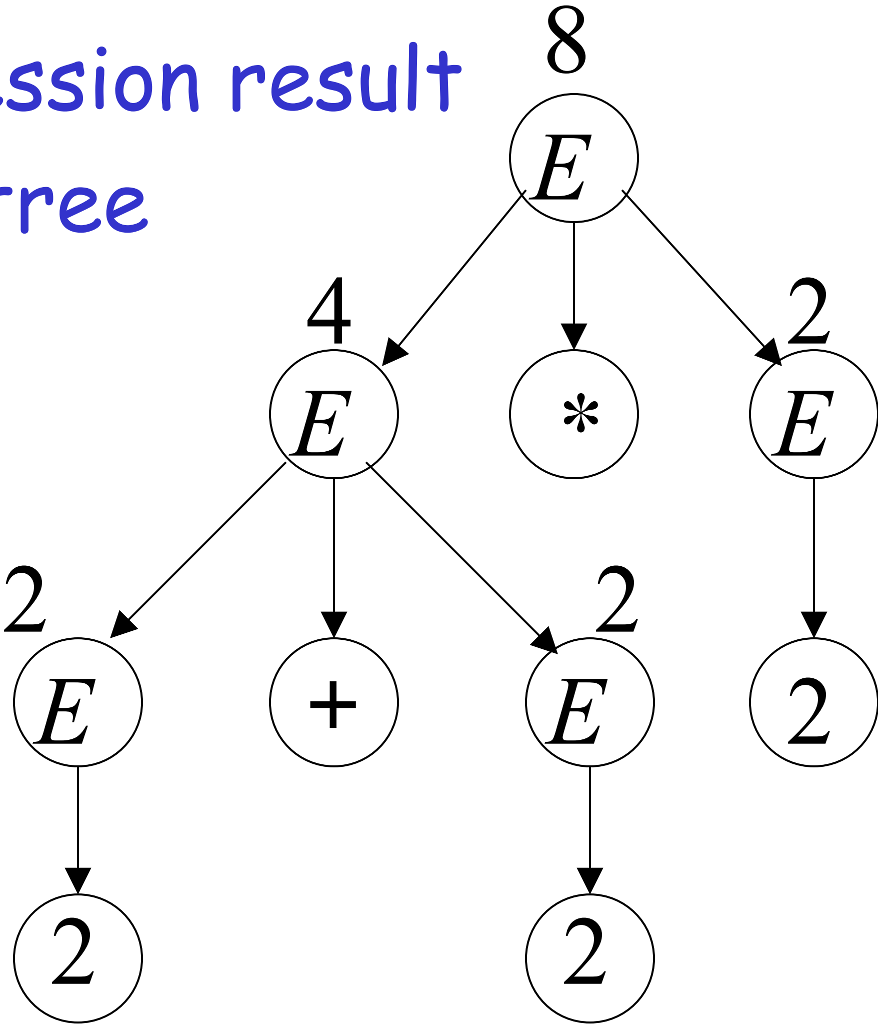
$$2+2*2=6$$

"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/parse trees

or

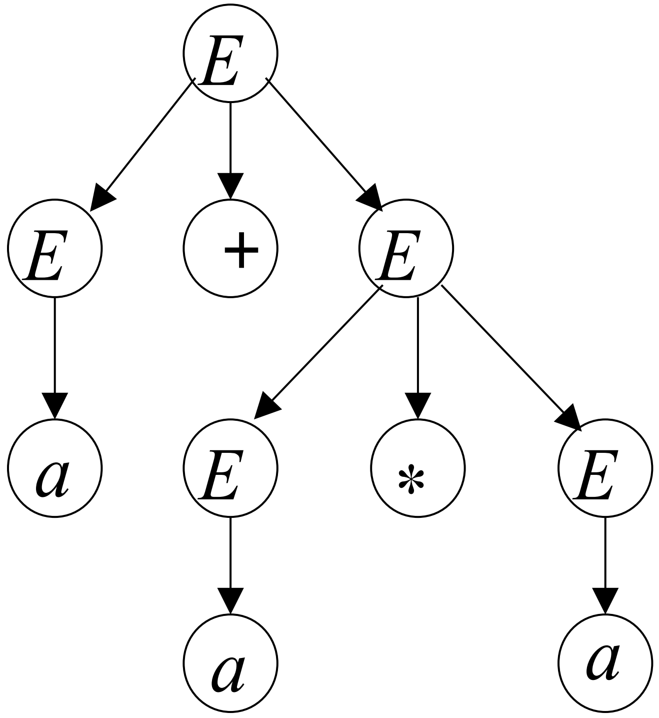
two leftmost derivations

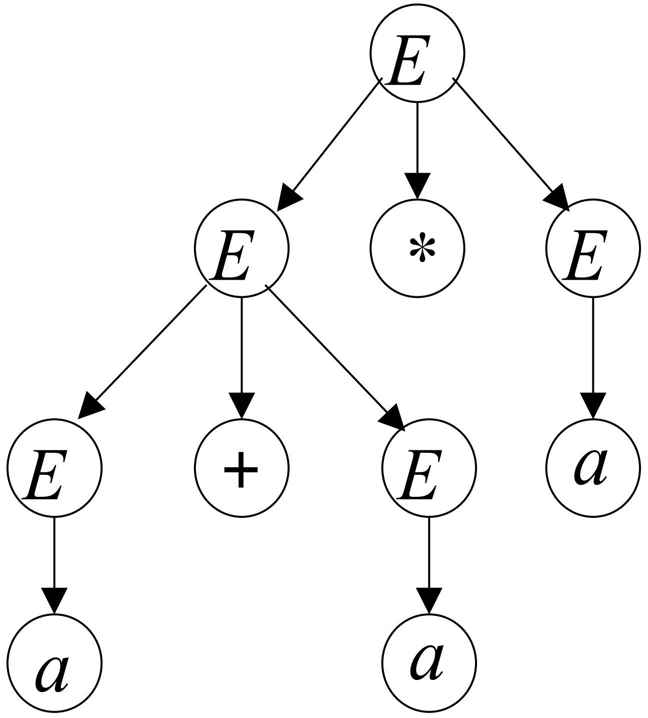
(Two different derivation trees give two different leftmost derivations and vice-versa)

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

this grammar is ambiguous since

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





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

this grammar is ambiguous also because

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

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

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

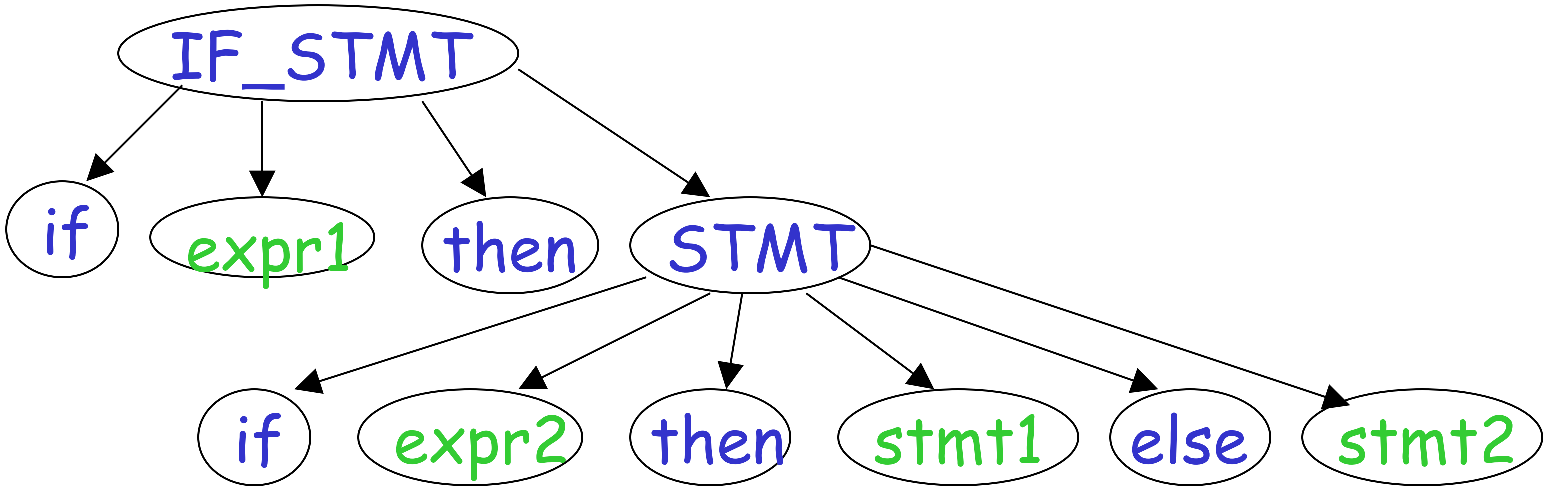
Another ambiguous grammar:

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

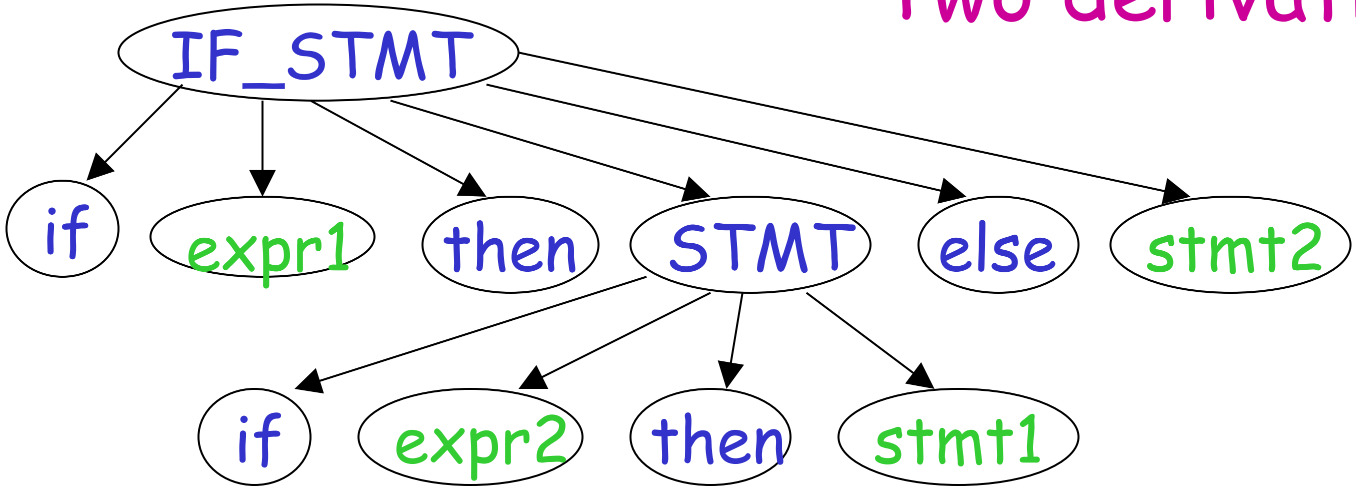
Variables Terminals

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

A successful example:

Ambiguous Grammar

$$\boxed{\begin{array}{l} E \to E + E \\ E \to E * E \\ E \to (E) \\ E \to a \end{array}}$$

Equivalent

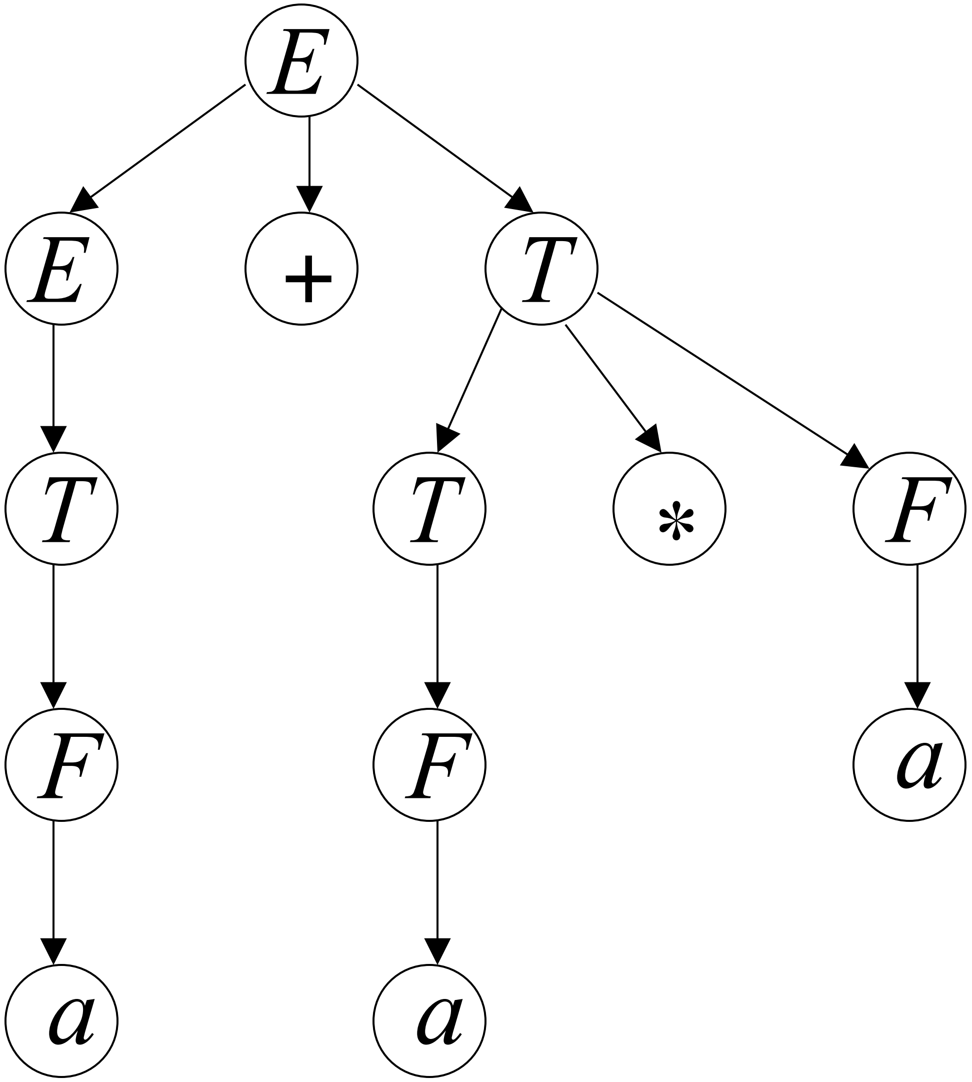
Non-Ambiguous Grammar

$$\boxed{\begin{array}{l} E \to E + T \mid T \\ T \to T * F \mid F \\ F \to (E) \mid a \end{array}}$$

generates the same language

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

$$\boxed{\begin{array}{l} E \rightarrow E+T \mid T \\ T \rightarrow T*F \mid F \\ F \rightarrow (E) \mid a \end{array}}$$



Unique derivation tree for $a+a*a$

An unsuccessful 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:

$$\begin{tikzcd}
L = \arrow[d] & \{a^n b^n c^m\} \cup \arrow[d] & \{a^n b^m c^m\} \arrow[d] \\
S \to S_1 \mid S_2 & S_1 \to S_1 c \mid A & S_2 \to a S_2 \mid B \\
 & A \to aAb \mid \varepsilon & B \to bBc \mid \varepsilon
\end{tikzcd}$$

Another unsuccessful example:

L human language = inherently ambiguous:

The boy saw the man on the hill
with the telescope