Context-Free Languages

Course Instructor: Hashim Ayub

Book: Prof. Sipser-MIT
Slides: Prof. Busch - LSU

context-free grammar (CFG)

In formal language theory, a context-free grammar (CFG) is a grammar in which every production rule is of the form

$V \rightarrow w$

where V is a single nonterminal symbol, and w is a string of terminals and/or nonterminals (possibly empty). The term "context-free" expresses the fact that nonterminals can be rewritten without regard to the context in which they occur. A formal language is context-free if some context-free grammar generates it.

Context-free grammars play a central role in the description and design of programming languages and compilers. They are also used for analyzing the syntax of natural languages.

(Ref. http://en.wikipedia.org/wiki/Content_free_grammar)

context-sensitive grammar (CSG)

is a grammar where each production has the form

$wAx \rightarrow wyx$,

where w and x are strings of terminals and nonterminals and y is also a string of terminals.

In other words, the productions give rules saying "if you see *A in a given context*, you may replace A by the string y." It's an unfortunate that these grammars are called "context-sensitive grammars" because it means that "context-free" and "context-sensitive" are not opposites, and it means that there are certain classes of grammars that arguably take a lot of contextual information into account but aren't formally considered to be context-sensitive.

context-sensitive grammar (CSG)

A context-sensitive grammar (CSG) is a formal grammar in which the left-hand sides and right-hand sides of any production rules may be surrounded by a context of terminal and nonterminal symbols. Context-sensitive grammars are more general than context-free grammars but still orderly enough to be parsed by a **linear bounded automaton**.

The concept of context-sensitive grammar was introduced by Noam Chomsky in the 1950s as a way to describe the syntax of natural language where it is indeed often the case that a word may or may not be appropriate in a certain place depending upon the context. A formal language that can be described by a context-sensitive grammar is called a context-sensitive language.

(Ref. http://en.wikipedia.org/wiki/Context-sensitive_grammar)

Recursively enumerable language

In mathematics, logic and computer science, a formal language is called **recursively enumerable** (also **recognizable**, **partially decidable**, **semidecidable**, **Turing-acceptable** or **Turing-recognizable**) if it is a recursively enumerable subset in the set of all possible words over the alphabet of the language, i.e., if there exists a Turing machine which will enumerate all valid strings of the language.

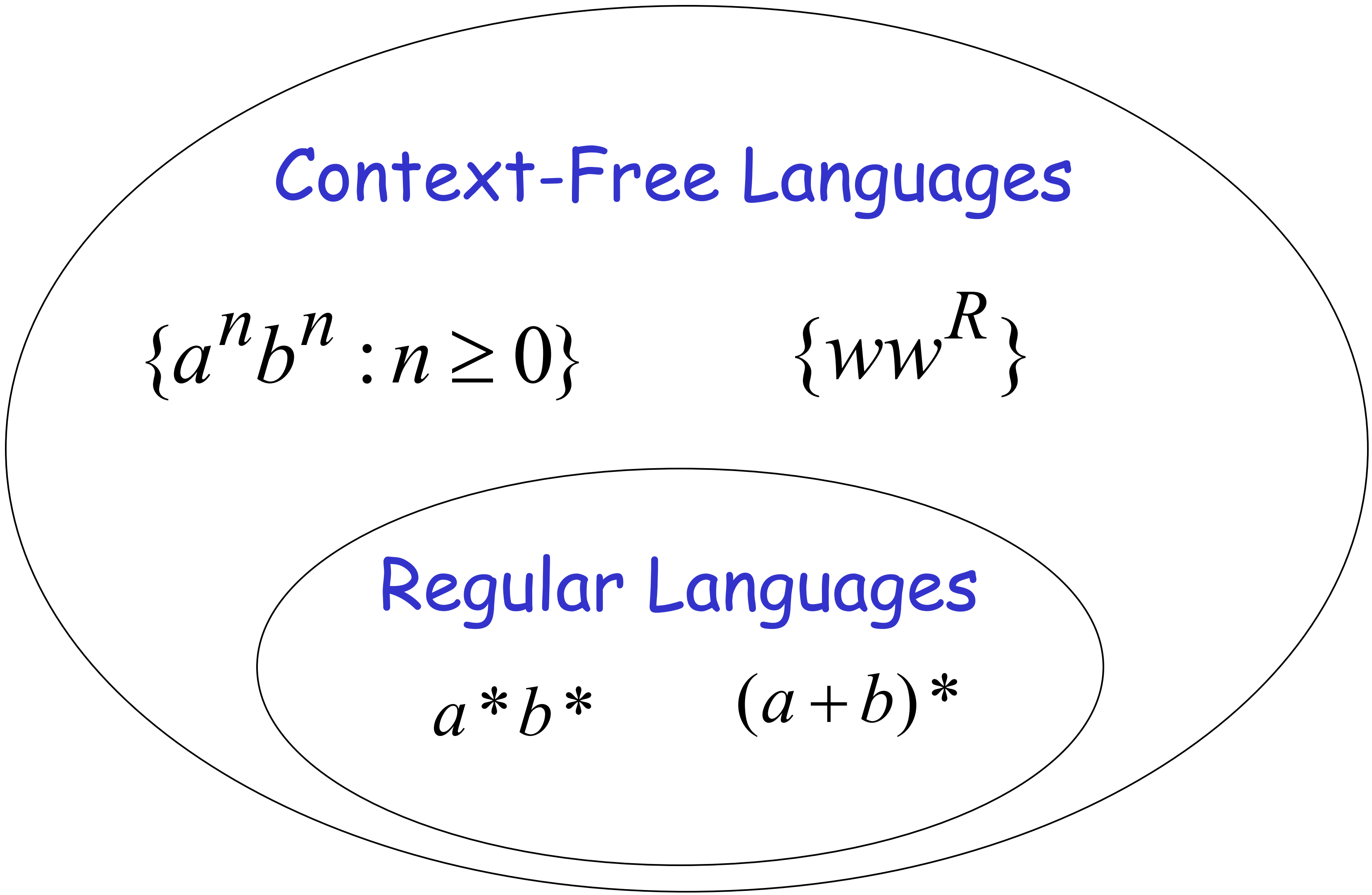
Recursively enumerable languages are known as **type-0** languages in the Chomsky hierarchy of formal languages. All regular, context-free, context-sensitive and recursive languages are recursively enumerable.

The class of all recursively enumerable languages is called RE.

(Ref. https://en.wikipedia.org/wiki/Recursively_enumerable_language)

Some Application Area of Context Free Languages

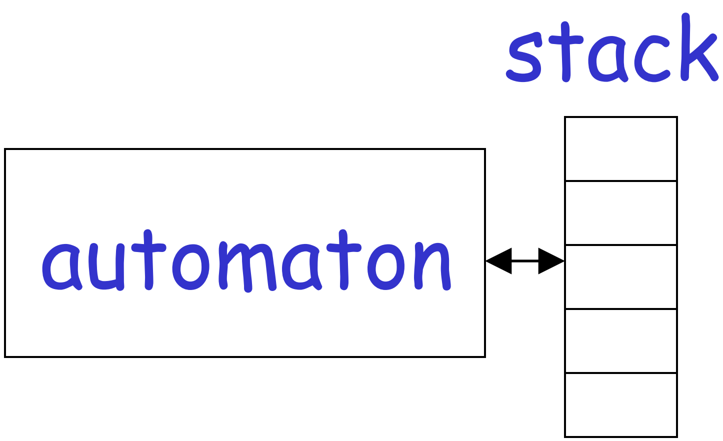
- Natural Language understanding (AI)
- Formulation of New Programming Languages
- Compiler Construction and Optimization


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

Context-Free Grammar (CFG) is a generator of the Context-free language (CFL)

&

A recognizer [Push down automata (PDA)] will be an acceptor

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 sleeps":

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

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

Sequence of Terminals (symbols)

$$\langle noun \rangle \rightarrow \overbrace{cat}$$

$$\langle sentence \rangle \rightarrow \underbrace{\langle noun\_phrase \rangle \ \langle predicate \rangle}$$

Variables

Sequence of Variables

Another Example

Sequence of
terminals and variables

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

$S \rightarrow \lambda$

Variable

The right side
may be λ

Grammar: $S \to aSb$

$S \to \lambda$

Derivation of string ab:

$$S \Rightarrow aSb \Rightarrow ab$$

$S \to aSb$ $\qquad\qquad$ $S \to \lambda$

Grammar: $S \rightarrow aSb$

$S \rightarrow \lambda$

Derivation of string $aabb$:

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

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

Grammar: $S \to aSb$

$S \to \lambda$

Other derivations:

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

$$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb$$
$$\Rightarrow aaaaSbbbb \Rightarrow aaaabbbb$$

Grammar:

$$S \to aSb$$

$$S \to \lambda$$

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

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

$$S \to aSb$$

$$S \to \lambda$$

Possible Derivations

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

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

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

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

Another convenient notation:

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

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

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

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

Formal Definitions

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

Set of variables

Set of terminals

Start variable

Set of productions

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

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

Consider the following example grammar with 5 productions:

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

Example of Context-Free Grammar

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

productions

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

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

$$V = \{S\}$$

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 λ

Example:

context-free grammar G : $\boxed{S \rightarrow aSb \mid \lambda}$

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

Since, there is derivation

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

Context-Free Language definition:

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 :

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

generates $L(G) = L$

Another Example

Context-free grammar G :

$$S \to aSa \mid bSb \mid \lambda$$

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:

$$S \rightarrow aSb \mid SS \mid \lambda$$

Example derivations:

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

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

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

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

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

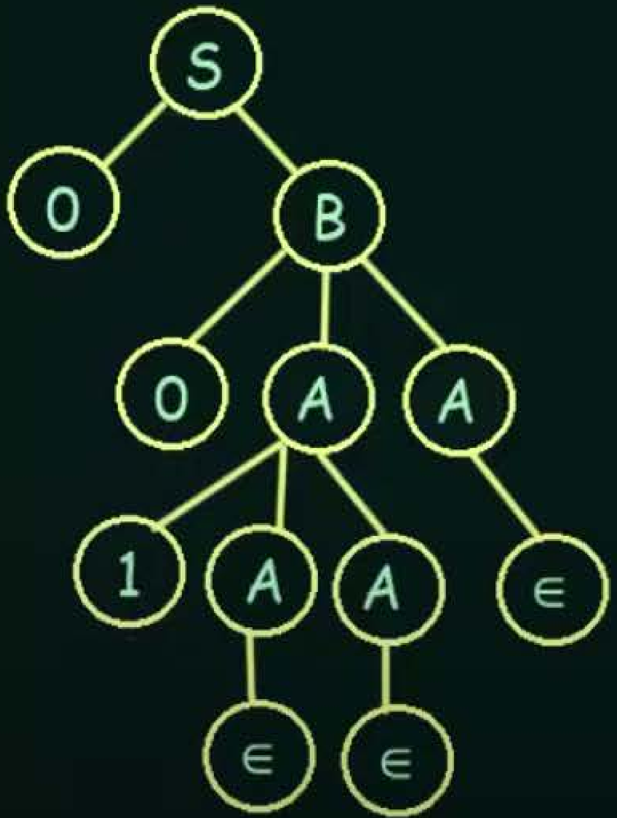
Describes matched parentheses: () ((( ))) (( )) $a = ($, $\mathrm{b} =)$

Derivation Order
and
Derivation Trees

Derivation Tree

A Derivation Tree or Parse Tree is an ordered rooted tree that graphically represents the semantic information of strings derived from a Context Free Grammar

Example: For the Grammar $G = \{V, T, P, S\}$ where $S \rightarrow 0B$, $A \rightarrow 1AA \mid \epsilon$, $B \rightarrow 0AA$



Root Vertex: Must be labelled by the Start Symbol

Vertex: Labelled by Non-Terminal Symbols

Leaves: Labelled by Terminal Symbols or ϵ

Left Derivation Tree

A Left Derivation Tree is obtained by applying production to the leftmost variable in each step.

Right Derivation Tree

A Right Derivation Tree is obtained by applying production to the rightmost variable in each step.

Eg. For generating the string aabaa from the Grammar $S \rightarrow aAS \mid aSS \mid \epsilon$, $A \rightarrow SbA \mid ba$

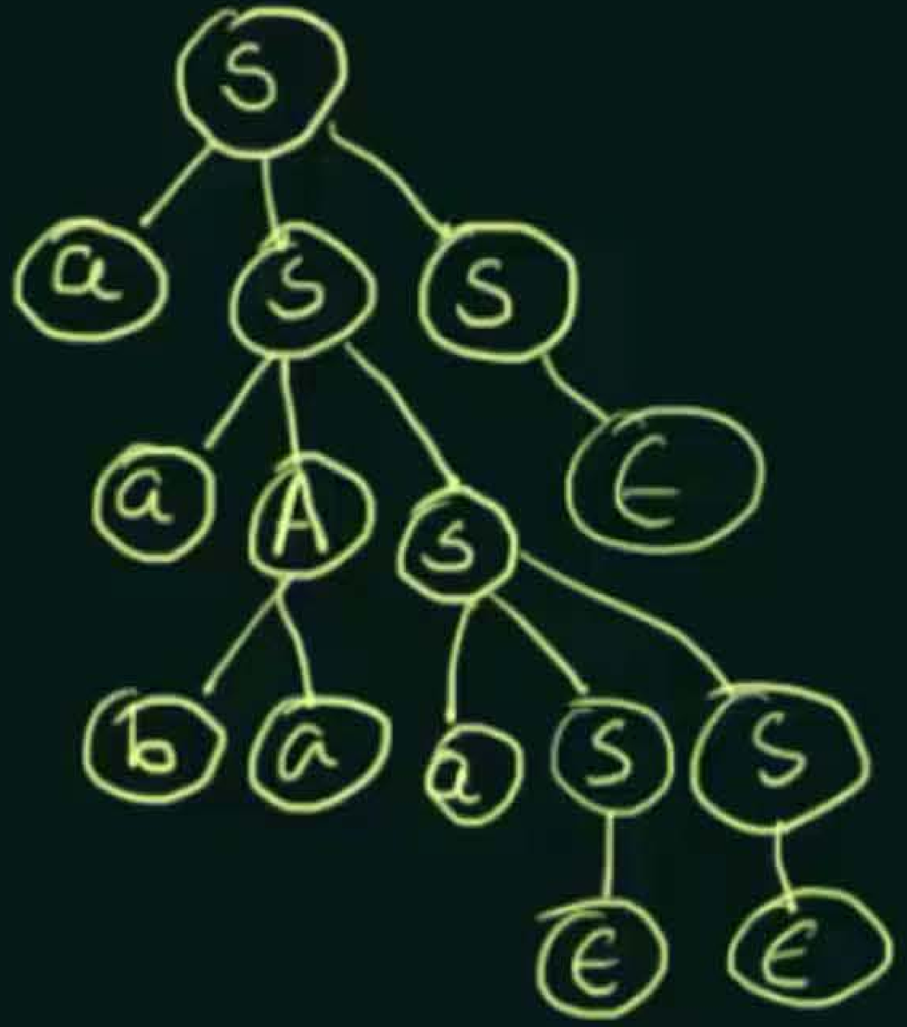
Left Derivation Tree

A Left Derivation Tree is obtained by applying production to the leftmost variable in each step.

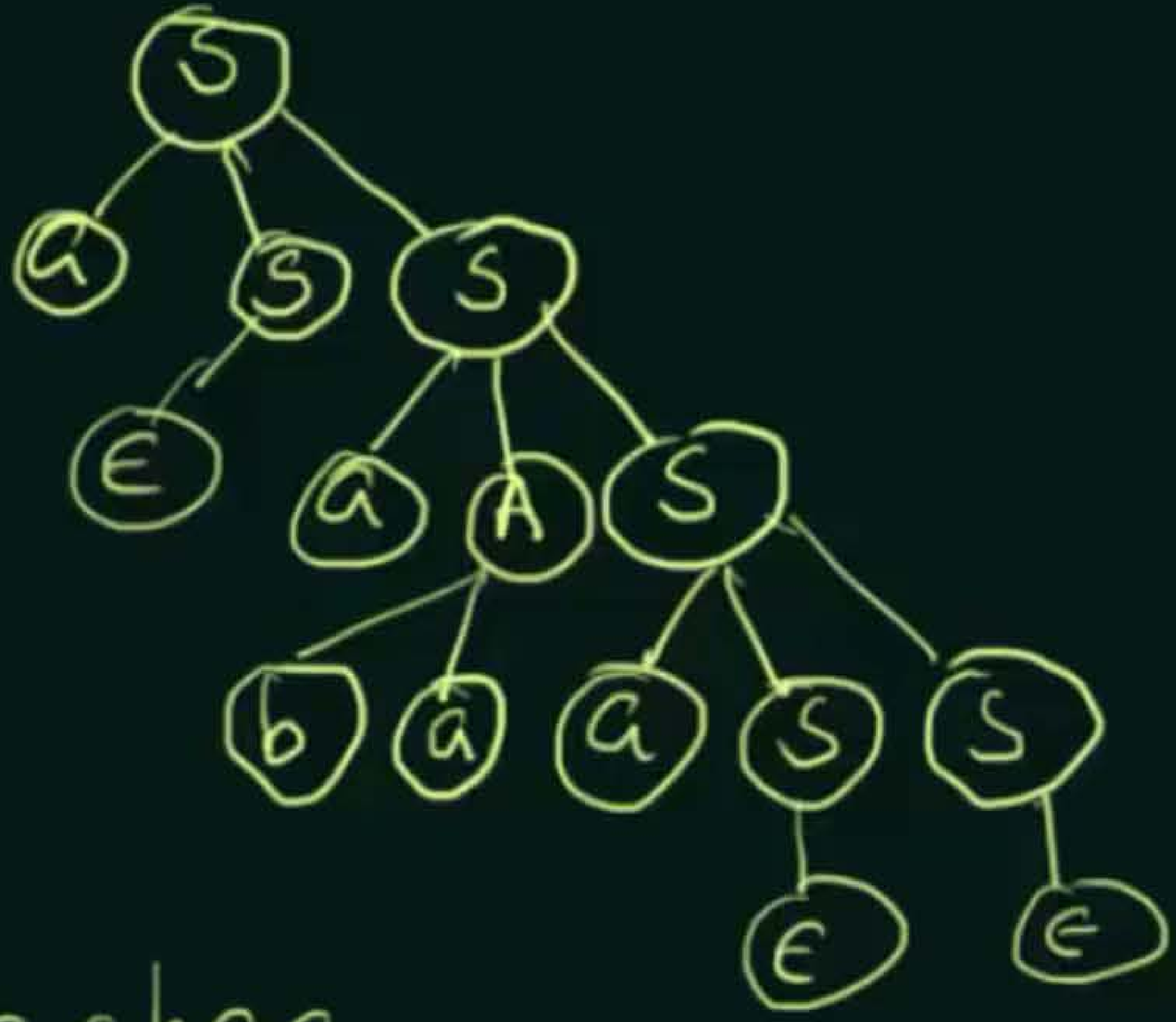
Right Derivation Tree

A Right Derivation Tree is obtained by applying production to the rightmost variable in each step.

Eg. For generating the string aabaa from the Grammar $S \to aAS \mid aSS \mid \epsilon$, $A \to SbA \mid ba$



aabaa



aabaa

Derivation Order

Consider the following example grammar with 5 productions:

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

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

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

$3.\ A \rightarrow \lambda$ $\quad 5.\ B \rightarrow \lambda$

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

$$1.\ S \to AB \qquad 2.\ A \to aaA \qquad 4.\ B \to Bb$$
$$3.\ A \to \lambda \qquad 5.\ B \to \lambda$$

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

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

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:

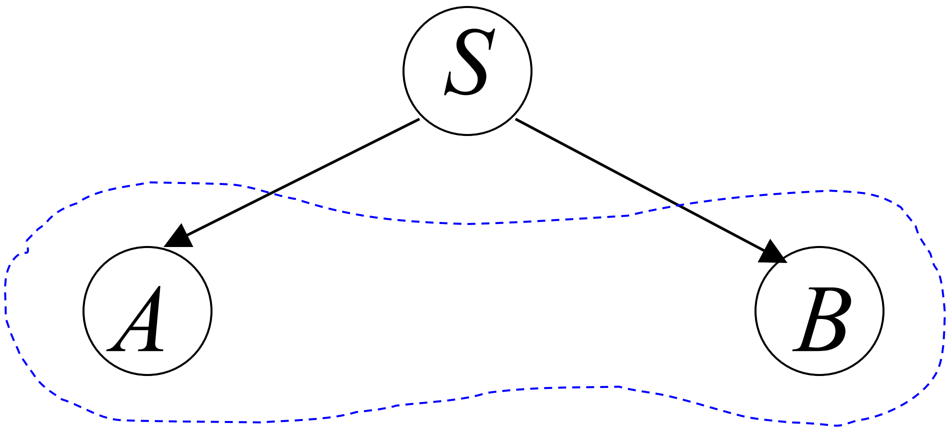
$$S \rightarrow AB \qquad A \rightarrow aaA \mid \lambda \qquad B \rightarrow Bb \mid \lambda$$

And a derivation of aab:

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

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

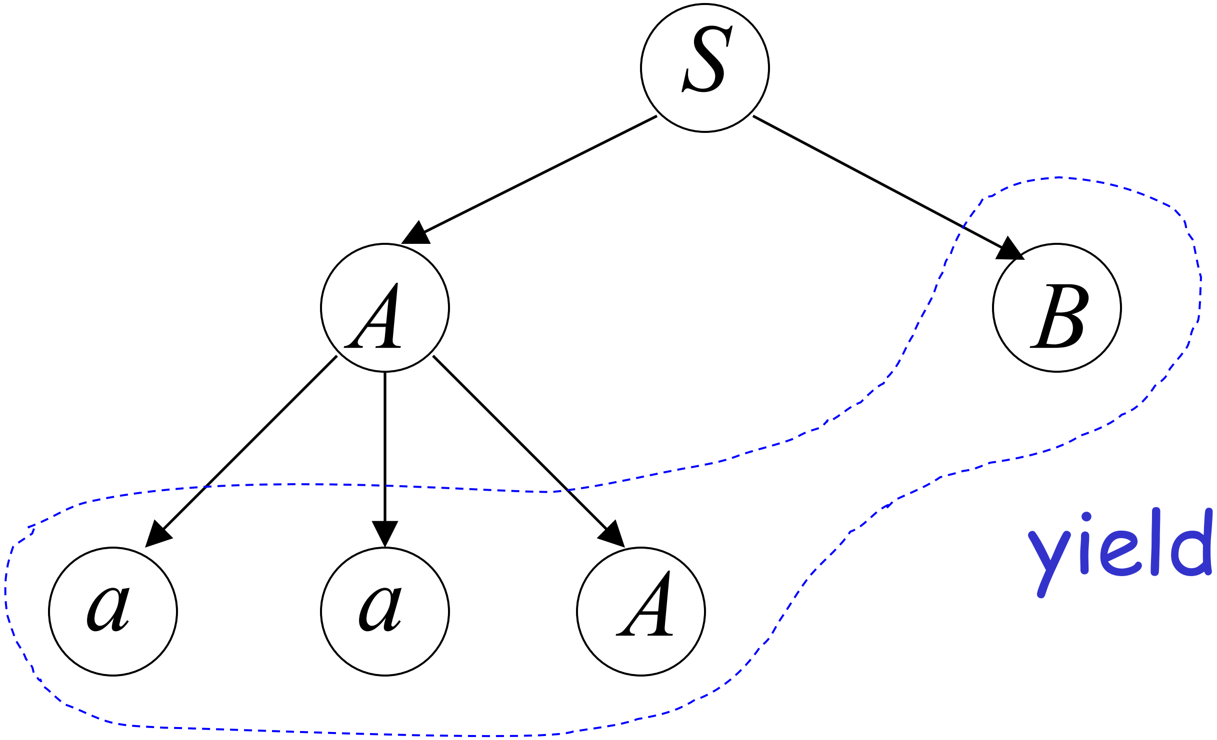
$$S \Rightarrow AB$$



yield AB

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

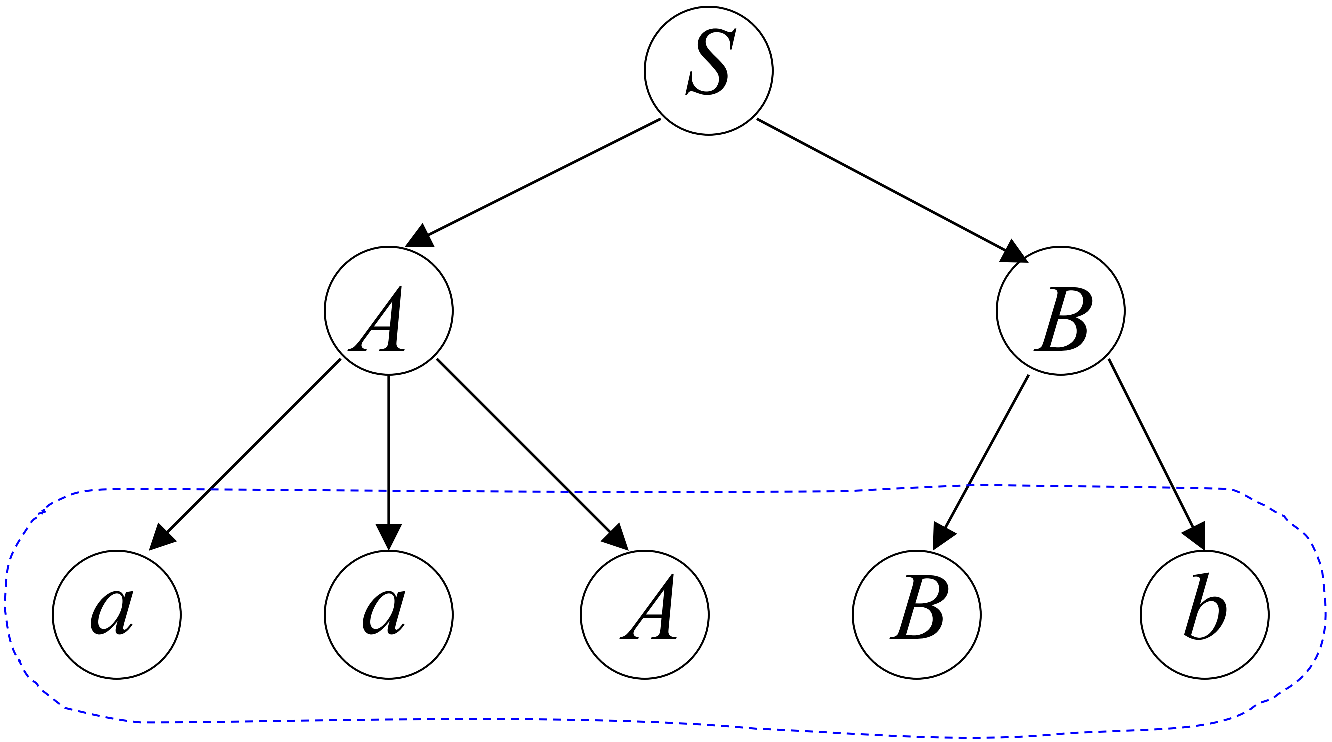
$$S \Rightarrow AB \Rightarrow aaAB$$



yield $aaAB$

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

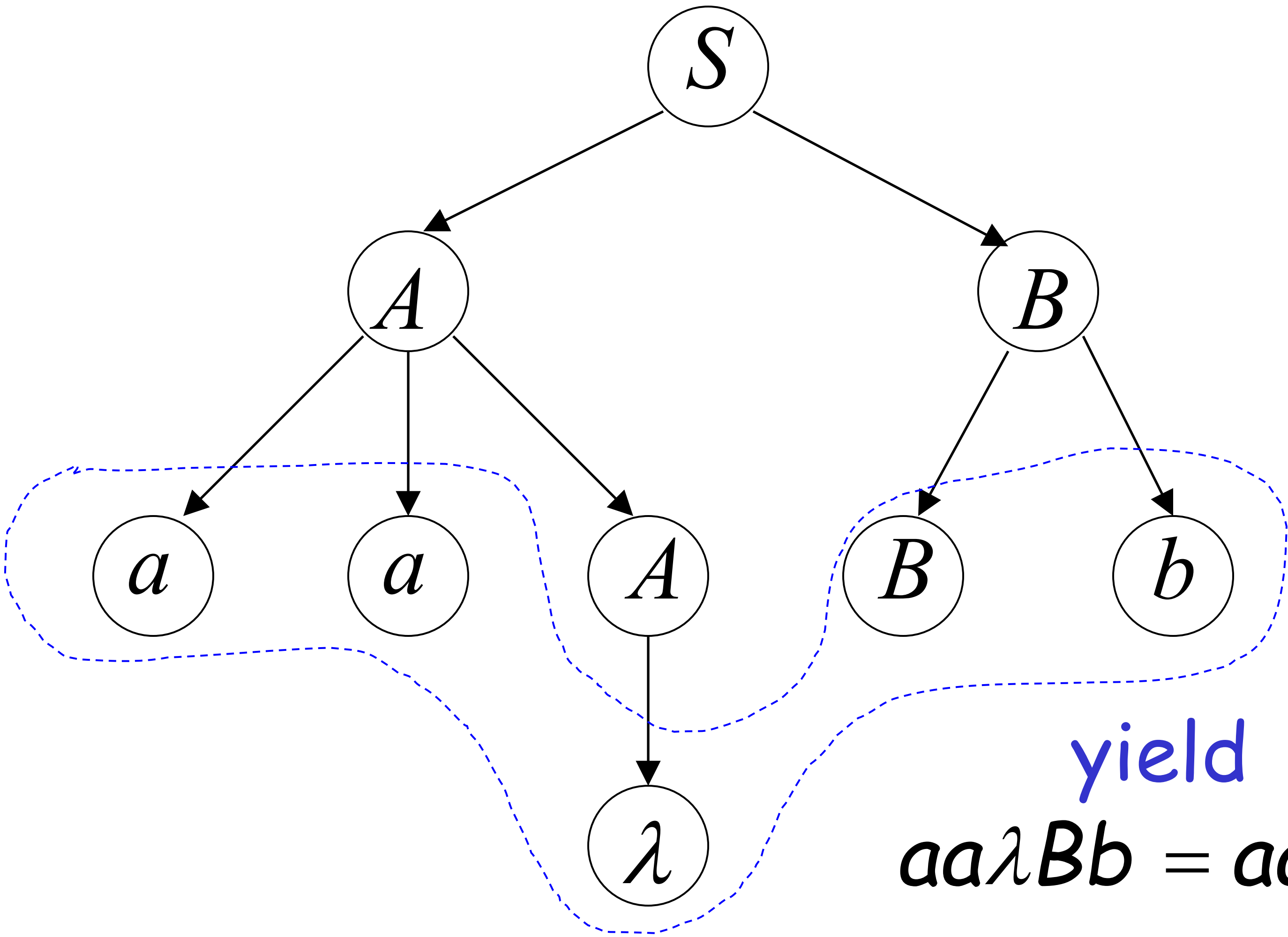
$$S \Rightarrow AB \Rightarrow aaAB \Rightarrow aaABb$$



yield *aaABb*

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

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



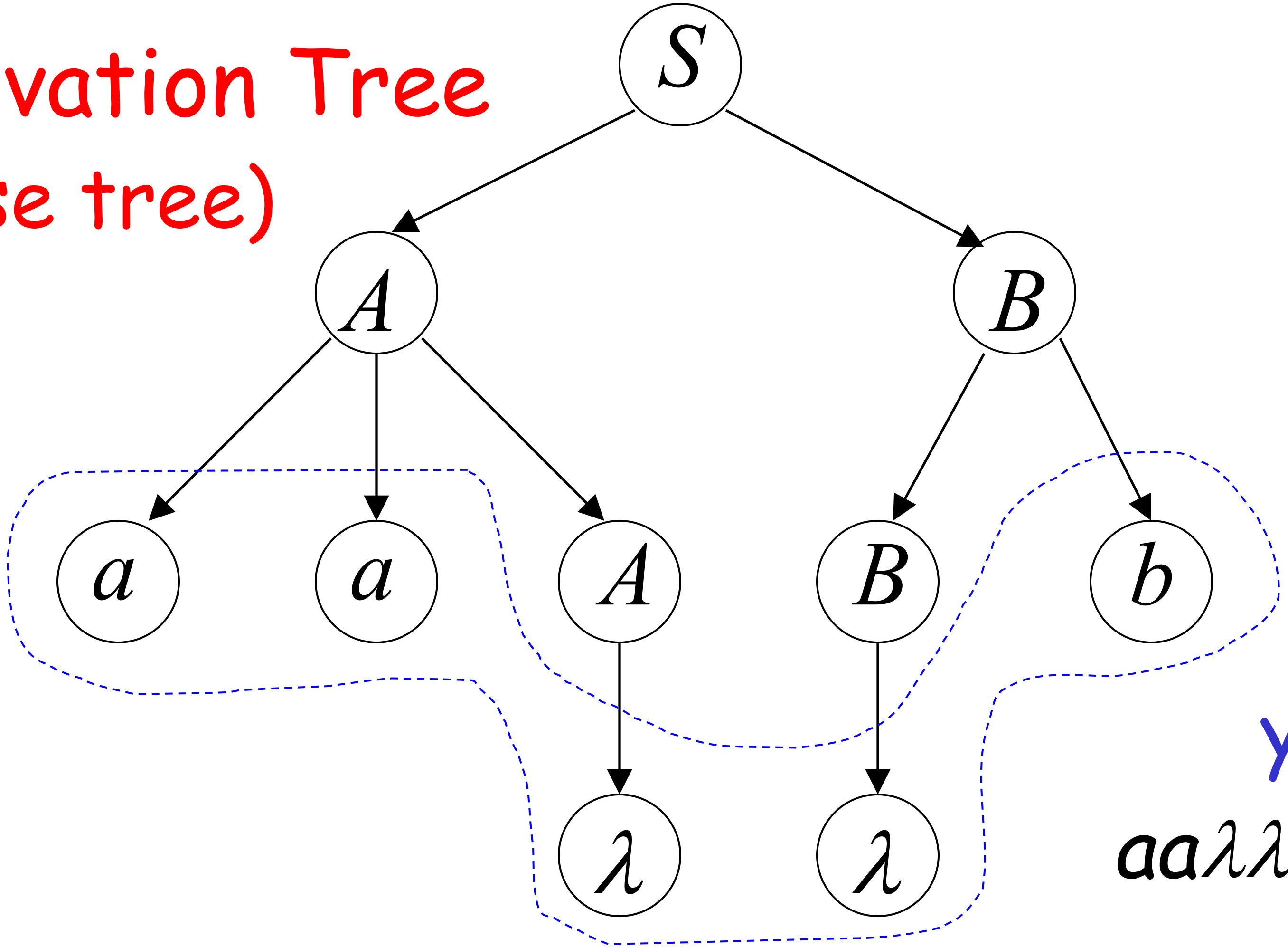
yield

$aa\lambda Bb = aaBb$

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

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

Derivation Tree
(parse tree)



yield

$aa\lambda\lambda b = aab$

Sometimes, derivation order doesn't matter

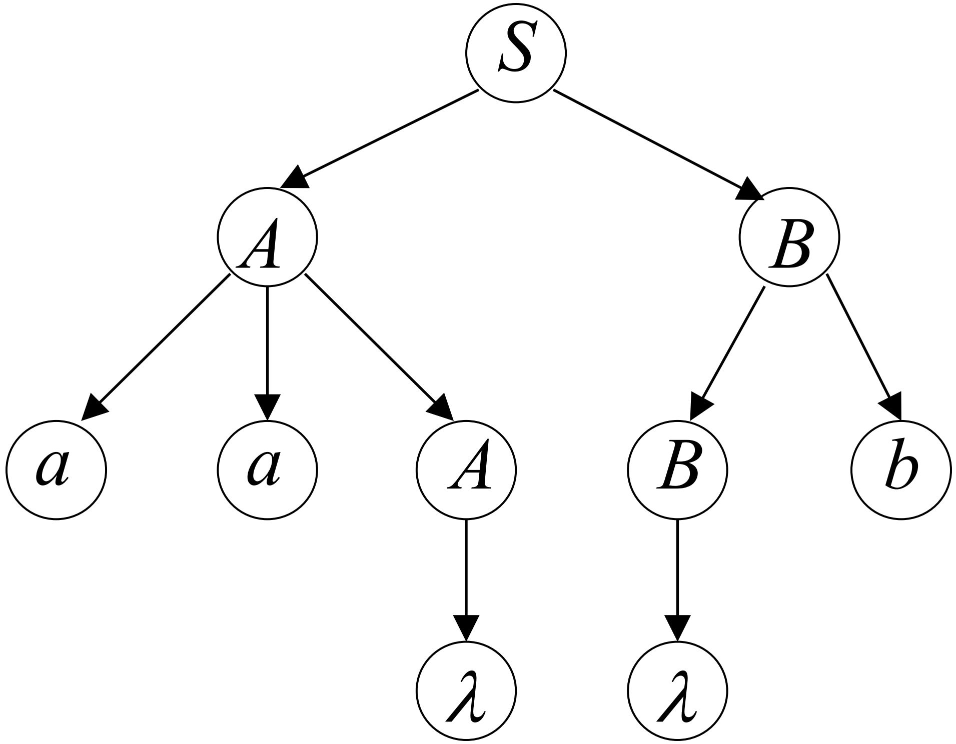
Leftmost derivation:

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

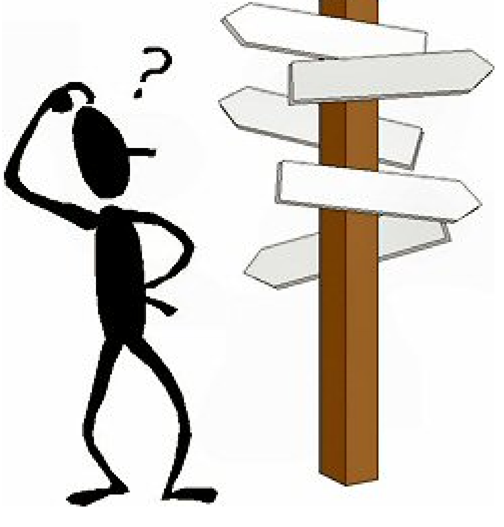
Rightmost derivation:

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

Give same derivation tree



Ambiguity



Grammar for mathematical expressions

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

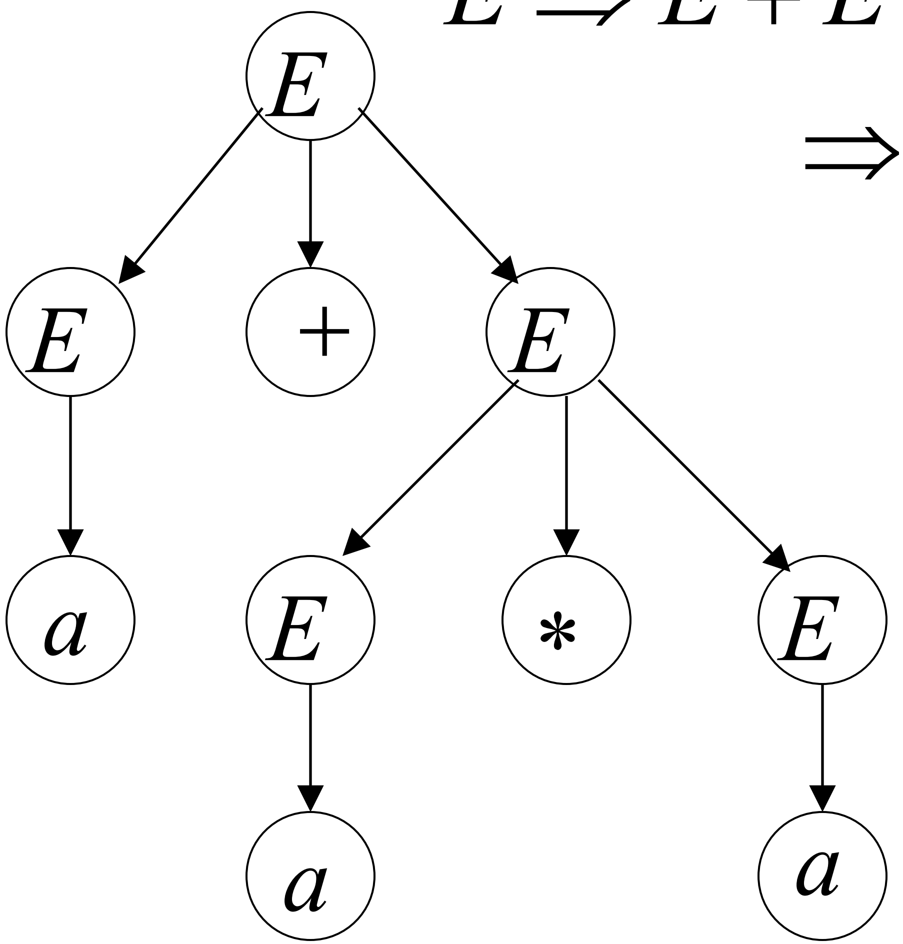
Example strings:

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

Denotes any number

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

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

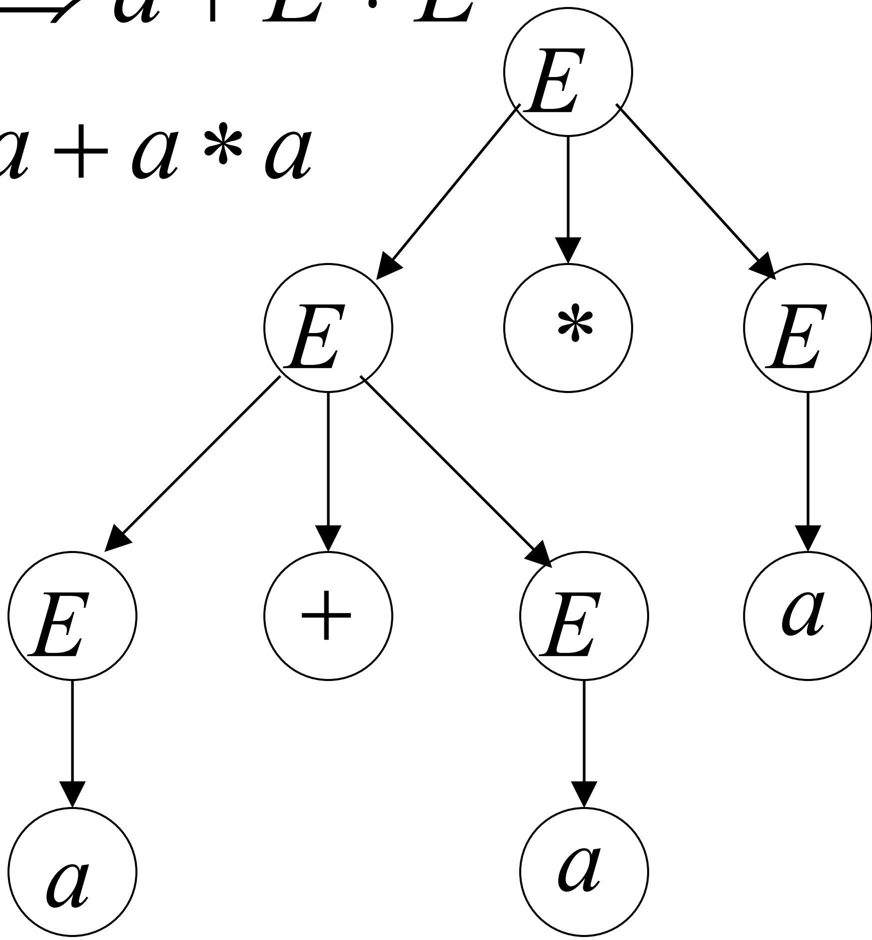


A leftmost derivation
for $a + a * a$

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

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

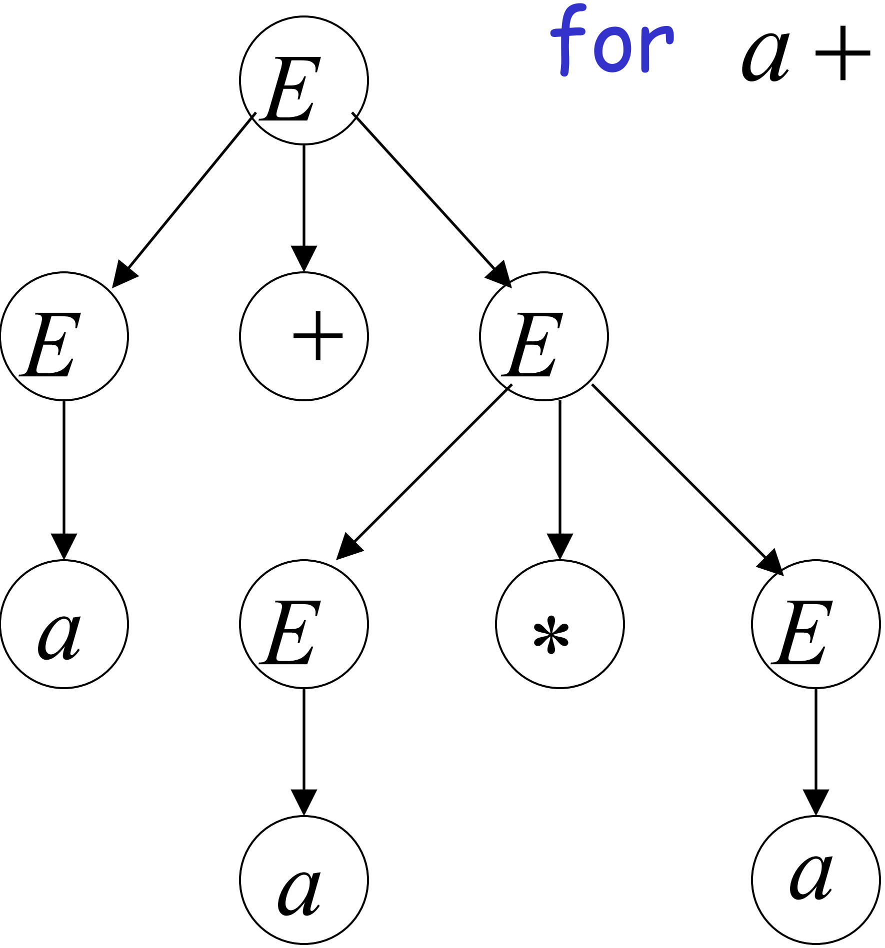
Another leftmost derivation for $a + a * a$

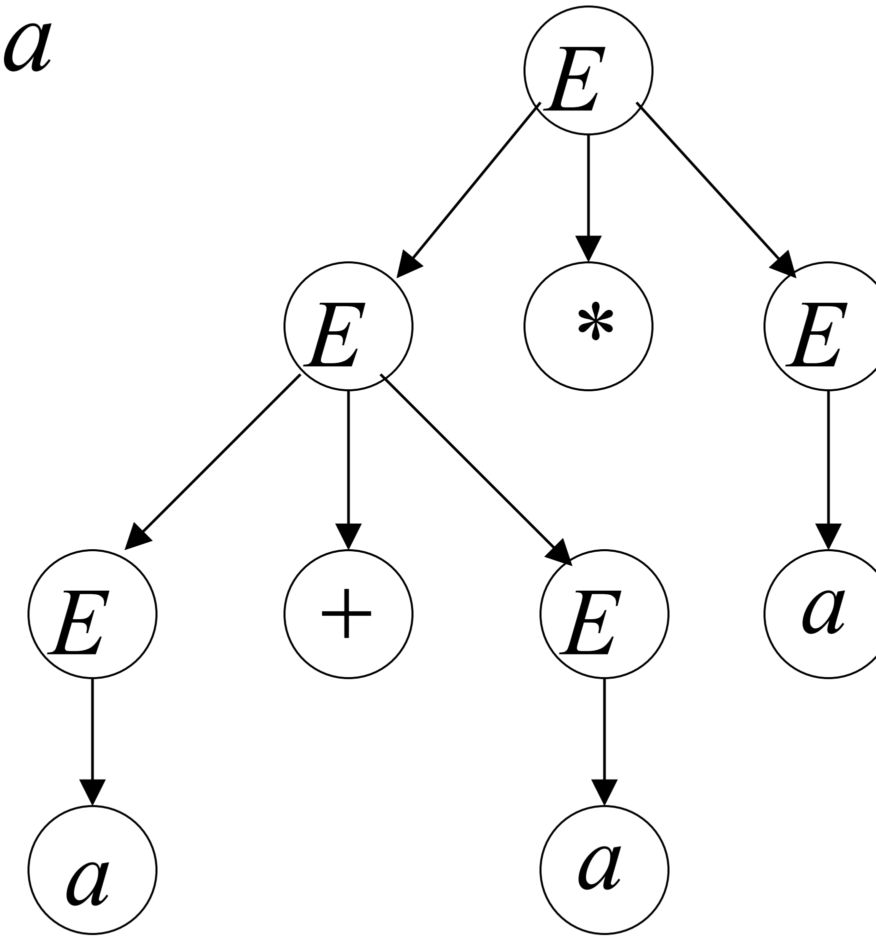


$$E \to E + E \mid E * E \mid (E) \mid a$$

Two derivation trees

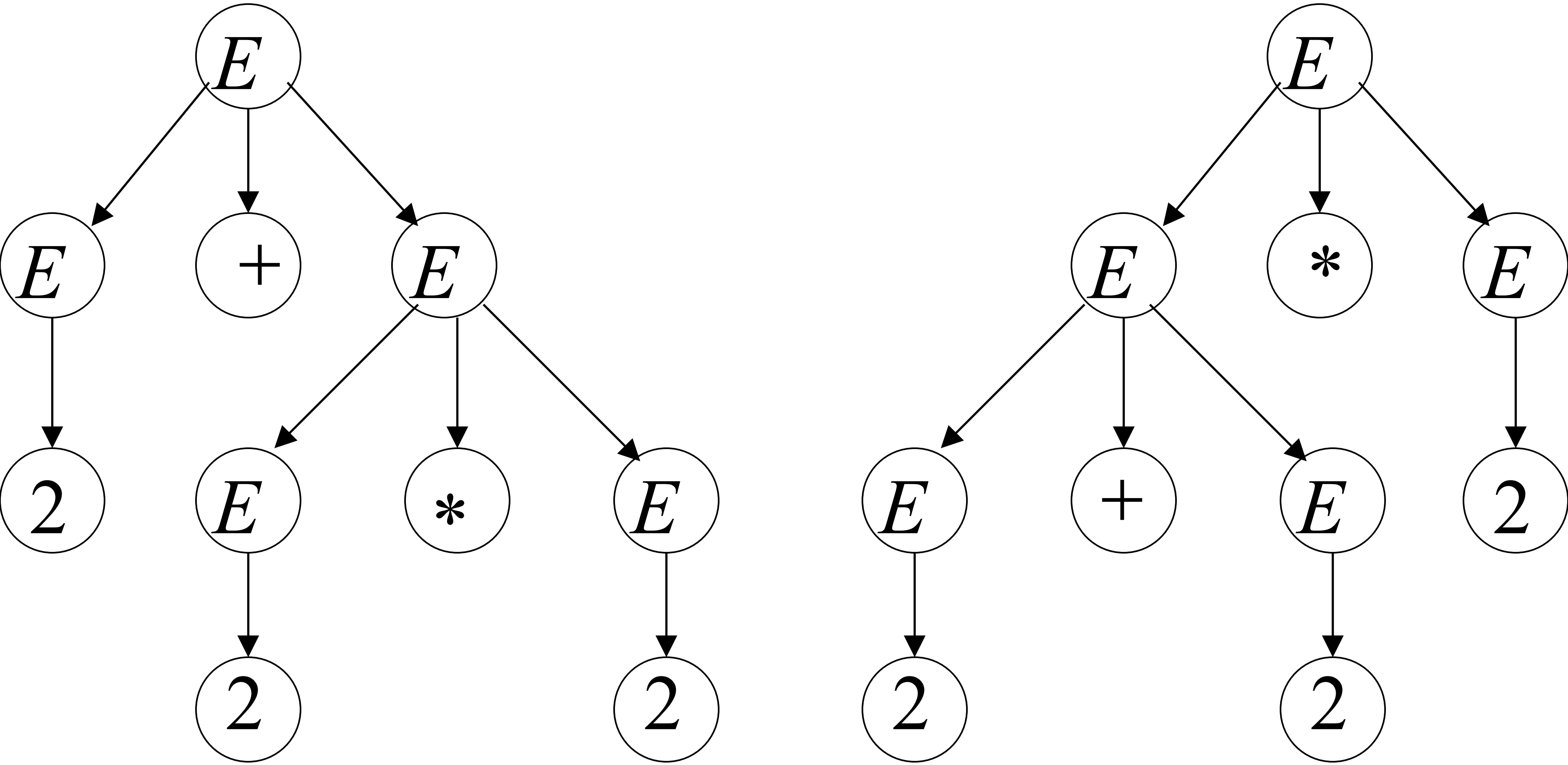
for $a + a * a$





take $a = 2$

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



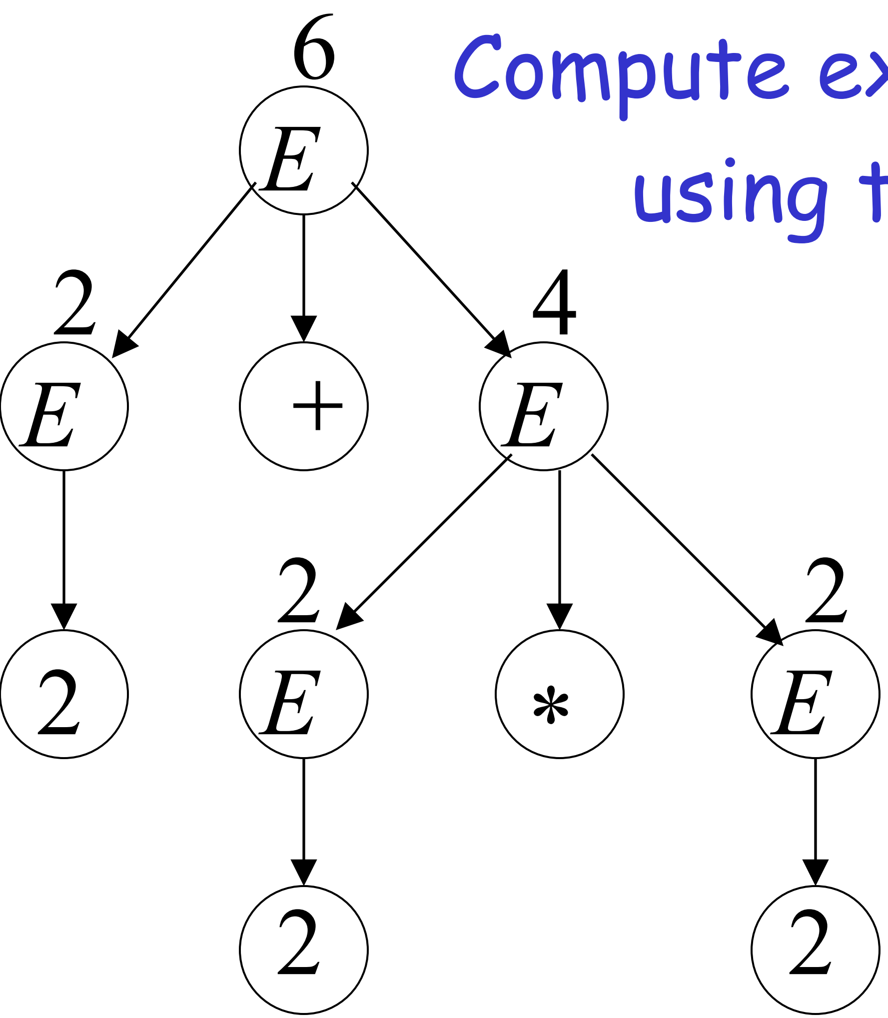
Good Tree

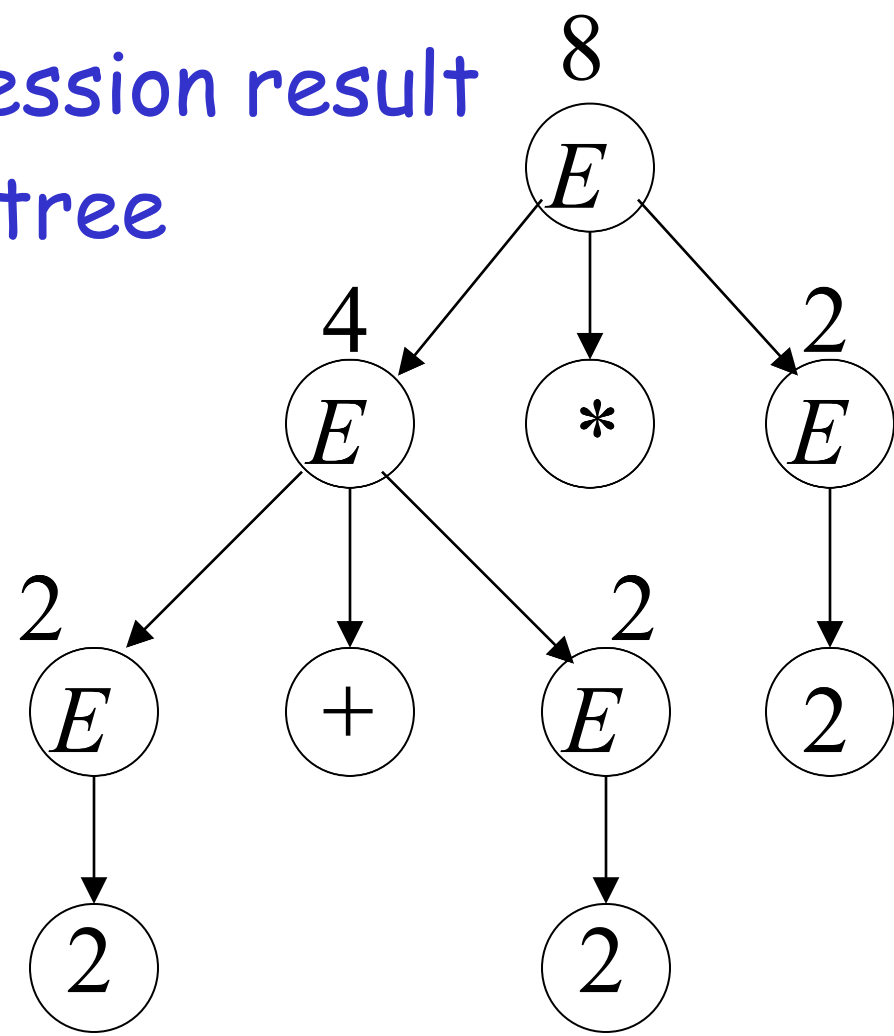
$$2+2*2=6$$

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:

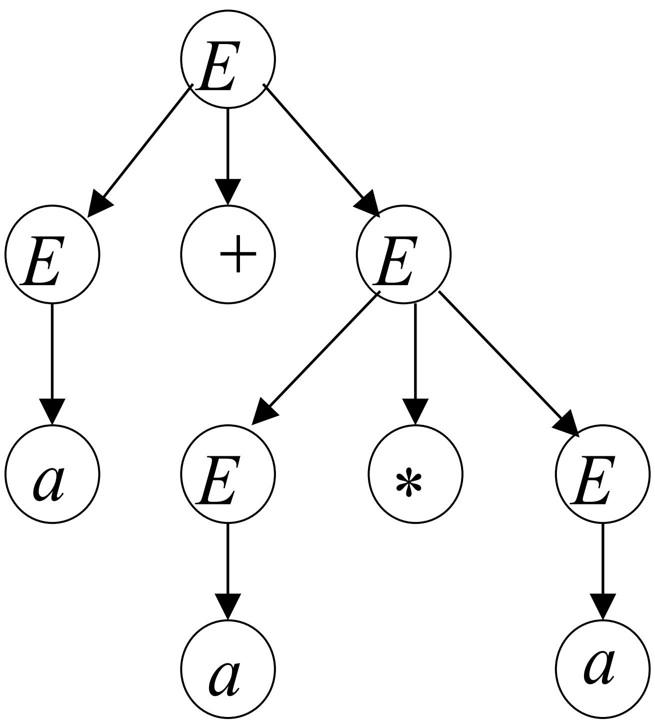
two different derivation trees
or
two leftmost derivations

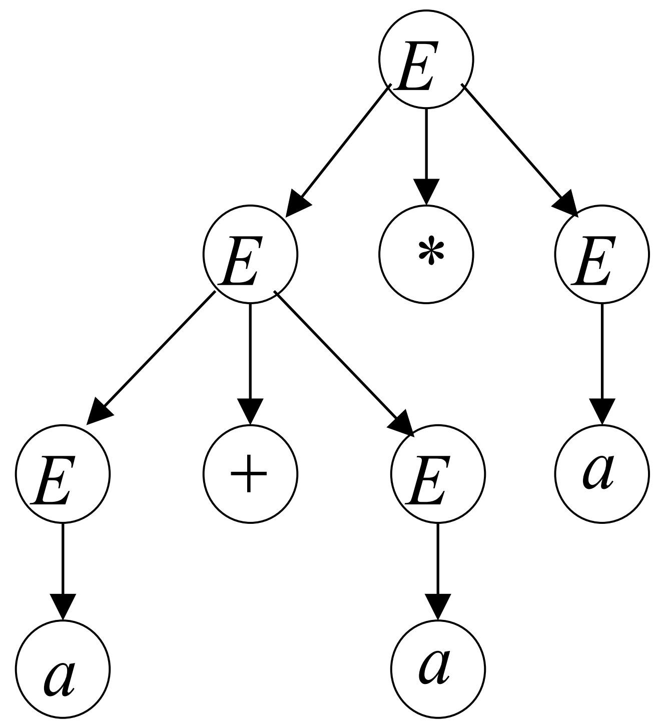
(Two different derivation trees give two different leftmost derivations and vice-versa)

Example: $E \to E + E \;|\; E * E \;|\; (E) \;|\; a$

this grammar is ambiguous since

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





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

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:

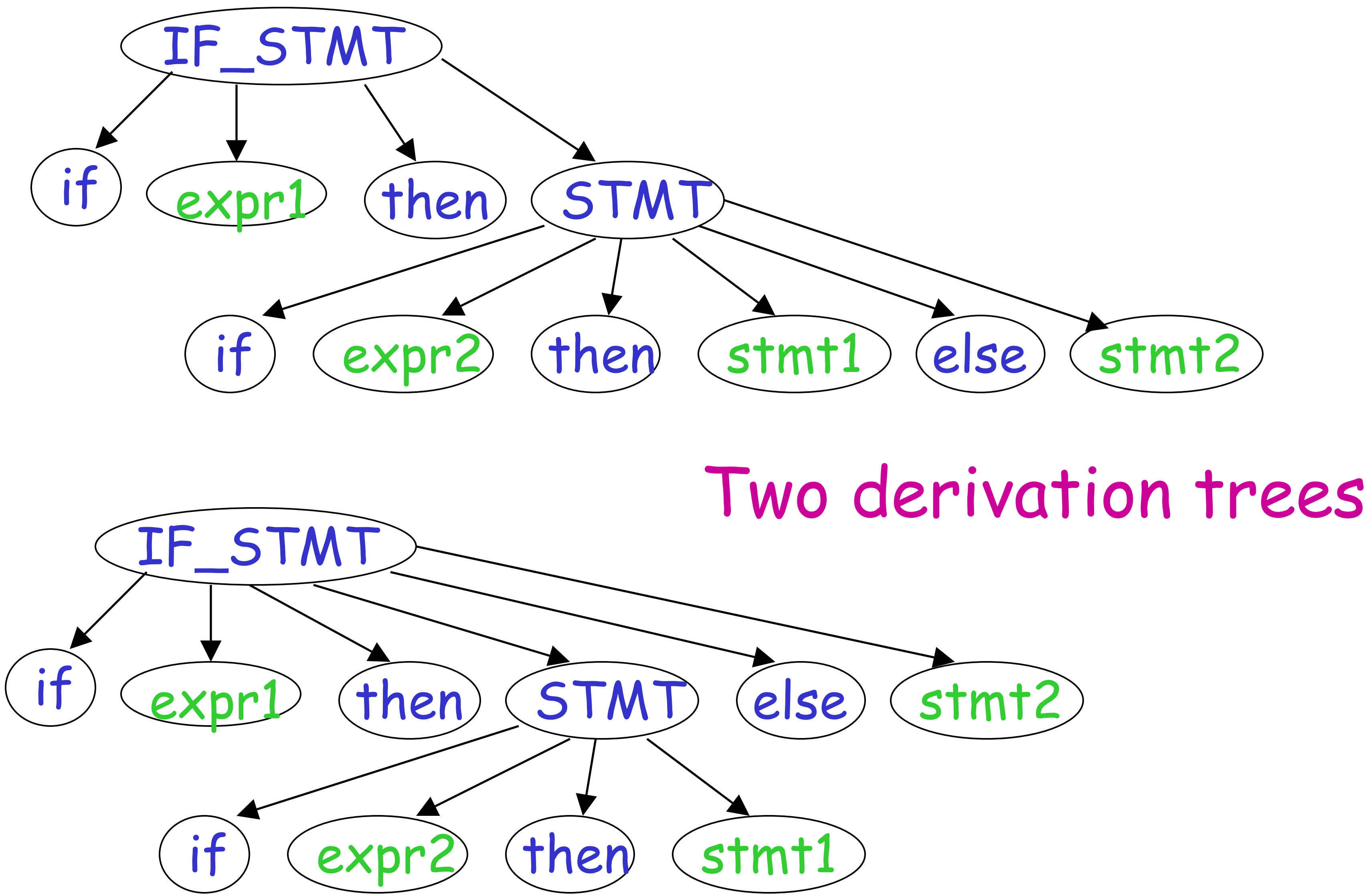
$$\begin{aligned} IF\_STMT \rightarrow & \text{ if } EXPR \text{ then } STMT \\ | & \text{ if } EXPR \text{ then } STMT \text{ else } STMT \end{aligned}$$

Variables

Terminals

Very common piece of grammar in programming languages

If expr1 then if expr2 then stmt1 else stmt2



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 it is difficult to achieve this

A successful example:

Ambiguous Grammar

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

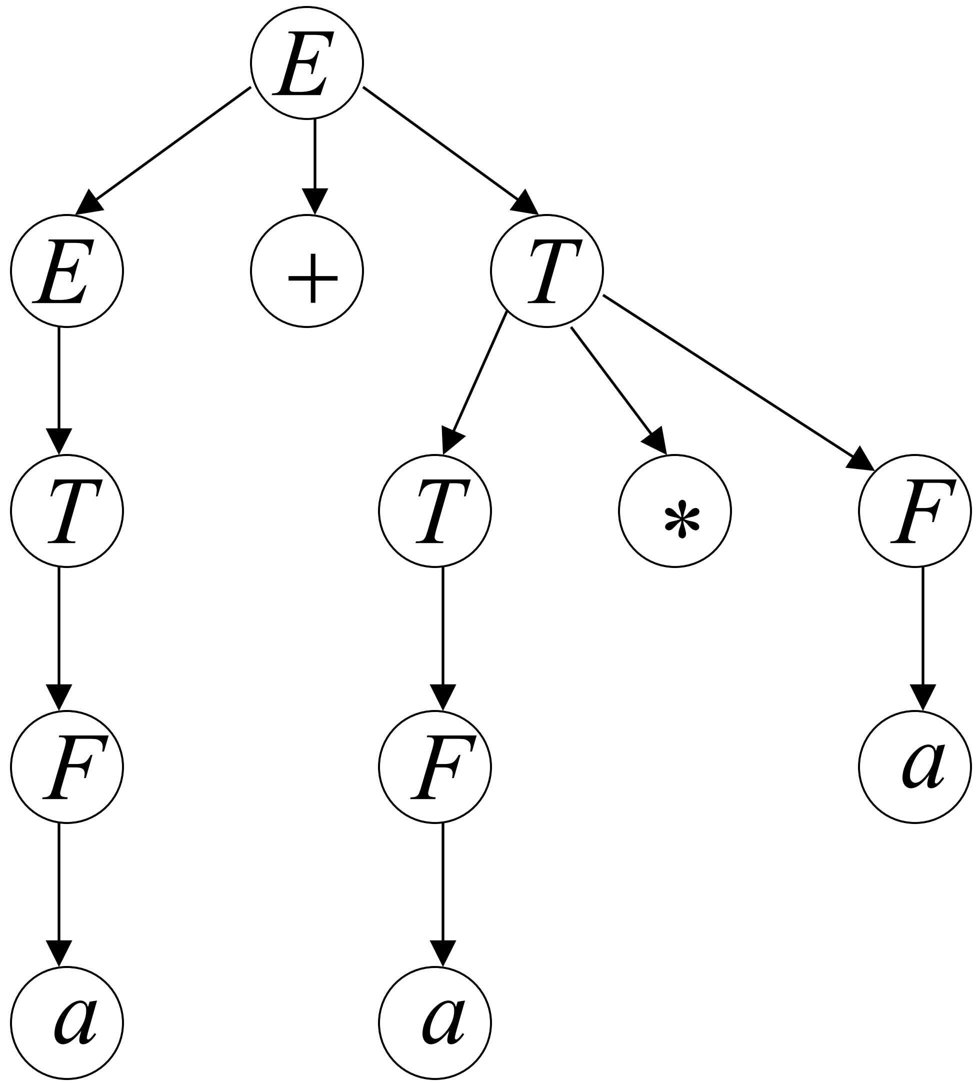
Equivalent Non-Ambiguous Grammar

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

generates the same language

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

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



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:

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

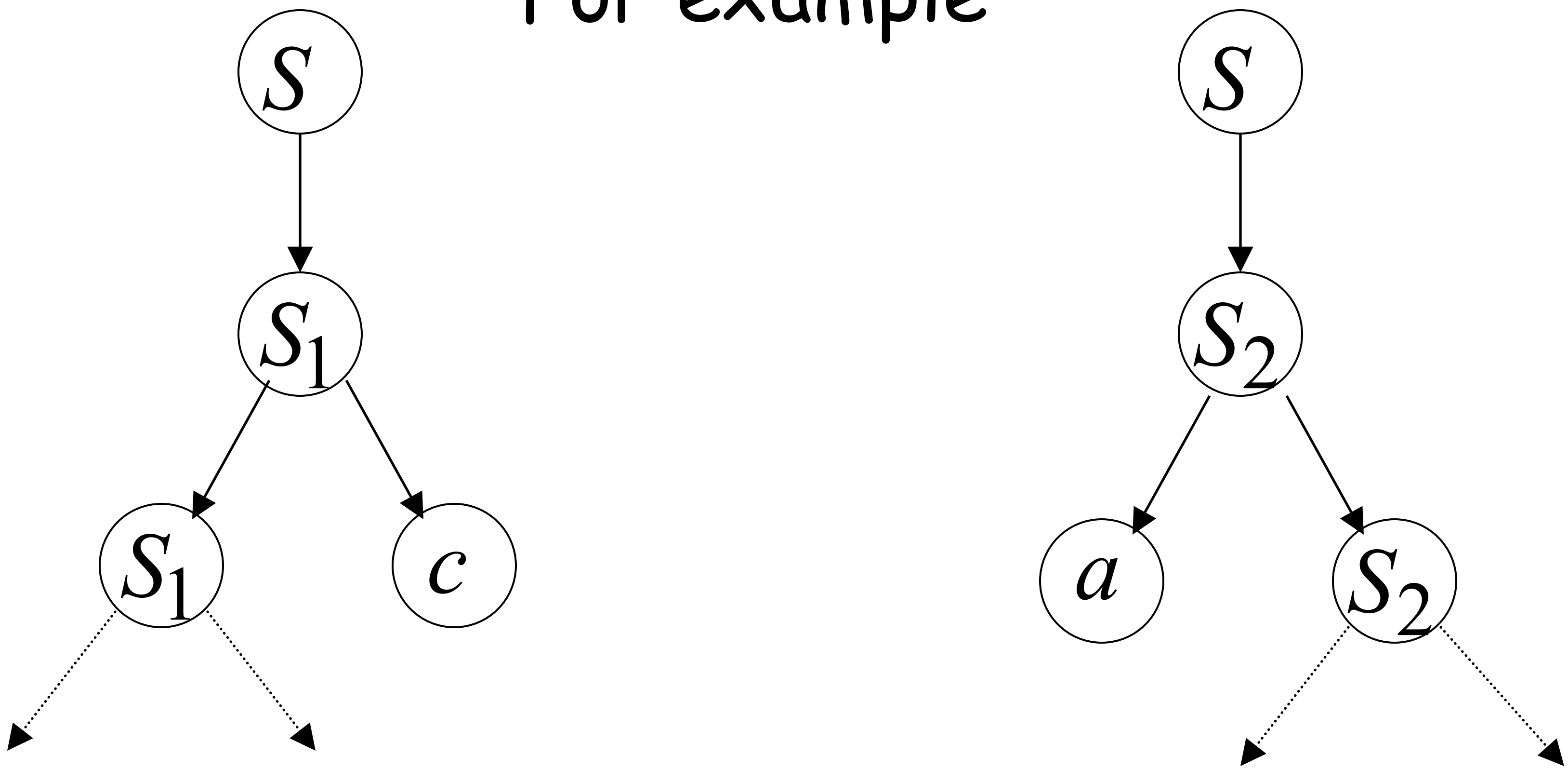
$$S \rightarrow S_1 \mid S_2 \qquad S_1 \rightarrow S_1 c \mid A \qquad S_2 \rightarrow a S_2 \mid B$$

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

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

has always two different derivation trees (for any grammar)

For example



Practical Activity 1

- The language L contains all strings over the alphabet {a,b} that begin with **a** and end with **b**, ie:

$$L = \{ab, aab, abb, aaab, aabb, abab, abbb, aaaab, \ldots\}$$

- **Write context-free grammar rules that generate the language L.**

Practical Activity 1: Possible Solution

$S \rightarrow \mathtt{a}T\mathtt{b}$, $T \rightarrow \lambda$, $T \rightarrow \mathtt{a}T$, $T \rightarrow \mathtt{b}T$ $\qquad$ S är startsymbol.

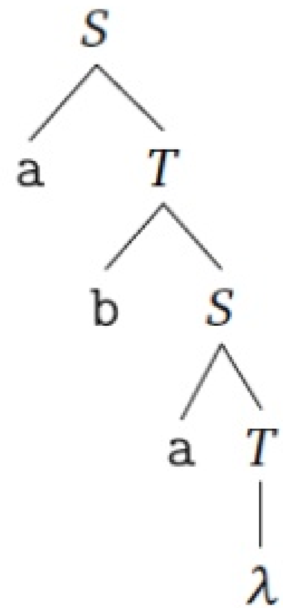
Practical Activity 2

- Look at the following CFG rules:

$$S \rightarrow \mathrm{a}T,\ T \rightarrow \lambda,\ T \rightarrow \mathrm{b}S$$

- **Draw a parse tree for the string *aba*.**

Practical Activity 2: Possible Solution



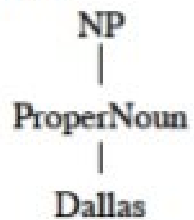
Practical Activity 3

Draw tree structures for the following phrases and sentences:

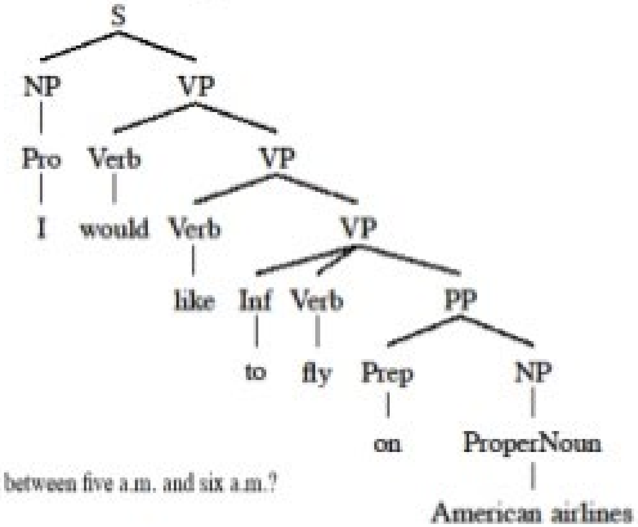
1. Dallas
2. I would like to fly on American airlines.
3. after five p.m.
4. Does American 487 have a first class section?
5. early flights
6. any delays in Denver

Practical Activity 3: Possible Solutions

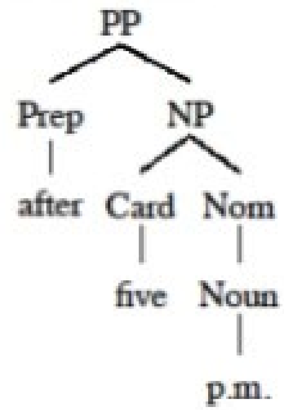
Dallas



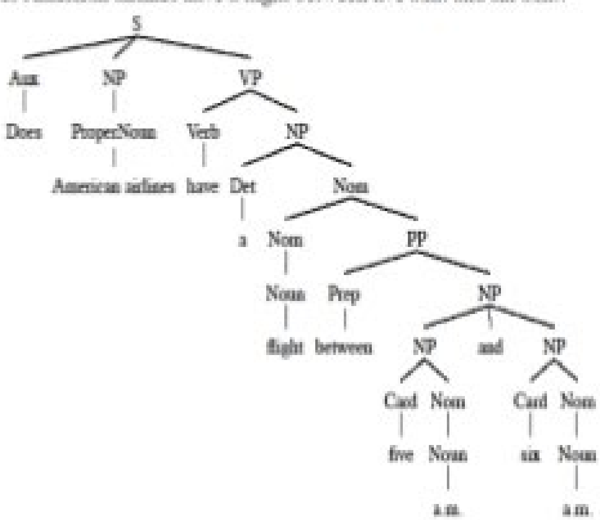
I would like to fly on American airlines.



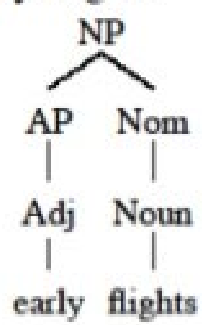
after five p.m.



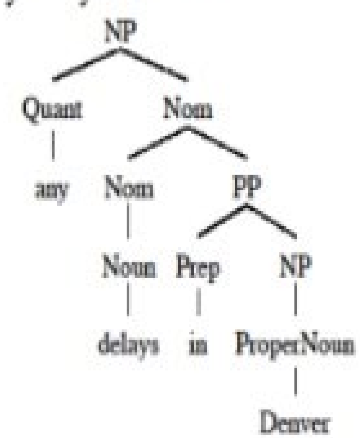
Does American airlines have a flight between five a.m. and six a.m.?



early flights



any delays in Denver



References

- **http://matt.might.net/articles/grammars-bnf-ebnf/**
- https://en.wikipedia.org/wiki/Chomsky_hierarchy

Thank You!