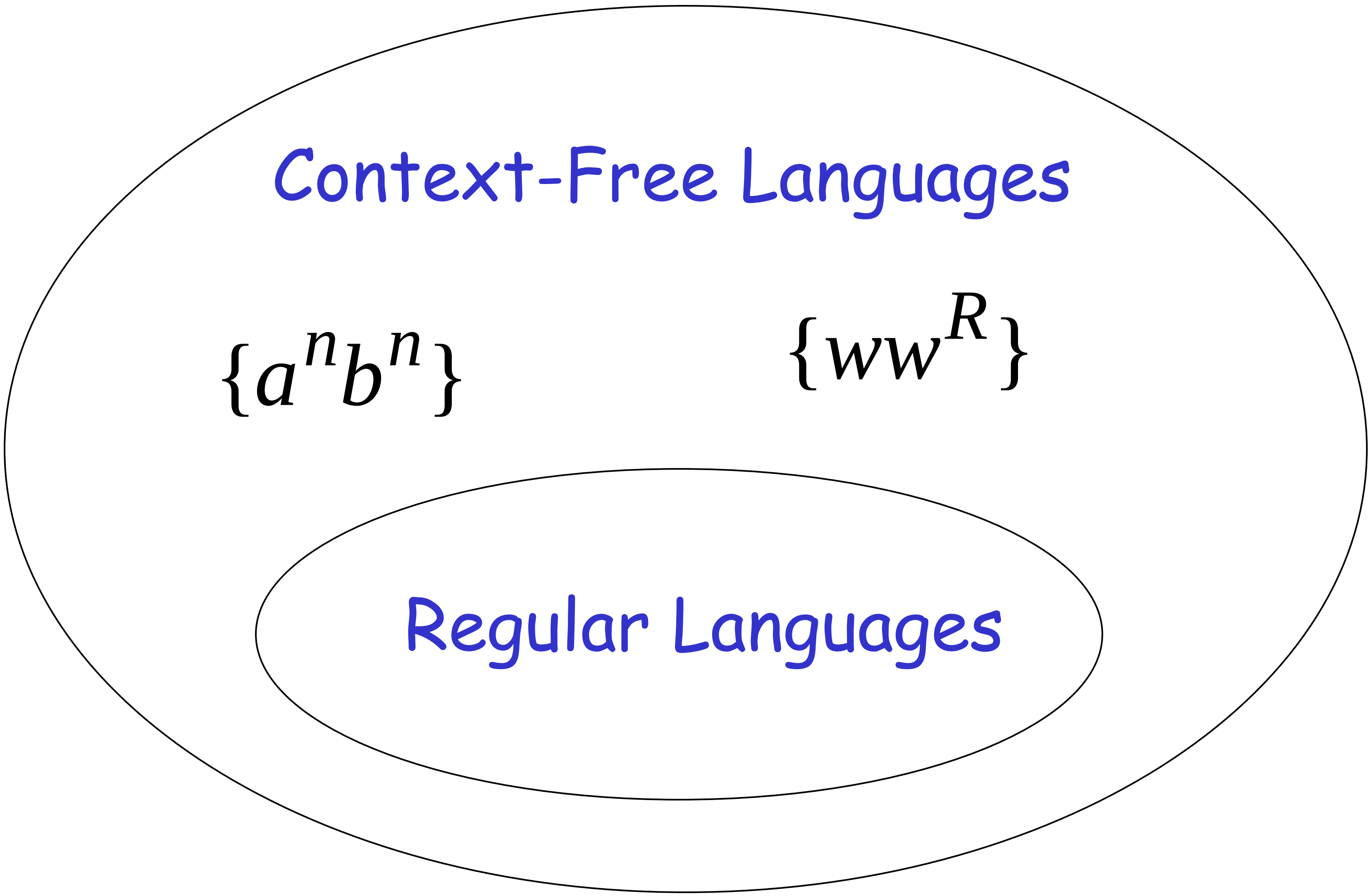
Context-Free Languages

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

Regular Languages

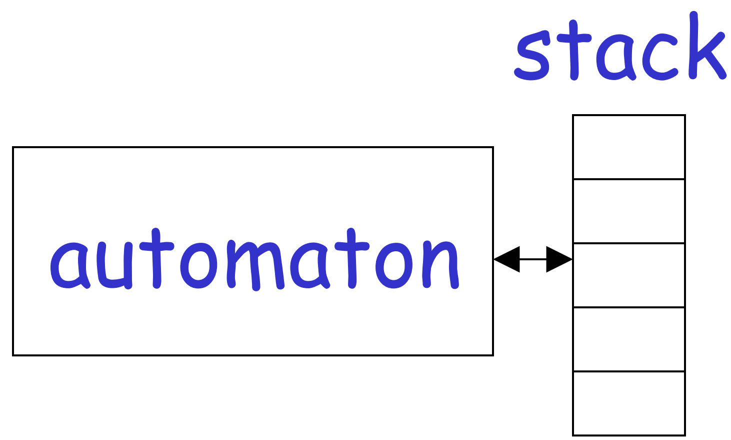
$$a^*b^* \qquad (a+b)^*$$


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

Context-Free Languages

Context-Free Grammars

Pushdown Automata



Context-Free Grammars

Grammars

Grammars express languages

Example: the English language

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

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

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

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

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

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

A derivation of "the dog walks":

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

A derivation of "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 walks",
"the cat runs",
"the cat walks",
"a dog runs",
"a dog walks",
"the dog runs",
"the dog walks" }

Notation

Production Rules

$$\underbrace{\qquad\qquad\qquad\qquad}$$

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

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

Variable

Terminal

Another Example

Grammar: $S \to aSb$

$S \to \lambda$

Derivation of sentence ab:

$$S \Rightarrow aSb \Rightarrow ab$$

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

Grammar: $S \to aSb$

$S \to \lambda$

Derivation of sentence $aabb$:

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

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

Other derivations:

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

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

Language of the grammar

$$S \to aSb$$

$$S \to \lambda$$

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

More Notation

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

$V:$ Set of variables

$T:$ Set of terminal symbols

$S:$ Start variable

$P:$ Set of Production rules

Example

Grammar G : $S \to aSb$

$S \to \lambda$

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

$$V = \{S\} \qquad T = \{a, b\}$$

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

More Notation

Sentential Form:

A sentence that contains variables and terminals

Example:

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

Sentential Forms   sentence

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

Instead of:

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

In general we write: $w_1 \stackrel{*}{\Rightarrow} w_n$

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

By default: $w \overset{*}{\Rightarrow} w$

Example

Grammar

$$S \rightarrow aSb$$

$$S \rightarrow \lambda$$

Derivations

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

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

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

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

Example

Grammar

$$S \to aSb$$

$$S \to \lambda$$

Derivations

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

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

Another Grammar Example

Grammar G :

$$S \rightarrow Ab$$

$$A \rightarrow aAb$$

$$A \rightarrow \lambda$$

Derivations:

$$S \Rightarrow Ab \Rightarrow b$$

$$S \Rightarrow Ab \Rightarrow aAbb \Rightarrow abb$$

$$S \Rightarrow Ab \Rightarrow aAbb \Rightarrow aaAbbb \Rightarrow aabbb$$

More Derivations

$$S \Rightarrow Ab \Rightarrow aAbb \Rightarrow aaAbbb \Rightarrow aaaAbbbb$$

$$\Rightarrow aaaaAbbbbb \Rightarrow aaaabbbbb$$

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

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

$$S \overset{*}{\Rightarrow} a^n b^n b$$

Language of a Grammar

For a grammar G
with start variable S :

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

String of terminals

Example

For grammar G :

$$S \to Ab$$

$$A \to aAb$$

$$A \to \lambda$$

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

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

A Convenient Notation

$$A \to aAb$$
$$A \to \lambda$$

$$\Longrightarrow \quad A \to aAb \mid \lambda$$

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

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

Example

A context-free grammar G:

$$S \rightarrow aSb$$

$$S \rightarrow \lambda$$

A derivation:

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

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

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:

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

$$S \to aSa$$

$$S \to bSb$$

$$S \to \lambda$$

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

Example

A context-free grammar G:

$$S \to aSb$$

$$S \to SS$$

$$S \to \lambda$$

A derivation:

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

A context-free grammar G:

$$S \to aSb$$

$$S \to SS$$

$$S \to \lambda$$

A derivation:

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

$$S \to aSb$$

$$S \to SS$$

$$S \to \lambda$$

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

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

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

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

Definition: Context-Free Grammars

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

Variables

Terminal symbols

Start variable

Productions of the form:

$$A \rightarrow x$$

Variable

String of variables and terminals

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

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

Definition: Context-Free Languages

A language L is context-free

if and only if

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

Derivation Order

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

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

Leftmost derivation:

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

Rightmost derivation:

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

$$S \to aAB$$

$$A \to bBb$$

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

Leftmost derivation:

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

$$\Rightarrow abbbbB \Rightarrow abbbb$$

Rightmost derivation:

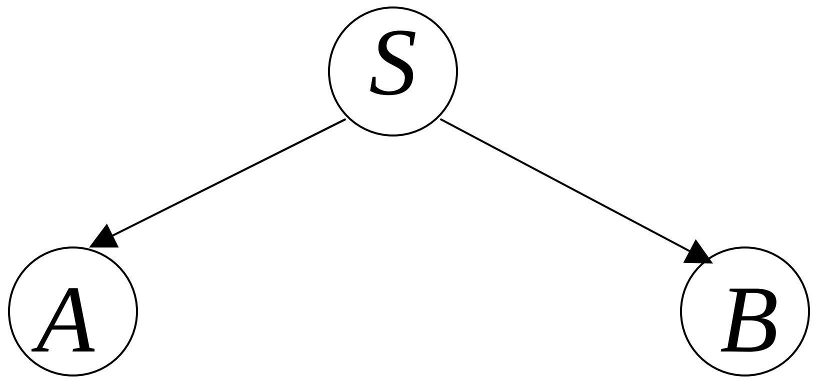
$$S \Rightarrow aAB \Rightarrow aA \Rightarrow abBb \Rightarrow abAb$$

$$\Rightarrow abbBbb \Rightarrow abbbb$$

Derivation Trees

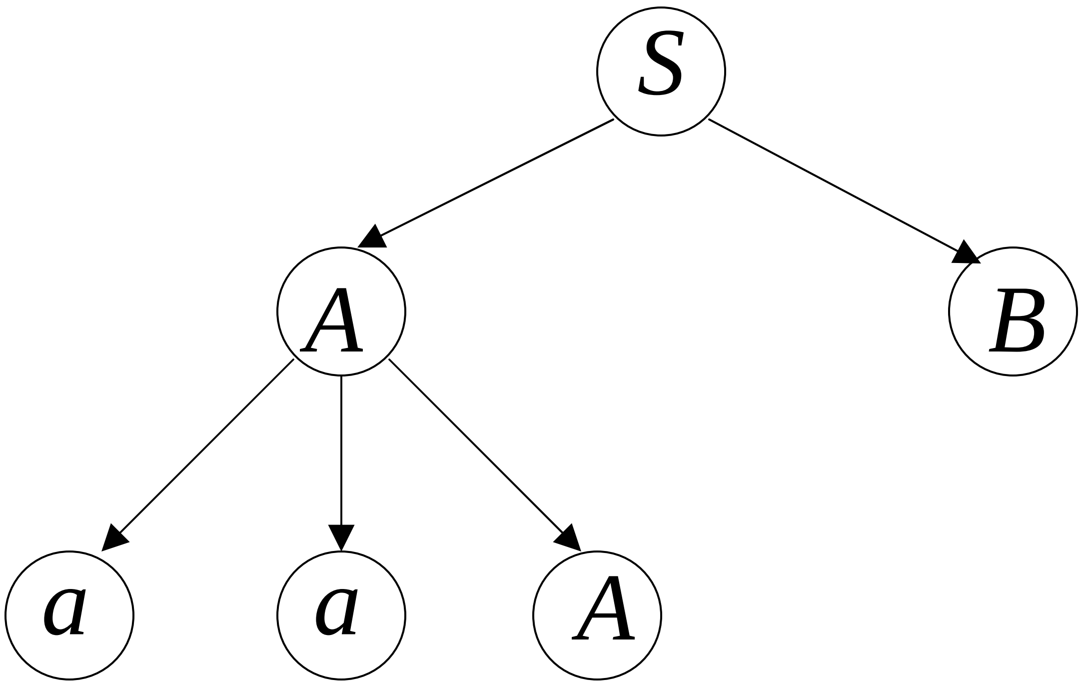
$S \to AB \qquad A \to aaA \mid \lambda \qquad B \to Bb \mid \lambda$

$$S \Rightarrow AB$$



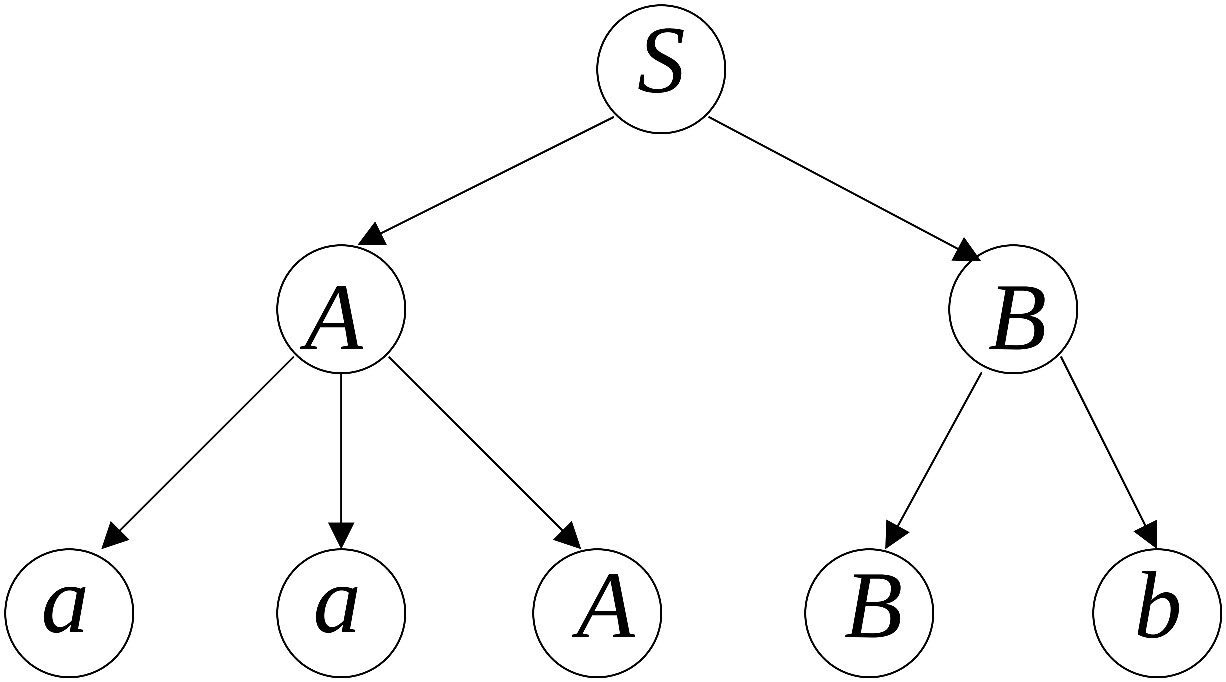
$$S \to AB \qquad A \to aaA \mid \lambda \qquad B \to Bb \mid \lambda$$

$$S \Rightarrow AB \Rightarrow aaAB$$



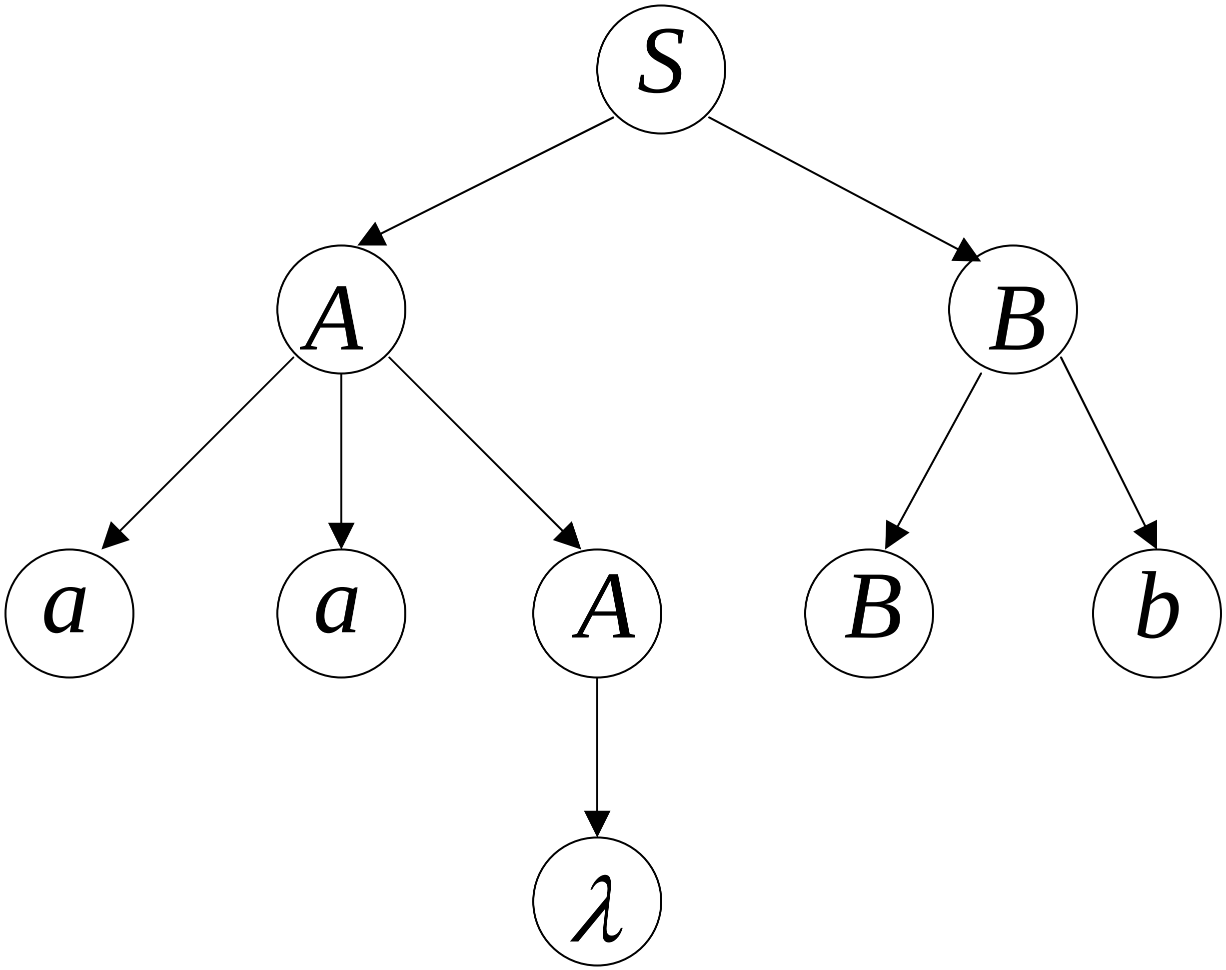
$$S \to AB \qquad A \to aaA \mid \lambda \qquad B \to Bb \mid \lambda$$

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



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

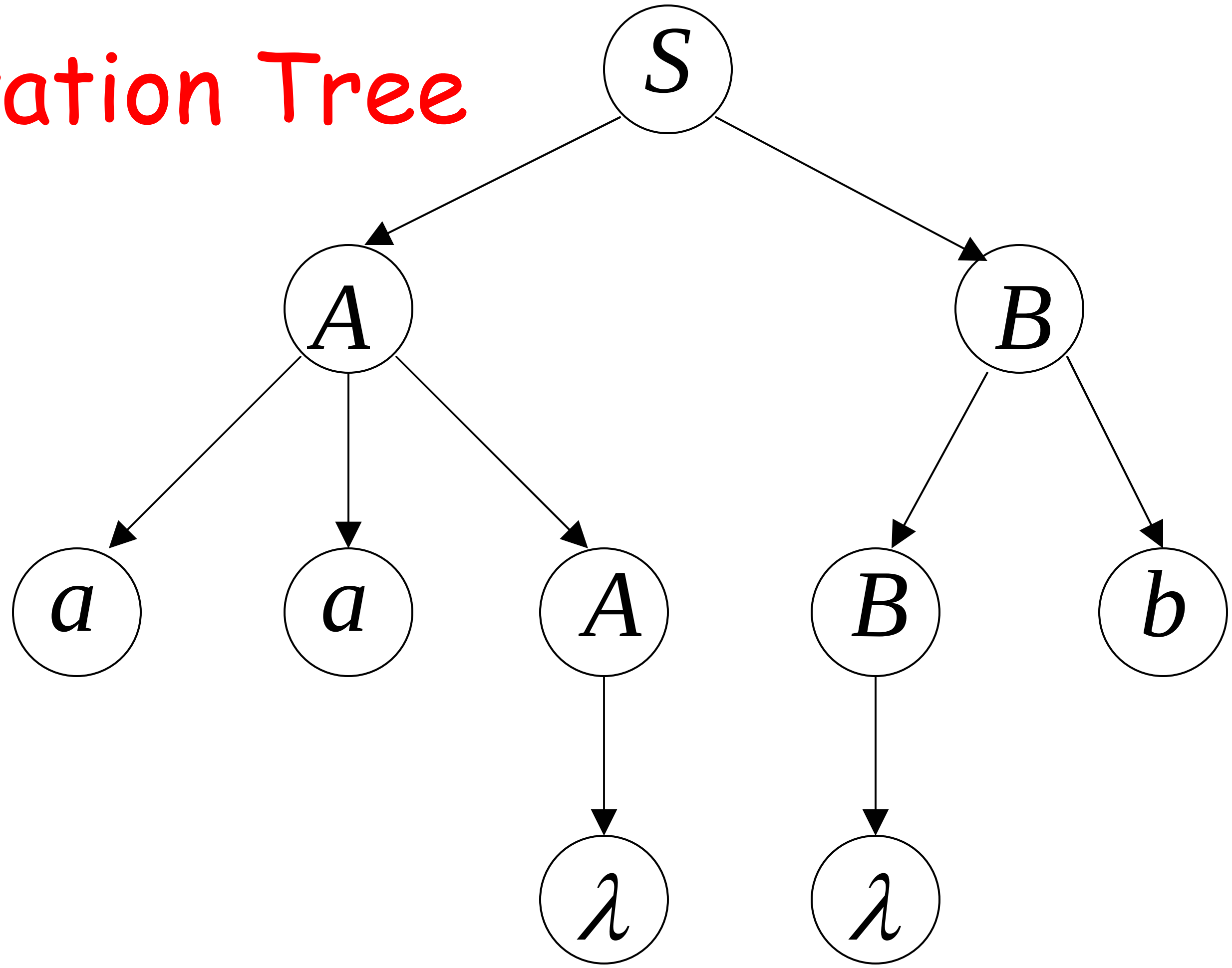
$$S \Rightarrow AB \Rightarrow aaAB \Rightarrow aaABb \Rightarrow aaBb$$



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

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

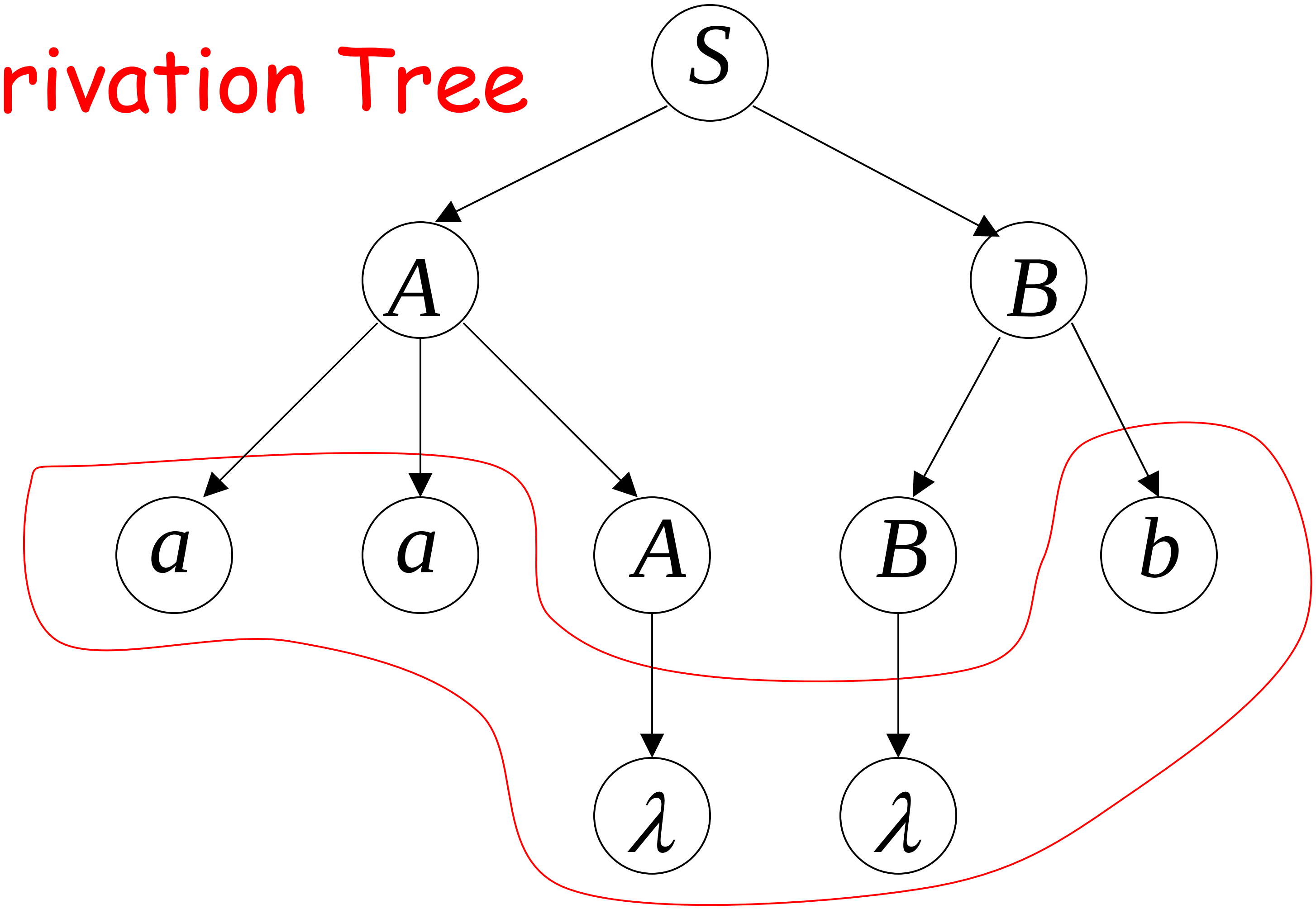
Derivation Tree



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

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

Derivation Tree



yield

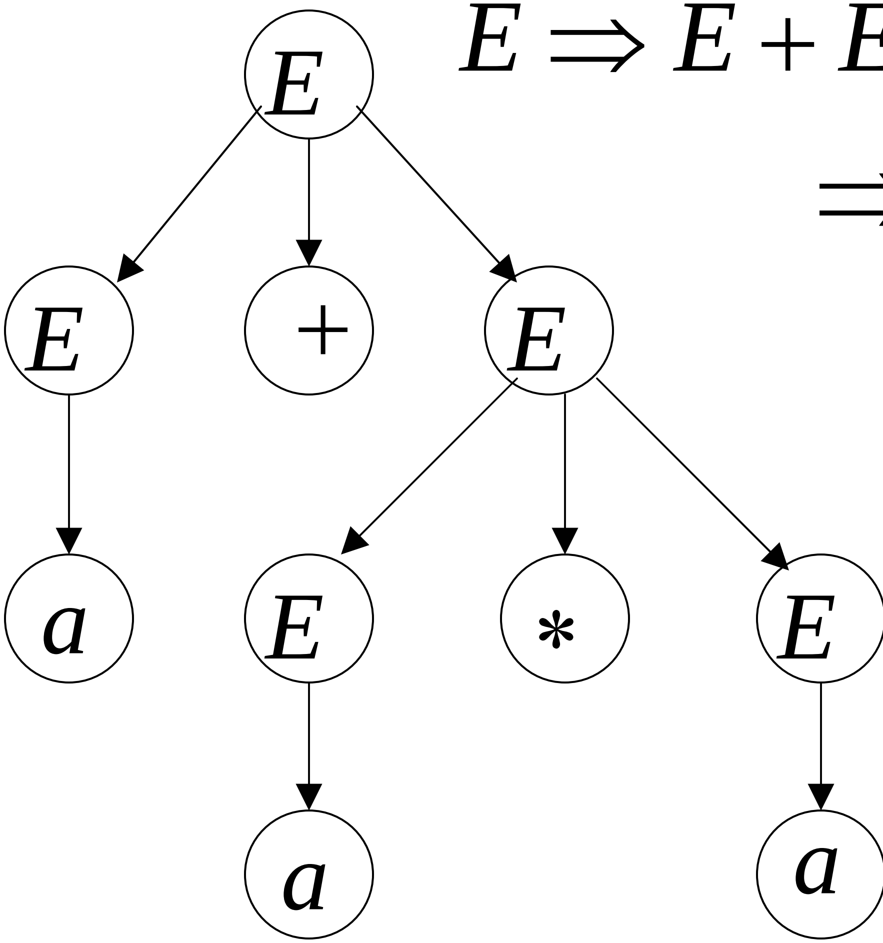
$aa\lambda\lambda b$

$= aab$

Ambiguity

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

$$a + a * a$$



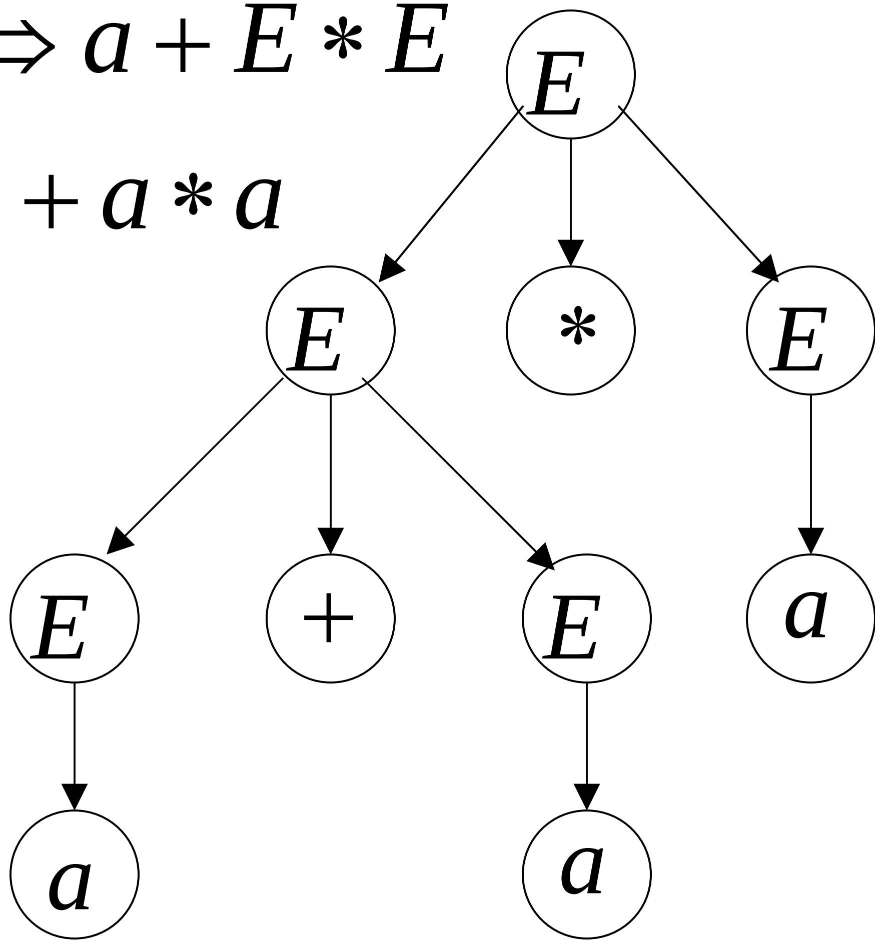
$$E \Rightarrow E + E \Rightarrow a + E \Rightarrow a + E * E$$
$$\Rightarrow a + a * E \Rightarrow a + a * a$$

leftmost derivation

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

$$a + a * a$$

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

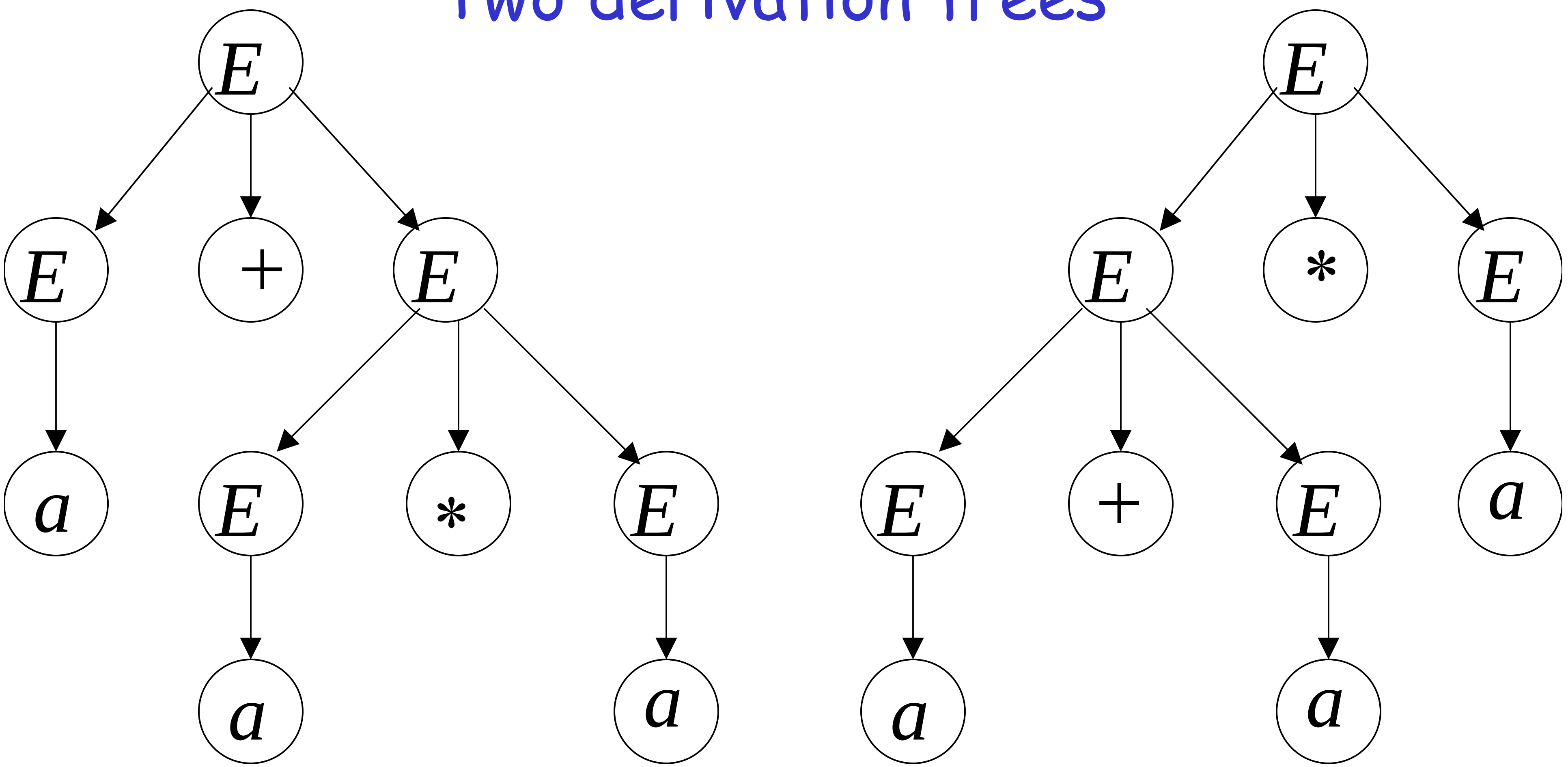


leftmost derivation

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

$$a + a * a$$

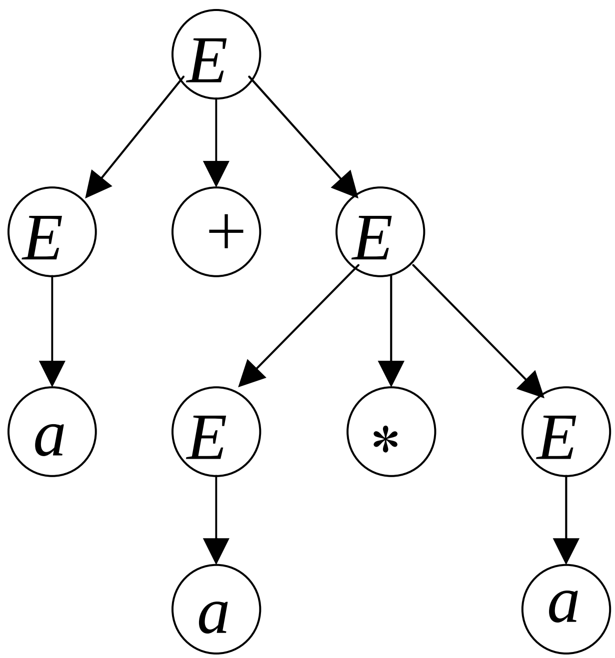
Two derivation trees

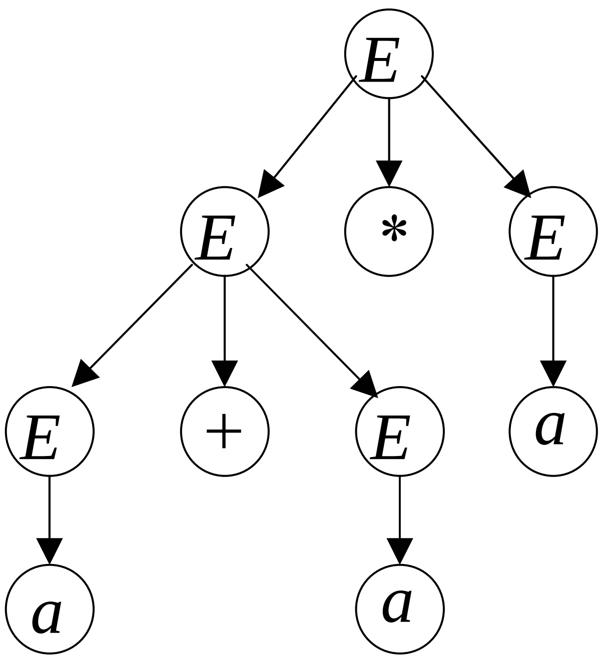


The grammar $E \to E + E \;\mid\; E * E \mid (E) \mid a$

is **ambiguous:**

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





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:

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:

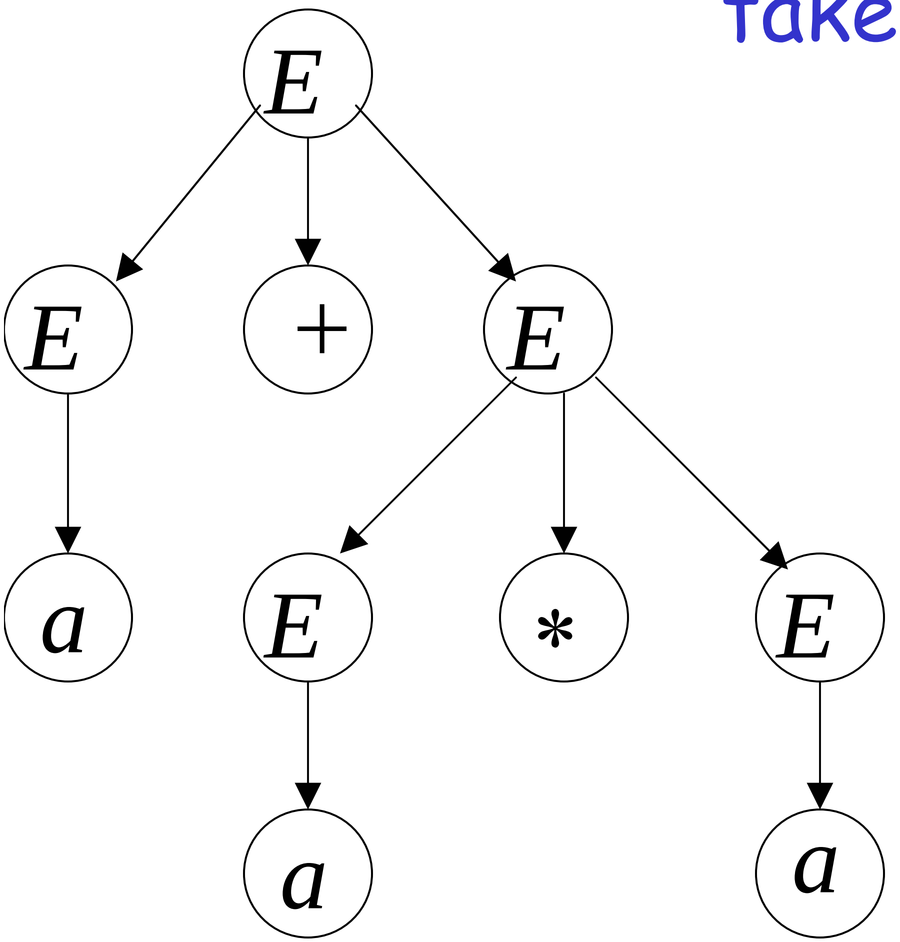
two or more leftmost derivations

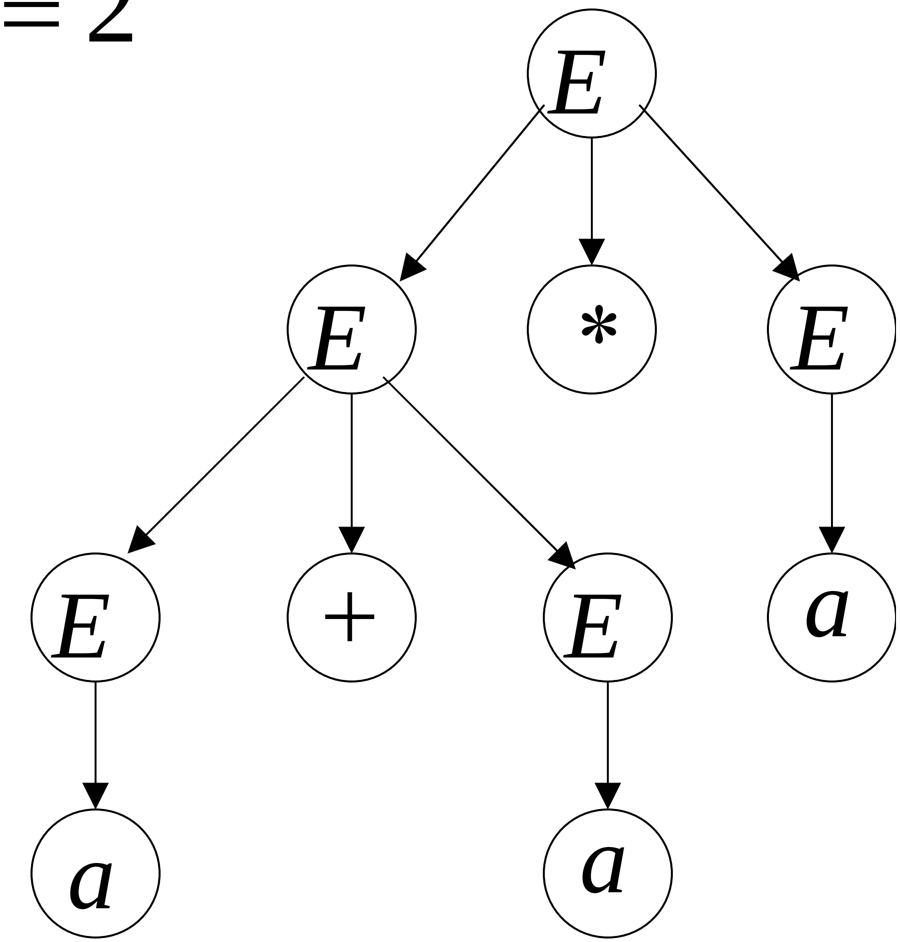
(or rightmost)

Why do we care about ambiguity?

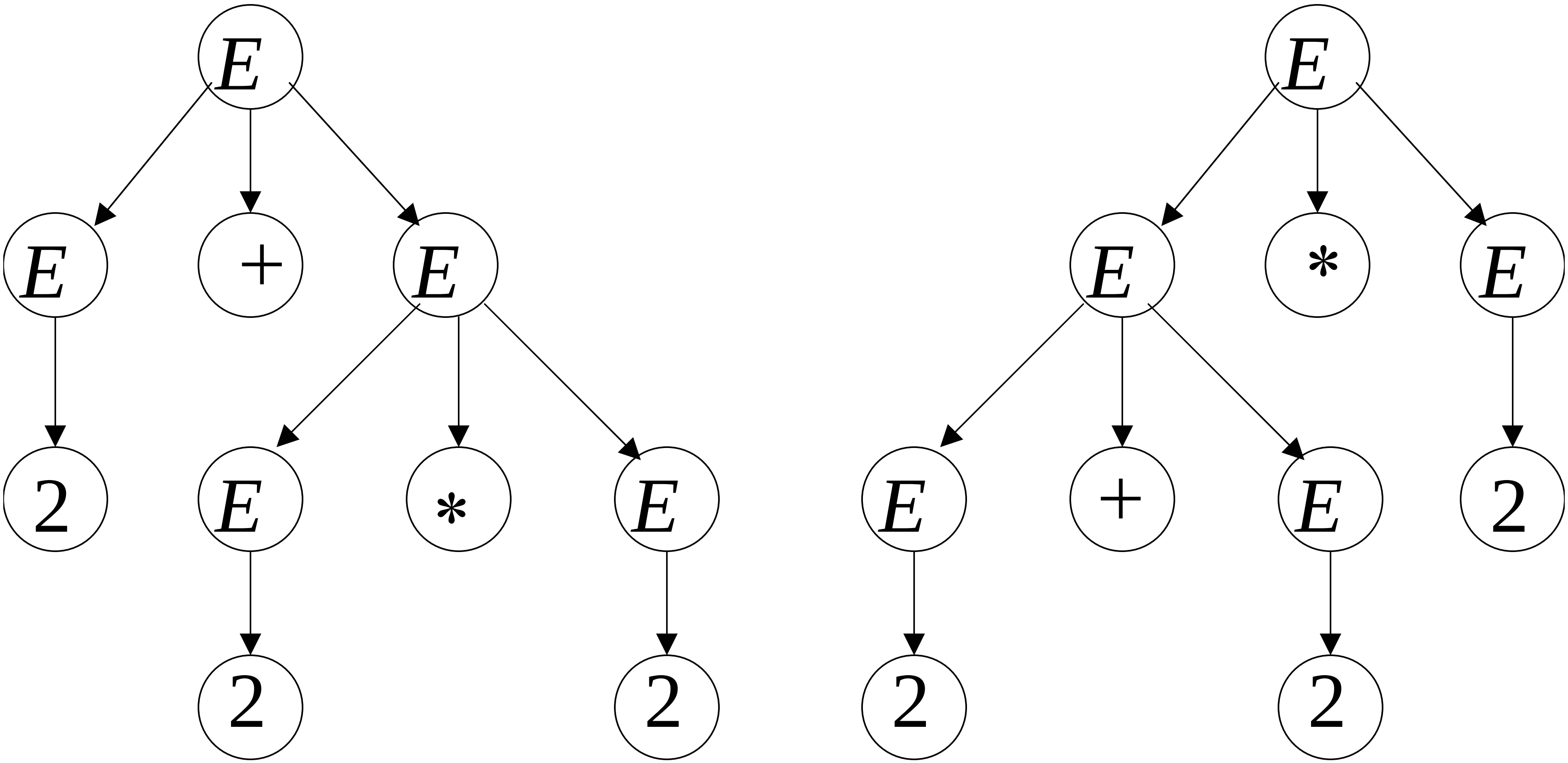
$$a + a * a$$

take $a = 2$

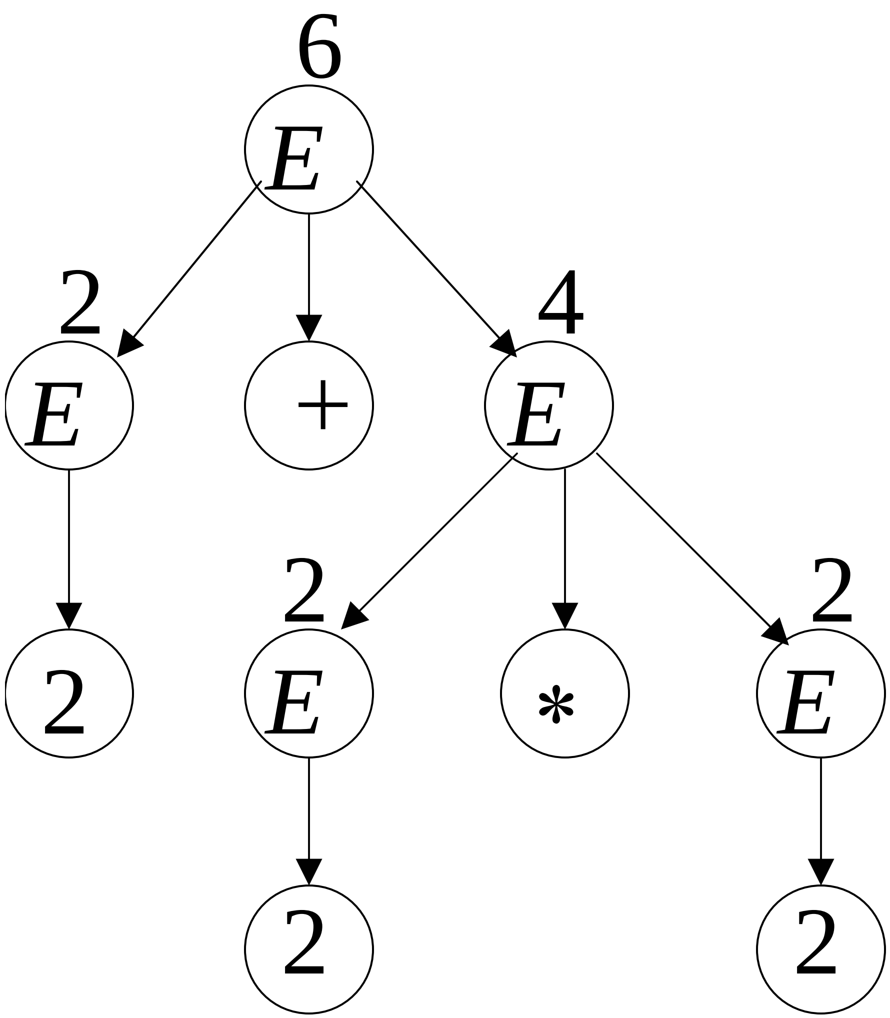




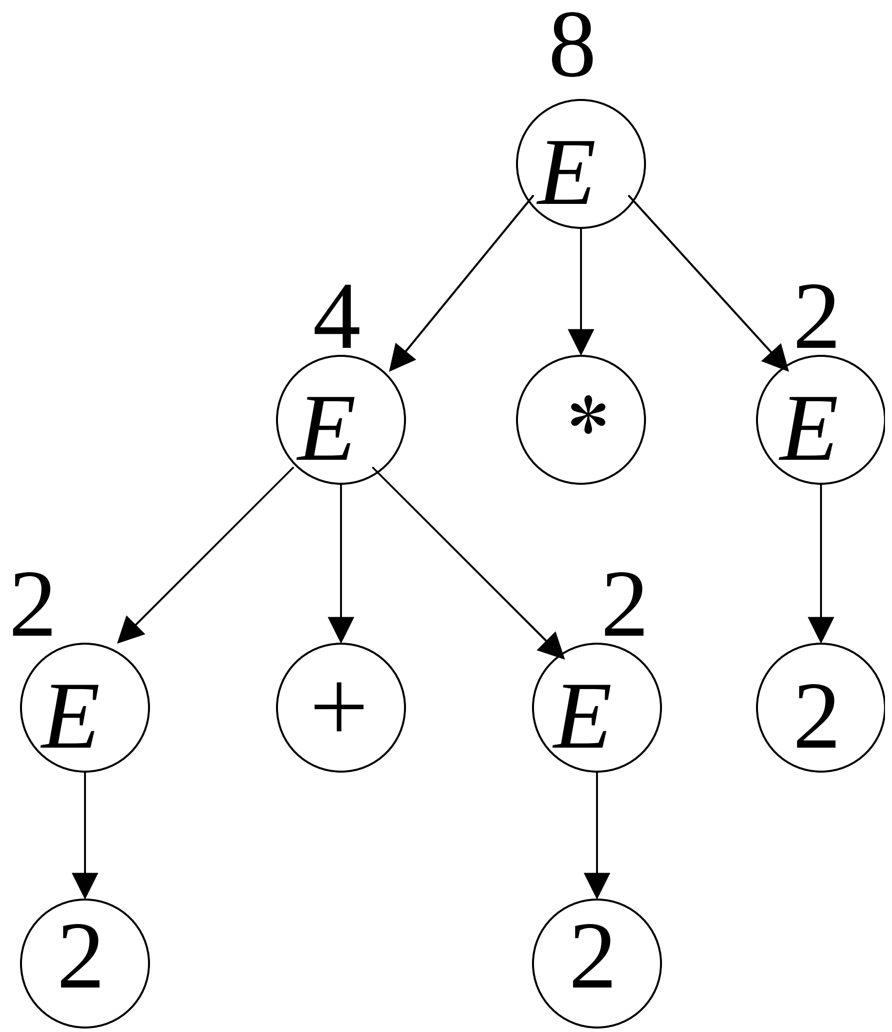
$$2 + 2 * 2$$



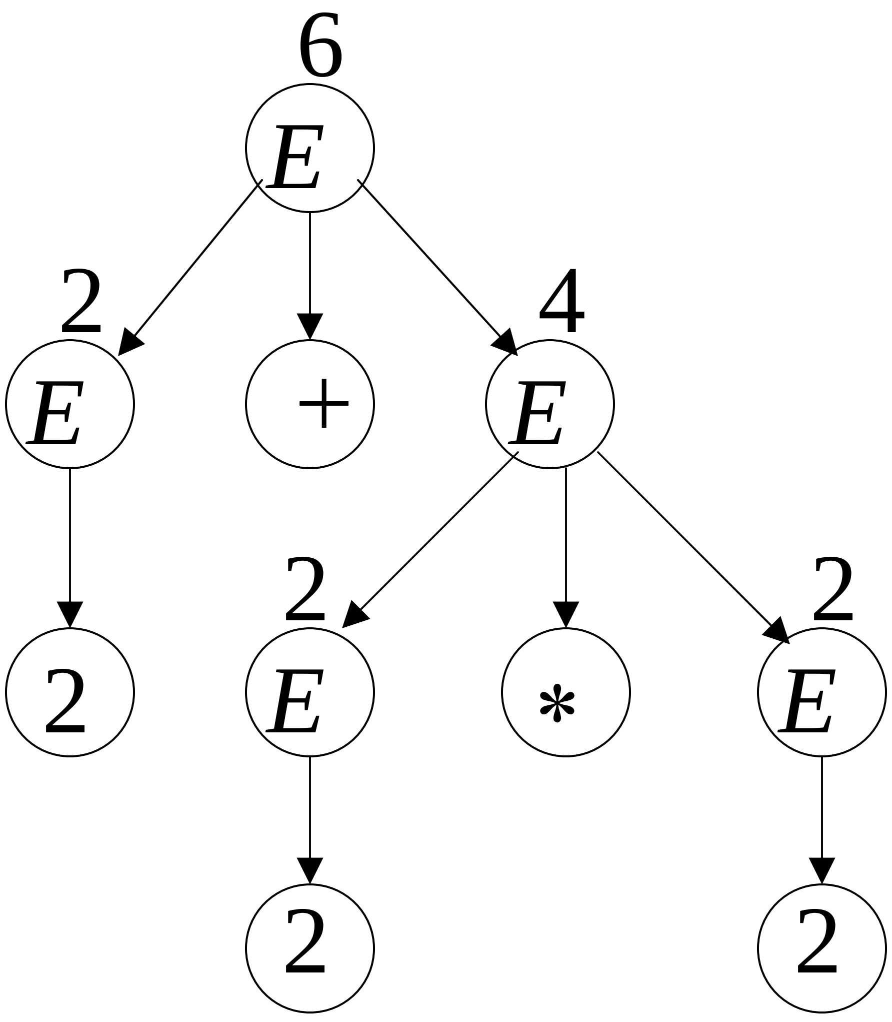
$$2 + 2 * 2 = 6$$



$$2 + 2 * 2 = 8$$



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



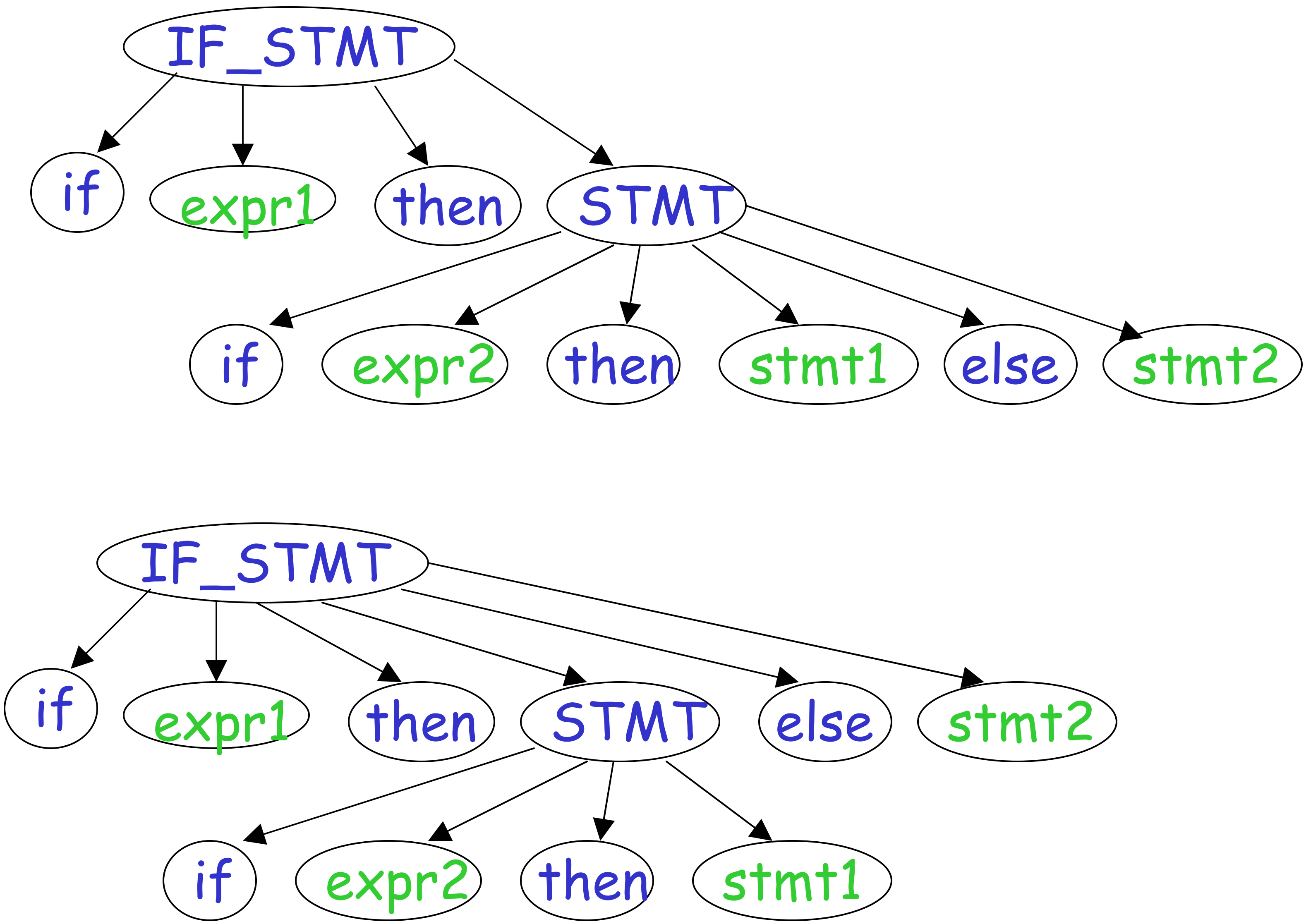
- Ambiguity is **bad** for programming languages

- We want to remove ambiguity

Another Ambiguous Grammar

$$\text{IF\_STMT} \rightarrow \text{if EXPR then STMT}$$
$$\quad | \ \text{if EXPR then STMT else STMT}$$

If expr1 then if expr2 then stmt1 else stmt2



Inherent Ambiguity

Some context free languages
have only ambiguous grammars

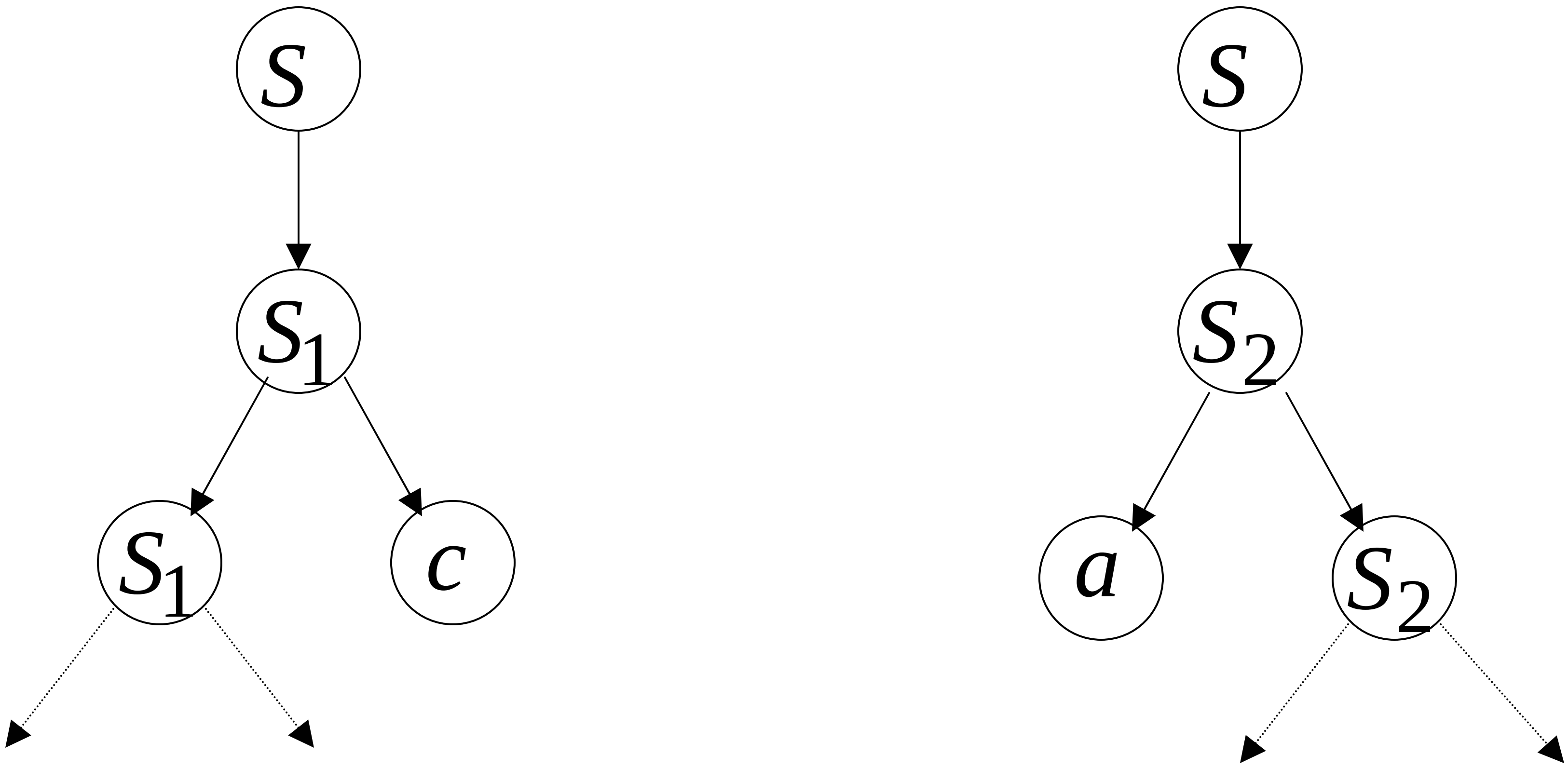
Example: $L = \{a^n b^n c^m\} \cup \{a^n b^m c^m\}$

$$S \rightarrow S_1 \mid S_2 \qquad S_1 \rightarrow S_1 c \mid A \qquad S_2 \rightarrow aS_2 \mid B$$

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

The string $a^n b^n c^n$

has two derivation trees



Simplifications of Context-Free Grammars

A Substitution Rule

$$S \to aB$$
$$A \to aaA$$
$$A \to abBc$$
$$B \to aA$$
$$B \to b$$

Substitute

$$B \to b$$

Equivalent grammar

$$S \to aB \mid ab$$
$$A \to aaA$$
$$A \to abBc \mid abbc$$
$$B \to aA$$

A Substitution Rule

$$S \rightarrow aB \mid ab$$
$$A \rightarrow aaA$$
$$A \rightarrow abBc \mid abbc$$
$$B \rightarrow aA$$

Substitute

$$B \rightarrow aA$$

$$S \rightarrow \cancel{aB} \mid ab \mid aaA$$
$$A \rightarrow aaA$$
$$A \rightarrow \cancel{abBc} \mid abbc \mid abaAc$$

Equivalent grammar

In general:

$$A \rightarrow xBz$$

$$B \rightarrow y_1$$

Substitute

$$B \rightarrow y_1$$

$$A \rightarrow xBz \mid xy_1z$$

equivalent grammar

Nullable Variables

$\lambda-\text{production}:$ $\qquad A \rightarrow \lambda$

Nullable Variable: $\qquad A \Rightarrow \ldots \Rightarrow \lambda$

Removing Nullable Variables

Example Grammar:

$$S \to aMb$$

$$M \to aMb$$

$$M \to \lambda$$

Nullable variable

$$S \to aMb$$

$$M \to aMb$$

$$\cancel{M \to \lambda}$$

Substitute

$$M \to \lambda$$

Final Grammar

$$S \to aMb$$

$$S \to ab$$

$$M \to aMb$$

$$M \to ab$$

Unit-Productions

Unit Production: $A \to B$

(a single variable in both sides)

Removing Unit Productions

Observation:

$$A \to A$$

Is removed immediately

Example Grammar:

$$S \to aA$$
$$A \to a$$
$$A \to B$$
$$B \to A$$
$$B \to bb$$

$$S \to aA$$

$$A \to a$$

$$\cancel{A \to B}$$

$$B \to A$$

$$B \to bb$$

Substitute

$$A \to B$$

$$S \to aA \mid aB$$

$$A \to a$$

$$B \to A \mid B$$

$$B \to bb$$

$$S \to aA \mid aB$$

$$A \to a$$

$$B \to A \mid \cancel{B}$$

$$B \to bb$$

Remove

$$B \to B$$

$$S \to aA \mid aB$$

$$A \to a$$

$$B \to A$$

$$B \to bb$$

$$S \to aA \mid aB$$

$$A \to a$$

$$\cancel{B \to A}$$

$$B \to bb$$

Substitute

$$B \to A$$

$$S \to aA \mid aB \mid aA$$

$$A \to a$$

$$B \to bb$$

Remove repeated productions

Final grammar

$$S \to aA \mid aB \mid \cancel{aA}$$

$$A \to a$$

$$B \to bb$$

$$S \to aA \mid aB$$

$$A \to a$$

$$B \to bb$$

Useless Productions

$$S \to aSb$$

$$S \to \lambda$$

$$S \to A$$

$$A \to aA$$ Useless Production

Some derivations never terminate...

$$S \Rightarrow A \Rightarrow aA \Rightarrow aaA \Rightarrow \ldots \Rightarrow aa\ldots aA \Rightarrow \ldots$$

Another grammar:

$$S \to A$$

$$A \to aA$$

$$A \to \lambda$$

$$B \to bA$$ Useless Production

Not reachable from S

In general:

contains only terminals

if $S \Rightarrow \ldots \Rightarrow xAy \Rightarrow \ldots \Rightarrow w$

$w \in L(G)$

then variable A is useful

otherwise, variable A is useless

A production $A \to x$ is useless
if any of its variables is useless

$$S \to aSb$$

$$S \to \lambda$$

Productions

Variables

$$S \to A$$ useless

useless $$A \to aA$$ useless

useless $$B \to C$$ useless

useless $$C \to D$$ useless

Removing Useless Productions

Example Grammar:

$$S \to aS \mid A \mid C$$

$$A \to a$$

$$B \to aa$$

$$C \to aCb$$

First: find all variables that can produce strings with only terminals

$$S \to aS \mid A \mid C$$
$$A \to a$$
$$B \to aa$$
$$C \to aCb$$

Round 1: $\{A, B\}$

$$S \to A$$

Round 2: $\{A, B, S\}$

Keep only the variables
that produce terminal symbols: $\{A, B, S\}$

(the rest variables are useless)

$$S \to aS \mid A \mid \cancel{C}$$

$$A \to a$$

$$B \to aa$$

$$\cancel{C \to aCb}$$

$$S \to aS \mid A$$

$$A \to a$$

$$B \to aa$$

Remove useless productions

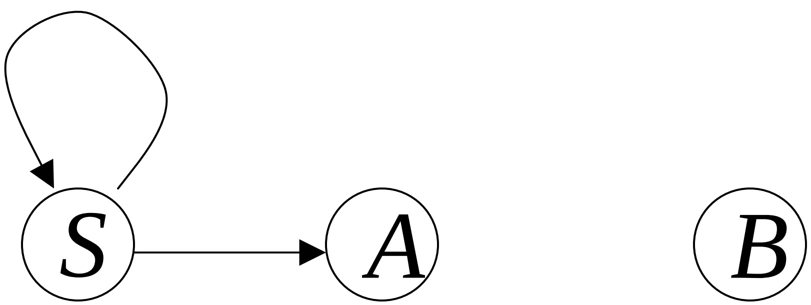
Second: Find all variables reachable from S

Use a Dependency Graph

$$S \rightarrow aS \mid A$$

$$A \rightarrow a$$

$$B \rightarrow aa$$



not reachable

Keep only the variables reachable from S

(the rest variables are useless)

Final Grammar

$$S \to aS \mid A$$
$$A \to a$$
$$\cancel{B \to aa}$$

$$S \to aS \mid A$$
$$A \to a$$

Remove useless productions

Removing All

Step 1: Remove Nullable Variables

Step 2: Remove Unit-Productions

Step 3: Remove Useless Variables

Normal Forms
for
Context-free Grammars

Chomsky Normal Form

Each productions has form:

$$A \to BC \qquad \text{or} \qquad A \to a$$

variable variable terminal

Examples:

$$S \to AS$$
$$S \to a$$
$$A \to SA$$
$$A \to b$$

Chomsky
Normal Form

$$S \to AS$$
$$S \to AAS$$
$$A \to SA$$
$$A \to aa$$

Not Chomsky
Normal Form

Convertion to Chomsky Normal Form

Example:

$$S \to ABa$$

$$A \to aab$$

$$B \to Ac$$

Not Chomsky
Normal Form

Introduce variables for terminals: T_a, T_b, T_c

$$S \to ABa$$
$$A \to aab$$
$$B \to Ac$$

$$S \to ABT_a$$
$$A \to T_a T_a T_b$$
$$B \to AT_c$$
$$T_a \to a$$
$$T_b \to b$$
$$T_c \to c$$

Introduce intermediate variable: V_1

$$S \to ABT_a$$
$$A \to T_a T_a T_b$$
$$B \to AT_c$$
$$T_a \to a$$
$$T_b \to b$$
$$T_c \to c$$

$$\Longrightarrow$$

$$S \to AV_1$$
$$V_1 \to BT_a$$
$$A \to T_a T_a T_b$$
$$B \to AT_c$$
$$T_a \to a$$
$$T_b \to b$$
$$T_c \to c$$

Introduce intermediate variable: V_2

$$S \to AV_1$$
$$V_1 \to BT_a$$
$$A \to T_aT_aT_b$$
$$B \to AT_c$$
$$T_a \to a$$
$$T_b \to b$$
$$T_c \to c$$

$\Longrightarrow$

$$S \to AV_1$$
$$V_1 \to BT_a$$
$$A \to T_aV_2$$
$$V_2 \to T_aT_b$$
$$B \to AT_c$$
$$T_a \to a$$
$$T_b \to b$$
$$T_c \to c$$

Final grammar in Chomsky Normal Form:

Initial grammar

$$S \to ABa$$

$$A \to aab$$

$$B \to Ac$$

$$S \to AV_1$$

$$V_1 \to BT_a$$

$$A \to T_a V_2$$

$$V_2 \to T_a T_b$$

$$B \to AT_c$$

$$T_a \to a$$

$$T_b \to b$$

$$T_c \to c$$

In general:

From any context-free grammar
(which doesn't produce λ)
not in Chomsky Normal Form

we can obtain:

An equivalent grammar
in Chomsky Normal Form

The Procedure

First remove:

Nullable variables

Unit productions

Then, for every symbol a:

Add production $T_a \to a$

In productions: replace a with T_a

New variable: T_a

Replace any production $A \to C_1 C_2 \cdots C_n$

with
$$A \to C_1 V_1$$
$$V_1 \to C_2 V_2$$
$$\cdots$$
$$V_{n-2} \to C_{n-1} C_n$$

New intermediate variables: $V_1, V_2, \ldots, V_{n-2}$

Theorem: For any context-free grammar (which doesn't produce λ) there is an equivalent grammar in Chomsky Normal Form

Observations

- Chomsky normal forms are good for parsing and proving theorems

- It is very easy to find the Chomsky normal form for any context-free grammar

Greinbach Normal Form

All productions have form:

$$A \to a\, V_1 V_2 \cdots V_k \qquad\qquad k \geq 0$$

symbol

variables

Observations

- Greinbach normal forms are very good for parsing

- It is hard to find the Greinbach normal form of any context-free grammar

Compilers

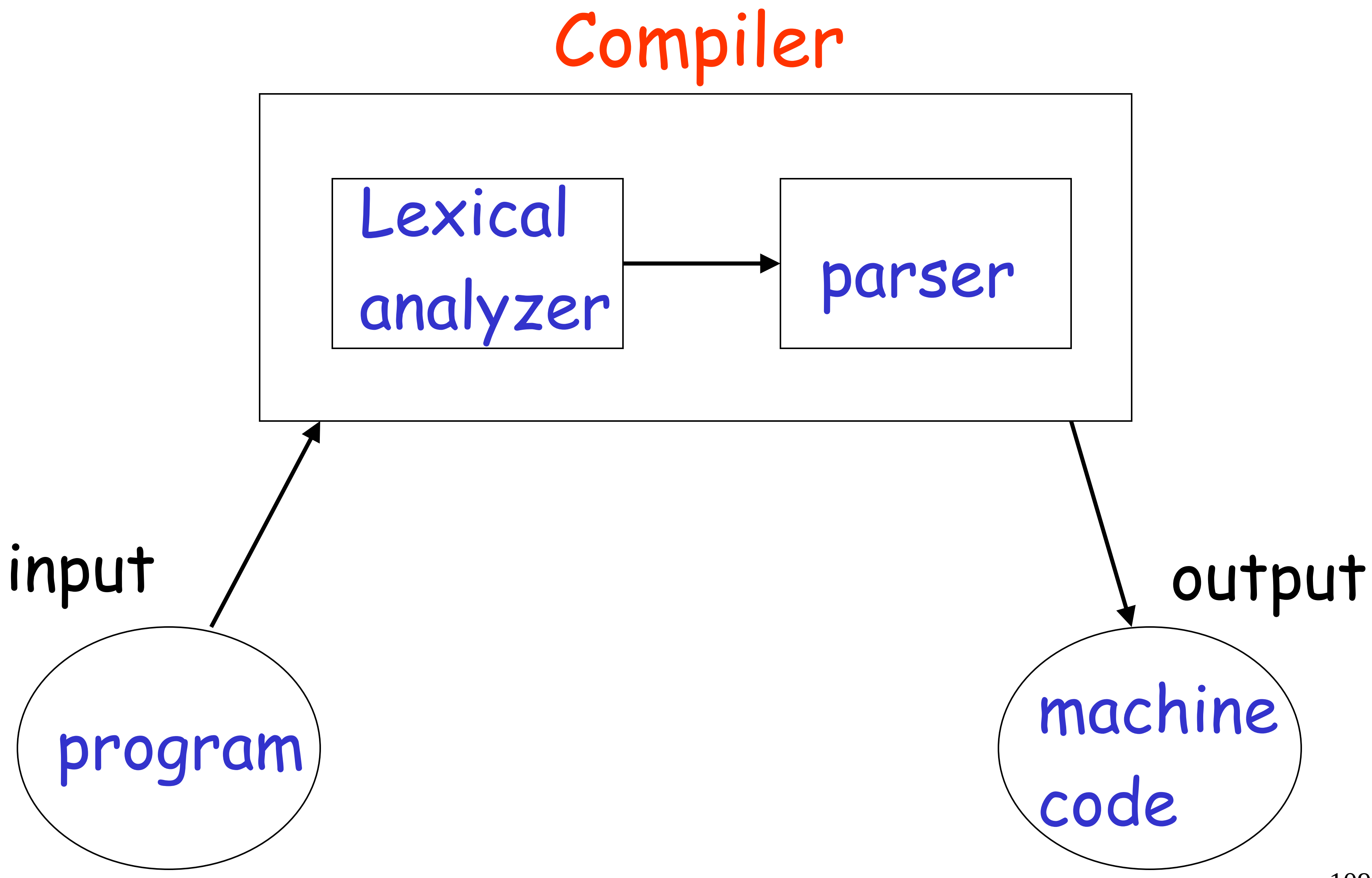
Program

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

Compiler

Machine Code

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


Compiler
Lexical analyzer
parser
input
program
output
machine code

A parser knows the grammar of the programming language

Parser

PROGRAM → STMT_LIST

STMT_LIST → STMT; STMT_LIST | STMT;

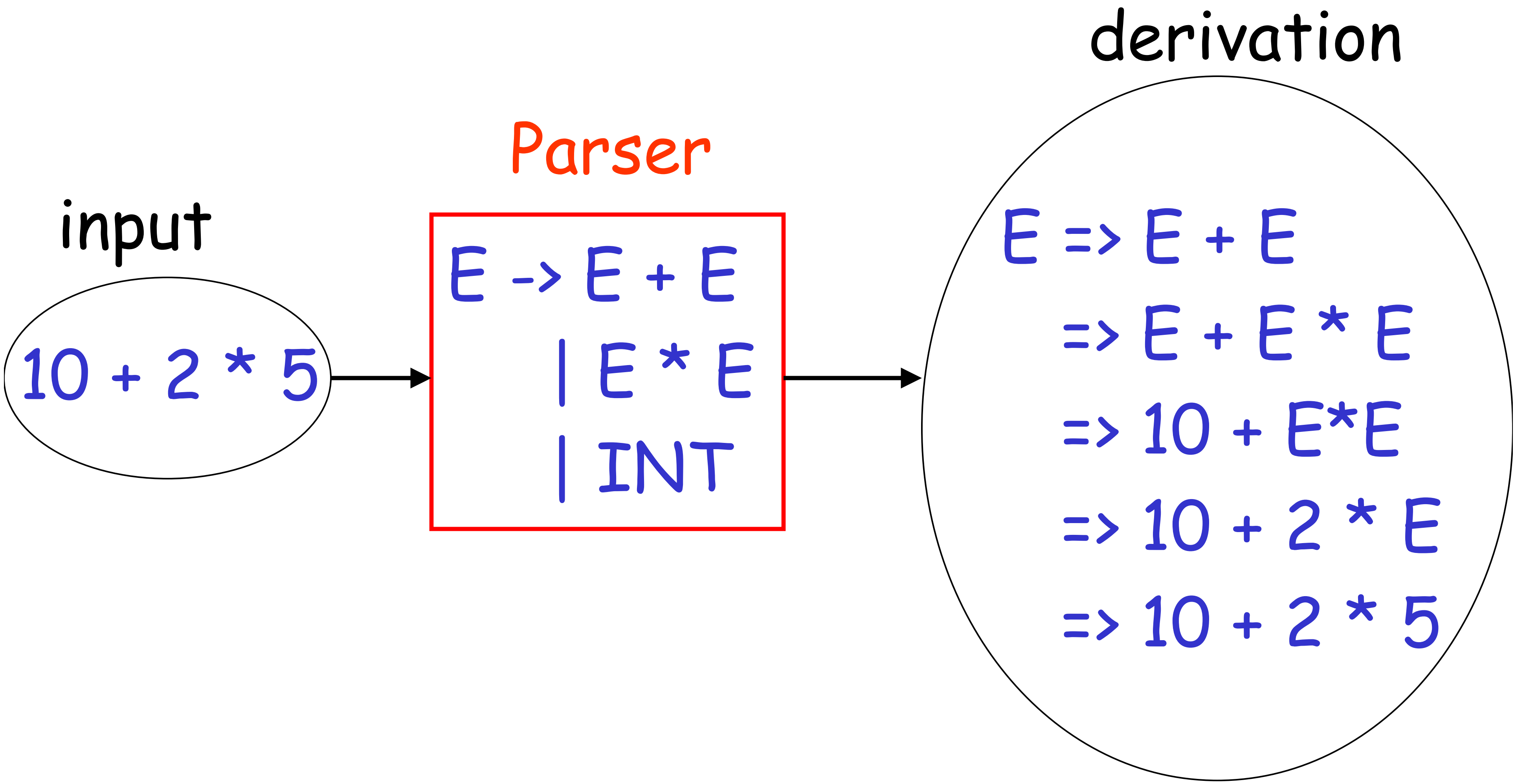
STMT → EXPR | IF_STMT | WHILE_STMT
| { STMT_LIST }

EXPR → EXPR + EXPR | EXPR - EXPR | ID

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

WHILE_STMT → while (EXPR) do STMT

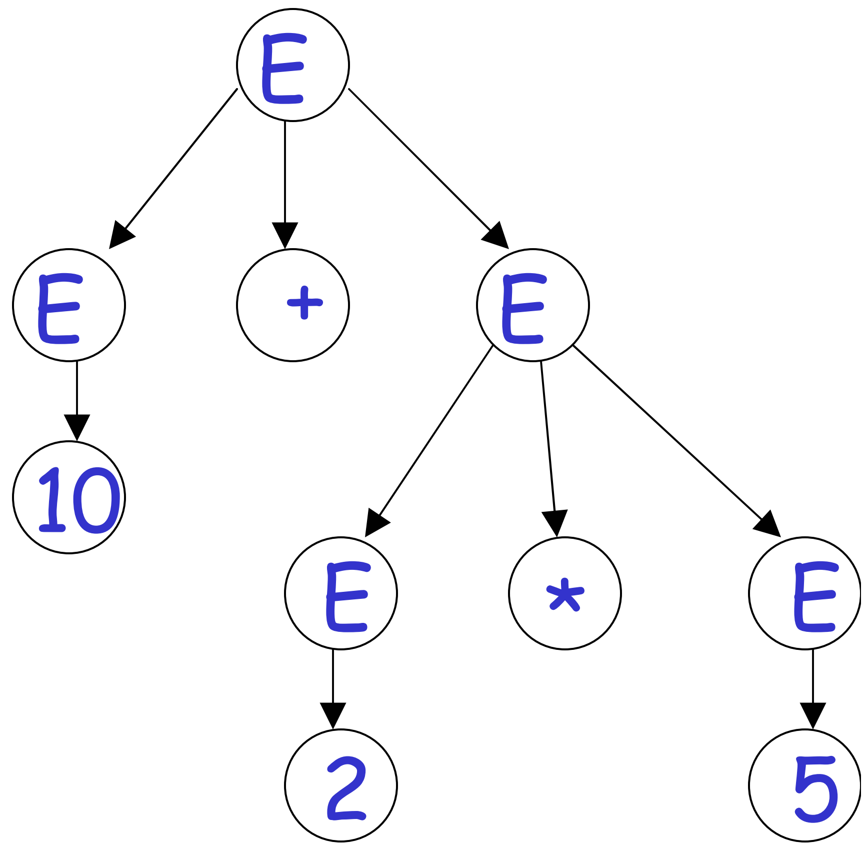
The parser finds the derivation of a particular input



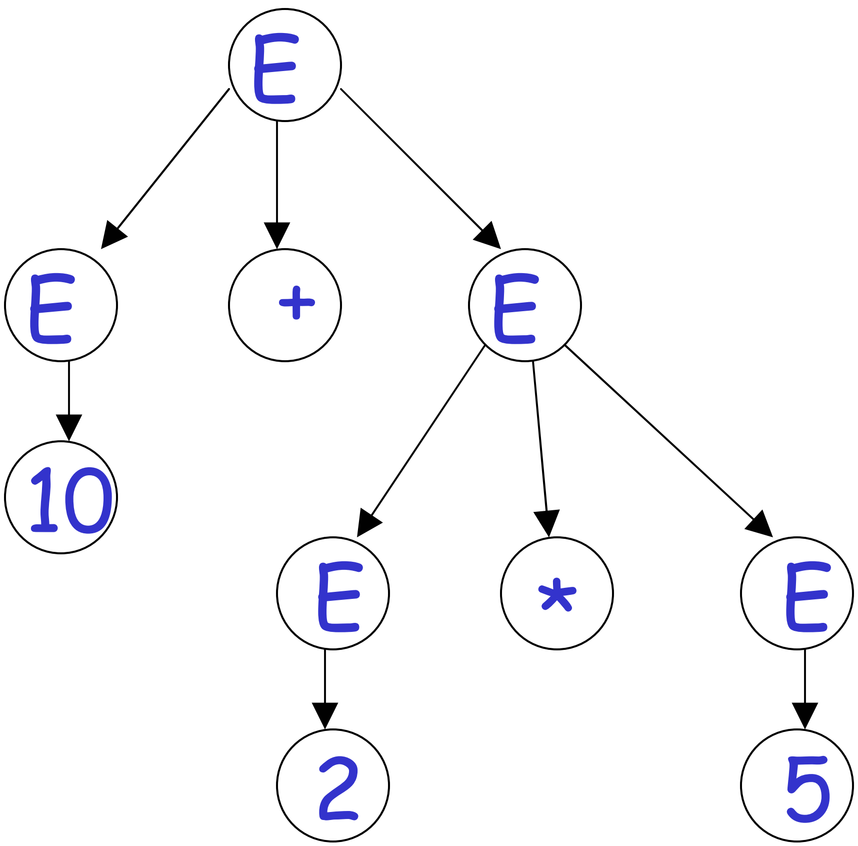
derivation

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

derivation tree



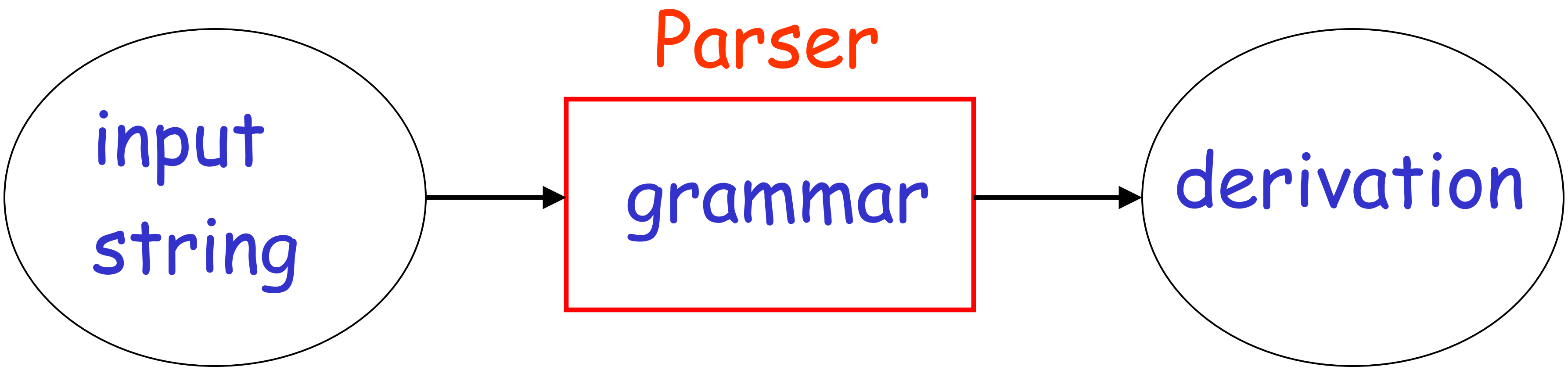
derivation tree



machine code

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

Parsing


Parser
input
string
grammar
derivation

Example:

input

$aabb$

Parser

$$S \rightarrow SS$$

$$S \rightarrow aSb$$

$$S \rightarrow bSa$$

$$S \rightarrow \lambda$$

derivation

?

Exhaustive Search

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

Phase 1:

$$S \Rightarrow SS$$

$$S \Rightarrow aSb$$

$$S \Rightarrow bSa$$

$$S \Rightarrow \lambda$$

Find derivation of

$$aabb$$

All possible derivations of length 1

$S \Rightarrow SS$

$S \Rightarrow aSb$

$\cancel{S \Rightarrow bSa}$

$\cancel{S \Rightarrow \lambda}$

$aabb$

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

$$S \Rightarrow SS \Rightarrow SSS$$

$$S \Rightarrow SS \Rightarrow aSbS$$

$aabb$

Phase 1

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

$$S \Rightarrow SS$$

$$S \Rightarrow SS \Rightarrow S$$

$$S \Rightarrow aSb$$

$$S \Rightarrow aSb \Rightarrow aSSb$$

$$S \Rightarrow aSb \Rightarrow aaSbb$$

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

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

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

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

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

Time complexity of exhaustive search

Suppose there are no productions of the form

$$A \to \lambda$$

$$A \to B$$

Number of phases for string w : approx. |w|

For grammar with k rules

Time for phase 1: k

k possible derivations

Time for phase 2: k^2

k^2 possible derivations

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

A total of $2^{|w|}$ possible derivations

Total time needed for string w:

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

phase 1

phase 2

phase |w|

Extremely bad!!!

For general context-free grammars:

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

The CYK parser