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

$$S \to aB \mid ab$$

$$A \to aaA$$

$$A \to abBc \mid abbc$$

$$B \to aA$$

Substitute

$$B \to aA$$

$$S \to \text{~~}aB\text{~~} \mid ab \mid aaA$$

$$A \to aaA$$

$$A \to \text{~~}abBc\text{~~} \mid abbc \mid abaAc$$

Equivalent grammar

In general:

$$A \to xBz$$

$$B \to y_1$$

Substitute $B \to y_1$

$$A \to xBz \mid xy_1z$$

equivalent grammar

Nullable Variables

$\lambda - \text{production}:$ $A \to \lambda$

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

Removing Nullable Variables

Example Grammar:

$$S \to aMb$$

$$M \to aMb$$

$$M \to \lambda$$

Nullable variable

$$S \to aMb$$
$$M \to aMb$$
~~$M \to \lambda$~~

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:

$$A \rightarrow A$$

Is removed immediately

Example Grammar:

$$S \to aA$$

$$A \to a$$

$$A \to B$$

$$B \to A$$

$$B \to bb$$

$$S \to aA$$

$$A \to a$$

~~$A \to B$~~

$$B \to A$$

$$B \to bb$$

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

$$B \to B$$

$$S \to aA \mid aB$$

$$A \to a$$

$$B \to A$$

$$B \to bb$$

$S \to aA \mid aB$

$A \to a$

~~$B \to A$~~

$B \to bb$

Substitute

$B \to A$

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

$A \to a$

$B \to bb$

Remove repeated productions

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

$$A \rightarrow a$$

$$B \rightarrow bb$$

Final grammar

$$S \rightarrow aA \mid aB$$

$$A \rightarrow a$$

$$B \rightarrow bb$$

Useless Productions

$$S \rightarrow aSb$$

$$S \rightarrow \lambda$$

$$S \rightarrow A$$

$$A \rightarrow aA$$ 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

$$S \rightarrow aS \mid A \mid C$$

$$A \rightarrow a$$

$$B \rightarrow aa$$

$$C \rightarrow aCb$$

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

$$S \rightarrow A$$

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

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

(the rest variables are useless)

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

$$A \to a$$

$$B \to aa$$

$$C \to aCb$$

$$S \to aS \mid A$$

$$A \to a$$

$$B \to aa$$

Remove useless productions

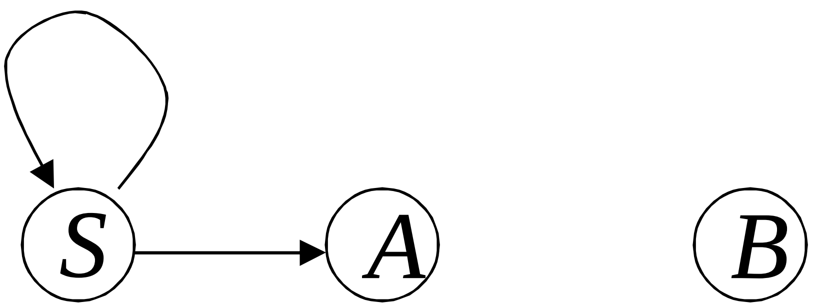
Second: Find all variables reachable from S

$$S \to aS \mid A$$

$$A \to a$$

$$B \to aa$$

Use a Dependency Graph



not reachable

Keep only the variables reachable from S

(the rest variables are useless)

$$S \to aS \mid A$$

$$A \to a$$

~~$$B \to aa$$~~

Final Grammar

$$S \to aS \mid A$$

$$A \to a$$

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:

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

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

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

$\Rightarrow$

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

Introduce intermediate variable: V_2

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

$\Rightarrow$

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

Final grammar in Chomsky Normal Form:

$$S \to AV_1$$

$$V_1 \to BT_a$$

$$A \to T_a V_2$$

$$V_2 \to T_a T_b$$

$$B \to AT_c$$

$$T_a \to a$$

$$T_b \to b$$

$$T_c \to c$$

Initial grammar

$$S \to ABa$$

$$A \to aab$$

$$B \to Ac$$

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:

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

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

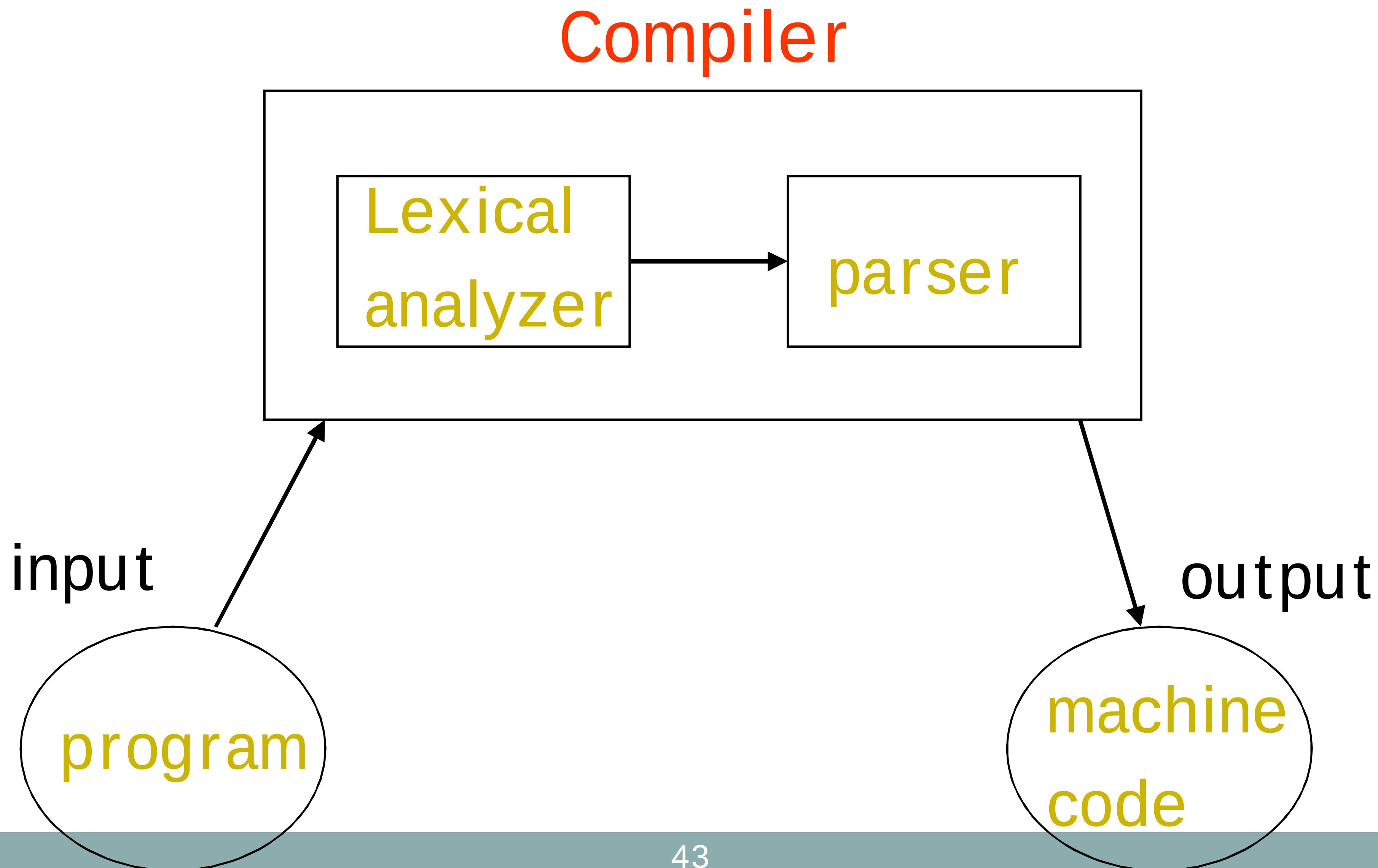
Program

```
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
output
program
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 }

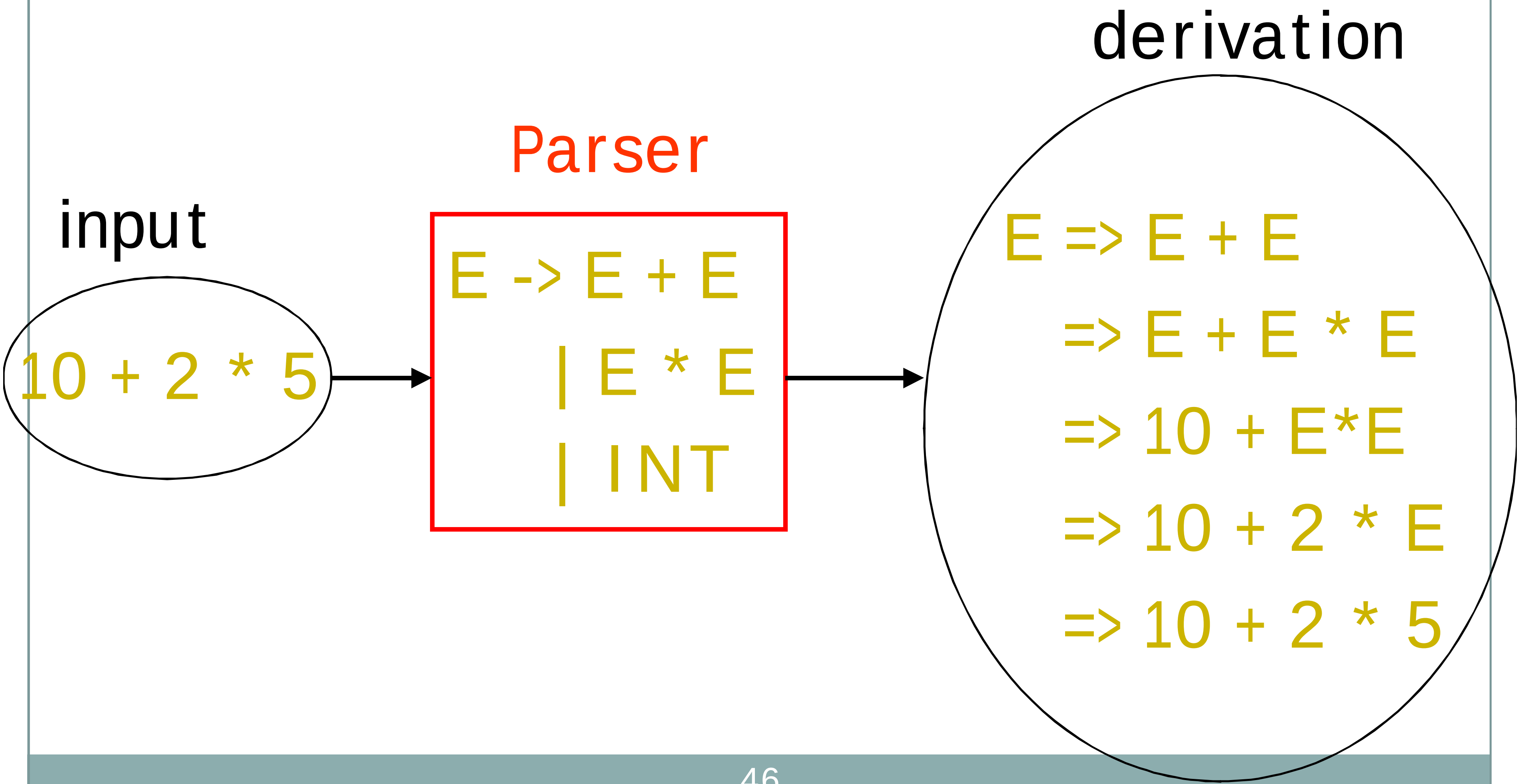
EXPR → EXPR + EXPR | EXPR - EXPR | ID

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

E => E + E

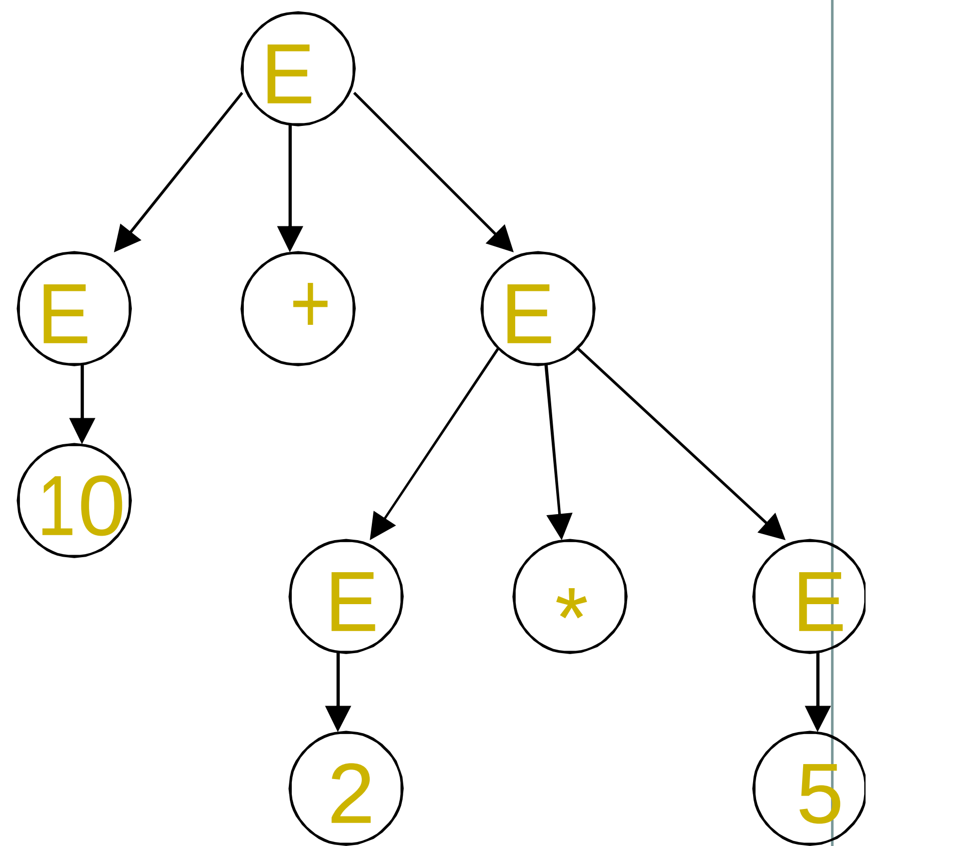
=> E + E * E

=> 10 + E*E

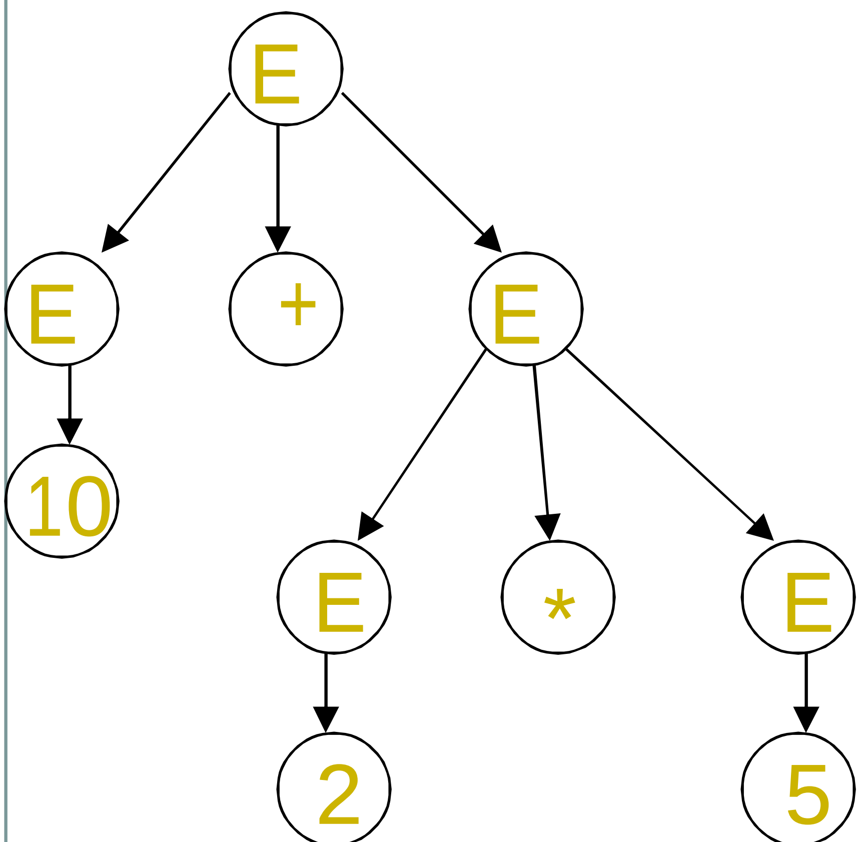
=> 10 + 2 * E

=> 10 + 2 * 5

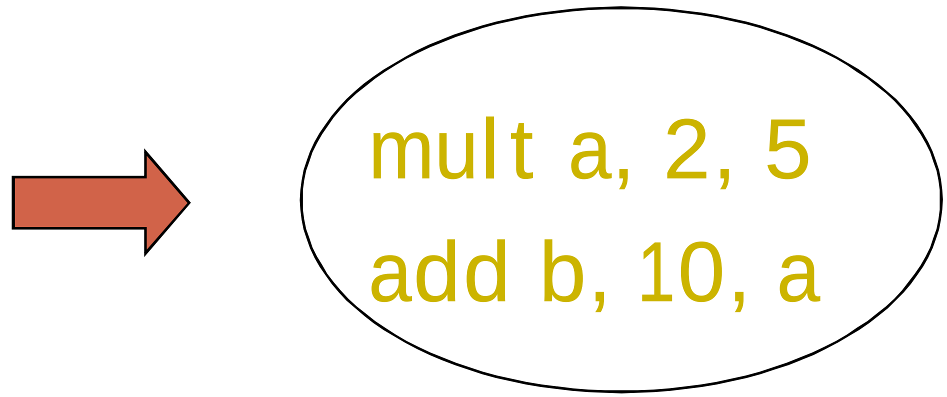
derivation tree



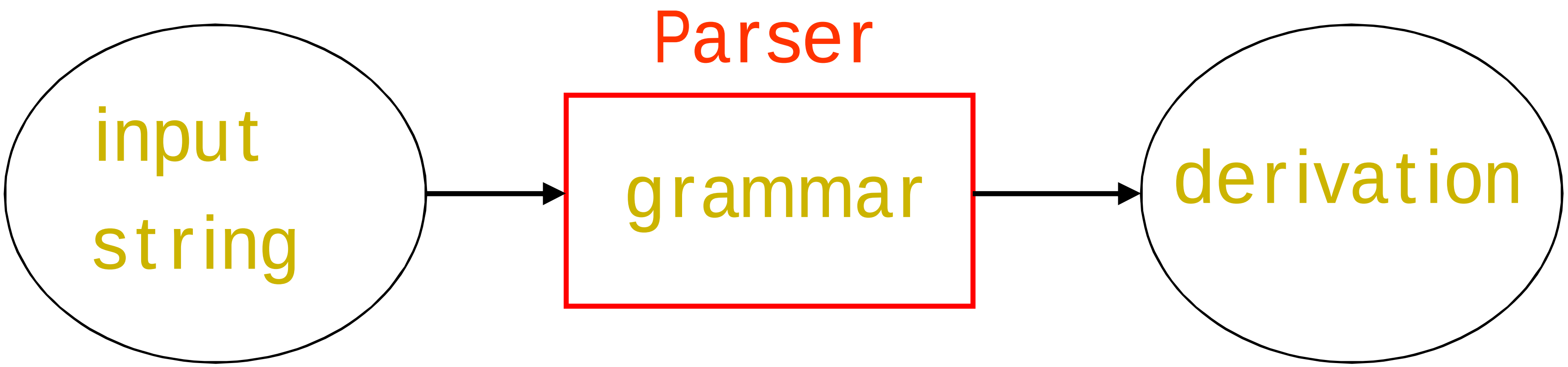
derivation tree



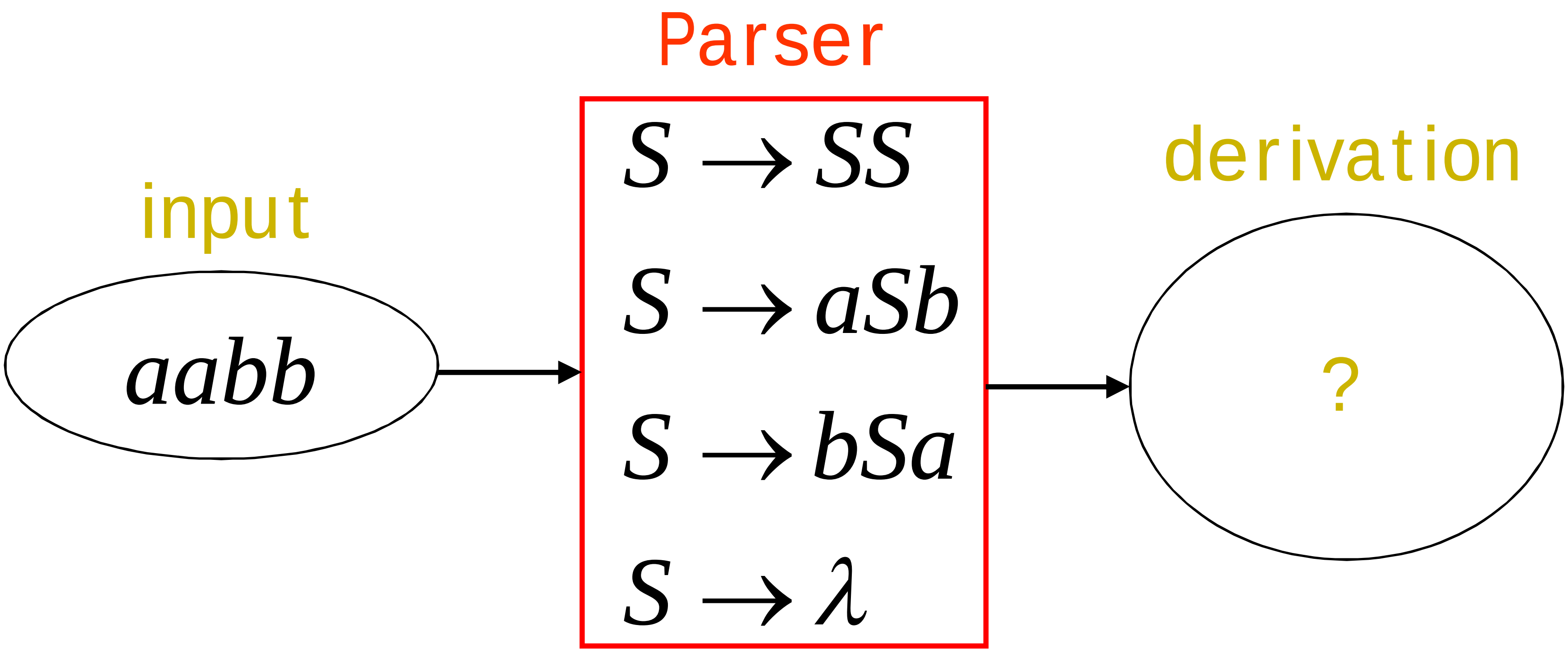
machine code



Parsing


Parser
input
string
grammar
derivation

Example:


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

$$S \Rightarrow SS$$

$$S \Rightarrow aSb$$

$$S \Rightarrow bSa$$

$$S \Rightarrow \lambda$$

$$aabb$$

Phase 2

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

$$S \Rightarrow SS \Rightarrow SSS$$

$$S \Rightarrow SS \Rightarrow aSbS$$

$aabb$

Phase 1

~~$S \Rightarrow SS \Rightarrow bSaS$~~

$$S \Rightarrow SS$$

$$S \Rightarrow SS \Rightarrow S$$

$$S \Rightarrow aSb$$

$$S \Rightarrow aSb \Rightarrow aSSb$$

$$S \Rightarrow aSb \Rightarrow aaSbb$$

~~$S \Rightarrow aSb \Rightarrow abSab$~~

~~$S \Rightarrow aSb \Rightarrow ab$~~

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

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)

input

$$aabb$$

Parser

$$S \rightarrow SS$$

$$S \rightarrow aSb$$

$$S \rightarrow bSa$$

$$S \rightarrow \lambda$$

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 $k^{|w|}$:

A total of $k^{|w|}$ 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