Compilers

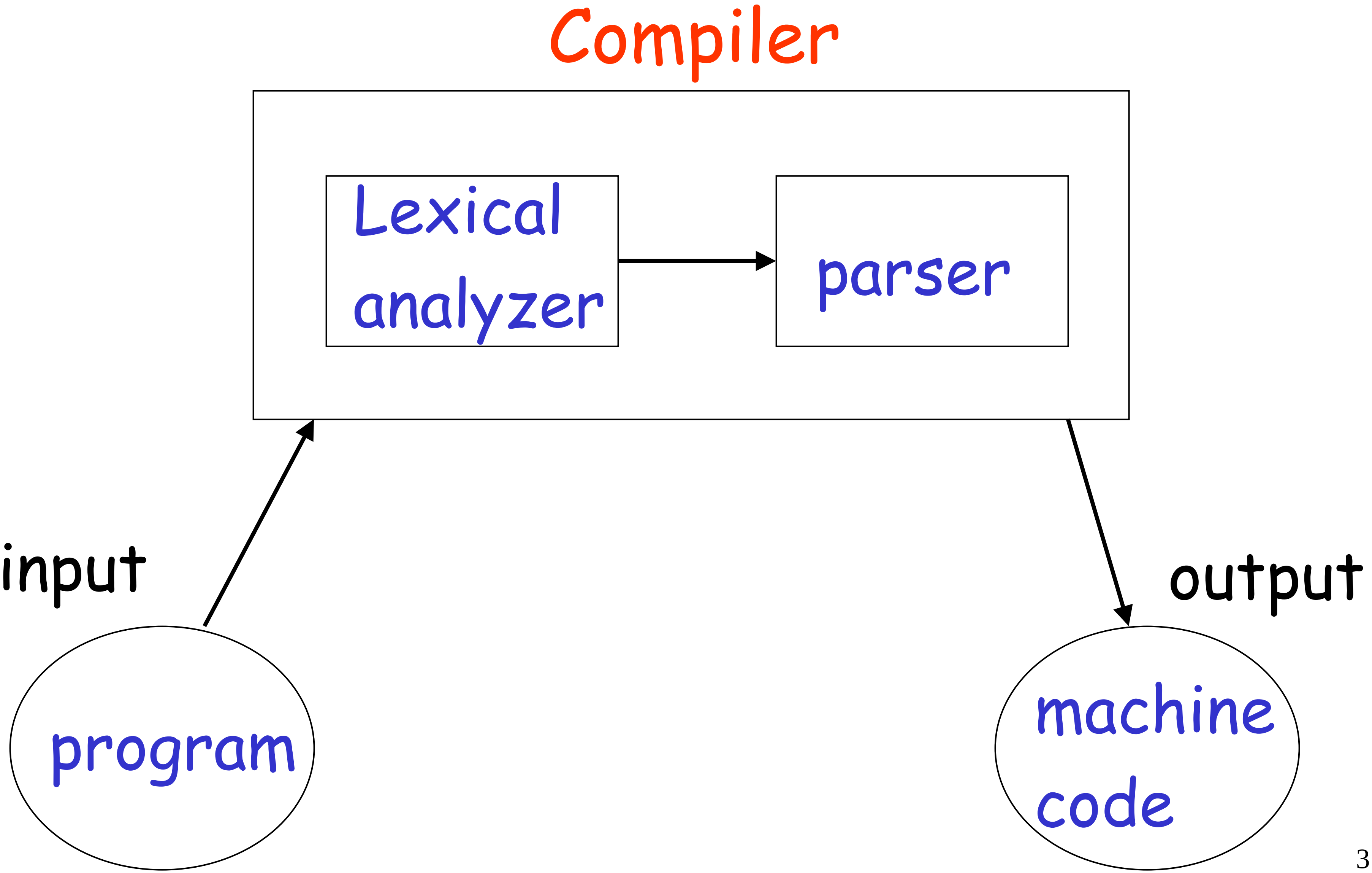
Program

```
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 }

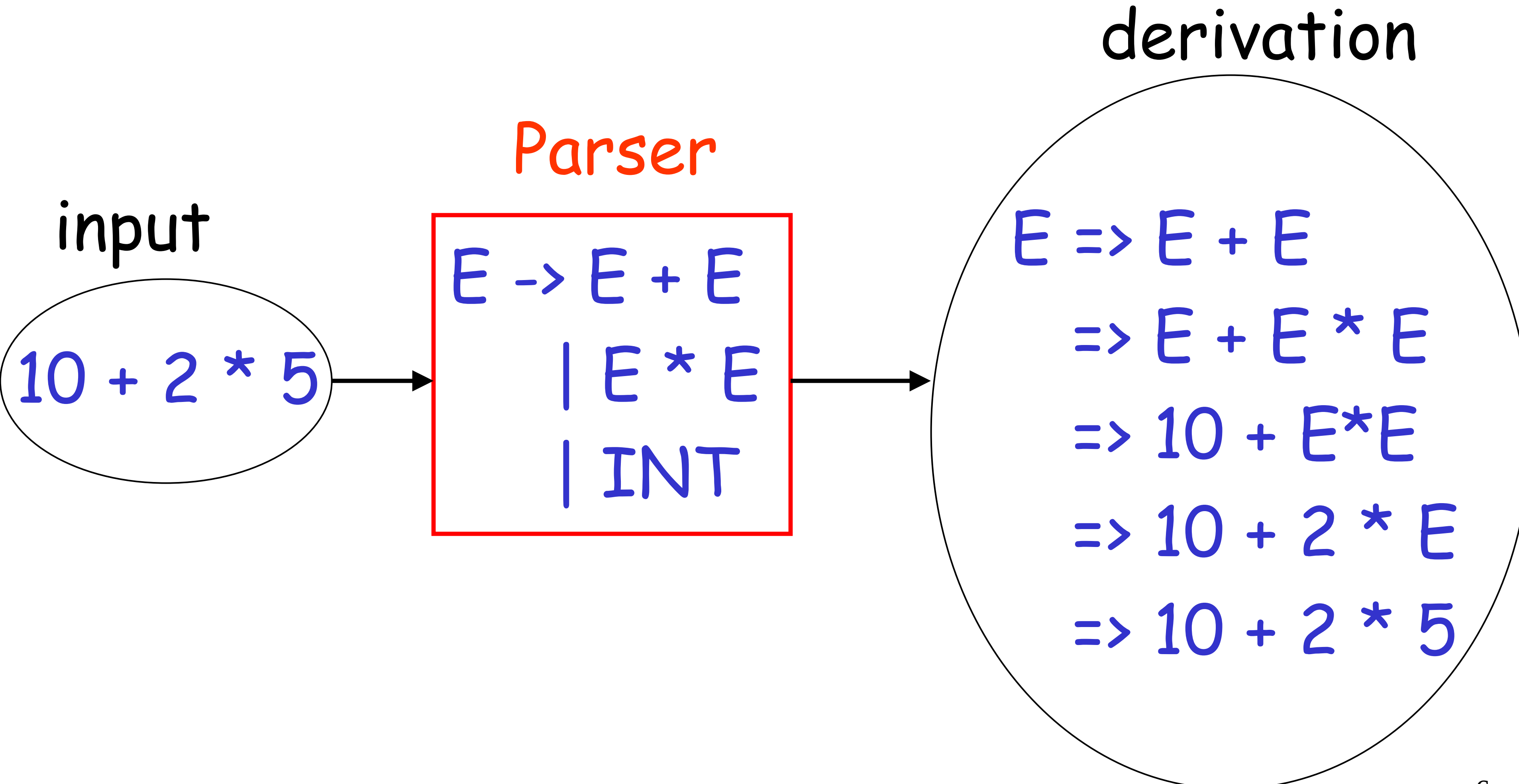
EXPR → EXPR + EXPR | EXPR - EXPR | ID

IF_STMT → if (EXPR) then STMT

| if (EXPR) then STMT else STMT

WHILE_STMT → while (EXPR) do STMT

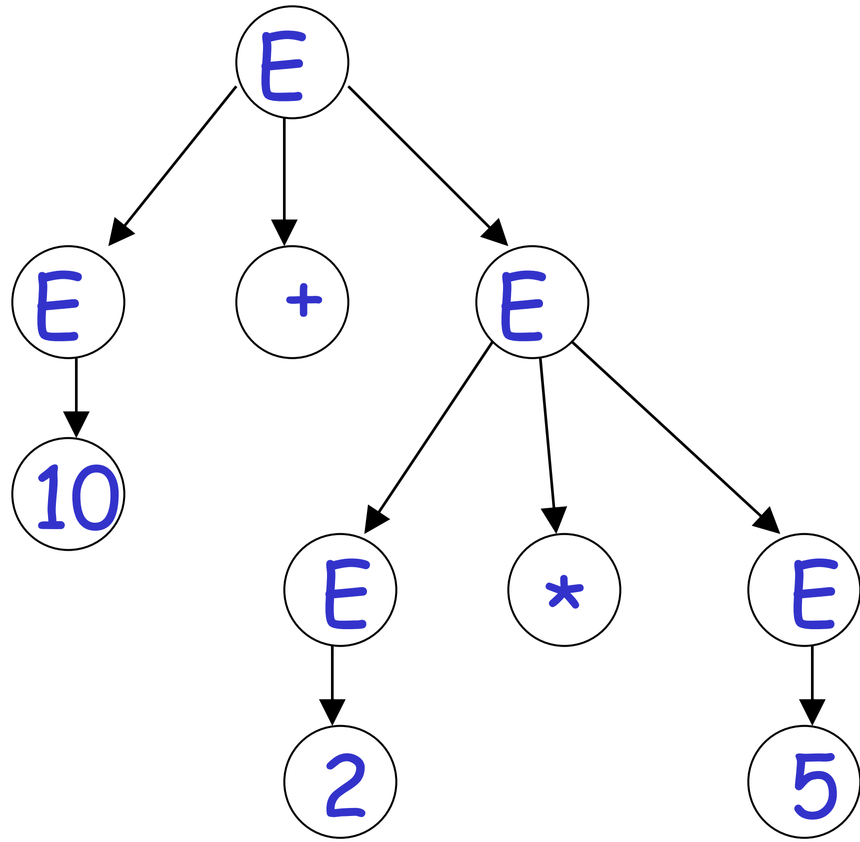
The parser finds the derivation of a particular input



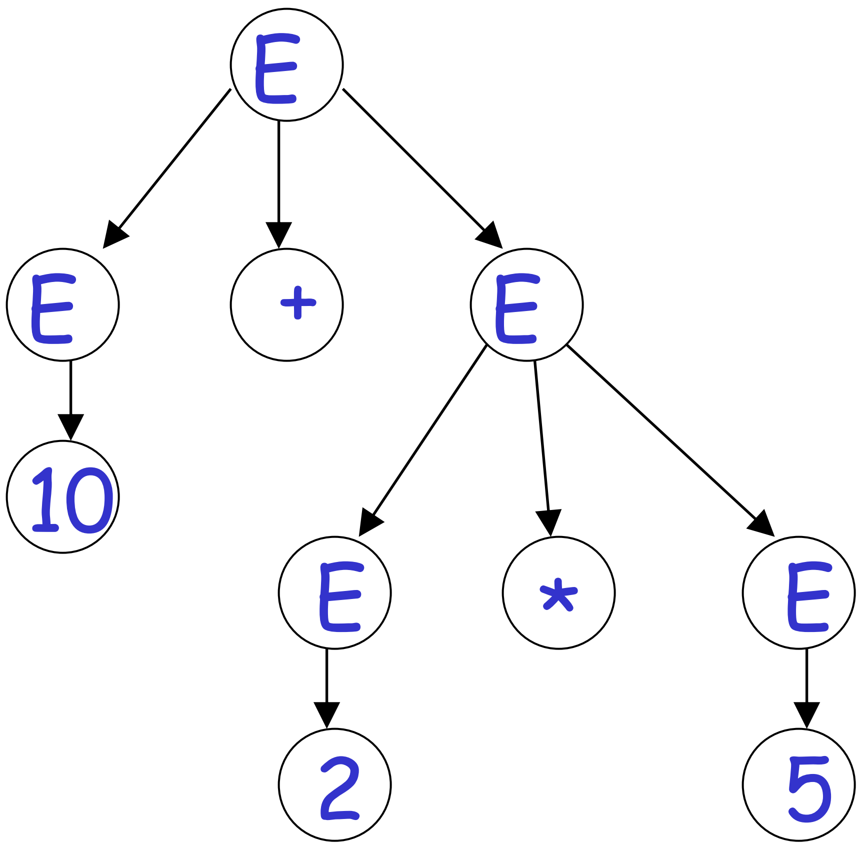
derivation

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

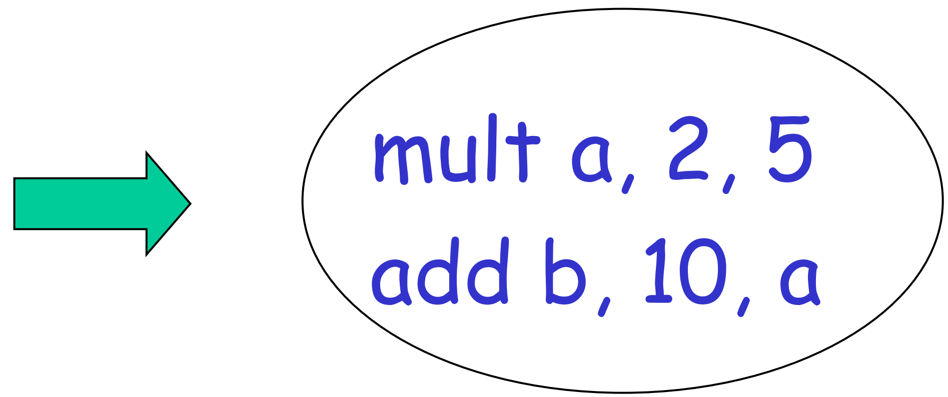
derivation tree



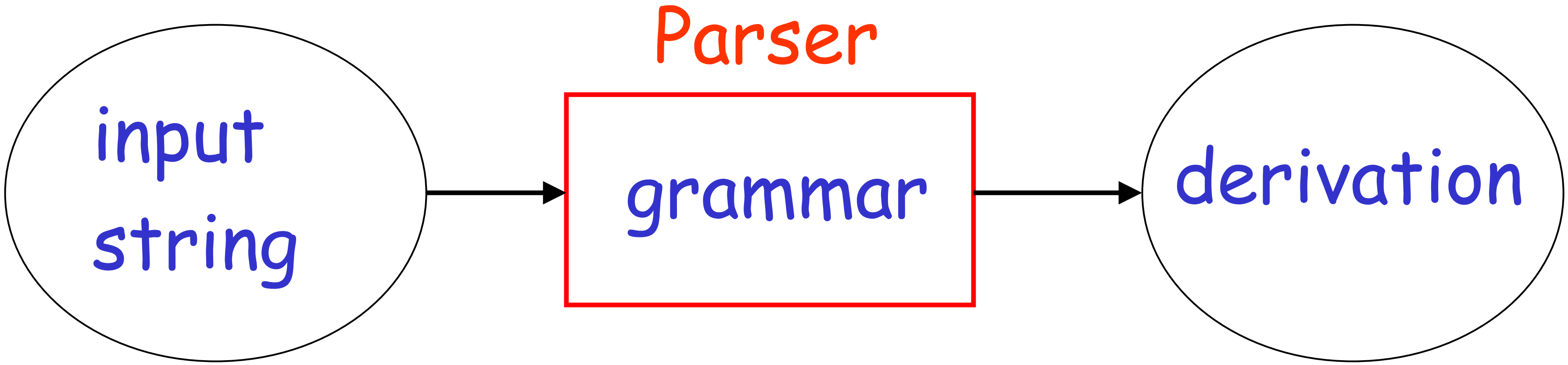
derivation tree



machine code



Parsing


Parser
input string
grammar
derivation

Example:

input

$$aabb$$

Parser

$$S \to SS$$

$$S \to aSb$$

$$S \to bSa$$

$$S \to \lambda$$

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

$S \Rightarrow SS$

$aabb$

$S \Rightarrow aSb$

$\cancel{S \Rightarrow bSa}$

$\cancel{S \Rightarrow \lambda}$

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

$$A \rightarrow \lambda$$

$$A \rightarrow B$$

Number of phases for string w : $2|w|$

Each step in the derivation increases the length of the sentential form or the number of nonterminal symbols.

Why 2|w| and not just |w|?

Because $S \Rightarrow AS \Rightarrow ABS \Rightarrow ABC \Rightarrow aBC \Rightarrow abC \Rightarrow abc$

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 $2|w|$: $k^{2|w|}$

$k^{2|w|}$ possible derivations

Total time needed for string w:

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

phase 1

phase 2

phase $2|w|$

Extremely bad!!!

There exist faster algorithms
for specialized grammars

S-grammar: $A \rightarrow ax$

symbol

string
of variables

Pair (A, a) appears once

S-grammar example:

$$S \to aS$$

$$S \to bSS$$

$$S \to c$$

Each string has a unique derivation...

$$S \Rightarrow aS \Rightarrow abSS \Rightarrow abcS \Rightarrow abcc$$

For S-grammars:

In the exhaustive search parsing
there is only one choice in each phase

Time for a phase: 1

Total time for parsing string w: $|w|$

For general context-free grammars:

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

Linz 6th, Theorem 5.3, page 144.

Linz 6th, Section 6.3, page 178ff – the CYK algorithm.

Simplifications of Context-Free Grammars

A Substitution Rule

Equivalent grammar

$$A \to a$$
$$A \to aaA$$
$$A \to abBc$$
$$B \to abbA$$
$$B \to b$$

Substitute B

$$A \to a$$
$$A \to aaA$$
$$A \to ababbAc$$
$$A \to abbc$$

In general:

$$A \to xBz$$

$$B \to y_1 \mid y_2 \mid \cdots \mid y_n$$

Substitute
B

$$A \to xy_1z \mid xy_2z \mid \cdots \mid xy_nz$$

equivalent grammar

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:

$$S \rightarrow A$$

$$A \rightarrow aA$$

$$A \rightarrow \lambda$$

$$B \rightarrow bA$$ Useless Production

Not reachable from S

In general:

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 $A \rightarrow x$ is useful if all its variables are useful

Removing Useless Productions

Example Grammar:

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

$$A \rightarrow a$$

$$B \rightarrow aa$$

$$C \rightarrow aCb$$

First: find all variables that produce strings with only terminals

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

$$A \rightarrow a$$

$$B \rightarrow aa$$

$$C \rightarrow aCb$$

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

$\Downarrow$

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

Keep only the variables that produce terminal symbols

$$\{A, B, S\}$$

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

$$A \to a$$

$$B \to aa$$

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

$$\Rightarrow$$

$$S \to aS \mid A$$

$$A \to a$$

$$B \to aa$$

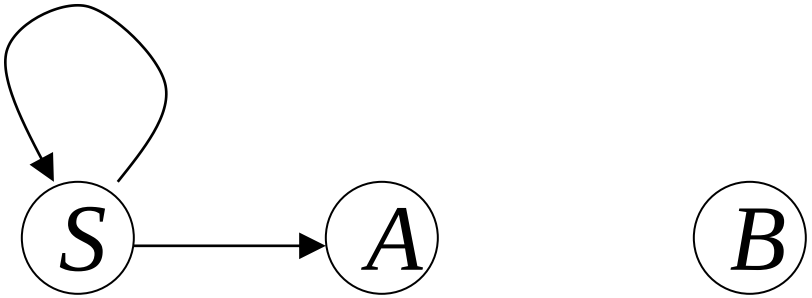
Second: Find all variables reachable from S

$$S \rightarrow aS \mid A$$

$$A \rightarrow a$$

$$B \rightarrow aa$$

Dependency Graph



not reachable

Keep only the variables reachable from S

$$S \rightarrow aS \mid A$$

$$A \rightarrow a$$

$$\cancel{B \rightarrow aa}$$

Final Grammar

$$S \rightarrow aS \mid A$$

$$A \rightarrow a$$

Nullable Variables

$\lambda -$ production : $\quad A \to \lambda$

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

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: $A \rightarrow B$

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

$$B \to B$$

$$S \to aA \mid aB$$

$$A \to a$$

$$B \to A$$

$$B \to bb$$

$$S \rightarrow aA \mid aB$$

$$A \rightarrow a$$

$$\cancel{B \rightarrow A}$$

$$B \rightarrow bb$$

Substitute

$$B \rightarrow A$$

$$S \rightarrow aA \mid aB \mid aA$$

$$A \rightarrow a$$

$$B \rightarrow bb$$

Remove repeated productions

Final grammar

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

$$A \rightarrow a$$

$$B \rightarrow bb$$

$$S \rightarrow aA \mid aB$$

$$A \rightarrow a$$

$$B \rightarrow bb$$

Removing All

Step 1: Remove Nullable Variables

Step 2: Remove Unit-Productions

Step 3: Remove Useless Variables