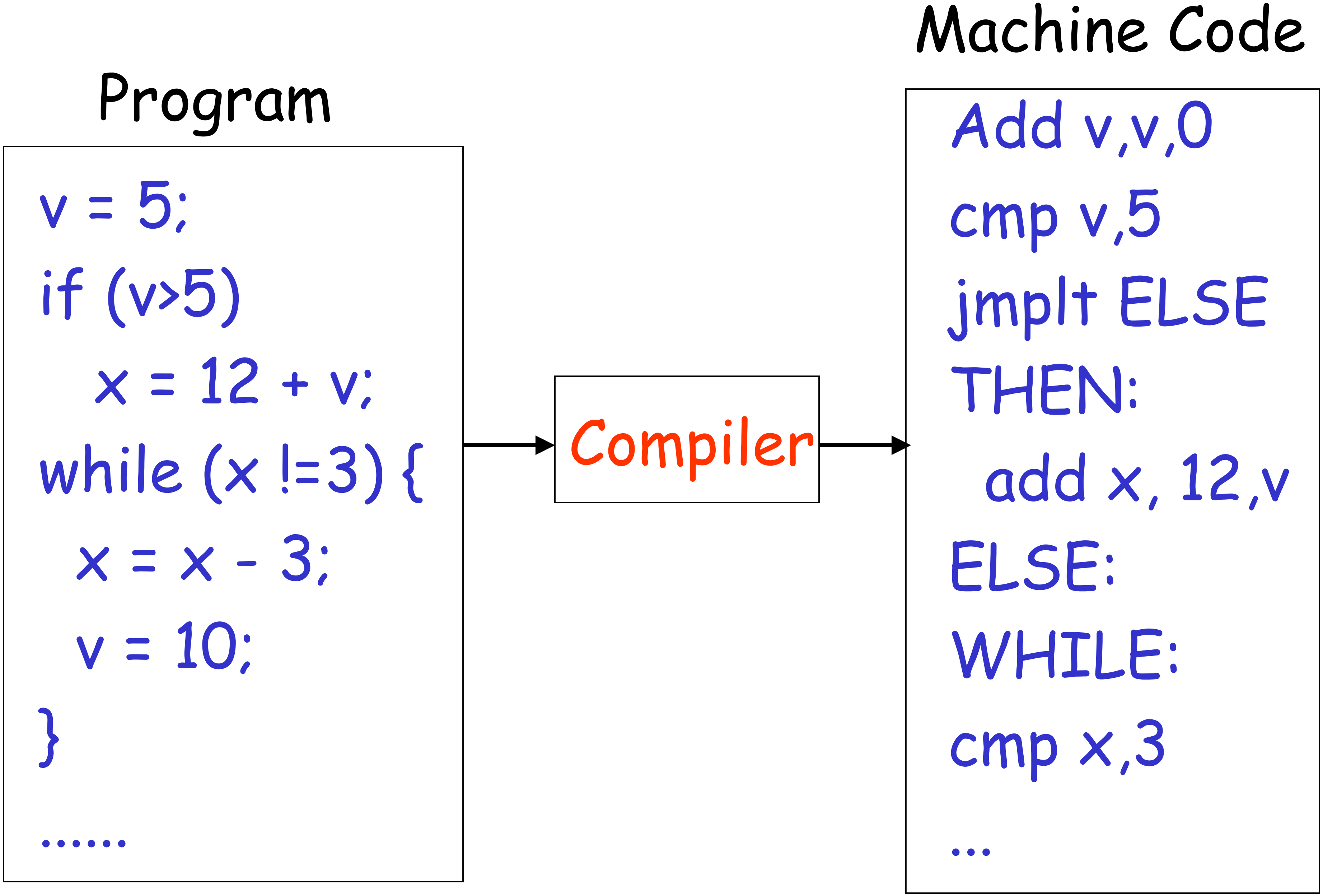
Compilers



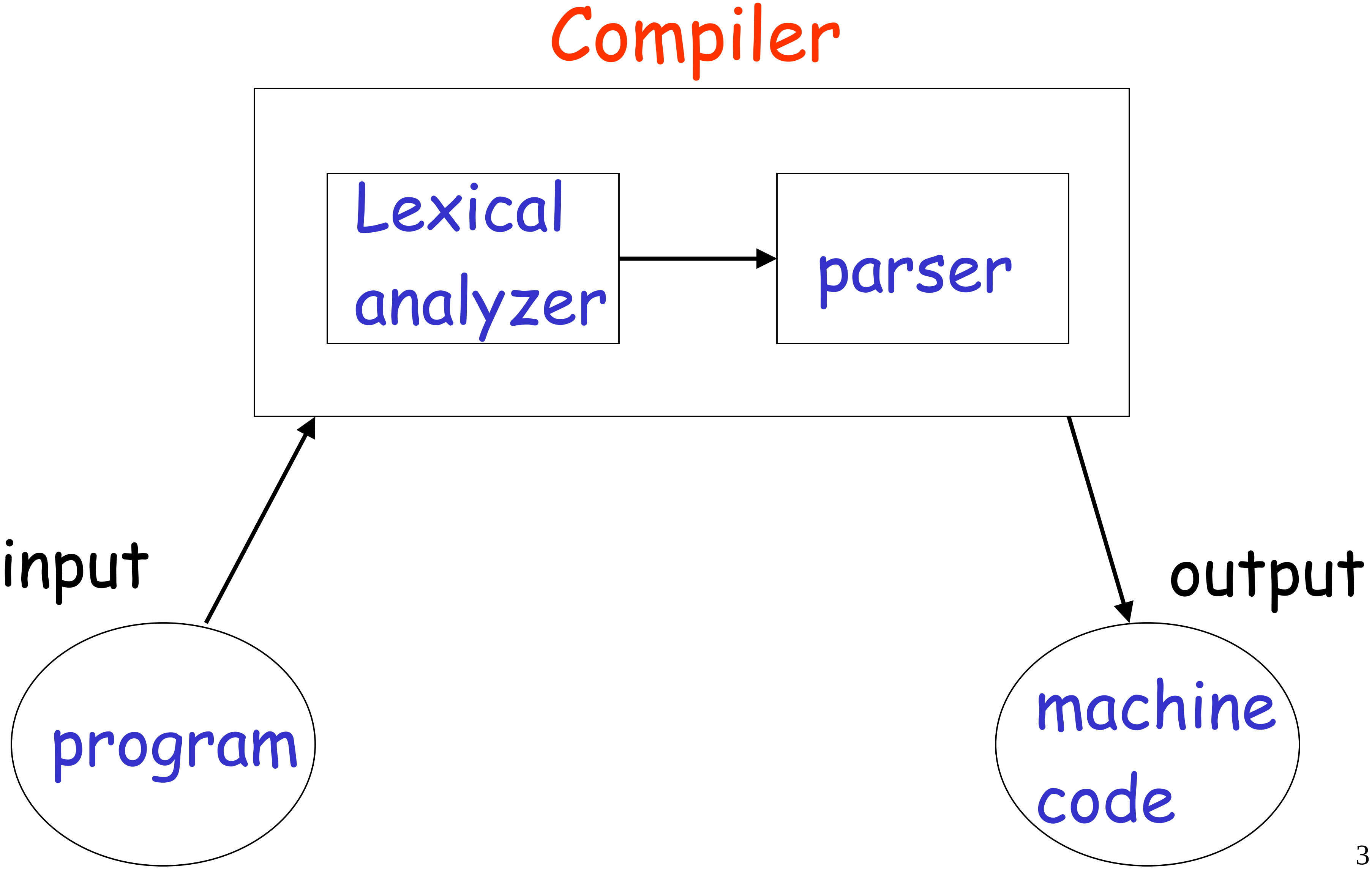
Program

```
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;

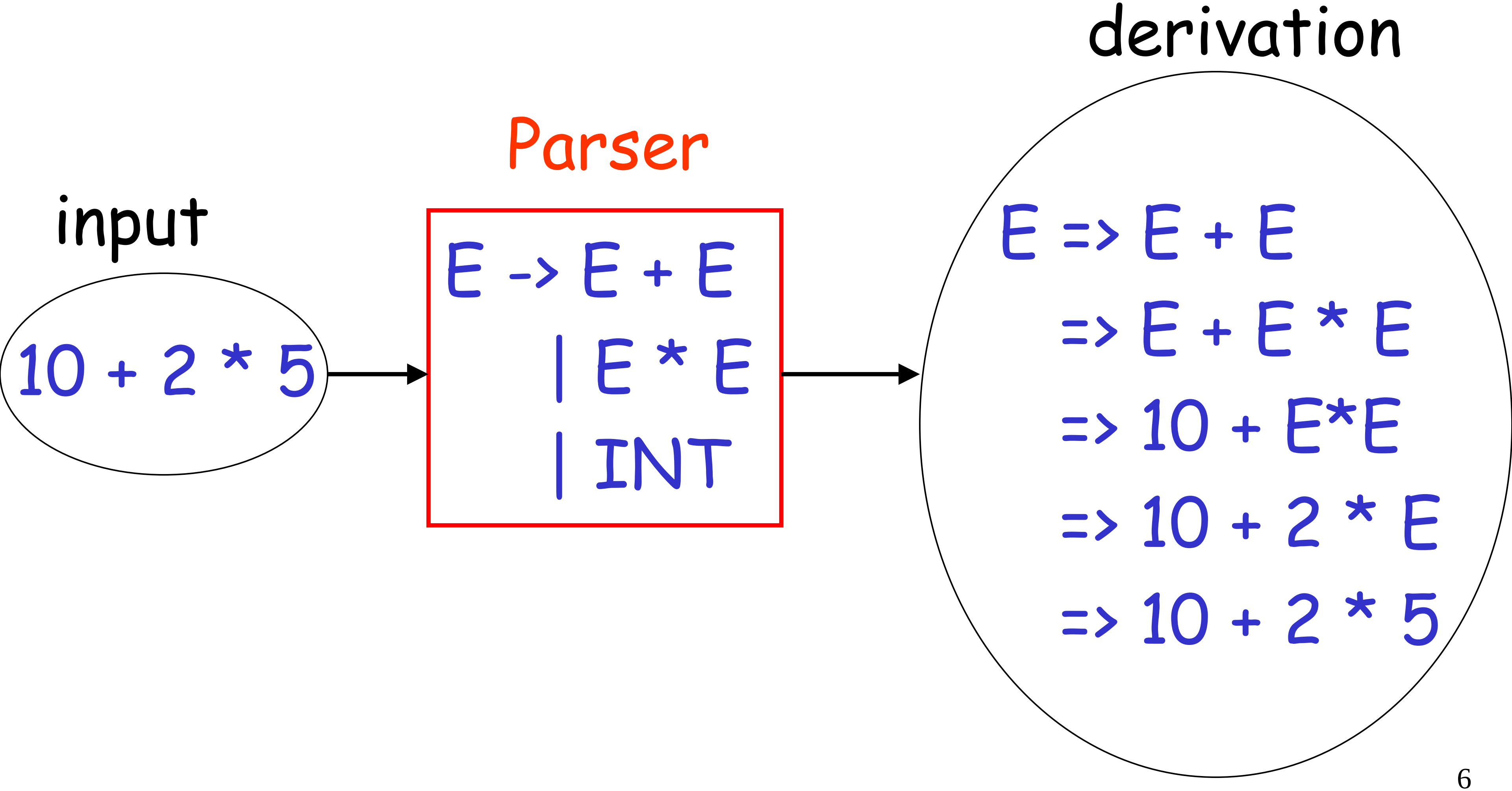
STMT → EXPR | IF_STMT | WHILE_STMT
| { STMT_LIST }

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



input

10 + 2 * 5

Parser

```
E -> E + E
   | E * E
   | INT
```

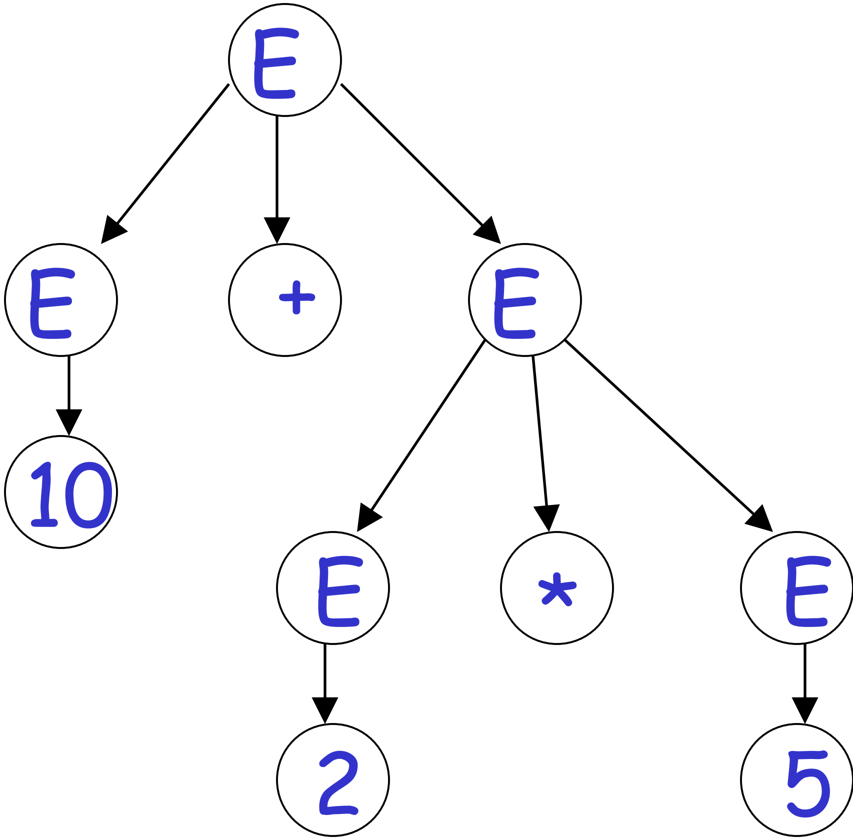
derivation

```
E => E + E
  => E + E * E
  => 10 + E*E
  => 10 + 2 * E
  => 10 + 2 * 5
```

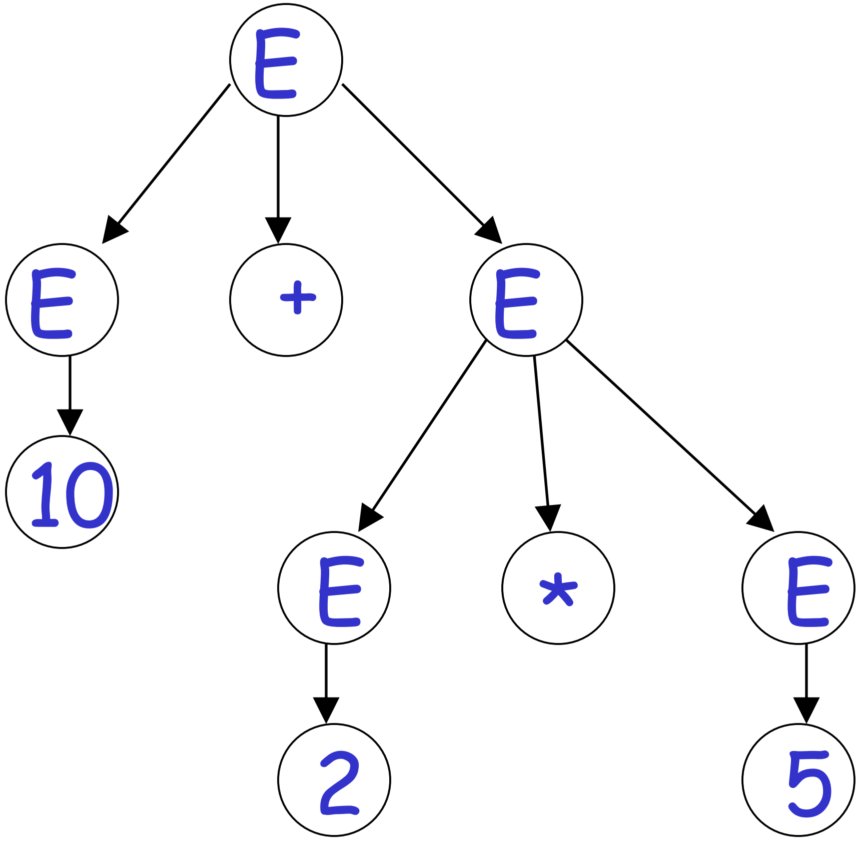
derivation

E => E + E
=> E + E * E
=> 10 + E*E
=> 10 + 2 * E
=> 10 + 2 * 5

derivation tree



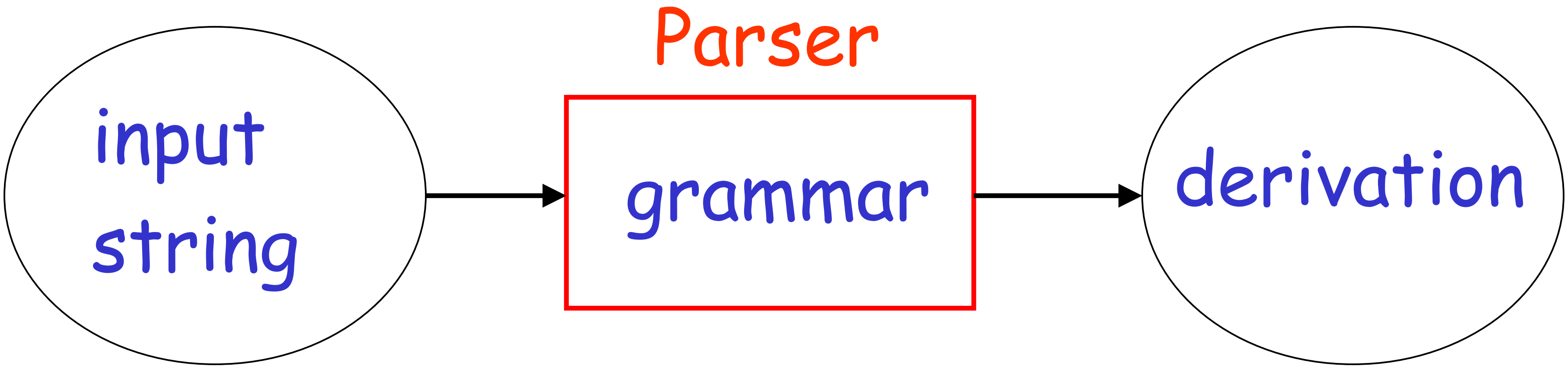
derivation tree



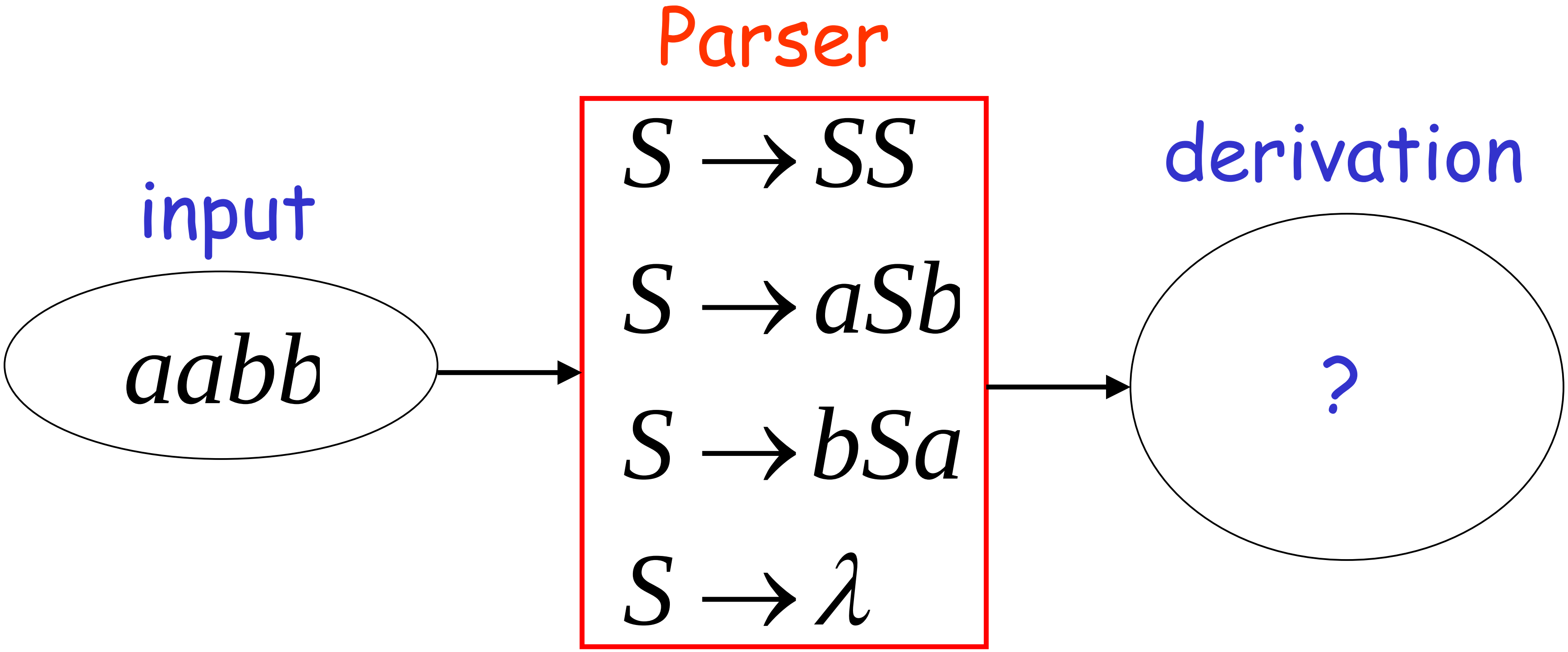
machine code

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

Parsing


Parser
input string
grammar
derivation

Example:



Parser

$S \rightarrow SS$

$S \rightarrow aSb$

$S \rightarrow bSa$

$S \rightarrow \lambda$

Exhaustive Search

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

Phase 1:

$$S \Rightarrow SS$$

$$S \Rightarrow aSb$$

$$S \Rightarrow bSa$$

$$S \Rightarrow \lambda$$

Find derivation of

$aabb$

All possible derivations of length 1

$S \Rightarrow SS$ $aabb$

$S \Rightarrow aSb$

$\xcancel{S \Rightarrow bSa}$

$\xcancel{S \Rightarrow \lambda}$

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

$aabb$

Phase 1

$S \Rightarrow SS$

$S \Rightarrow SS \Rightarrow SSS$

$S \Rightarrow SS \Rightarrow aSbS$

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

$S \Rightarrow SS \Rightarrow S$

$S \Rightarrow aSb$

$S \Rightarrow aSb \Rightarrow aSSb$

$S \Rightarrow aSb \Rightarrow aaSbb$

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

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

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

Phase 2

$$S \Rightarrow SS \Rightarrow SSS$$

$$S \Rightarrow SS \Rightarrow aSbS$$

$$aabb$$

$$S \Rightarrow SS \Rightarrow S$$

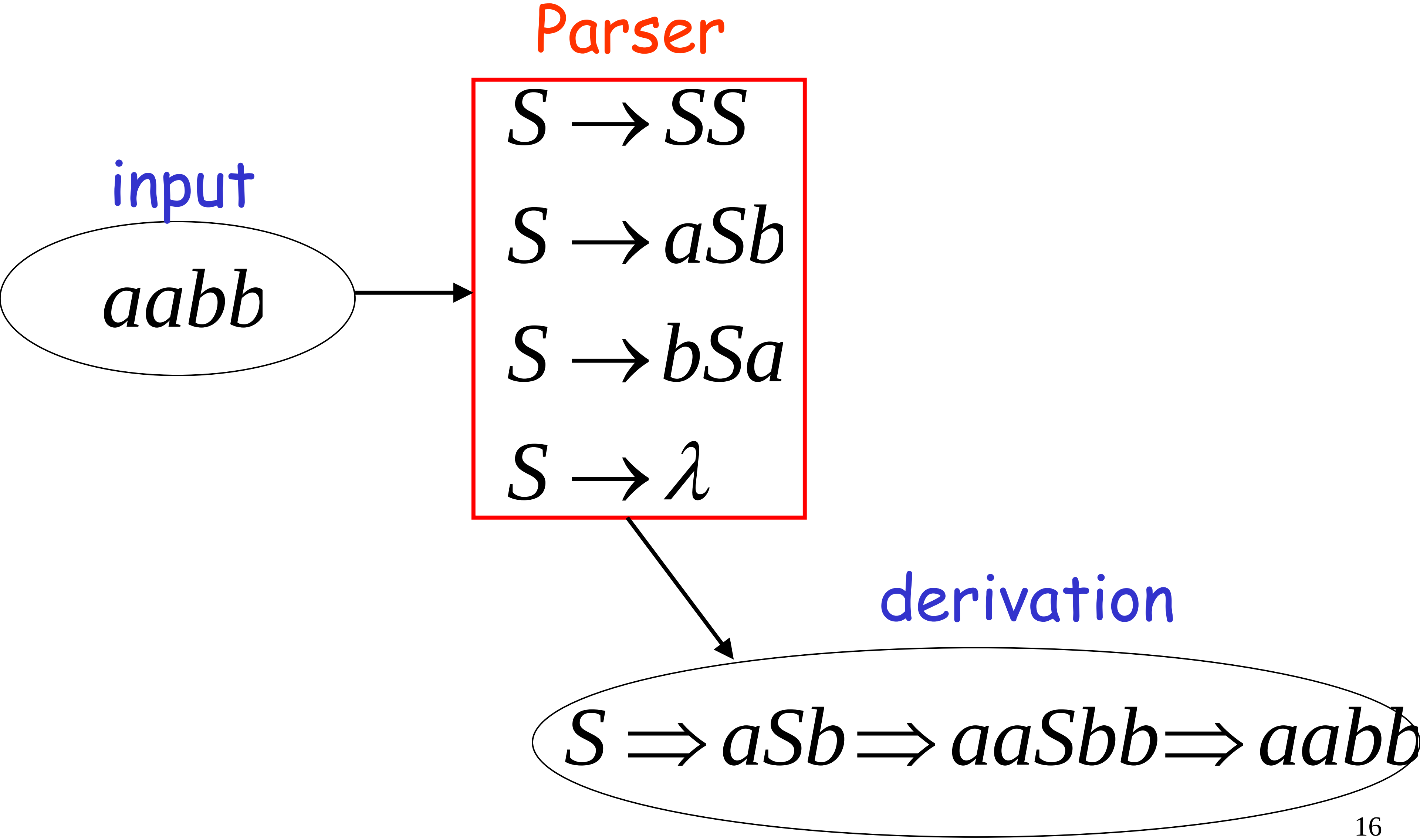
$$S \Rightarrow aSb \Rightarrow aSSb$$

$$S \Rightarrow aSb \Rightarrow aaSbb$$

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 : $2|w|$

For grammar with k rules

Time for phase 1: k

k possible derivations

Time for phase 2: k^2

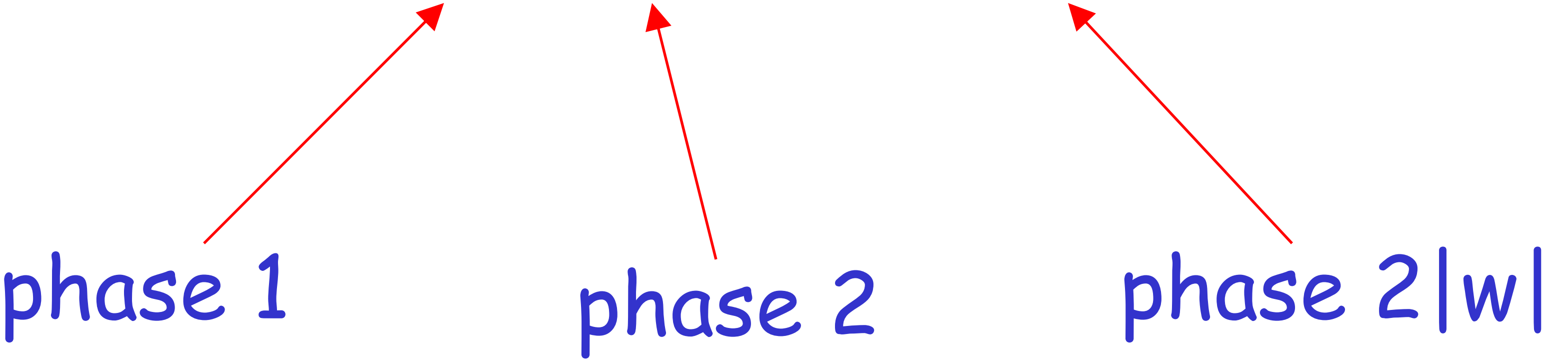
k^2 possible derivations

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

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

Total time needed for string w:

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



phase 1

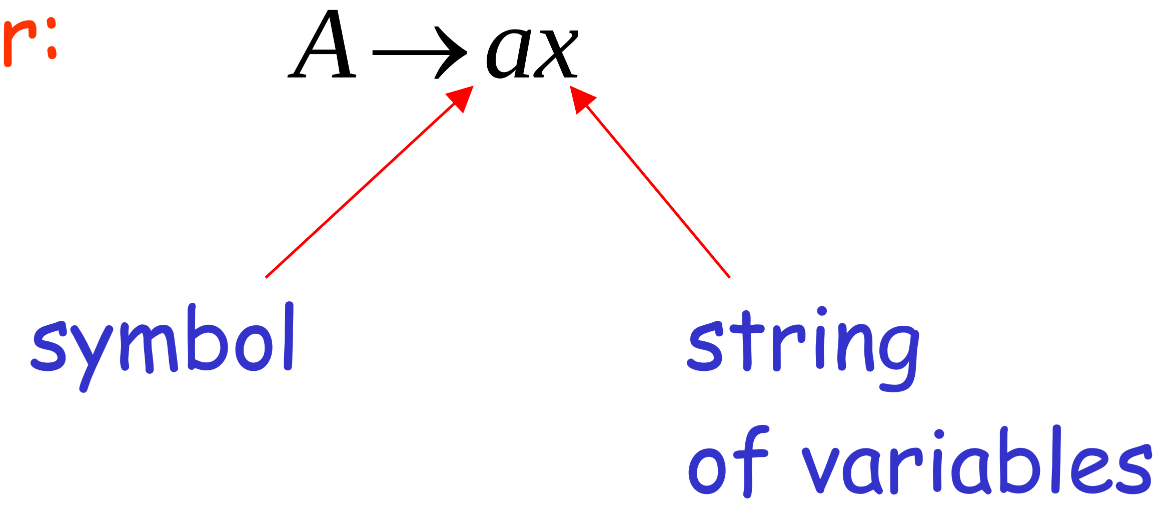
phase 2

phase 2|w|

Extremely bad!!!

There exist faster algorithms for specialized grammars

S-grammar:



$A \to ax$

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 $|w|^3$

Simplifications
of
Context-Free Grammars

A Substitution Rule

$$A \to a$$

$$A \to aaA$$

$$A \to abBc$$

$$B \to abbA$$

$$B \to b$$

Substitute B

Equivalent grammar

$$A \to a$$

$$A \to aaA$$

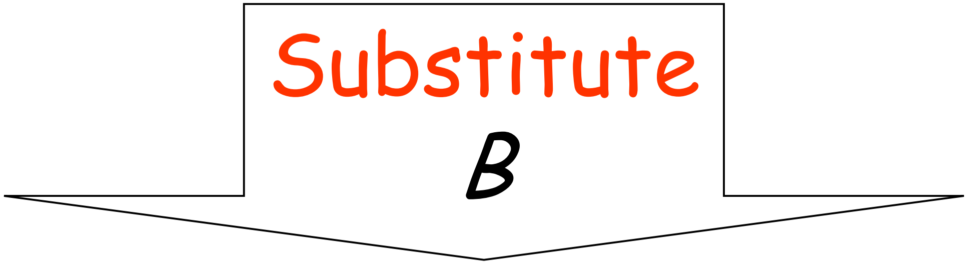
$$A \to ababbAc$$

$$A \to abbc$$

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

$$S \to aSb$$

$$S \to \lambda$$

$$S \to A$$

$$\boxed{A \to aA}$$ Useless Production

Some derivations never terminate...

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

Another grammar:

$$S \to A$$

$$A \to aA$$

$$A \to \lambda$$

$$\boxed{B \to bA}$$ Useless Production

Not reachable from S

In general:

If $$S \Rightarrow \ldots \Rightarrow xAy \Rightarrow \ldots \Rightarrow \underset{\substack{\uparrow \\ w \in L(G)}}{w}$$

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:

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

$$A \to a$$

$$B \to aa$$

$$C \to aCb$$

First: find all variables that produce strings with only terminals

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

$$A \to a$$

$$B \to aa$$

$$C \to aCb$$

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

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

Keep only the variables that produce terminal symbols

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

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

$$A \to a$$

$$B \to aa$$

$$\xcancel{C \to aCb}$$

➡

$$S \to aS \mid A$$

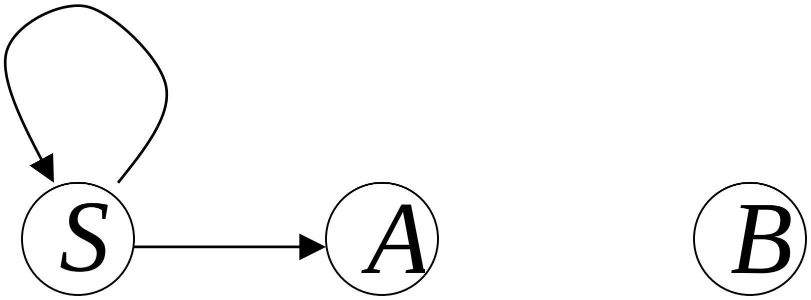
$$A \to a$$

$$B \to aa$$

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

$$S \to aS \mid A$$

$$A \to a$$

$$\xcancel{B \to aa}$$

$$\Longrightarrow$$

Final Grammar

$$S \to aS \mid A$$

$$A \to a$$

Nullable Variables

$\lambda-$production: $A \rightarrow \lambda$

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

Removing Nullable Variables

Example Grammar:

$$S \to aMb$$

$$M \to aMb$$

$$M \to \lambda$$

Nullable variable

$$S \to aMb$$

$$M \to aMb$$

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

Substitute

$$M \to \lambda$$

Final Grammar

$$S \to aMb$$

$$S \to ab$$

$$M \to aMb$$

$$M \to ab$$

Unit-Productions

Unit Production: $A \to B$

Removing Unit Productions

Observation:

$$A \rightarrow A$$

Is removed immediately

Example Grammar:

$$S \to aA$$

$$A \to a$$

$$A \to B$$

$$B \to A$$

$$B \to bb$$

$S \to aA$

$A \to a$

$\xcancel{A \to B}$

$B \to A$

$B \to bb$

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

$$B \to B$$

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

$$S \rightarrow aA \mid aB$$

$$A \rightarrow a$$

$$\xcancel{B \rightarrow A}$$

$$B \rightarrow bb$$

Substitute

$$B \rightarrow A$$

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

$$A \rightarrow a$$

$$B \rightarrow bb$$

Remove repeated productions

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

$$A \to a$$

$$B \to bb$$

Final grammar

$$S \to aA \mid aB$$

$$A \to a$$

$$B \to bb$$

Removing All

Step 1: Remove Nullable Variables

Step 2: Remove Unit-Productions

Step 3: Remove Useless Variables