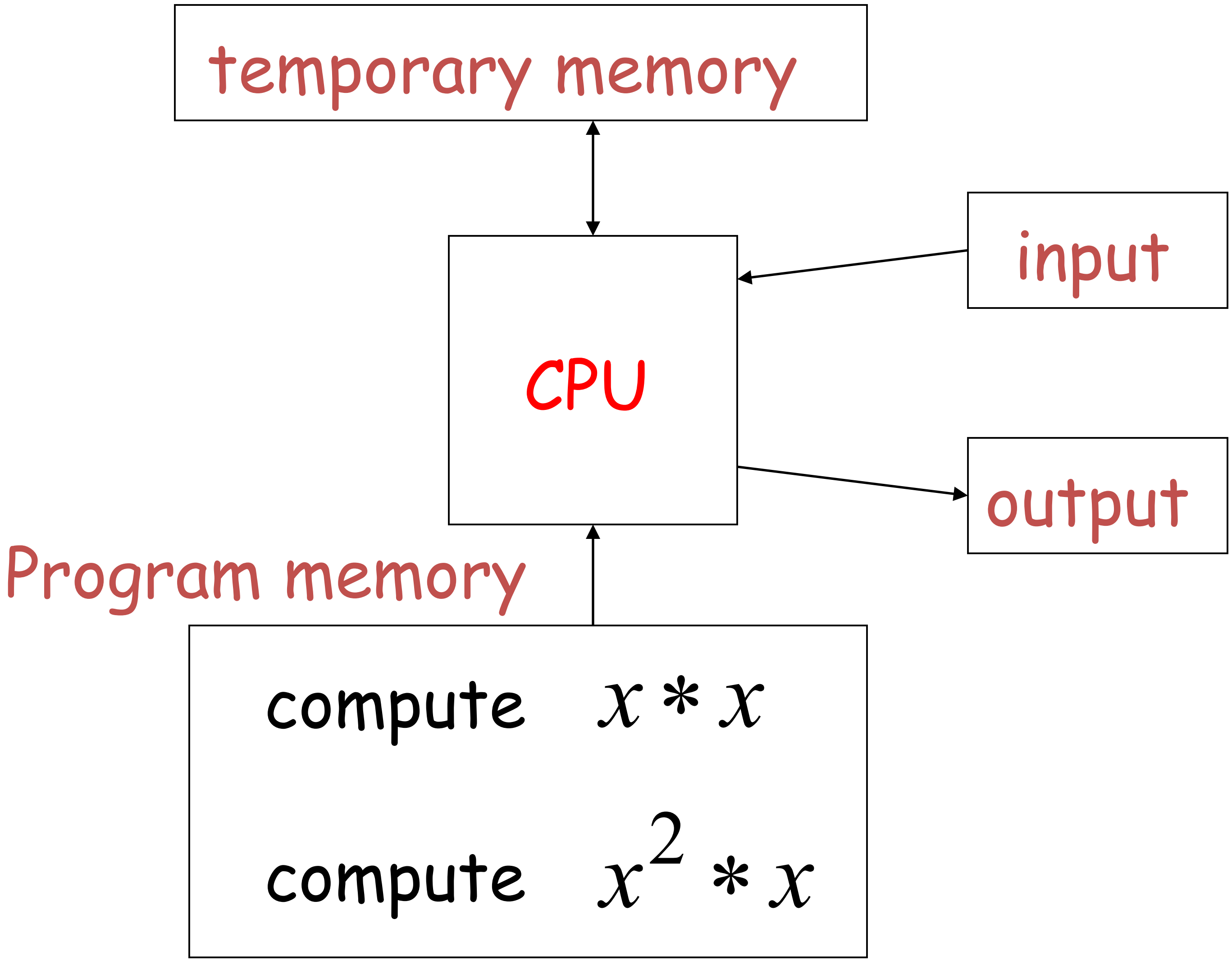
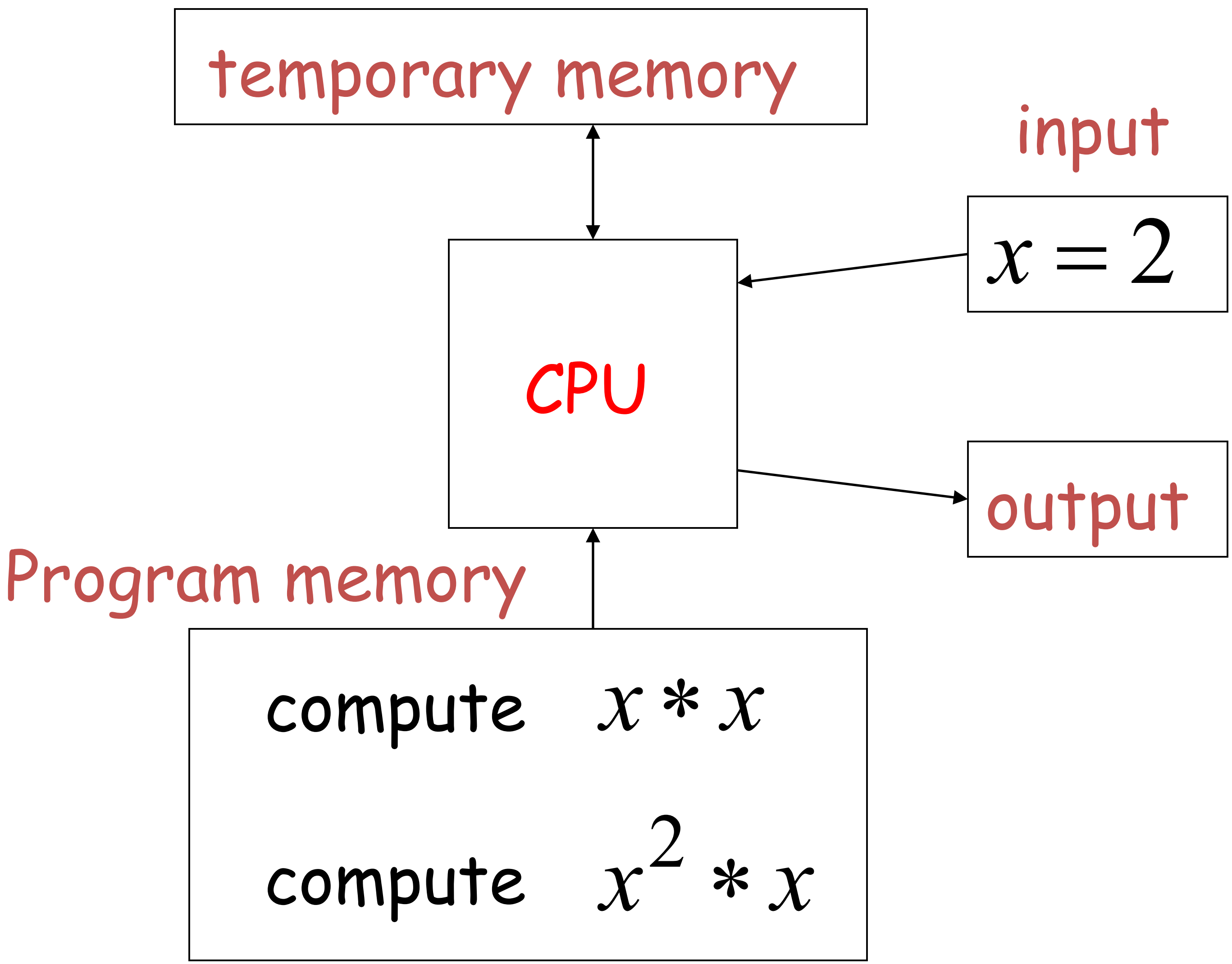
Context-Free Grammars

Example: $f(x) = x^3$

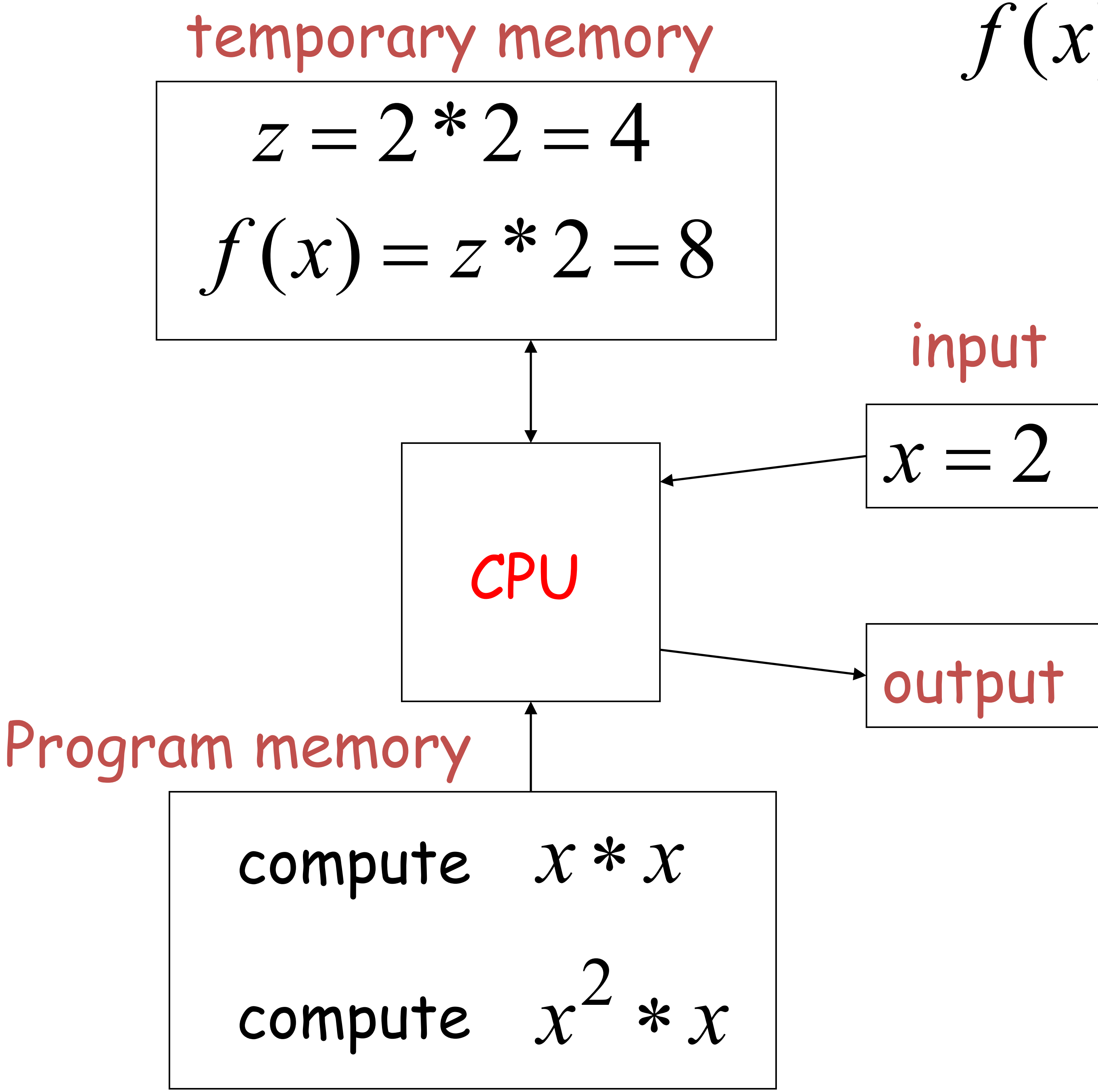


$$f(x) = x^3$$

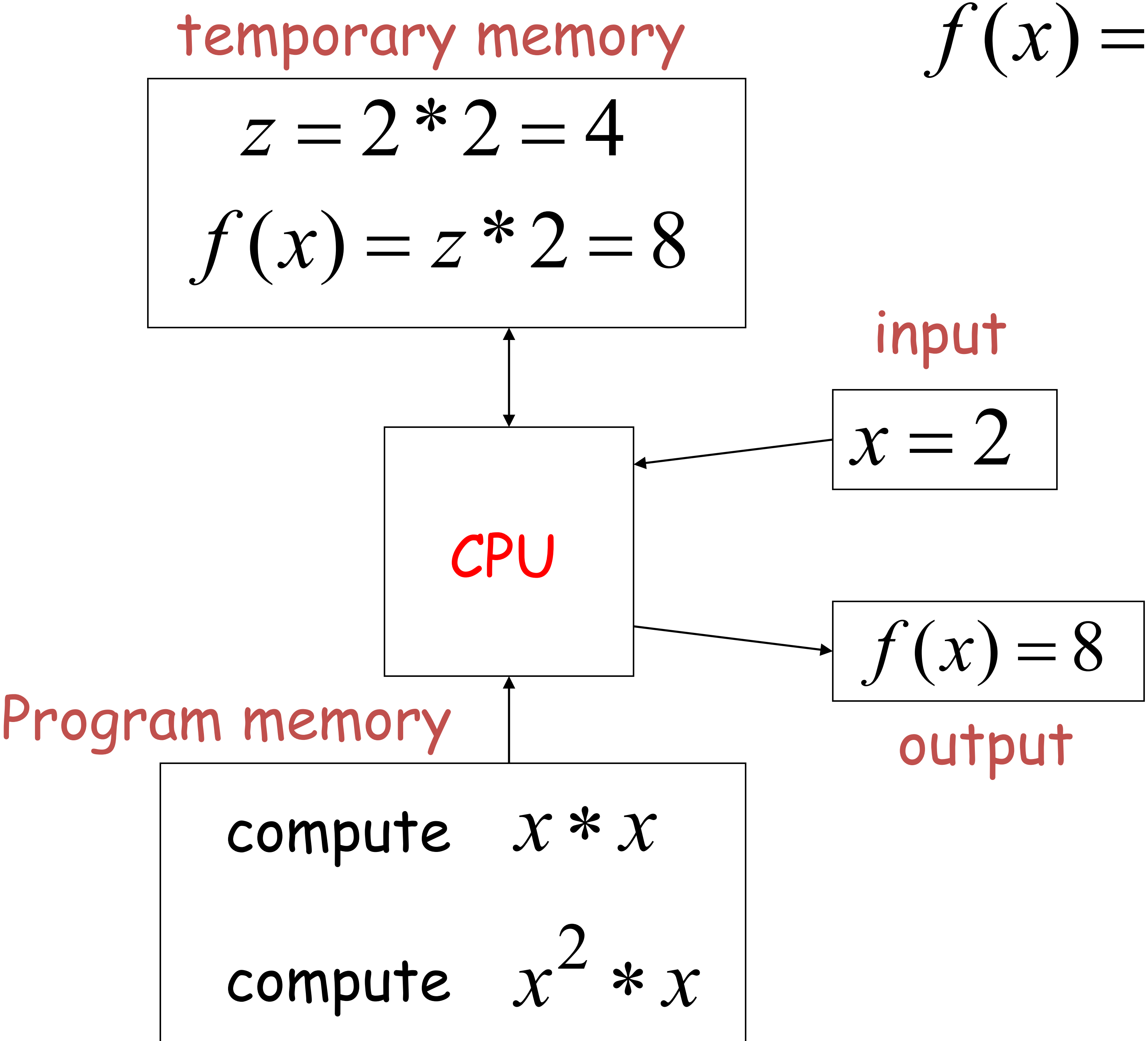


temporary memory
input
$x = 2$
CPU
output
Program memory
compute $x * x$
compute $x^2 * x$

$$f(x) = x^3$$

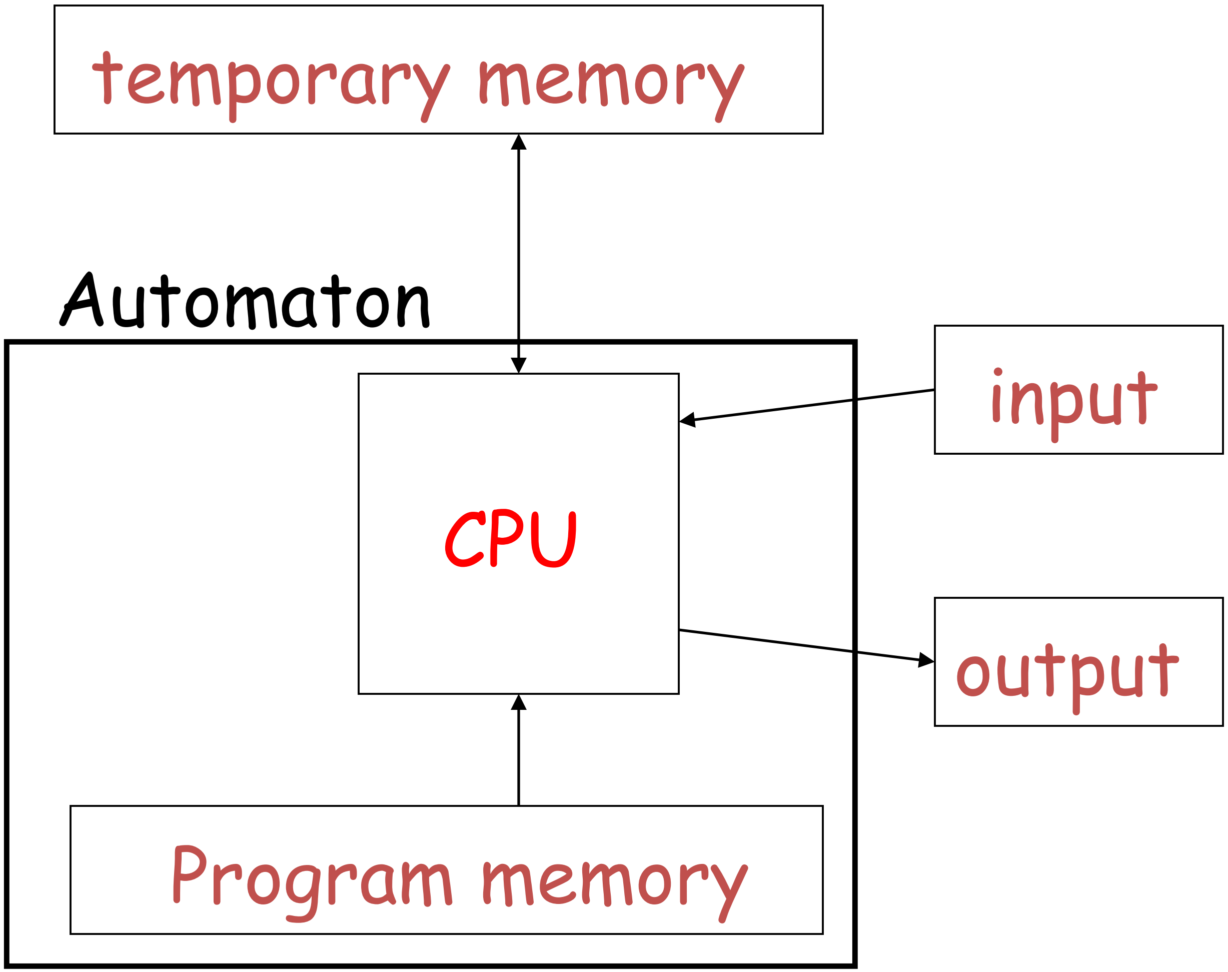

temporary memory
$z = 2 * 2 = 4$
$f(x) = z * 2 = 8$
input
$x = 2$
CPU
output
Program memory
compute $x * x$
compute $x^2 * x$

$$f(x) = x^3$$

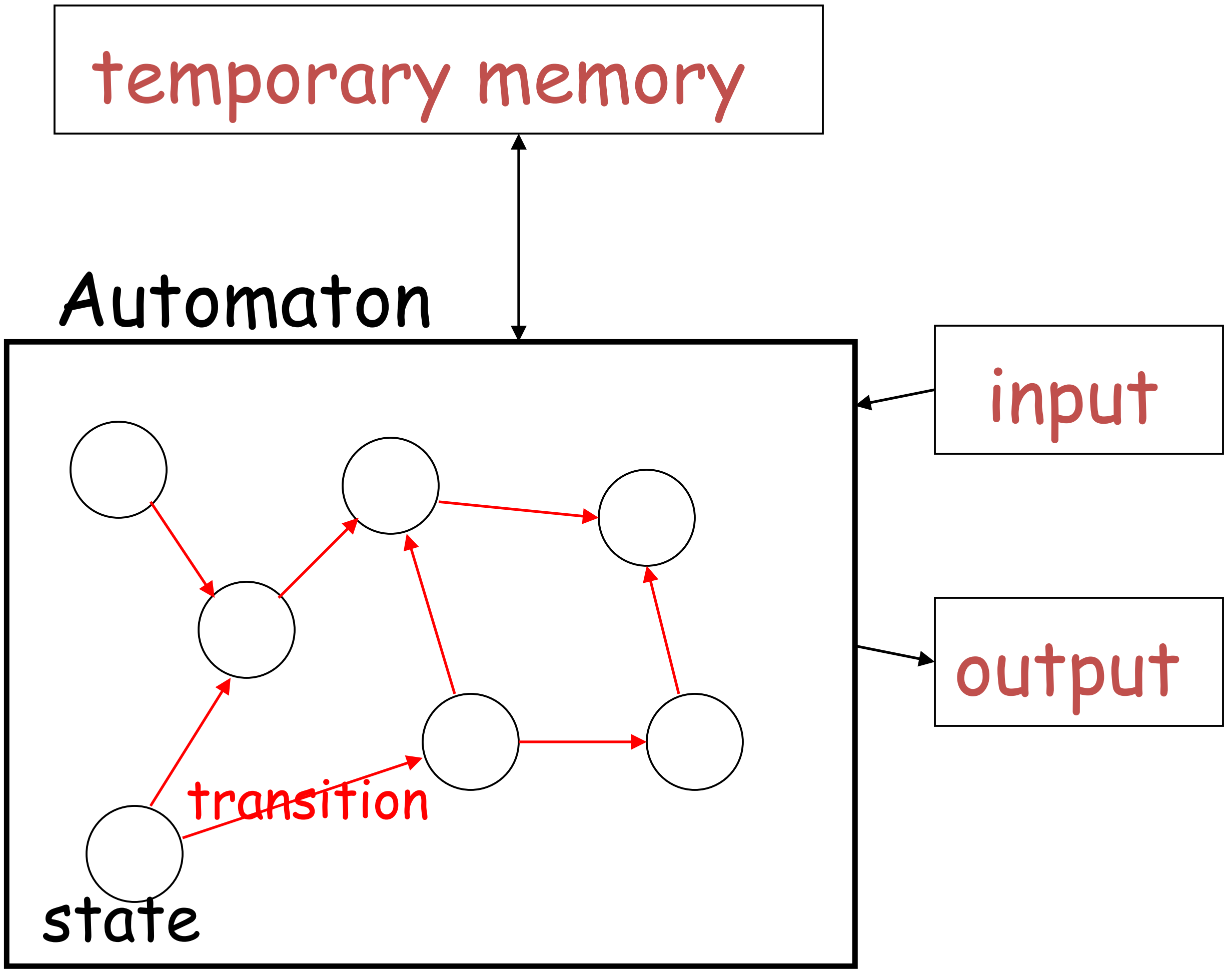


temporary memory
$z = 2 * 2 = 4$
$f(x) = z * 2 = 8$
input
$x = 2$
CPU
$f(x) = 8$
output
Program memory
compute $x * x$
compute $x^2 * x$

Automaton


temporary memory
Automaton
CPU
input
output
Program memory

Automaton

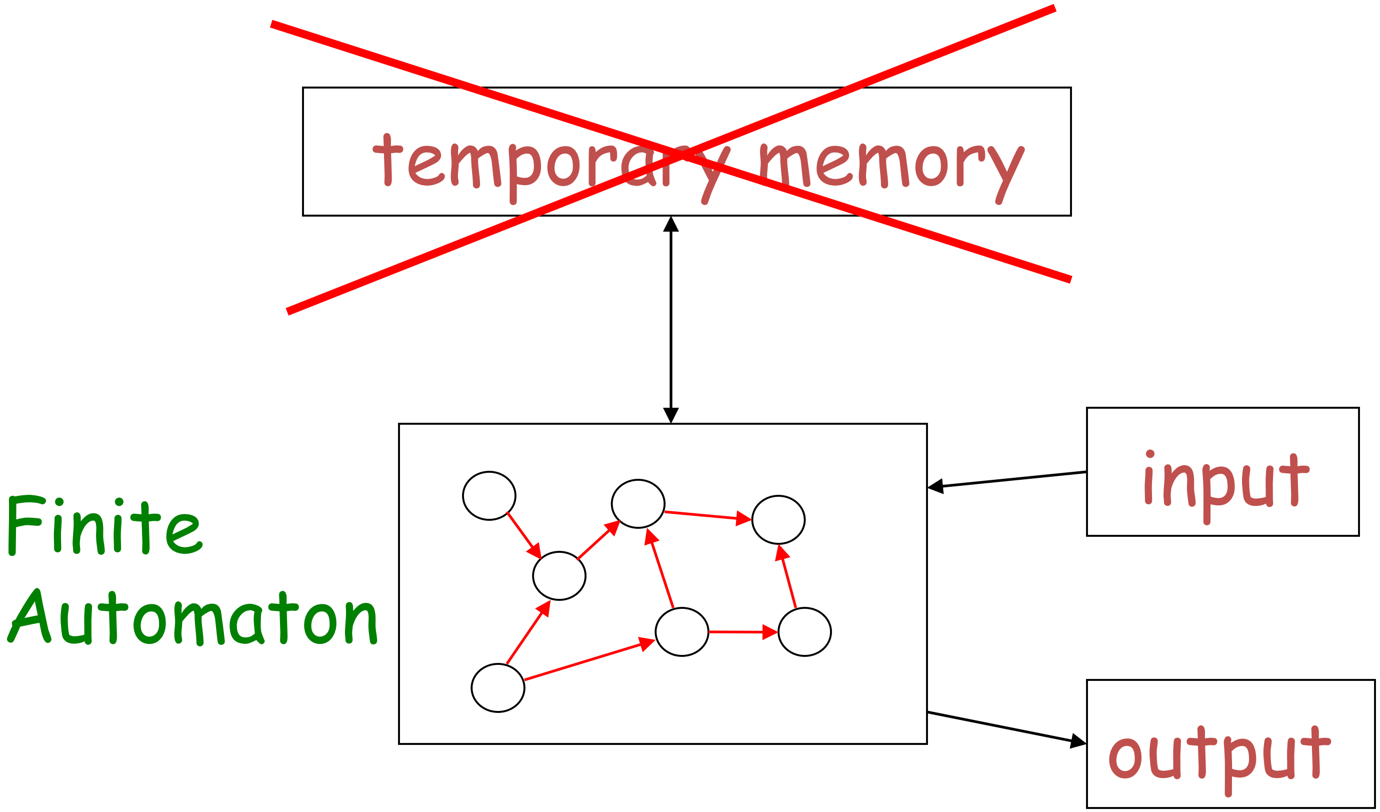

temporary memory
Automaton
input
output
transition
state

Different Kinds of Automata

Automata are distinguished by the temporary memory

- **Finite Automata:** no temporary memory
- **Pushdown Automata:** stack
- **Turing Machines:** random access memory

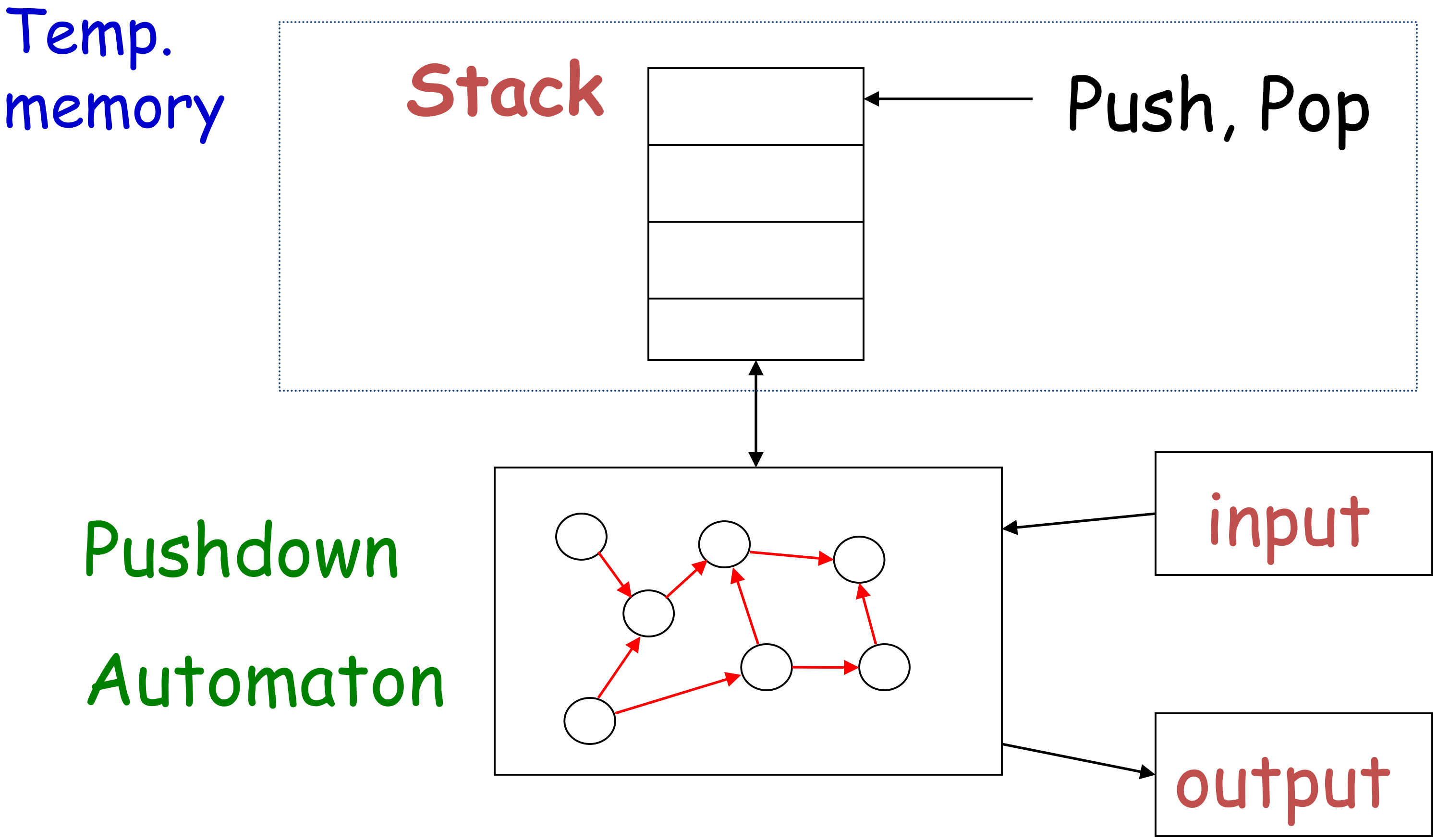
Finite Automaton



Example: Elevators, Vending Machines

(small computing power)

Pushdown Automaton



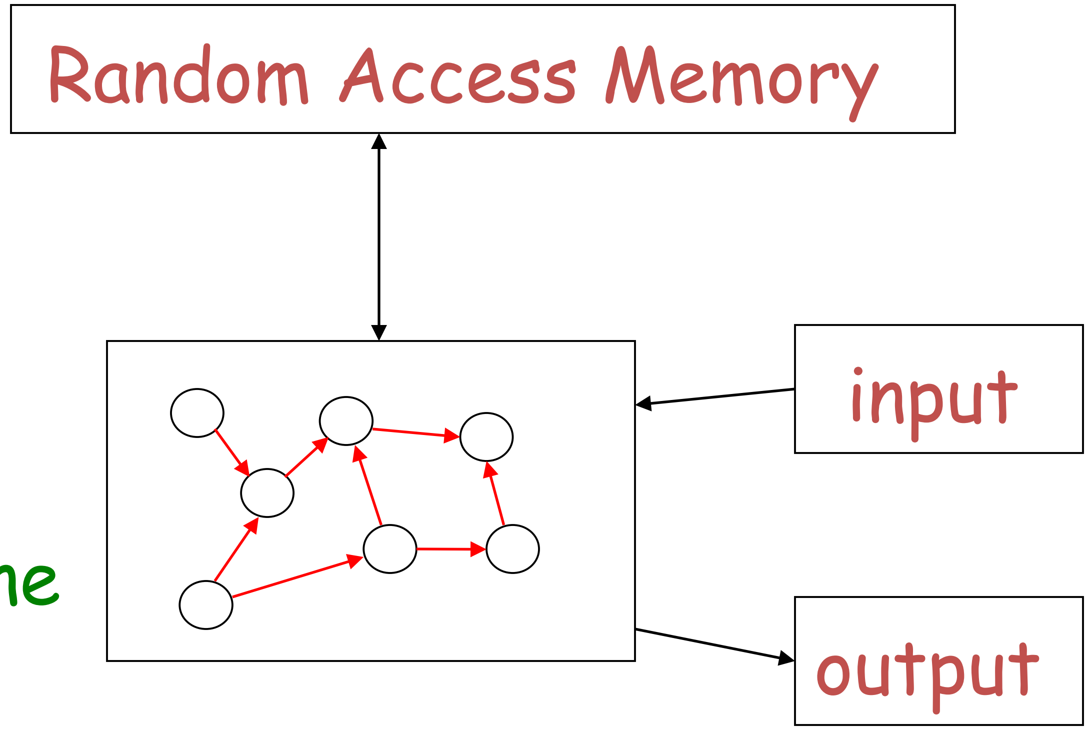
Example: Compilers for Programming Languages

(medium computing power)

Turing Machine

Temp.
memory

Turing
Machine



Examples: Any Algorithm

(highest computing power)

Power of Automata

Simple problems	More complex problems	Hardest problems
Finite Automata $<$	Pushdown Automata $<$	Turing Machine

Less power $\longrightarrow$ More power

Solve more computational problems

Introduction

In this lecture, we turn to ***Context Free Grammars*** and ***Context Free Languages***.

The class of Context Free Languages is an intermediate class between the class of regular languages and the class of **Decidable Languages** (To be defined).

Introduction

A ***Context Free Grammar (CFG)*** is a “machine” that creates a language.

A language created by a CF grammar is called ***A Context Free Language (CFL)***.

(We will show that) The class of Context Free Languages ***Properly Contains*** the class of Regular Languages.

Context Free Grammar - Example

Consider grammar G_1 :

$$I \to aIb$$

$$I \to \varepsilon$$

A CFG consists of **substitution rules** also called ***productions*****.**

The capital letters are the ***variables***.

One variable is designated as the **start variable**.

The other symbols are the ***terminals.***

Context Free Grammar - Example

Consider grammar G_1 :

$$I \to aIb$$

$$I \to \varepsilon$$

The grammar G_1 ***generates*** the language $B = \{a^n b^n \mid n \geq 0\}$ called ***the language of*** G_1 , denoted by $L(G_1)$.

The rules can be written as $I \to aIb \mid \varepsilon$

(using symbol "|" as an "or")

Context Free Grammar - Example

Consider grammar G_1 :

$$I \to aIb$$

$$I \to \varepsilon$$

This is a ***derivation*** of the word $aaabbb$ by G_1:

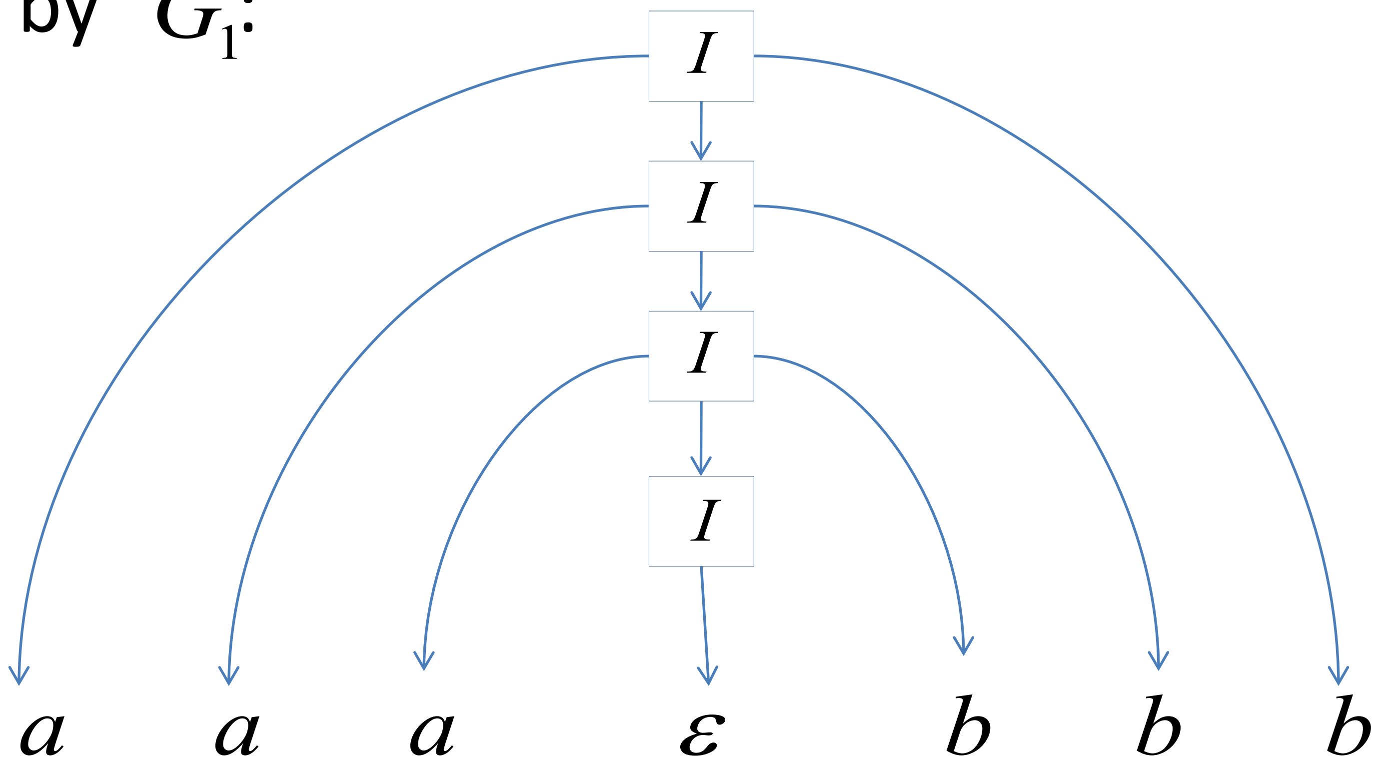
$$I \Rightarrow aIb \Rightarrow aaIbb \Rightarrow aaaIbbb \Rightarrow aaabbb$$

On each step, a single rule is activated. This mechanism is **nondeterministic.**

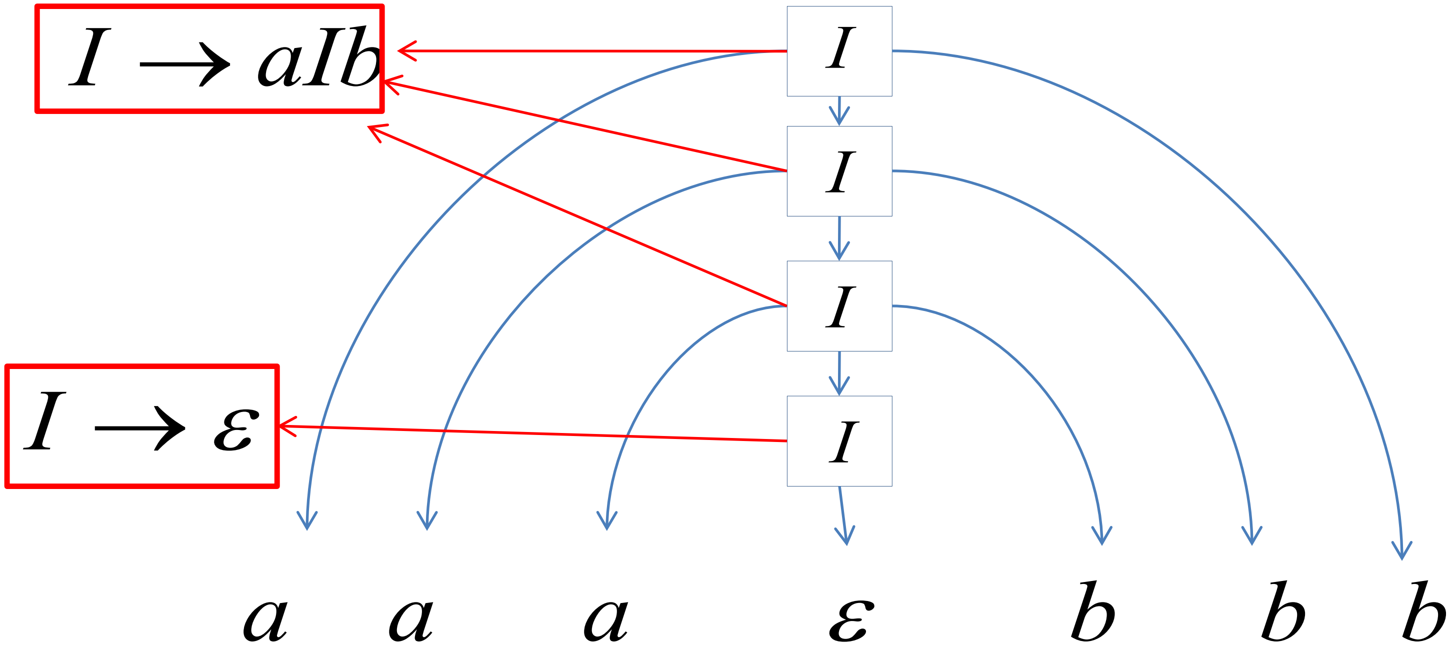
Context Free Grammar - Example

This is ***a parse tree*** of the word $aaabbb$

by G_1:



Context Free Grammar - Example



Each internal node of the tree is associated with a single production.

CF Grammar – A Formal Definition

A ***Context Free Grammar*** is a 4-tupple (V, Σ, R, S) where:

1. V is a finite set called the ***variables***.
2. Σ is a finite set, disjoint from V called the ***terminals.***
3. R is a set of ***rules***, where a rule is a variable and a string of variables and terminals, and
4. $S \in V$ is the ***start variable*** .

A Derivation – A Formal Definition

A word is a string of ***terminals**.*

A ***derivation*** of a word w from a context free grammar $G=(V,\Sigma,R,S)$ is a sequence of strings $S=s_0 \Rightarrow s_1 \Rightarrow ... \Rightarrow s_l = w$, over $V \cup \Sigma$, where:

1. $s_0 = S$ is the start variable of G.
2. For each $1 \le i \le l$, s_i is obtained by activating a single production (rule) of G on one of the variables of s_{i-1}.

CF Grammar – A Formal Definition

A word w is in **the language of grammar G**, denoted by $w \in L(G)$, if there exists a derivation whose rightmost string is w .

Thus,

$$L(G) = \{w \mid w \text{ can be derived from } G\}$$

Example2: Arithmetical EXPS

Grammar G_2 :

$V = \{<EXPR>, <TERM>, <FACTOR>\}$

$\Sigma = \{a, b, +, \times, (,)\}$

Rules:

1. $<EXPR> \to <EXPR> + <TERM> | <TERM>$
2. $<TERM> \to <TERM> \times <FACTOR> | <FACTOR>$
3. $<FACTOR> \to (<EXPR>) | a | b$

$S = <EXPR>$

Example2: Arithmetical EXPS

Derivation of $(a+b)\times a$ by Grammar G_2 :

$< EXPR >$

input

Example2: Arithmetical EXPS

Derivation of $(a+b)\times a$ by Grammar G_2 :

$< EXPR >$

input

rule $< EXPR > \rightarrow < TERM >$

Example2: Arithmetical EXPS

Derivation of $(a+b)\times a$ by Grammar G_2 :

$$<EXPR> \rightarrow <TERM>$$

input output

rule $<EXPR> \rightarrow <TERM>$

Example2: Arithmetical EXPS

Derivation of $(a+b)\times a$ by Grammar G_2 :

$$< EXPR > \to < TERM >$$

input

Example2: Arithmetical EXPS

Derivation of $(a+b)\times a$ by Grammar G_2 :

$$<EXPR> \rightarrow <TERM>$$

input

rule $<TERM> \rightarrow <TERM> \times <FACTOR>$

Example2: Arithmetical EXPS

Derivation of $(a+b)\times a$ by Grammar G_2 :

$$< EXPR > \rightarrow < TERM > \rightarrow < TERM > \times < FACTOR >$$

input output

rule $< TERM > \rightarrow < TERM > \times < FACTOR >$

Example2: Arithmetical EXPS

Derivation of $(a+b)\times a$ by Grammar G_2 :

$$<EXPR> \rightarrow <TERM> \rightarrow <TERM> \times <FACTOR>$$

input

Example2: Arithmetical EXPS

Derivation of $(a+b)\times a$ by Grammar G_2 :

$$< EXPR > \rightarrow < TERM > \rightarrow < TERM > \times < FACTOR >$$

input

rule $< FACTOR > \rightarrow a$

Example2: Arithmetical EXPS

Derivation of $(a+b)\times a$ by Grammar G_2 :

$$<EXPR> \to <TERM> \to <TERM> \times <FACTOR>$$

$$\to <TERM> \times a$$

input

output

rule $<FACTOR> \to a$

Example2: Arithmetical EXPS

Derivation of $(a+b)\times a$ by Grammar G_2 :

$$< EXPR > \rightarrow < TERM > \rightarrow < TERM > \times < FACTOR >$$

$$\rightarrow < TERM > \times a$$

input

Example2: Arithmetical EXPS

Derivation of $(a+b)\times a$ by Grammar G_2 :

$$< EXPR > \to < TERM > \to < TERM > \times < FACTOR >$$

$$\to < TERM > \times a$$

input

rule $< TERM > \to < FACTOR >$

Example2: Arithmetical EXPS

Derivation of $(a+b)\times a$ by Grammar G_2 :

$$< EXPR > \to < TERM > \to < TERM > \times < FACTOR >$$

$$\to < TERM > \times a \to < FACTOR > \times a$$

input output

rule $< TERM > \to < FACTOR >$

Example2: Arithmetical EXPS

Derivation of $(a+b)\times a$ by Grammar G_2 :

$$< EXPR > \rightarrow < TERM > \rightarrow < TERM > \times < FACTOR >$$

$$\rightarrow < TERM > \times a \rightarrow < FACTOR > \times a$$

input

Example2: Arithmetical EXPS

Derivation of $(a+b)\times a$ by Grammar G_2 :

$$< EXPR > \to < TERM > \to < TERM > \times < FACTOR >$$

$$\to < TERM > \times a \to < FACTOR > \times a$$

input

rule $< FACTOR > \to (< EXPR >)$

Example2: Arithmetical EXPS

Derivation of $(a+b)\times a$ by Grammar G_2 :

$$< EXPR > \to < TERM > \to < TERM > \times < FACTOR >$$

$$\to < TERM > \times a \to < FACTOR > \times a$$

$$\to (< EXPR >) \times a$$

input

output

rule $< FACTOR > \to (< EXPR >)$

Example2: Arithmetical EXPS

Derivation of $(a+b)\times a$ by Grammar G_2 :

$$< EXPR > \rightarrow < TERM > \rightarrow < TERM > \times < FACTOR >$$

$$\rightarrow < TERM > \times a \rightarrow < FACTOR > \times a$$

$$\rightarrow (< EXPR >) \times a$$

input

Example2: Arithmetical EXPS

Derivation of $\left(a+b\right)\times a$ by Grammar G_2 :

$$< EXPR > \rightarrow < TERM > \rightarrow < TERM > \times < FACTOR >$$

$$\rightarrow < TERM > \times a \rightarrow < FACTOR > \times a$$

$$\rightarrow \left(< EXPR >\right) \times a$$

input

rule $< EXPR > \rightarrow < EXPR > + < TERM >$

Example2: Arithmetical EXPS

Derivation of $(a+b)\times a$ by Grammar G_2 :

$$< EXPR > \to < TERM > \to < TERM > \times < FACTOR >$$

$$\to < TERM > \times a \to < FACTOR > \times a$$

$$\to (< EXPR >) \times a \to (< EXPR > + < TERM >) \times a$$

input output

rule $< EXPR > \to < EXPR > + < TERM >$

Example2: Arithmetical EXPS

Derivation of $(a+b)\times a$ by Grammar G_2 :

$$< EXPR > \to < TERM > \to < TERM > \times < FACTOR >$$

$$\to < TERM > \times a \to < FACTOR > \times a$$

$$\to (< EXPR >) \times a \to (< EXPR > + < TERM >) \times a$$

input

Example2: Arithmetical EXPS

Derivation of $(a+b)\times a$ by Grammar G_2 :

$$<EXPR> \to <TERM> \to <TERM> \times <FACTOR>$$

$$\to <TERM> \times a \to <FACTOR> \times a$$

$$\to (<EXPR>) \times a \to (<EXPR> + <TERM>) \times a$$

input

rule $<EXPR> \to <TERM>$

Example2: Arithmetical EXPS

Derivation of $(a+b)\times a$ by Grammar G_2 :

$$< EXPR > \to < TERM > \to < TERM > \times < FACTOR >$$

$$\to < TERM > \times a \to < FACTOR > \times a$$

$$\to (< EXPR >) \times a \to (< EXPR > + < TERM >) \times a$$

$$\to (< TERM > + < TERM >) \times a$$

output rule $< EXPR > \to < TERM >$

Example2: Arithmetical EXPS

Derivation of $(a+b)\times a$ by Grammar G_2 :

$$< EXPR > \to < TERM > \to < TERM > \times < FACTOR >$$

$$\to < TERM > \times a \to < FACTOR > \times a$$

$$\to (< EXPR >) \times a \to (< EXPR > + < TERM >) \times a$$

$$\to (< TERM > + < TERM >) \times a$$

input

Example2: Arithmetical EXPS

Derivation of $(a+b)\times a$ by Grammar G_2 :

$$< EXPR > \to < TERM > \to < TERM > \times < FACTOR >$$

$$\to < TERM > \times a \to < FACTOR > \times a$$

$$\to (< EXPR >) \times a \to (< EXPR > + < TERM >) \times a$$

$$\to (< TERM > + < TERM >) \times a$$

input

rule $< TERM > \to < FACTOR >$

Example2: Arithmetical EXPS

Derivation of $(a+b)\times a$ by Grammar G_2 :

$$<EXPR> \to <TERM> \to <TERM> \times <FACTOR>$$

$$\to <TERM> \times a \to <FACTOR> \times a$$

$$\to (<EXPR>) \times a \to (<EXPR> + <TERM>) \times a$$

$$\to (<TERM> + <TERM>) \times a$$

$$\to (<FACTOR> + <FACTOR>) \times a$$

output rule $<TERM> \to <FACTOR>$

Example2: Arithmetical EXPS

Derivation of $(a+b)\times a$ by Grammar G_2 :

$$< EXPR > \to < TERM > \to < TERM > \times < FACTOR >$$

$$\to < TERM > \times a \to < FACTOR > \times a$$

$$\to (< EXPR >) \times a \to (< EXPR > + < TERM >) \times a$$

$$\to (< TERM > + < TERM >) \times a$$

$$\to (< FACTOR > + < FACTOR >) \times a$$

input

Example2: Arithmetical EXPS

Derivation of $(a+b)\times a$ by Grammar G_2 :

$$<EXPR> \rightarrow <TERM> \rightarrow <TERM> \times <FACTOR>$$

$$\rightarrow <TERM> \times a \rightarrow <FACTOR> \times a$$

$$\rightarrow (<EXPR>) \times a \rightarrow (<EXPR> + <TERM>) \times a$$

$$\rightarrow (<TERM> + <TERM>) \times a$$

$$\rightarrow (<FACTOR> + <FACTOR>) \times a$$

input

rule $<FACTOR> \rightarrow a$

Example2: Arithmetical EXPS

Derivation of $(a+b)\times a$ by Grammar G_2 :

$$< EXPR > \to < TERM > \to < TERM > \times < FACTOR >$$

$$\to < TERM > \times a \to < FACTOR > \times a$$

$$\to (< EXPR >) \times a \to (< EXPR > + < TERM >) \times a$$

$$\to (< TERM > + < TERM >) \times a$$

$$\to (< FACTOR > + < FACTOR >) \times a$$

$$\to (a + < FACTOR >) \times a$$

output rule $< FACTOR > \to a$

Example2: Arithmetical EXPS

Derivation of $(a+b)\times a$ by Grammar G_2 :

$$< EXPR > \to < TERM > \to < TERM > \times < FACTOR >$$

$$\to < TERM > \times a \to < FACTOR > \times a$$

$$\to (< EXPR >)\times a \to (< EXPR > + < TERM >)\times a$$

$$\to (< TERM > + < TERM >)\times a$$

$$\to (< FACTOR > + < FACTOR >)\times a$$

$$\to (a + < FACTOR >)\times a$$

input

Example2: Arithmetical EXPS

Derivation of $(a+b)\times a$ by Grammar G_2 :

$$< EXPR > \rightarrow < TERM > \rightarrow < TERM > \times < FACTOR >$$

$$\rightarrow < TERM > \times a \rightarrow < FACTOR > \times a$$

$$\rightarrow (< EXPR >)\times a \rightarrow (< EXPR > + < TERM >)\times a$$

$$\rightarrow (< TERM > + < TERM >)\times a$$

$$\rightarrow (< FACTOR > + < FACTOR >)\times a$$

$$\rightarrow (a + < FACTOR >)\times a$$

input rule $< FACTOR > \rightarrow b$

Example2: Arithmetical EXPS

Derivation of $(a+b)\times a$ by Grammar G_2 :

$$<EXPR> \to <TERM> \to <TERM> \times <FACTOR>$$

$$\to <TERM> \times a \to <FACTOR> \times a$$

$$\to (<EXPR>) \times a \to (<EXPR> + <TERM>) \times a$$

$$\to (<TERM> + <TERM>) \times a$$

$$\to (<FACTOR> + <FACTOR>) \times a$$

$$\to (a + <FACTOR>) \times a \to (a+b) \times a$$

Example2: Arithmetical EXPS

Derivation of $a+b\times a$ by Grammar G_2 :

$$< EXPR > \to < EXPR > + < TERM > \to$$

$$\to < TERM > + < TERM > \to < FACTOR > + < TERM >$$

$$\to a + < TERM > \to a + < TERM > \times < FACTOR >$$

$$\to a + < FACTOR > \times < FACTOR >$$

$$\to a + b \times < FACTOR >$$

$$\to a + b \times a$$

Note: There is more than one derivation.

Derivation Order

Consider the following example grammar with 5 productions:

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

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

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

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

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

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

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

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 :

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

Ambiguity

A string w is derived **ambiguously** in CFG *G* if it has two or more different leftmost derivations.

Grammar G is **ambiguous** if it generates some strings ambiguously.

Example4: Similar to Arith. EXPS

Grammar G_4 :

V = {<EXPR>} $\quad\Sigma$ *= {a,b,+,×,(,)}*

Rules:

<EXPR>$\rightarrow$<EXPR> + <EXPR> | <EXPR> × <EXPR>

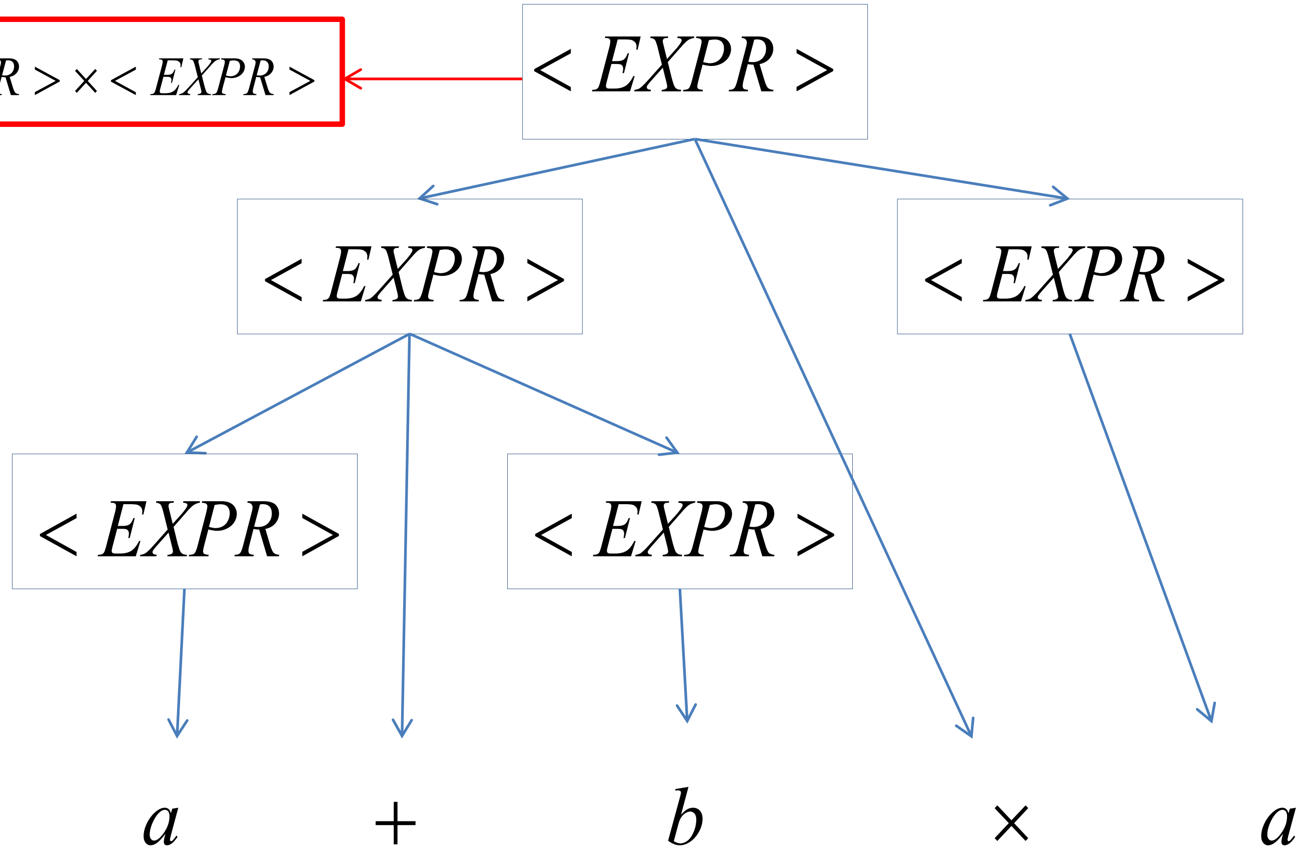
<EXPR> $\rightarrow$ (*<EXPR>*) | *a* | *b*

S = <EXPR>

Grammar G_4 is ambiguous. It has different parse trees for $a + b \times a$ (shown next).

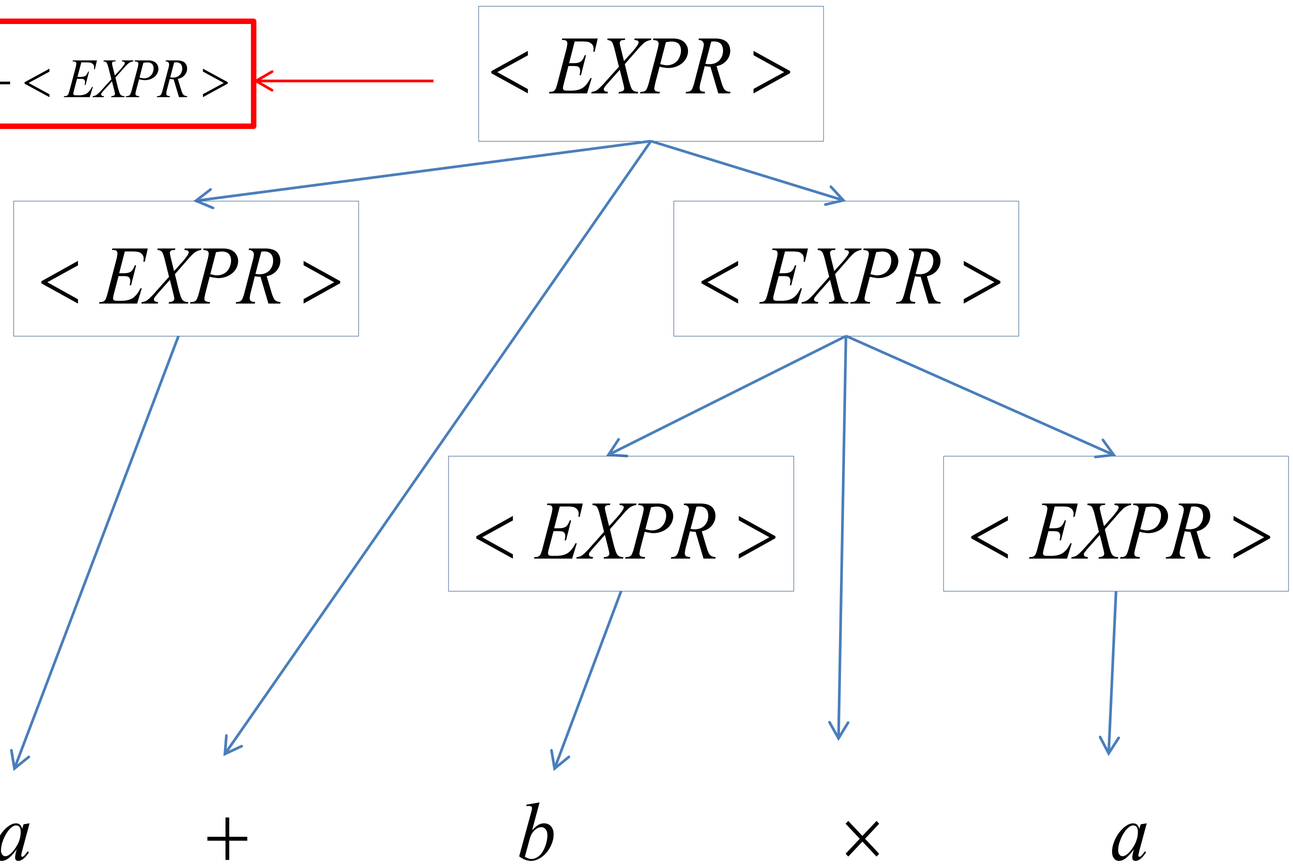
Example4: 1st Parse Tree for $a + b \times a$

$$< EXPR > \rightarrow < EXPR > \times < EXPR >$$



Example4: 2nd Parse Tree for $a+b\times a$

$$\boxed{<EXPR> \to <EXPR> + <EXPR>}$$



Two different derivation trees
may cause problems in applications which
use the derivation trees:

- Evaluating expressions

- In general, in compilers
  for programming languages

Ambiguity

Note: Some ambiguous grammars may have an unambiguous equivalent grammar.

But: There exist ***inherently ambiguous grammars*** , i.e. an ambiguous grammar that does not have an equivalent unambiguous one.

Converting a DFA into an equivalent CFG

- Make a variable R_i for each state q_i of DFA
- Add the rule $R_i \to aR_j$ if $\delta(q_i, a) = q_j$ is a transition in DFA
- Add the rule $R_i \to \varepsilon$ if q_i is an accept state of DFA
- Make R_0 the start variable of CFG if q_0 is the start state of DFA

Discussion

Q: From a computational point of view, how strong are context free languages?

A: Since the language $B = \{a^n b^n \mid n \geq 0\}$ is not regular and it is CF, we conclude that $CFL \not\subset RL$.

Q: Can one prove $CFL \supset RL$?

A: Yes.

Discussion

Q: A language is regular if it is recognized by a DFA (or NFA). Does there exist a type of machine that characterizes CFL?

A: Yes, those are the **Push-Down Automata** (Next Lecture). .

Q: Can one prove a language not to be CFL ?

A: Yes, by the **Pumping Lemma for CFL-s** . For example: $L = \left\{a^n b^n c^n \mid n \geq 0\right\}$ is not CFL.

Chomsky Normal Form (CNF)

A CFG is said to be in *Chomsky Normal Form (CNF)* if every production is of one of these two forms:

1. A -> BC (right side is two variables).
2. A -> a (right side is a single terminal).

In addition we permit the rule S -> ε, where S is the start variable.

Theorem: If L is a CFL, then L has a CFG in CNF.