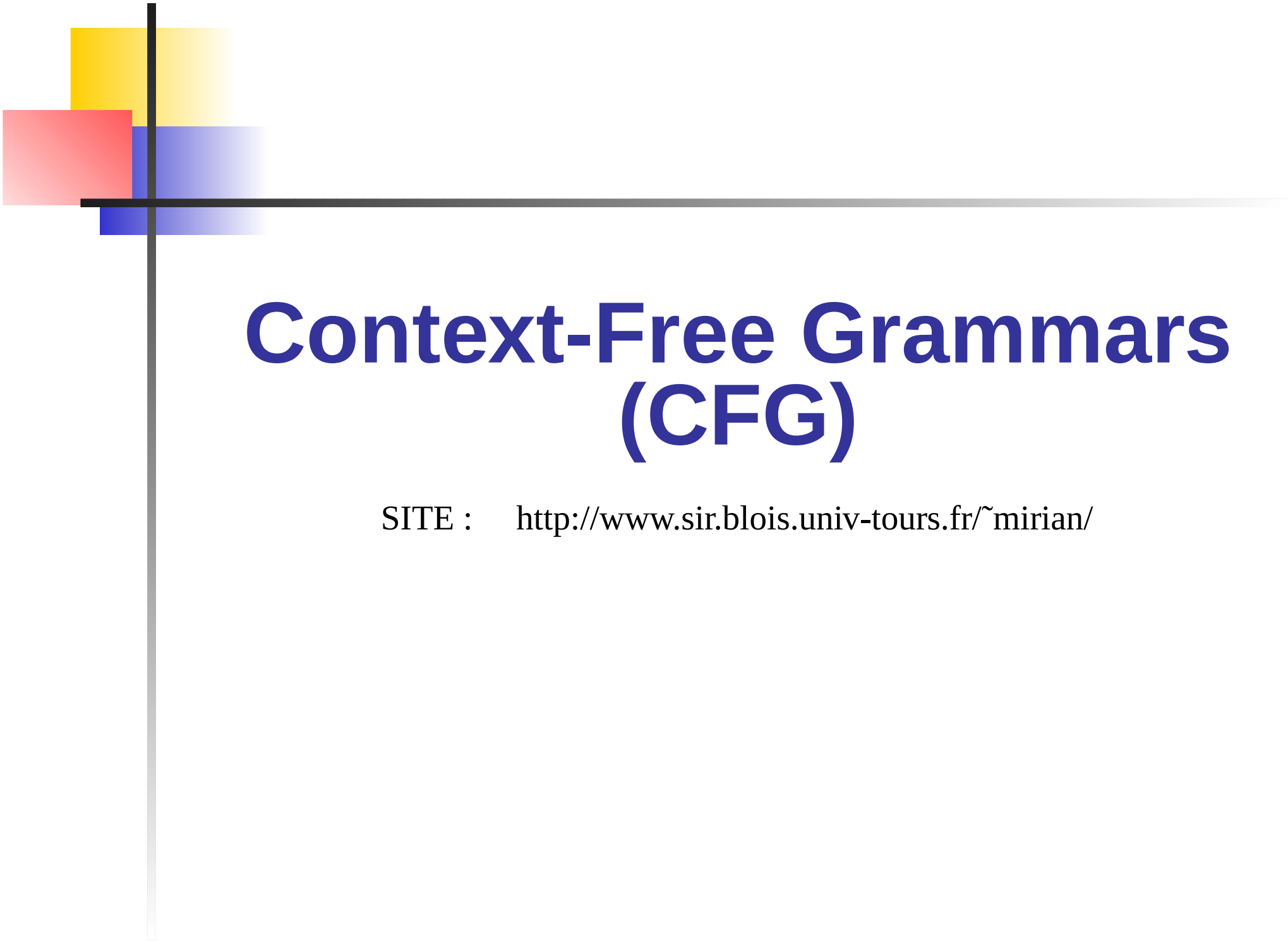


Context-Free Grammars (CFG)

SITE : http://www.sir.blois.univ-tours.fr/~mirian/

An informal example

- Language of palindromes: L_{pal}
  - A palindrome is a string that reads the same forward and backward
  - Ex: $otto$, $madamimadam$, 0110, 11011, ϵ
- L_{pal} is not a regular language (can be proved by using the pumping lemma)
- We consider $\Sigma = \{0, 1\}$. There is a natural, recursive definition of when a string of 0 and 1 is in L_{pal}.
  Basis: ϵ, 0 and 1 are palindromes
  Induction: If w is a palindrome, so are $0w0$ and $1w1$. No string is palindrome of 0 and 1, unless it follows from this basis and inductive rule.
- A CFG is a formal notation for expressing such recursive definitions of languages

What is a grammar?

- A grammar consists of one or more variables that represent classes of strings (*i.e.*, languages)
- There are rules that say how the strings in each class are constructed. The construction can use :
  1. symbols of the alphabet
  2. strings that are already known to be in one of the classes
  3. or both

A grammar for palindromes

- In the example of palindromes, we need one variable P which represents the set of palindromes; *i.e.*, the class of strings forming the language L_{pal}
- Rules:

$$\begin{aligned} P &\rightarrow \epsilon \\ P &\rightarrow 0 \\ P &\rightarrow 1 \\ P &\rightarrow 0P0 \\ P &\rightarrow 1P1 \end{aligned}$$

- The first three rules form the **basis**.
  - They tell us that the class of palindromes includes the strings ϵ, 0 and 1
  - None of the right sides of theses rules contains a variable, which is why they form a basis for the definition
- The last two rules form the **inductive** part of the definition.
  For instance, rule 4 says that if we take any string w from the class P, then $0w0$ is also in class P.

Definition of Context-Free Grammar

A GFG (or just a grammar) G is a tuple $G = (V, T, P, S)$ where

1. V is the (finite) set of variables (or nonterminals or syntactic categories). Each variable represents a language, *i.e.*, a set of strings
2. T is a finite set of terminals, *i.e.*, the symbols that form the strings of the language being defined
3. P is a set of production rules that represent the recursive definition of the language.
4. S is the start symbol that represents the language being defined. Other variables represent auxiliary classes of strings that are used to help define the language of the start symbol.

Production rules

Each production rule consists of:

1. A variable that is being (partially) defined by the production. This variable is often called the *head* of the production.
2. The production symbol $\longrightarrow$.
3. A string of zero or more terminals and variables. This string, called the *body* of the production, represents one way to form strings in the language of the variable of the head.
   In doing so, we leave terminals unchanged and substitute for each variable of the body any string that is known to be in the language of that variable

Compact Notation for Productions

- We often refers to the production whose head is A as "productions for A" or "A-productions"
- Moreover, the productions

$$A \to \alpha_1, A \to \alpha_2 \ldots A \to \alpha_n$$

can be replaced by the notation

$$A \to \alpha_1 \mid \alpha_2 \mid \ldots \mid \alpha_n$$

Examples: CFG for palindromes

$$G_{pal} = (\{P\}, \{0, 1\}, A, P)$$

where A represents the production rules:

$$\begin{aligned} P &\rightarrow \epsilon \\ P &\rightarrow 0 \\ P &\rightarrow 1 \\ P &\rightarrow 0P0 \\ P &\rightarrow 1P1 \end{aligned}$$

We can also write: $P \rightarrow \epsilon \mid 0 \mid 1 \mid 0P0 \mid 1P1$

Examples: CFG for expressions in a typical programming language

Operators: $+$ (addition) and $*$ (multiplication)

Identifiers: must begin with a or b, which may be followed by any string in $\{a, b, 0, 1\}^*$

We need two variables:

E: represents expressions. It is the start symbol.

I: represents the identifiers. Its language is regular and is the language of the regular expression: $(\mathbf{a} + \mathbf{b})(\mathbf{a} + \mathbf{b} + \mathbf{0} + \mathbf{1})^*$

Exemples (cont.): The grammar

Grammar $G_1 = (\{E, I\}, T, P, E)$ where: $T = \{+, *, (,), a, b, 0, 1\}$ and P is the set of productions:

1	E	$\rightarrow$	I
2	E	$\rightarrow$	$E + E$
3	E	$\rightarrow$	$E * E$
4	E	$\rightarrow$	(E)
5	I	$\rightarrow$	a
6	I	$\rightarrow$	b
7	I	$\rightarrow$	Ia
8	I	$\rightarrow$	Ib
9	I	$\rightarrow$	$I0$
10	I	$\rightarrow$	$I1$

Derivations Using a Grammar

- We apply the productions of a CFG to infer that certain strings are in the language of a certain variable
- Two inference approaches:
  1. **Recursive inference**, using productions from body to head
  2. **Derivations**, using productions from head to body

Recursive inference - an exemple

We consider some inferences we can make using G_1

Recall G_1:

$$E \rightarrow I \mid E + E \mid E * E \mid (E)$$
$$I \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1$$

	String	Lang	Prod	String(s) used
(i)	a	I	5	-
(ii)	b	I	6	-
(iii)	b0	I	9	(ii)
(iv)	b00	I	9	(iii)
(v)	a	E	1	(i)
(vi)	b00	E	1	(iv)
(vii)	a + b00	E	2	(v), (vi)
(viii)	(a + b00)	E	4	(vii)
(ix)	a (a + b00)	E	3	(v), (viii)

Derivations

- Applying productions from head to body requires the definition of a new relational symbol: $\Rightarrow$
- Let:
  - $G = (V, T, P, S)$ be a CFG
  - $A \in V$
  - $\alpha, \beta \subset (V \cup T)^*$ and
  - $A \rightarrow \gamma \in P$

Then we write

$$\alpha A \beta \quad \Rightarrow_G \quad \alpha \gamma \beta$$

or, if G is understood

$$\alpha A \beta \quad \Rightarrow \quad \alpha \gamma \beta$$

and say that $\alpha A \beta$ derives $\alpha \gamma \beta$.

Zero or more derivation steps

We define $\stackrel{*}{\Rightarrow}$ to be the reflexive and transitive closure of $\Rightarrow$ (*i.e.*, to denote zero or more derivation steps):

Basis: Let $\alpha \in (V \cup T)^*$. Then $\alpha \stackrel{*}{\Rightarrow} \alpha$.

Induction: If $\alpha \stackrel{*}{\Rightarrow} \beta$ and $\beta \Rightarrow \gamma$, then $\alpha \stackrel{*}{\Rightarrow} \gamma$.

Examples of derivation

Derivation of $a * (a + b000)$ by G_1

$E \Rightarrow E * E \Rightarrow I * E \Rightarrow a * E \Rightarrow a * (E) \Rightarrow$
$a * (E + E) \Rightarrow a * (I + E) \Rightarrow a * (a + E) \Rightarrow a * (a + I) \Rightarrow$
$a * (a + I0) \Rightarrow a * (a + I00) \Rightarrow a * (a + b00)$

Note 1: At each step we might have several rules to choose from, e.g.
$I * E \Rightarrow a * E \Rightarrow a * (E)$, versus
$I * E \Rightarrow I * (E) \Rightarrow a * (E)$.
Note 2: Not all choices lead to successful derivations of a particular string, for instance
$E \Rightarrow E + E$ (at the first step)
won't lead to a derivation of $a * (a + b000)$.
Important: Recursive inference and derivation are equivalent. A string of terminals w is infered to be in the language of some variable A iff $A \stackrel{*}{\Rightarrow} w$

Leftmost and Rightmost derivation

- In other to restrict the number of choices we have in deriving a string, it is often useful to require that at each step we replace the leftmost (or rightmost) variable by one of its production rules
- Leftmost derivation $\Rightarrow_{lm}$: Always replace the left-most variable by one of its rule-bodies
- Rightmost derivation $\Rightarrow_{rm}$: Always replace the rightmost variable by one of its rule-bodies.

EXAMPLES

$1-$ Leftmost derivation: previous example

$2-$ Rightmost derivation:

$$E \Rightarrow_{rm} E*E \Rightarrow_{rm} E*(E) \Rightarrow_{rm}$$
$$E*(E+E) \Rightarrow_{rm} E*(E+I) \Rightarrow_{rm} E*(E+I0) \Rightarrow_{rm}$$
$$E*(E+I00) \Rightarrow_{rm} E*(E+b00) \Rightarrow_{rm} E*(I+b00) \Rightarrow_{rm}$$
$$E*(a+b00) \Rightarrow_{rm} I*(a+b00) \Rightarrow_{rm} a*(a+b00)$$

We can conclude that $E \Rightarrow^{*}_{rm} a*(a+b00)$

The Language of the Grammar

If $G(V, T, P, S)$ is a CFG, then the language of G is

$$L(G) = \{w \ \ in \ \ T^* \mid S \Rightarrow_G^* \ \ w\}$$

i.e., the set of strings over T derivable from the start symbol.

If G is a CFG, we call $L(G)$ a context-free language.

Example: $L(G_{pal})$ is a context-free language.

Theorem

A string $w \in \{0,1\}^*$ **is in** $L(G_{pal})$ **iff** $w = w^R$.

Proof: ($\supseteq$-direction.) Suppose $w = w^R$, *i.e.*, that w is a palindrome. We show by induction on $|w|$ that $w \in L(G_{pal})$

Basis: Basis: $|w| = 0$, or $|w| = 1$. Then w is ϵ , 0, or 1. Since $P \to \epsilon$, $P \to 0$, and $P \to 1$ are productions, we conclude that $P \stackrel{*}{\Rightarrow} w$ in all base cases.

Induction: Suppose $|w| \geq 2$. Since $w = w^R$, we have $w = 0x0$, or $w = 1x1$, and $x = x^R$.

If $w = 0x0$ we know from the IH that $P \stackrel{*}{\Rightarrow} x$. Then

$$P \Rightarrow 0P0 \stackrel{*}{\Rightarrow} 0x0 = w$$

Thus $w \in L(Gpal)$. The case for w = 1x1 is similar.

Proof (cont.)

($\subseteq$-direction.) We assume $w \in L(G_{pal})$ and we prove that $w = w^R$.
Since $w \in L(G_{pal})$, we have $P \stackrel{*}{\Rightarrow} w$. We do an induction of the length of $\stackrel{*}{\Rightarrow}$.

Basis: The derivation $P \stackrel{*}{\Rightarrow} w$ is done in one step. Then w must be ϵ,0, or 1, all palindromes.

Induction: Let $n \geq 1$, and suppose the derivation takes $n+1$ steps. Then we must have
$w = 0x0 \stackrel{*}{\Leftarrow} 0P0 \Leftarrow P$
or
$w = 1x1 \stackrel{*}{\Leftarrow} 1P1 \Leftarrow P$
where the second derivation is done in n steps. By the IH x is a palindrome, and the inductive proof is complete.

Sentential Forms

Let $G = (V, T, P, S)$ be a CFG, and $\alpha \in (V \cup T)^*$. If

$$S \stackrel{*}{\Rightarrow} \alpha$$

we say α is a sentential form.

If $S \Rightarrow_{lm} \alpha$ we say that α is a **left-sentential form**, and if $S \Rightarrow_{rm} \alpha$ we say that is a right-sentential form.

Note: $L(G)$ is those sentential forms that are in T^*.

Example

Recall G_1:

$$\begin{array}{lcl} E & \rightarrow & I \mid E+E \mid E*E \mid (E) \\ I & \rightarrow & a \mid b \mid Ia \mid Ib \mid I0 \mid I1 \end{array}$$

$1-$ Then $E*(I+E)$ is a sentential form since

$$E \Rightarrow E*E \Rightarrow E*(E) \Rightarrow E*(E+E) \Rightarrow E(I+E)$$

This derivation is neither leftmost, nor right-most.

$2-$ $a*E$ left-sentential form, since

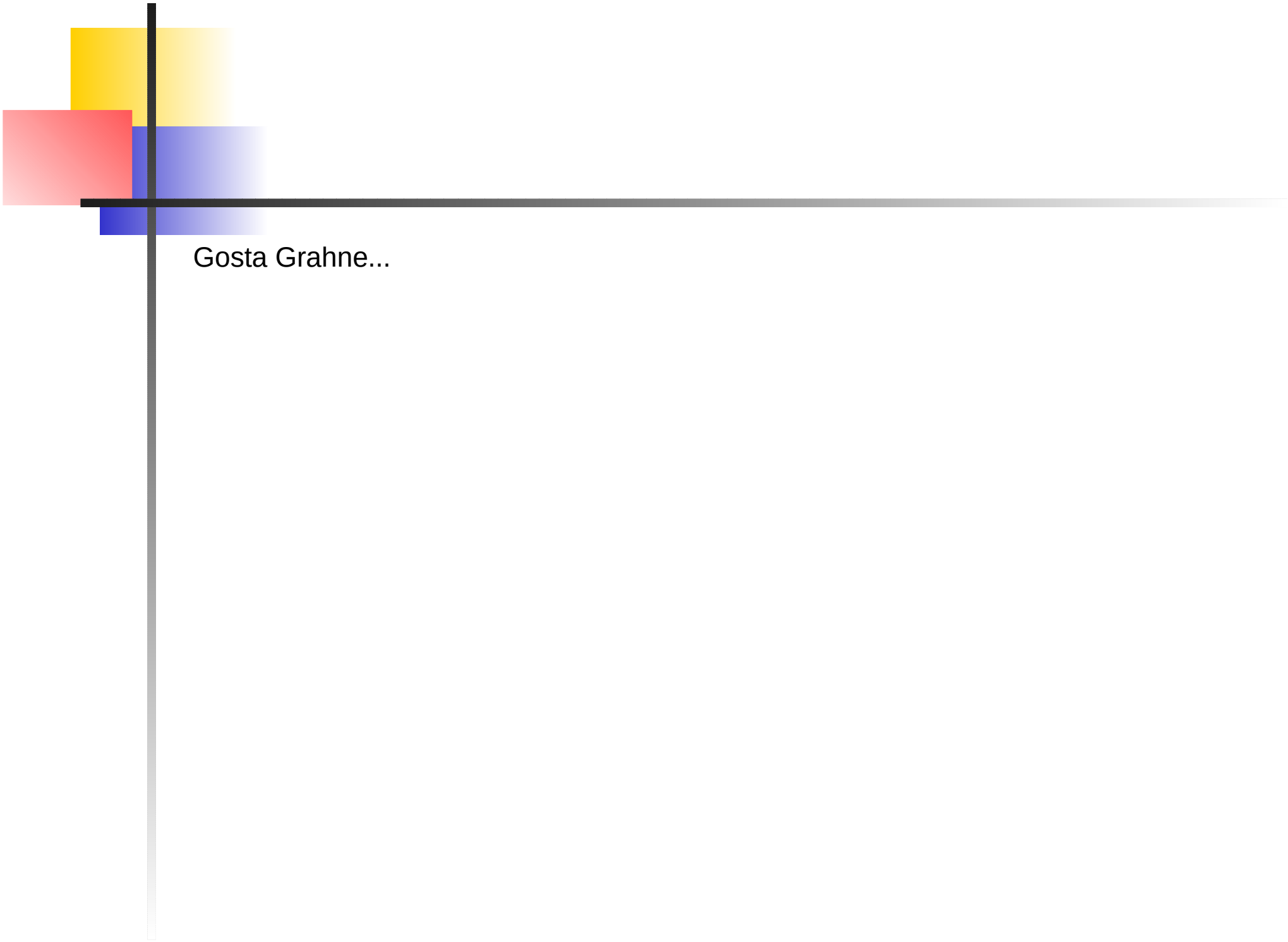
$$E \Rightarrow E*E \Rightarrow I*E \Rightarrow a*E$$

$3-$ $E*(E+E)$ is a right-sentential form since

$$E \Rightarrow E*E \Rightarrow E*(E) \Rightarrow E*(E+E)$$

Parse Trees

- If $w \in L(G)$, for some CFG, then w has a parse tree, which tells us the (syntactic) struc- ture of w.
- w could be a program, a SQL-query, an XML- document, etc.
- Parse trees are an alternative representation to derivations and recursive inferences.
- There can be several parse trees for the same string.
- Ideally there should be only one parse tree (the "true" structure) for each string, i.e. the language should be unambiguous.
- Unfortunately, we cannot always remove the ambiguity.



Gosta Grahne...