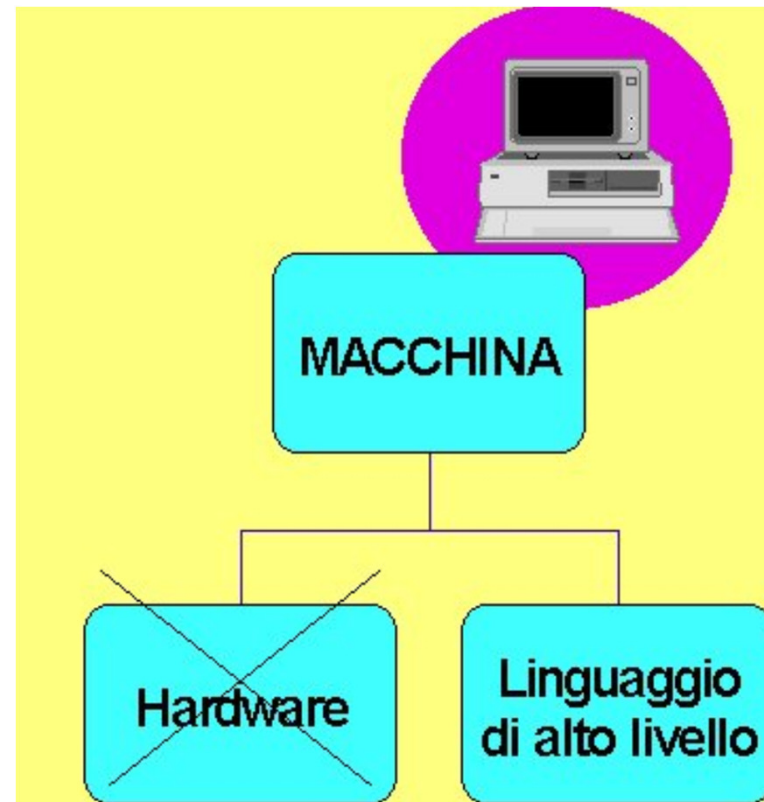


# LINGUAGGI DI ALTO LIVELLO

---

Si basano su una *macchina virtuale* le cui “mosse” non sono quelle della macchina hardware



# Un po' di storia sui linguaggi

1957

**John Backus** e colleghi della IBM rilasciano la **prima versione del compilatore** per il linguaggio di programmazione **FORTRAN** (Formula Translator) alla Westinghouse.

1959


Si forma il Comitato per i linguaggi di sistemi di dati e **nasce il COBOL** (Common Business Oriented Language).

1959

**John McCarthy** sviluppa il **linguaggio LISP** (List Processing) per le applicazioni di Intelligenza Artificiale.

1964

Nasce il linguaggio BASIC (Beginner's All-purpose Symbolic Instruction Code). E' sviluppato a Dartmouth da **John Kemeny** e **Thomas Kurtz**. Ne deriveranno molte varianti.



```
C
C Hello, world.
C
      Program Hello

      implicit none
      logical DONE

      DO while (.NOT. DONE)
        write(*,10)
      END DO
10 format('Hello, world.')
END
```

http://www.latec.edu/~scv/HelloWorld.shtml

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID. HELLOWORLD.
000300 DATE-WRITTEN.02/05/96 21:04.
000400* AUTHOR JOHN JONES
000500 ENVIRONMENT DIVISION.
000600 CONFIGURATION SECTION.
000700 SOURCE-COMPUTER. RM-COBOL.
000800 OBJECT-COMPUTER. RM-COBOL.
000900
001000 DATA DIVISION.
001100 FILE SECTION.
001200
100000 PROCEDURE DIVISION.
100100
100200 MAIN-LOGIC SECTION.
100300 BEGIN.
100400 DISPLAY " " LINE 1 POSITION 1 ERASE
100500 DISPLAY "HELLO, WORLD." LINE 15 POSITION 1
100600 STOP RUN.
100700 MAIN-LOGIC-EXIT.
100800 EXIT.
```

http://www.latec.edu/~scv/HelloWorld.shtml

```
10 print "Hello World!"
20 goto 10
```

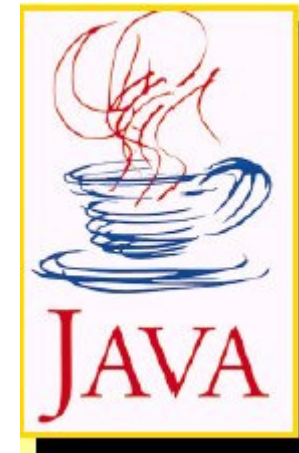
# Un po' di storia sui linguaggi

**1967**

**Ole-Johan Dahl e Kristen Nygaard** del Centro Computer Norvegese, completano una versione general-purpose del linguaggio **SIMULA**, il **primo linguaggio object-oriented**.

**1995**

Nasce il linguaggio di programmazione **Java**, piattaforma indipendente per sviluppo di applicazioni.

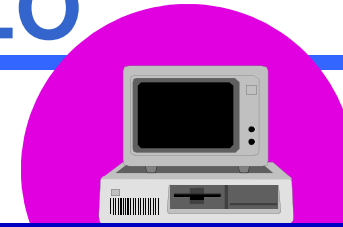


**1972**

**Dennis Ritchie** sviluppa il **linguaggio "C"** ai laboratori Bell. Così chiamato semplicemente perchè il suo predecessore era stato battezzato "B".

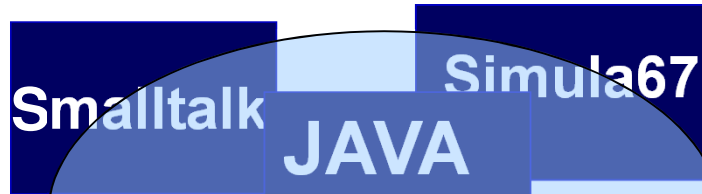
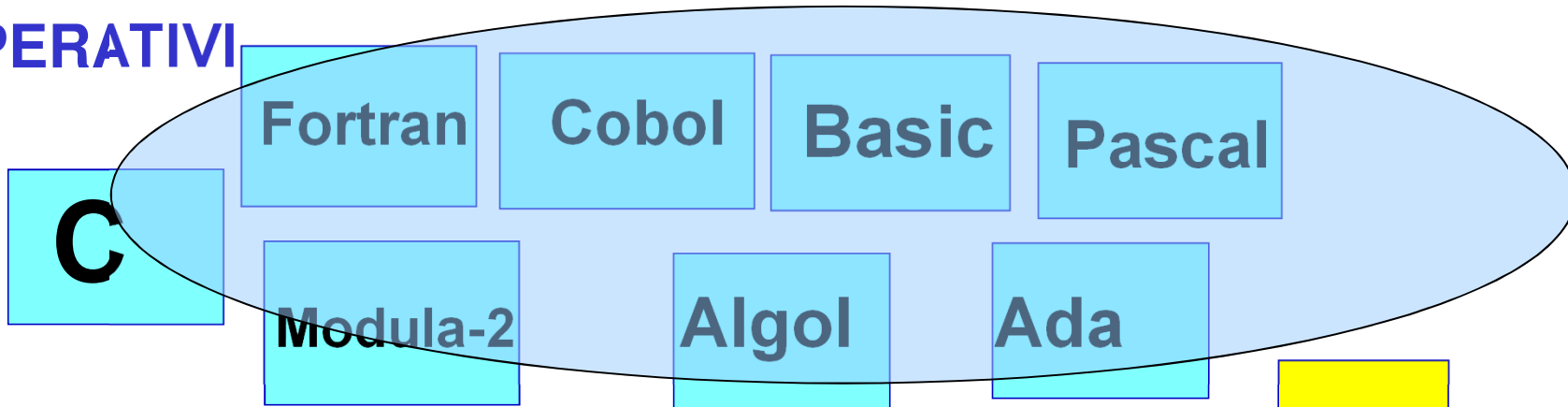
```
#include
main()
{
  for(;;)
  {
    printf ("Hello World!\n");
  }
}
```

# LINGUAGGI DI ALTO LIVELLO

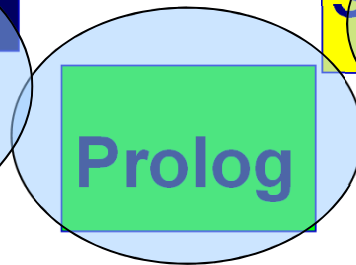


## Barriera di astrazione

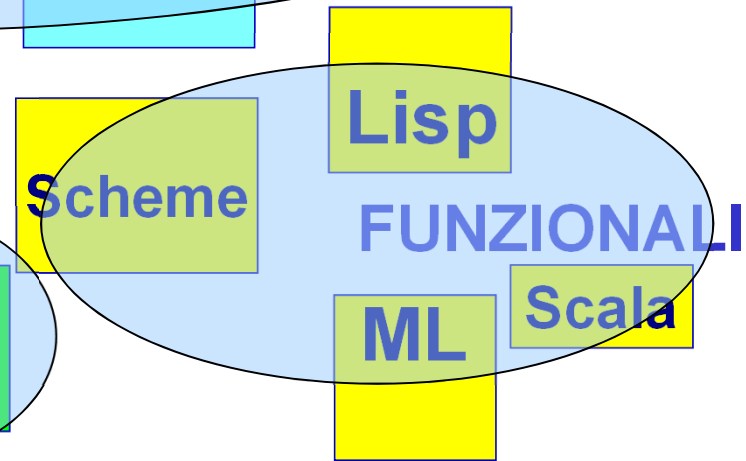
IMPERATIVI



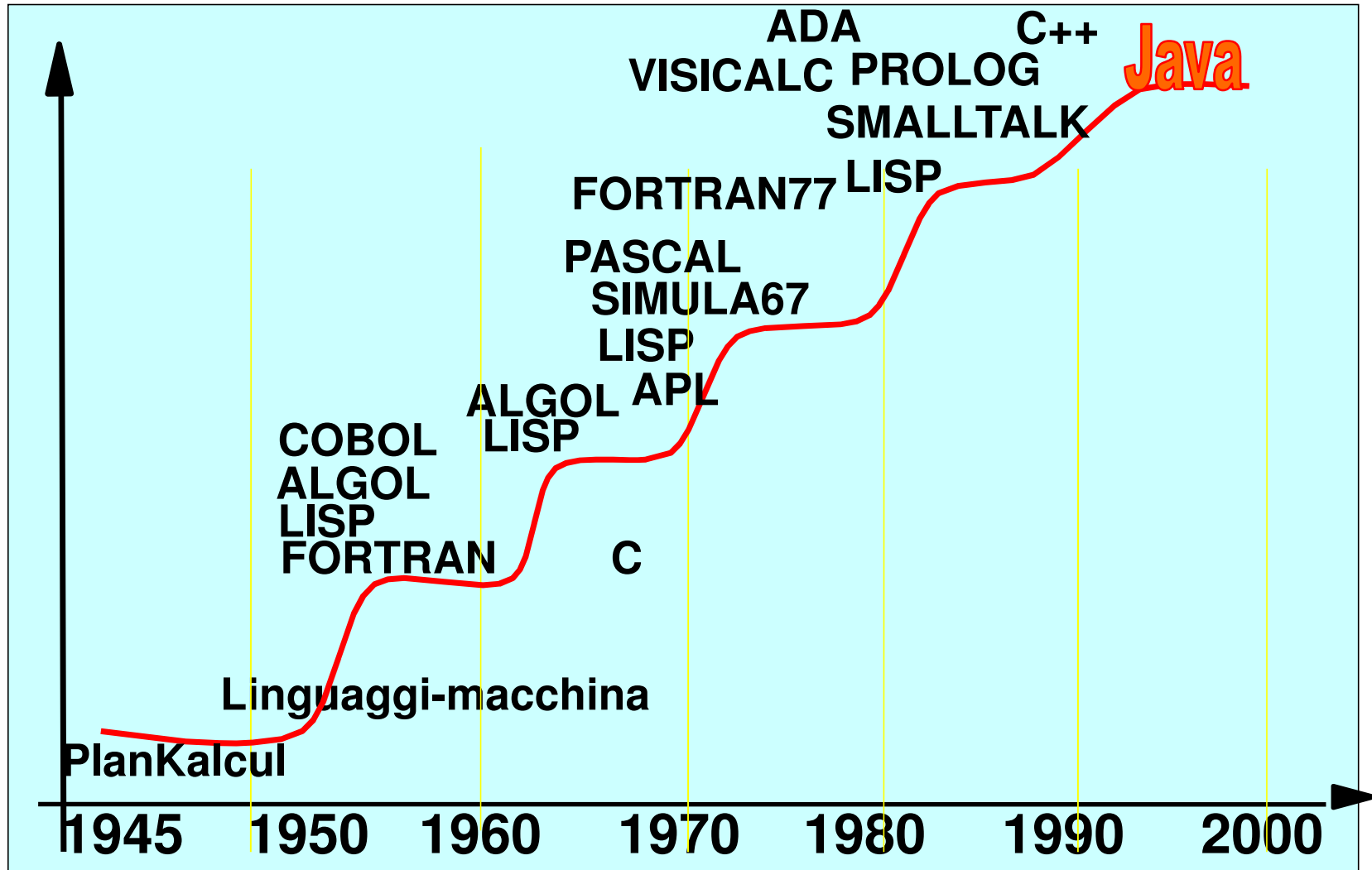
A OGGETTI



DICHIARATIVI

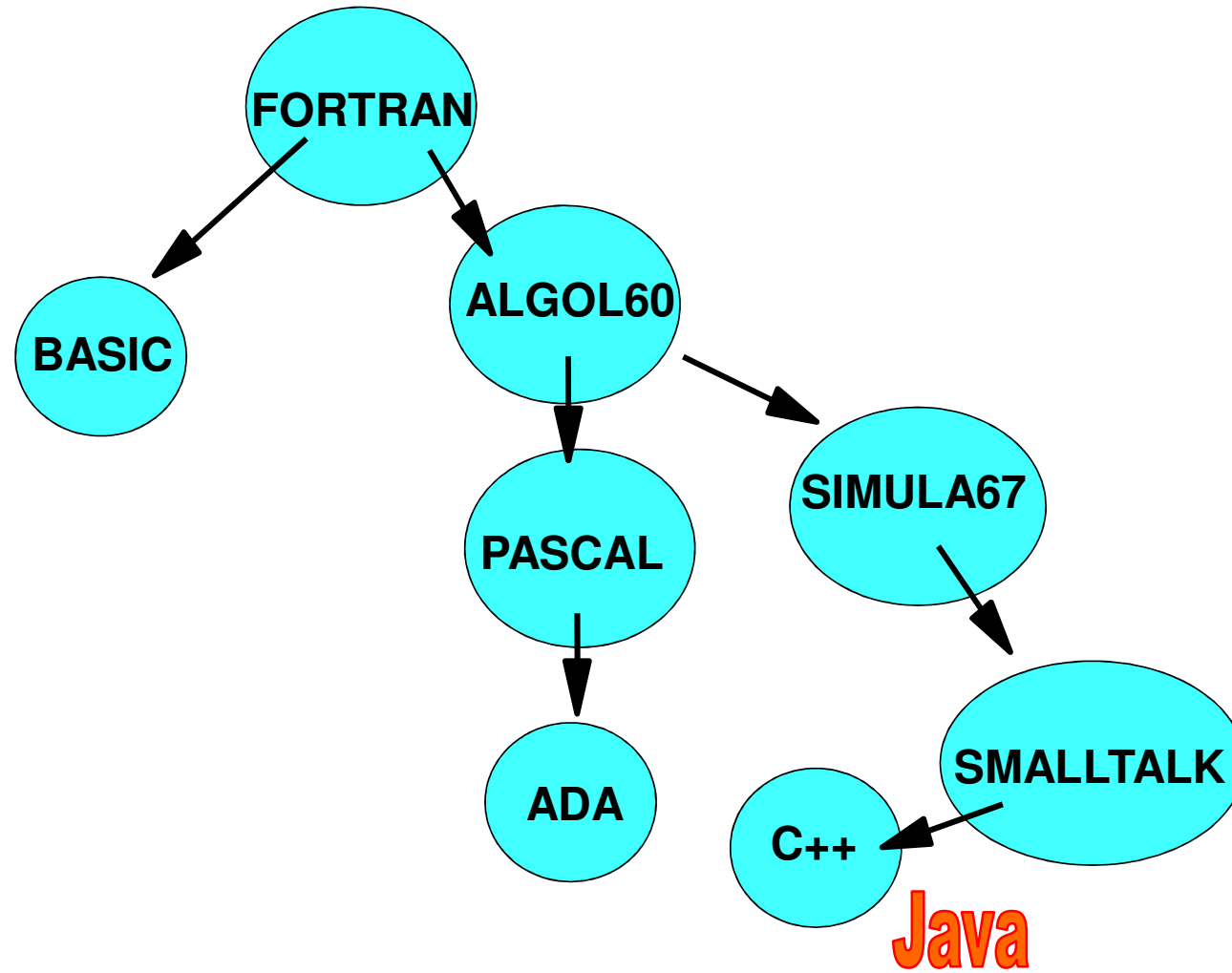


# EVOLUZIONE DEI LINGUAGGI




# EVOLUZIONE DEI LINGUAGGI

---



# TIOBE Indice di popolarità dei linguaggi

Sep 2018	Sep 2017	Programming Language	Ratings	Change
1	1	Java	17.436%	+4.75%
2	2	C	15.447%	+8.06%
3	5 	Python	7.653%	+4.67%
4	3	C++	7.394%	+1.83%
5	8	Visual Basic .NET	5.308%	+3.33%
6	4	C#	3.295%	-1.48%
7	6	PHP	2.775%	+0.57%
8	7	JavaScript	2.131%	+0.11%
9	-	SQL	2.062%	+2.06%
10	18	Objective-C	1.509%	+0.00%
11	12	Delphi/Object Pascal	1.292%	-0.49%
12	10	Ruby	1.291%	-0.64%
13	16	MATLAB	1.276%	-0.35%
14	15	Assembly language	1.232%	-0.41%
15	13	Swift	1.223%	-0.54%
16	17	Go	1.081%	-0.49%
17	9	Perl	1.073%	-0.88%
18	11	R	1.016%	-0.80%
19	19	PL/SQL	0.850%	-0.63%
20	14	Visual Basic	0.682%	-1.07%

# CHE COS'È UN LINGUAGGIO?

---

*“Un linguaggio è un **insieme di parole e di metodi di combinazione delle parole** usate e comprese da una comunità di persone”*

- È una definizione **poco precisa**:
  - *non evita le ambiguità* dei linguaggi naturali
  - non si presta a descrivere processi computazionali *meccanizzabili*
  - *non aiuta a stabilire proprietà*



# LA NOZIONE DI LINGUAGGIO

---

- Occorre una **nozione di linguaggio *più precisa***
- **Linguaggio come *sistema matematico***  
che consenta di rispondere a domande come:
  - quali sono le ***frasi lecite***?
  - si può stabilire se una frase ***appartiene al linguaggio***?
  - come si stabilisce il **significato** di una frase?
  - **quali elementi linguistici primitivi**?

# LINGUAGGIO & PROGRAMMA

---

- Dato un algoritmo, **un programma** è la sua **descrizione *in un particolare linguaggio*** di programmazione
- **Un linguaggio di programmazione** è una **notazione formale** che può essere usata per descrivere algoritmi. Due aspetti del linguaggio:
  - SINTASSI
  - SEMANTICA

# SINTASSI & SEMANTICA

---

- **Sintassi**: l'insieme di regole formali per la **scrittura di programmi** in un linguaggio, che dettano le *modalità per costruire frasi corrette* nel linguaggio stesso
- **Semantica**: l'insieme dei **significati** da attribuire alle frasi (sintatticamente corrette) costruite nel linguaggio

**NB**: una frase può essere **sintatticamente corretta** e tuttavia *non avere significato!*

# SINTASSI

---

Le regole sintattiche sono espresse attraverso *notazioni formali*:

- ◆ **BNF (Backus-Naur Form)**
- ◆ **EBNF (Extended BNF)**
- ◆ **diagrammi sintattici**

# SINTASSI EBNF: ESEMPIO

---

## Sintassi di un *numero naturale*

`<naturale> ::=`

`0 | <cifra-non-nulla>{<cifra>}`

`<cifra-non-nulla> ::=`

`1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9`

`<cifra> ::=`

`0 | <cifra-non-nulla>`

# ESEMPIO DI SINTASSI: numeri naturali

---

`<naturale> ::=`

`0 | <cifra-non-nulla>{<cifra>}`

*Intuitivamente significa che un numero naturale si può riscrivere come 0 oppure (|) come una cifra non nulla seguita da zero o più ({}) cifre*

`<cifra-non-nulla> ::=`

`1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9`

*una cifra non nulla si può riscrivere come 1 oppure 2 oppure 3...*

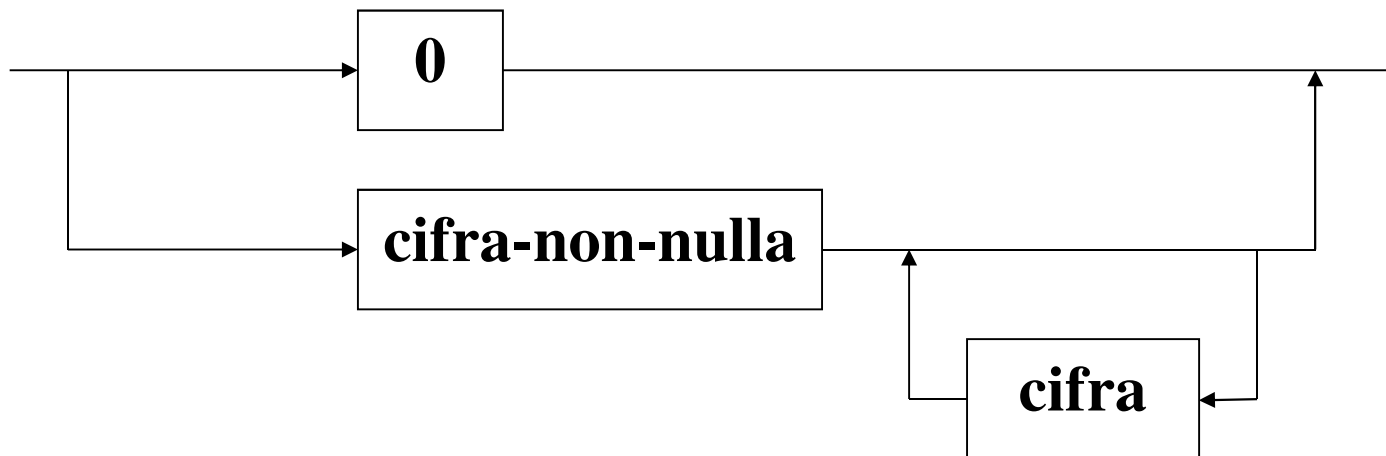
`<cifra> ::= 0 | <cifra-non-nulla>`

*una cifra si può riscrivere come 0 oppure come una cifra non nulla (definita precedentemente)*

# DIAGRAMMI SINTATTICI: ESEMPIO

---

## Sintassi di un *numero naturale*



# SEMANTICA

---

**La semantica è esprimibile:**

- ◆ ***a parole*** (poco precisa e ambigua)
- ◆ mediante **azioni**
  - ***semantica operativa***
- ◆ mediante **funzioni matematiche**
  - ***semantica denotazionale***
- ◆ mediante **formule logiche**
  - ***semantica assiomatica***



# DEFINIZIONE DI LINGUAGGIO

---

- Un linguaggio è un **insieme di frasi**
- Una frase è una **sequenza di simboli** appartenenti a un certo alfabeto

Proprietà desiderabili:

- Un linguaggio deve essere **effettivamente generabile**
- Un linguaggio di programmazione deve essere **decidibile**

# ALCUNE DEFINIZIONI

---

## **Alfabeto $V$ (o *vocabolario* o *lessico*)**

- È *l'insieme dei simboli* con cui si costruiscono le frasi

## **Universo linguistico $V^*$ di un alfabeto $V$**

- È *l'insieme di tutte le frasi* (*sequenze finite di lunghezza arbitraria*) di elementi di  $V$

## **Linguaggio $L$ su un alfabeto $V$**

- È *un sottoinsieme di  $V^*$*

## ESEMPIO

---

$V = \{ \text{if, else, ==, A, 0, =, +, 1, 2, (, ) } \}$

Allora:

$V^* = \{$   
    **if (A == 0) A = A + 2,**  
    **if else A,**  
    **do =A,**  
    ...  
}


## ESEMPIO

---

$V = \{ \text{if, else, ==, A, 0, =, +, 1, 2, (, ) } \}$

Allora:

$V^* = \{$   
    **if (A == 0) A = A + 2,**  
    **if else A,**  
    **do =A,**  
    ...  
}



Non tutte queste  
frasi faranno  
parte del  
linguaggio!

# LINGUAGGI E GRAMMATICHE

---

- **Come specificare il sottoinsieme di  $V^*$  che definisce il linguaggio?**
- **Specificando il modo *formale e preciso* la sintassi delle frasi del linguaggio**

## TRAMITE

una *grammatica formale*:  
una **notazione matematica** che  
consente di esprimere *in modo*  
***rigoroso la sintassi di un linguaggio***

# GRAMMATICA FORMALE

---

Una *quadrupla*  $\langle VT, VN, P, S \rangle$  dove:

- **VT** è un *insieme finito di simboli terminali*
- **VN** è un *insieme finito di simboli non terminali*
- **P** è un *insieme finito di produzioni, ossia di regole di riscrittura*
- **S** è un particolare *simbolo non-terminale* detto *simbolo iniziale* o *scopo* della grammatica

# GRAMMATICA B.N.F.

---

Una *Grammatica B.N.F.* è una grammatica in cui le produzioni hanno la forma

$$X ::= A$$

- $X \in VN$  è un simbolo non terminale
- $A$  è una **sequenza di simboli** ciascuno appartenente all'alfabeto  $VN \cup VT$
- Una *Grammatica B.N.F.* definisce quindi un **linguaggio sull'alfabeto terminale  $VT$**  mediante un **meccanismo di derivazione** (o **riscrittura**)

# GRAMMATICA E LINGUAGGIO

---

Data una grammatica  $G$ , si dice perciò

***Linguaggio  $L_G$  generato da  $G$***

l'insieme delle frasi di  $V$

- derivabili dal **simbolo iniziale  $S$**
- applicando le **produzioni  $P$**

Le frasi di un linguaggio di programmazione vengono dette ***programmi*** di tale linguaggio



# DERIVAZIONE

---

Siano

- G una grammatica
- $\beta, \gamma$  due *stringhe*, cioè due elementi dell'universo linguistico  $(VN \cup VT)^*$

$\gamma$  deriva direttamente da  $\beta$  (e si scrive  $\beta \rightarrow \gamma$ ) se

- le stringhe *si possono decomporre* in
$$\beta = \eta A \delta \qquad \gamma = \eta \alpha \delta$$
- ed esiste la produzione  $A ::= \alpha$

In generale,  $\gamma$  deriva da  $\beta$  se esiste una sequenza di  $N$  derivazioni *dirette* che da  $\beta$  possono produrre  $\gamma$

$$\beta = \beta_0 \rightarrow \beta_1 \rightarrow \dots \rightarrow \beta_n = \gamma$$

# FORMA B.N.F. COMPATTA

---

- In una grammatica BNF spesso ***esistono più regole con la stessa parte sinistra:***

–  $X ::= A_1$

– ...

–  $X ::= A_N$

- Per comodità si stabilisce allora di poterle ***compattare in un'unica regola:***

$X ::= A_1 \mid A_2 \mid \dots \mid A_N$

dove **il simbolo | indica l'alternativa**

# ESEMPIO COMPLESSIVO

---

**G** =  $\langle$  VT, VN, P, S  $\rangle$

dove:

**VT** = { il, gatto, topo, sasso, mangia, beve }

**VN** = { <frase>, <soggetto>, <verbo>,  
<compl-ogg>, <articolo>, <nome> }

**S** = <frase>

**P** = ...

# ESEMPIO COMPLESSIVO

---

P = {  
    <frase> ::= <soggetto> <verbo> <compl-ogg>  
    <soggetto> ::= <articolo><nome>  
    <articolo> ::= il  
    <nome> ::= gatto | topo | sasso  
    <verbo> ::= mangia | beve  
    <compl-ogg> ::= <articolo> <nome>  
}

# ESEMPIO COMPLESSIVO

---

**ESEMPIO: derivazione della frase**

*“il sasso mangia il topo”*

(ammesso che tale frase *sia derivabile*, ossia faccia parte del linguaggio generato dalla nostra grammatica)

**DERIVAZIONE “LEFT-MOST”**

**A partire dallo scopo della grammatica, si riscrive sempre *il simbolo non-terminale più a sinistra***

# ESEMPIO COMPLESSIVO

---

<frase>

- <soggetto> <verbo> <compl-ogg>
- <articolo> <nome> <verbo> <compl-ogg>
- **il** <nome> <verbo> <compl-ogg>
- **il sasso** <verbo> <compl-ogg>
- **il sasso mangia** <compl-ogg>
- **il sasso mangia** <articolo><nome>
- **il sasso mangia il** <nome>
- **il sasso mangia il topo**

# ESEMPIO COMPLESSIVO

---

## ALBERO SINTATTICO

*un grafo che esprime il processo di derivazione di una frase* usando una data grammatica

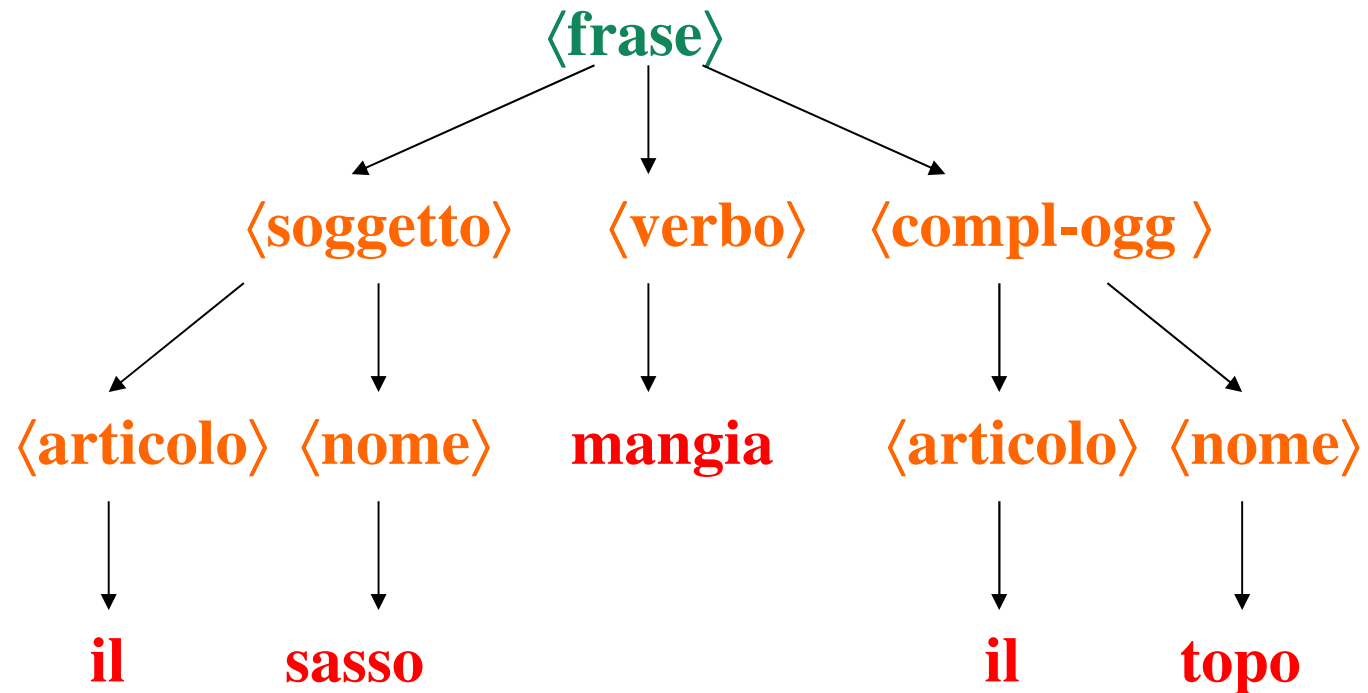
## ESEMPIO: derivazione della frase

*“il sasso mangia il topo”*

(ammesso che tale frase *sia derivabile*, ossia faccia parte del linguaggio generato dalla nostra grammatica)

# ESEMPIO COMPLESSIVO

---





## EXTENDED B.N.F. (E.B.N.F.)

---

Una forma *estesa* della notazione B.N.F. che introduce alcune ***notazioni compatte per alleggerire la scrittura*** delle regole di produzione

Forma EBNF	BNF equivalente	significato
$X ::= [a] B$	$X ::= B \mid aB$	a può comparire 0 o 1 volta
$X ::= \{a\}^n B$	$X ::= B \mid aB \mid \dots \mid a^n B$	a può comparire da 0 a n volte
$X ::= \{a\} B$	$X ::= B \mid aX$	a può comparire 0 o più volte

NOTA: la produzione  $X ::= B \mid aX$  è ricorsiva (a destra)

# EXTENDED B.N.F. - E.B.N.F.

---

Per raggruppare *categorie sintattiche*:

Forma EBNF	BNF equivalente	significato
$X ::= (a \mid b) D \mid c$	$X ::= a D \mid b D \mid c$	raggruppa categorie sintattiche

- Ci sono programmi che possono creare automaticamente analizzatori sintattici (parser) per linguaggi espressi tramite EBNF
- XML è definito da una grammatica EBNF di circa 80 regole

# ESEMPIO: I NUMERI NATURALI

---

$G = \langle VT, VN, P, S \rangle$

dove:

$VT = \{ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 \}$

$VN = \{ \langle \text{num} \rangle, \langle \text{cifra} \rangle, \langle \text{cifra-non-nulla} \rangle \}$

$S = \langle \text{num} \rangle$

$P = \{$   
     $\langle \text{num} \rangle ::= \langle \text{cifra} \rangle \mid \langle \text{cifra-non-nulla} \rangle \{ \langle \text{cifra} \rangle \}$   
     $\langle \text{cifra} \rangle ::= 0 \mid \langle \text{cifra-non-nulla} \rangle$   
     $\langle \text{cifra-non-nulla} \rangle ::= 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$   
}

## ESEMPIO: I NUMERI INTERI

---

- Sintassi analoga alla precedente
- ma con *la possibilità di un segno (+, -) davanti al numero naturale*

Quindi:

- **stesse regole di produzione  
più una per gestire il segno**
- **stesso alfabeto terminale  
più i due simboli + e -**

## ESEMPIO: I NUMERI INTERI

---

$G = \langle VT, VN, P, S \rangle$ , dove:

$VT = \{ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, \quad \}$

$VN = \{ \langle int \rangle, \langle num \rangle, \langle cifra \rangle, \langle cifra-non-nulla \rangle \}$

$P = \{$

$\langle int \rangle ::= [+|-] \langle num \rangle$

$\langle num \rangle ::= 0 \mid \langle cifra-non-nulla \rangle \{ \langle cifra \rangle \}$

$\langle cifra \rangle ::= 0 \mid \langle cifra-non-nulla \rangle$

$\langle cifra-non-nulla \rangle ::= 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$

$\}$

# ESEMPIO: IDENTIFICATORI

$G = \langle VT, VN, P, S \rangle$

- Nell'uso pratico, quasi sempre *si danno solo le regole di produzione*, definendo VT, VN e S implicitamente

- Quindi:

$P = \{$

$\langle id \rangle ::= \langle lettera \rangle \{ \langle lettera \rangle \mid \langle cifra \rangle \}$

$\langle lettera \rangle ::= A \mid B \mid C \mid D \mid \dots \mid Z$

$\langle cifra \rangle ::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$

$\}$

scopo

VN

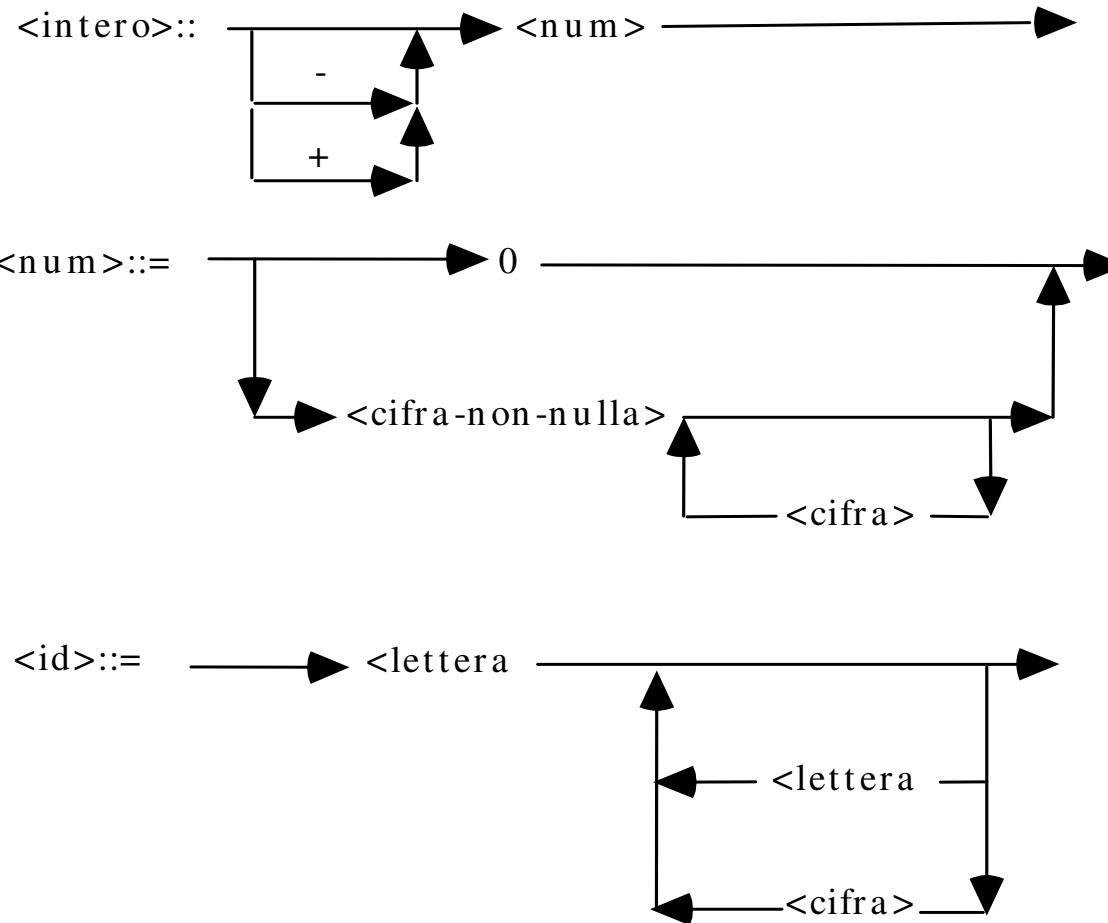
VN

VT

VT

# DIAGRAMMI SINTATTICI

---



## ESEMPIO DI ALBERO SINTATTICO

---

- Albero sintattico del numero **-3457**  
(grammatica EBNF dell'esempio 2)

- Attenzione

poiché  $X ::= \{a\} B$  equivale a  $X ::= B \mid aX$ ,  
e  $X ::= C \{a\}$  equivale a  $X ::= C \mid Xa$ ,

la regola:

$\langle \text{num} \rangle ::= \langle \text{cifra-non-nulla} \rangle \{ \langle \text{cifra} \rangle \}$

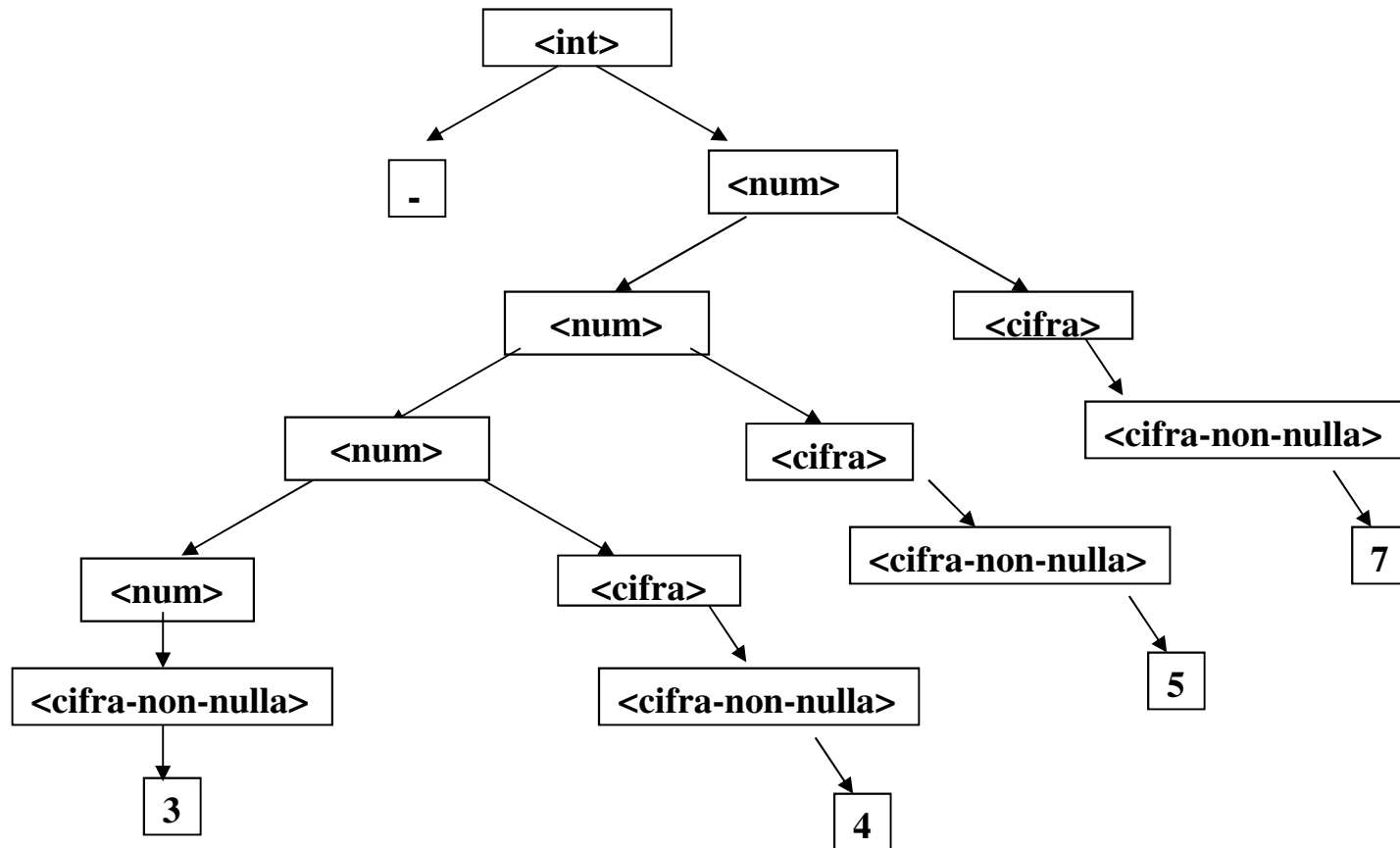
equivale a:

$\langle \text{num} \rangle ::= \langle \text{cifra-non-nulla} \rangle \mid \langle \text{num} \rangle \langle \text{cifra} \rangle$



# ALBERO SINTATTICO DI -3457

---



## ESERCIZIO Grammatiche 1

---

Data la grammatica **G** con scopo **S** e simboli terminali **{a,c,0,1}**

**S ::= a F c**

**F ::= a S c | E**

**E ::= 0 | 1**

si mostri (mediante derivazione left-most) che la stringa **aaa1ccc** appartiene alla grammatica

# ESERCIZIO 1: Soluzione

---

**S ::= a F c**

**F ::= a S c | E**

**E ::= 0 | 1**

**S** → aF**c** → aaS**cc** → aaaF**ccc** → aaaE**ccc**  
→ aaa**1ccc**

## ESERCIZIO Grammatiche 2

---

Si consideri la grammatica **G** con scopo **S** e simboli terminali {**il, la, Alice, regina, coniglio, sgrida, saluta, gioca**}

**S ::= T P | A T P**

**P ::= V | V T | V A T**

**T ::= Alice | regina | coniglio**

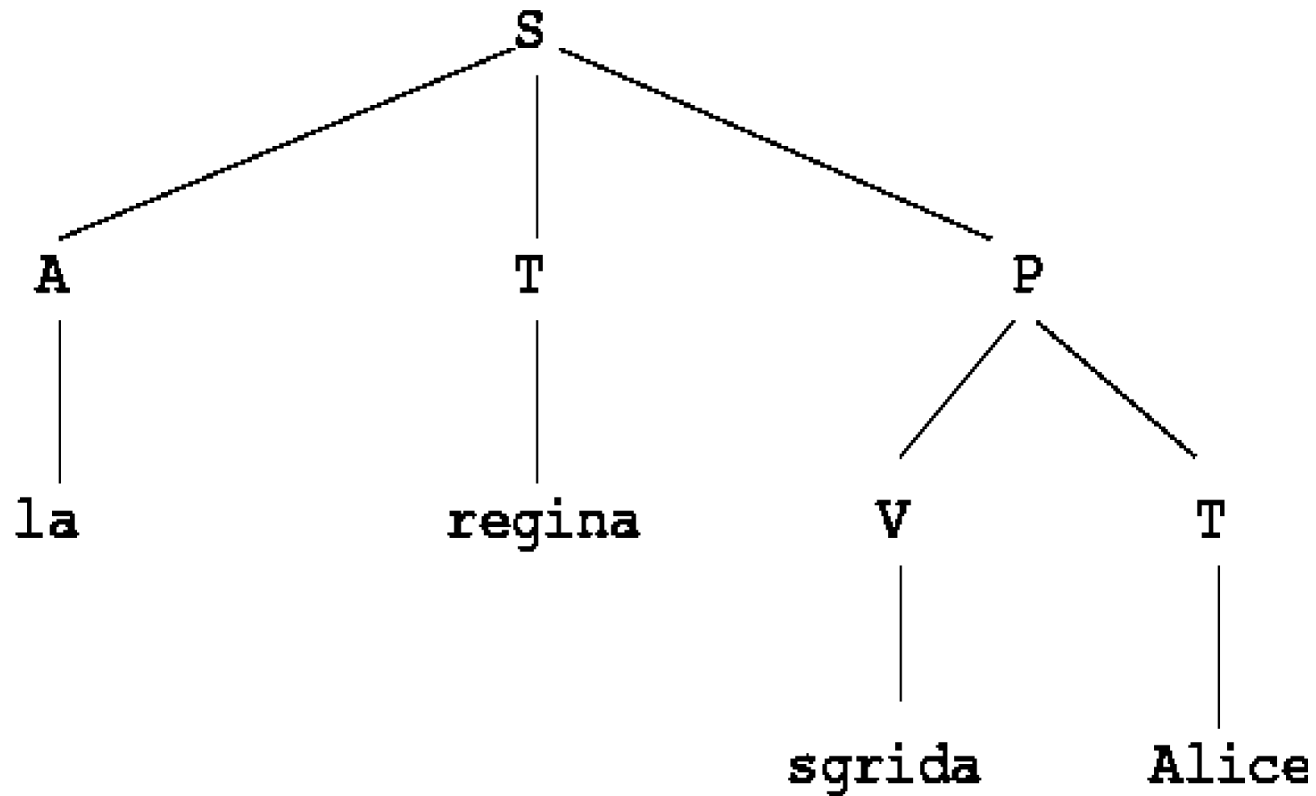
**A ::= il | la**

**V ::= sgrida | saluta | gioca**

Si dica se la stringa **la regina sgrida Alice** è sintatticamente corretta rispetto a tale grammatica e se ne mostri l'albero sintattico

## ESERCIZIO 2: Soluzione

---



# ESERCIZIO Grammatiche 3

## Espressioni algebriche

---

$G = \langle VT, VN, P, S \rangle$ , dove:

$VT = \{ +, -, *, /, (, ), 1, 2, 3, 4, 5, 6, 7, 8, 9, 0 \}$

$VN = \{ \langle E \rangle, \langle T \rangle, \langle F \rangle, \langle \text{num} \rangle, \langle \text{cifra} \rangle, \langle \text{cifra-non-nulla} \rangle \}$

$S = \langle E \rangle$

# ESERCIZIO Grammatiche 3

## Espressioni Algebriche

---

**P = {**

**$\langle E \rangle ::= \langle E \rangle + \langle T \rangle \mid \langle E \rangle - \langle T \rangle \mid \langle T \rangle$**

**$\langle T \rangle ::= \langle T \rangle * \langle F \rangle \mid \langle T \rangle / \langle F \rangle \mid \langle F \rangle$**

**$\langle F \rangle ::= \langle \text{cifra} \rangle \mid ( \langle E \rangle )$**

**$\langle \text{cifra} \rangle ::= 0 \mid \langle \text{cifra-non-nulla} \rangle$**

**$\langle \text{cifra-non-nulla} \rangle ::= 1|2|3|4|5|6|7|8|9$**

**}**

**Disegnare il diagramma sintattico di tale grammatica.  
Determinare poi se le seguenti frasi fanno parte del  
linguaggio generato da questa grammatica o no, e  
disegnarne l'albero di derivazione sintattica:**

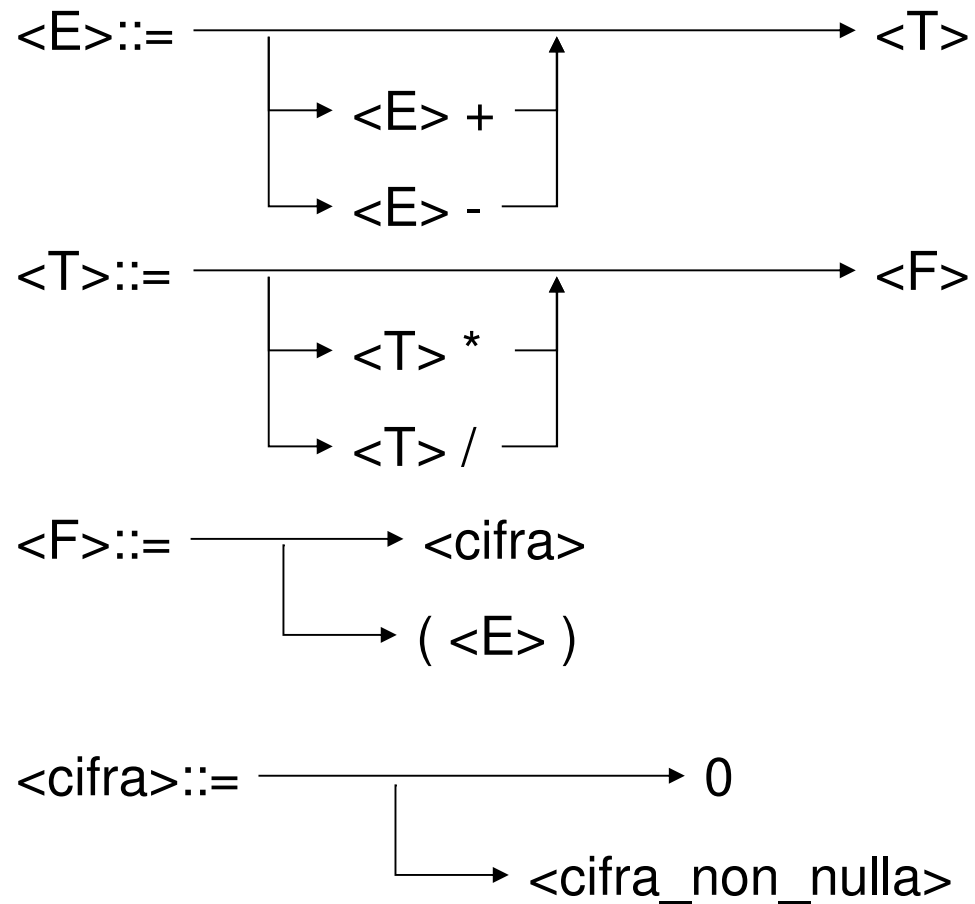
- 1.  $5 + 3 * 7$**
- 2.  $3 / 0 + 4$**

# ESERCIZIO Grammatiche 3

## Soluzione

---

**Diagramma sintattico:**



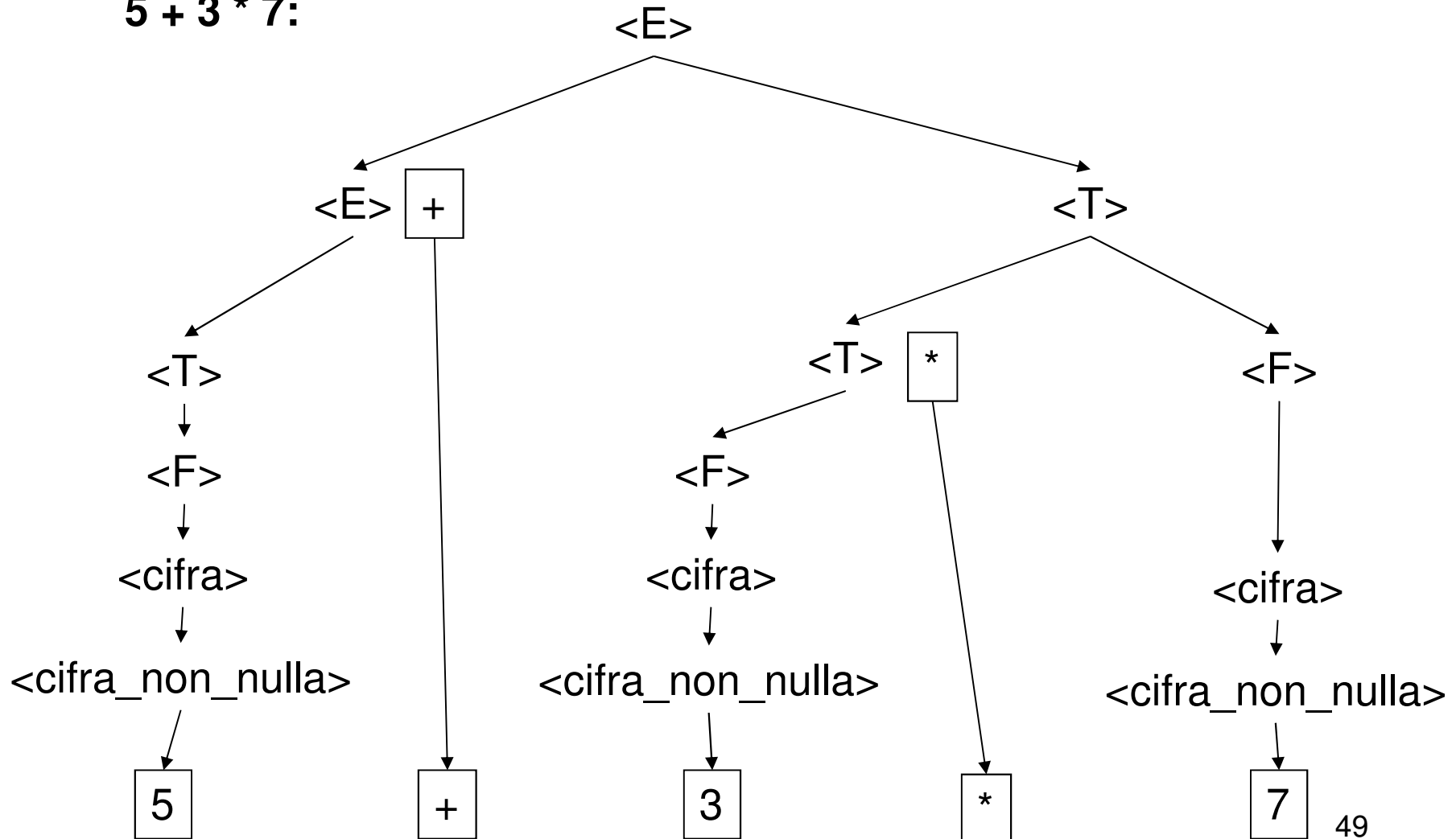


# ESERCIZIO Grammatiche 3

## Soluzione

---

5 + 3 \* 7:



# ESERCIZIO Grammatiche 3

## Soluzione

---

**3 / 0 + 4:**

