

LINGUAGGI DI ALTO LIVELLO

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

Linguaggi di alto livello

•Barriera di astrazione

- Fortran
- Cobol
- Basic
- Pascal
- C
- Modula-2
- Algol
- Ada
- Lisp
- Scheme
- Prolog
- ML

*AN - 1995

Linguaggi di alto livello

•Barriera di astrazione

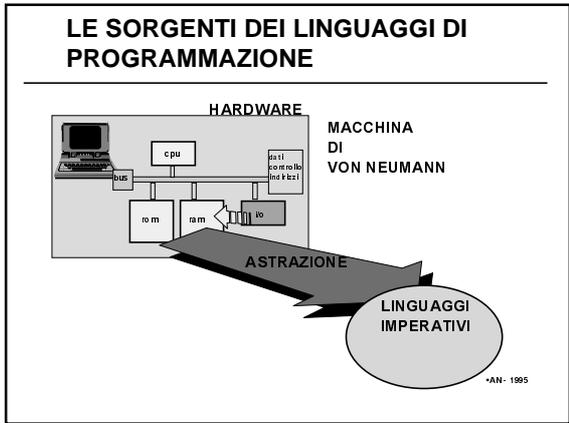
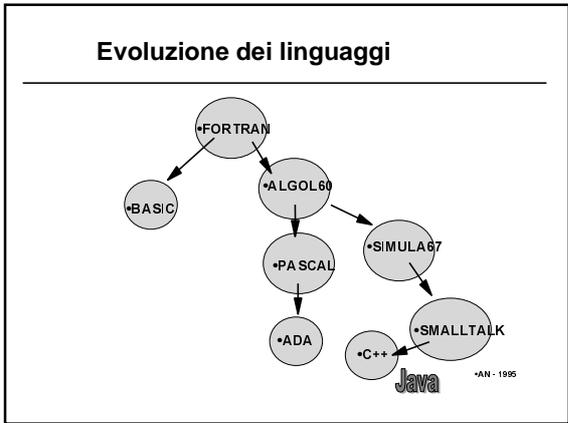
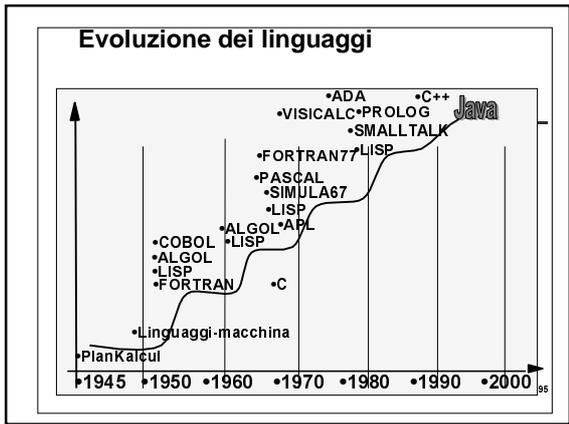
IMPERATIVI: •Fortran, •Cobol, •Basic, •Pascal, •C, •Modula-2, •Algol, •Ada

FUNZIONALI: •Lisp, •Scheme, •ML

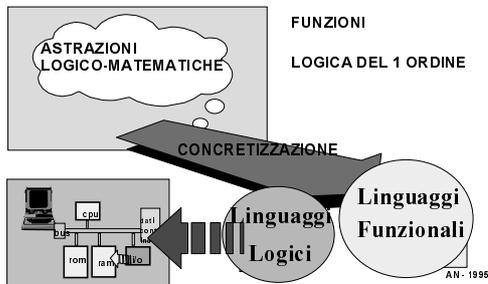
A OGGETTI: •Prolog

DICHIARATIVI: •Lisp, •ML

*AN - 1995



LE SORGENTI DEI LINGUAGGI DI PROGRAMMAZIONE



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*

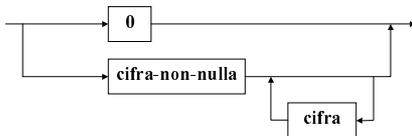
```
<naturale> ::=
  0 | <cifra-non-nulla>{<cifra>}
<cifra-non-nulla> ::=
  1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<cifra> ::=
  0 | <cifra-non-nulla>
```

SINTASSI DI UN NUMERO NATURALE

```
<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 una 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 prece
```

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 $V = VN \cup VT$.
- 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
- β, γ due *stringhe*, cioè due elementi dell'universo linguistico $(VN \cup VT)^*$.

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

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

In generale, γ deriva da β se esiste una sequenza di N derivazioni *dirette* che da β possono produrre γ
 $\beta = \beta_0 \rightarrow \beta_1 \rightarrow \dots \rightarrow \beta_n = \gamma$

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 \mid indica l'*alternativa*.

ESEMPIO COMPLESSIVO

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

dove:

$VT = \{ \text{il, gatto, topo, sasso, mangia, beve} \}$

$VN = \{ \langle \text{frase} \rangle, \langle \text{soggetto} \rangle, \langle \text{verbo} \rangle, \langle \text{compl-ogg} \rangle, \langle \text{articolo} \rangle, \langle \text{nome} \rangle \}$

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

$P = \dots$

ESEMPIO COMPLESSIVO

$P = \{$

$\langle \text{frase} \rangle ::= \langle \text{soggetto} \rangle \langle \text{verbo} \rangle \langle \text{compl-ogg} \rangle$

$\langle \text{soggetto} \rangle ::= \langle \text{articolo} \rangle \langle \text{nome} \rangle$

$\langle \text{articolo} \rangle ::= \text{il}$

$\langle \text{nome} \rangle ::= \text{gatto} \mid \text{topo} \mid \text{sasso}$

$\langle \text{verbo} \rangle ::= \text{mangia} \mid \text{beve}$

$\langle \text{compl-ogg} \rangle ::= \langle \text{articolo} \rangle \langle \text{nome} \rangle$

$\}$

ESEMPIO COMPLESSIVO

ESEMPIO: derivazione della frase

"il gatto 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

$\langle \text{frase} \rangle$

→ $\langle \text{soggetto} \rangle \langle \text{verbo} \rangle \langle \text{compl-ogg} \rangle$

→ $\langle \text{articolo} \rangle \langle \text{nome} \rangle \langle \text{verbo} \rangle \langle \text{compl-ogg} \rangle$

→ $\text{il} \langle \text{nome} \rangle \langle \text{verbo} \rangle \langle \text{compl-ogg} \rangle$

→ $\text{il gatto} \langle \text{verbo} \rangle \langle \text{compl-ogg} \rangle$

→ $\text{il gatto mangia} \langle \text{compl-ogg} \rangle$

→ $\text{il gatto mangia} \langle \text{articolo} \rangle \langle \text{nome} \rangle$

→ $\text{il gatto mangia il} \langle \text{nome} \rangle$

→ $\text{il gatto mangia il topo}$

ESEMPIO COMPLESSIVO

ALBERO SINTATTICO

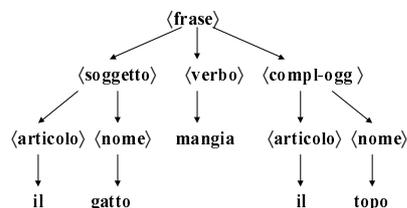
- un grafo che esprime il processo di derivazione di una frase usando una data grammatica.

ESEMPIO: derivazione della frase

“il gatto 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 (parsers) 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:
 $VT = \{ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, +, - \}$
 $VN = \{ \langle int \rangle, \langle num \rangle, \langle cifra \rangle, \langle cifra-non-nulla \rangle \}$
 $P = \{$
 $\langle int \rangle ::= [+|-] \langle num \rangle$
 $\langle num \rangle ::= 0 \mid \langle cifra-non-nulla \rangle \langle cifra \rangle$
 $\langle cifra \rangle ::= 0 \mid \langle cifra-non-nulla \rangle$
 $\langle cifra-non-nulla \rangle ::= 1|2|3|4|5|6|7|8|9$
 $\}$

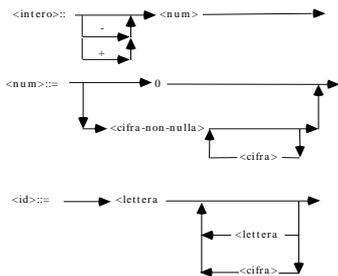
ESEMPIO: IDENTIFICATORI

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

- Nell'uso pratico, quasi sempre *si danno solo le regole di produzione*, definendo VT, VN e S *implicitamente*.
- Quindi: **scopo** (VN) \rightarrow $\langle id \rangle$ (VN)

$P = \{$
 $\langle id \rangle ::= \langle lettera \rangle \{ \langle lettera \rangle \mid \langle cifra \rangle \}$ (VT)
 $\langle lettera \rangle ::= A \mid B \mid C \mid D \mid \dots \mid Z$ (VT)
 $\langle cifra \rangle ::= 0|1|2|3|4|5|6|7|8|9$ (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:
 $\langle num \rangle ::= \langle cifra-non-nulla \rangle \langle cifra \rangle$
equivale a:
 $\langle num \rangle ::= \langle cifra-non-nulla \rangle \mid \langle num \rangle \langle cifra \rangle$

ALBERO SINTATTICO DI -3457

