Paradigmi e linguaggi di programmazione

Linguaggi formali e modelli per l'informatica

Linguaggi formali

- **Grammatiche**
  - Approccio generativista

Grammatiche

- ***Approccio generativo***

  - Metodo di costruzione delle stringhe del linguaggio basato sulla ***riscrittura***

    - riscrittura è la sostituzione di un oggetto al posto di una parte di un altro oggetto, secondo una precisa regola formale.

      - Un sistema di riscrittura è appunto costituito da un insieme di oggetti, per esempio termini, grafi o altri oggetti matematici, e un insieme finito di regole che definiscono una relazione sugli oggetti.

Grammatiche

Il concetto di grammatica in informatica, nasce dalla necessità di formalizzare e descrivere i processi legati all'uso del computer e agli strumenti di comunicazione:

- 1914 Axel Thue studia i primi problemi di riscrittura.
- 1920-40 Emil Post definisce sistemi di produzione.
- 1947 A.A. Markov definisce algoritmi basati su regole di riscrittura.
- 1956 N. Chomsky introduce le grammatiche formali nell'ambito degli studi sul linguaggio naturale.
- 1960 J. W. Backus e P. Naur introducono la BNF per descrivere la struttura sintattica dei programmi.
- 1990 T. Berners Lee introduce i linguaggi di marcatura per definire la struttura sintattica di pagine web e documenti

Le grammatiche formali

- Consentono di *descrivere linguaggi infiniti in modo finito*
  - quindi l'insieme delle frasi sarà numerabile

- Una grammatica definisce delle "frasi" lecite:
  - data una frase F, questa soddisfa la grammatica G?
- Definisce quindi una sorta di linguaggio
  - l'insieme delle frasi che soddisfano la grammatica

> ***Le grammatiche formali caratterizzano un linguaggio di programmazione come l'insieme di tutte le stringhe (programmi) derivabili dalla grammatica***

Le grammatiche formali

- Una grammatica è formata da un insieme di regole di trasformazione
  - Es.: G={X → YY, Y → a, Y → bc}
- Applicandole in tutte le loro possibili combinazioni si identificano tutte e sole le frasi del linguaggio di interesse
  - Es.:
  - X → YY → aY → aa
  - X → YY → aY → abc
  - X → YY → bcY → bca
  - X → YY → bcY → bcbc
- Il linguaggio di G ha 4 frasi:
  - L(G)={aa,abc,bca,bcbc}

Le grammatiche formali

- Es.: G={X → b, X → aXc}
  - X →b
  - X →aXc →abc
  - X→ aXc → aaXcc →aabcc
  - X → aXc → aaXcc → aaaXccc →aaabccc
  - ...

❖ L(G)={b,abc,aabcc,aaabccc,...}

- ***L(G) ha cardinalità infinita***
  - X è definito in termini di X

Le grammatiche formali

- ***Definizione matematica***

  Una grammatica è definita come una quadrupla formata dall'insieme dei simboli terminali di un linguaggio, simboli non terminali, simbolo iniziale, regole di produzione

- E' formata da 4 elementi
  - **VN** insieme finito di simboli non terminali
  - **VT** insieme finito di simboli terminali
  - **P** insieme di regole di riscrittura
  - **S** simbolo iniziale, detto "scopo"

Le grammatiche formali

Simboli (terminali e non)

- VT insieme finito di simboli terminali
  - è l'insieme delle sequenze dell'alfabeto A che possono comparire nella grammatica
- VN, insieme finito di simboli non terminali
  - sono meta-simboli di "appoggio"
  - rappresentano categorie sintattiche

Esempio:

- Sia A = {a,b,c} *e* X → YY, Y → a, Y → bc

➢ VT = {a,bc}, VN = {X,Y}

Le grammatiche formali

Regole di Produzione e Scopo

- **P**, insieme di regole di riscrittura
  - ognuna è del tipo **α→β**
  - dove α e β sono sequenze nell'insieme VT ∪ VN
- **S** ϵVN, simbolo iniziale, detto "scopo"
  - è il simbolo non terminale da cui si iniziano le trasformazioni

Esempio :

X → YY, Y → a, Y →bc

- P={X → YY, Y →a, Y →bc}, S=X

Le grammatiche formali

Trasformazioni

1. Si parte dal simbolo non terminale S
2. Si applica una regola di produzione $\alpha \rightarrow \beta$
3. si cerca la presenza della sotto-sequenza α
4. la si sostituisce con β
   - α e β sono chiamate "forme di frase"
5. Si procede finché non si ottiene una stringa con soli simboli terminali
   - tale stringa è una frase del linguaggio

- Tutte le frasi del linguaggio si ottengono con questo procedimento
  - scegliendo via via regole di produzione diverse

Le grammatiche formali

Esempio

- Sia una grammatica <VT,VN,P,S> su A={0,1}
  - VT={0,1} simboli terminali
  - VN={X} simboli non terminali
  - P = { X→0, X→1X} produzioni
  - S = X scopo
- *Qual è il linguaggio?*
  - L= {0,10,110,1110,11110,...}
- *Come si ottengono queste frasi?*
  - S = X →0
  - S = X →1X → 10
  - S = X → 1X →11X → 110

Le grammatiche formali

Esempio

- A={a,b}, L= {ab,aabb,aaabbb,...}
  - VT={a,b} simboli terminali
  - VN={F} simboli non terminali
  - P={F→ab, F→aFb} produzioni
  - S=F scopo della grammatica
- Frasi:
  - Es1: F→ ab
  - Es2: F→aFb →aabb
  - Es3: F→aFb→ aaFbb →aaabbb

Le grammatiche formali

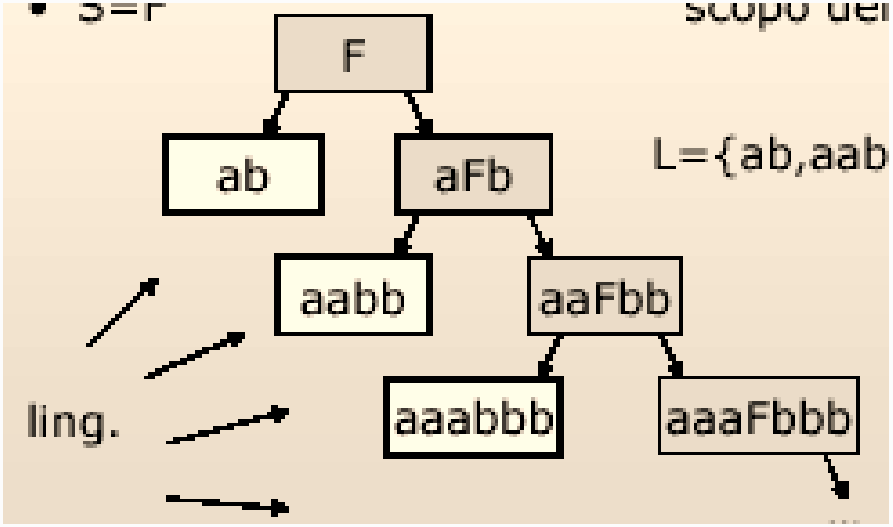
Albero delle frasi

- E’ una struttura visuale ad albero per rappresentare (una porzione finita) delle frasi di una grammatica
- La radice specifica lo scopo della grammatica
- Ogni nodo ha tanti figli
  - uno per ogni produzione applicabile
  - il figlio è ciò che si ottiene applicando la produzione al padre
- Ogni nodo specifica una forma di frase
- Le foglie sono frasi della grammatica

Le grammatiche formali

Esempio

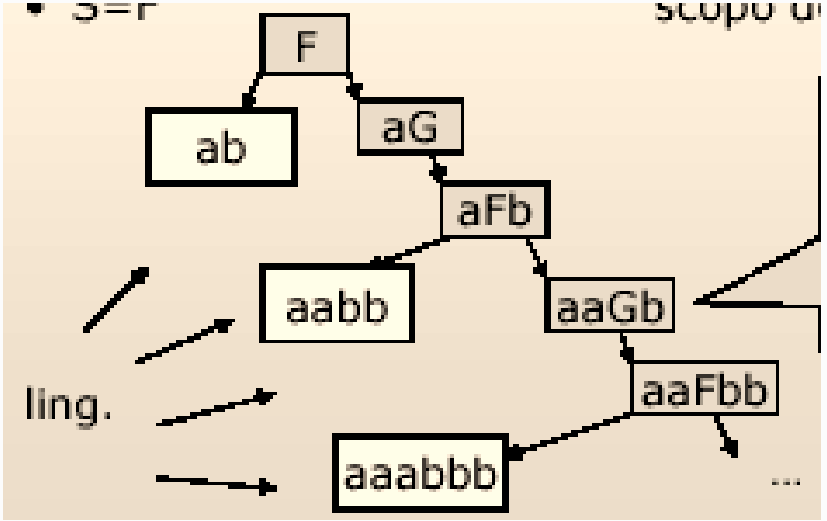
- VT={a,b}
  simboli terminali
- VN={F}
  simboli non terminali
- P={F → ab, F→aFb}
  produzioni
- S=F
  scopo della grammatica
- L={ab,aabb,aaabbb,..}



Le grammatiche formali

Esempio

- VT={a,b} simboli terminali
- VN={F,G} simboli non terminali
- P={F → ab, F→aG, G→Fb} produzioni
- S=F scopo della grammatica



➢ Stesso linguaggio di prima, diversa grammatica

Backus Naur Form (BNF)

- Sono una notazione un pò più conveniente per rappresentare grammatiche formali (anni ’60)
  - è una metasintassi, ovvero un formalismo attraverso cui è possibile descrivere la sintassi di linguaggi formali
  - fu proposta da John Backus nel corpo della definizione del linguaggio di programmazione ALGOL.
- Sono in grado di descrivere grammatiche *libere dal contesto* (di tipo 2 secondo la classificazione di Chomsky) o *algebriche*
  - *Rispetto alle grammatiche delle espressioni regolare (tipo 3) che sono non-counting*
- Una BNF definisce un linguaggio sull’alfabeto terminale mediante un meccanismo di derivazione o riscrittura

Backus Naur Form

- Simbologia di BNF
  - Le regole di produzione sono della forma

    $\alpha ::= \beta$ invece di $\alpha \rightarrow \beta$
  - Il meta-simbolo speciale | è usato per l'alternativa
    - da: $\alpha \rightarrow \beta 1$, $\alpha \rightarrow \beta 2$, …,$\alpha \rightarrow \beta n$
    - a: $\alpha ::= \beta 1 \mid \beta 2 \mid \ldots \mid \beta n$
  - Che si può leggere:
    - α produce β1 oppure β2 oppure ... oppure βn

Backus Naur Form

Esempio

- BNF per definire l'identificatore di variabile

<identificatore>::= <lettera>
<identificatore>::= <lettera><sequenza caratteri>
<lettera>::= a|b|…|z|_
<sequenza caratteri>::=<carattere><sequenza caratteri>
<carattere>::=<lettera> <cifra>
<cifra>::=0|1|2|3|4|5|6|7|8|9

Esempio: numero naturale

- BNF per definire 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>

BNF per un programma C

<Programma>::= main()<sequenza istruzioni>

<sequenza istruzioni>::=<istruzione><sequenza istruzioni>

<istruzione>::=<istruzione semplice>|<istruzione composta>

<istruzione semplice>::=<istruzioneIO>|<istruzione assegnazione>

<istruzioneIO>::=<istruzione lettura>|<istruzione scrittura>

<istruzione lettura>::=scanf(<indentificatore di variabile>);

<istruzione scrittura>::=printf(<identificatore di variabile>);

<istruzione assegnazione>::=<identificatore di

variabile>=<Espressione>;

<istruzione composta>::=<istruzione condizionale>|<istruzione iterativa>

<istruzione condizionale>::=……

Backus Naur Form

- Esempio: descrivere in modo formale, preciso e non ambiguo le regole da seguire per scrivere un indirizzo su una lettera:

1. un indirizzo postale include un destinatario, seguito da un indirizzo, seguito da una indicazione di località;
2. il destinatario comprende sicuramente un cognome, a cui si possono far precedere, nell'ordine, un titolo (come Sig. o Dott. ecc.) e un nome o una iniziale;
3. l'indirizzo comprende necessariamente una indicazione di via (o piazza, viale, ecc.) e il numero civico;
4. l'indicazione della località comprende un codice di avviamento postale opzionale, seguito dal nome del comune e dalla provincia.

Backus Naur Form

<indirizzo postale> ::= <destinatario> <indirizzo> <località>
<destinatario> ::= [<titolo>] [<nome>|<iniziale>] <cognome> <a capo>
<indirizzo> ::= <via> <numero civico> <a capo>
<località> ::= [<CAP>] <comune> <provincia>

Interpretazione:

- <indirizzo postale>

  (applicando la regola 1 diventa)
- <destinatario> <indirizzo> <località>

  (applicando la regola 2 diventa)
- [<titolo>] [<nome>|<iniziale>] <cognome> <a capo> <indirizzo> <località>

  (applicando la regola 3 diventa)
- [<titolo>] [<nome>|<iniziale>] <cognome> <a capo> <via> <numero civico> <a capo> <località>

Backus Naur Form

Esempio con più categorie sintattiche

- A=VT={il,gatto,topo,sasso,mangia,beve}
- P={

  <frase>::=<soggetto><verbo><compl-ogg>

  <soggetto>::=<articolo><nome>

  <articolo>::=il

  <nome>::=gatto|topo|sasso

  <verbo>::=mangia|beve

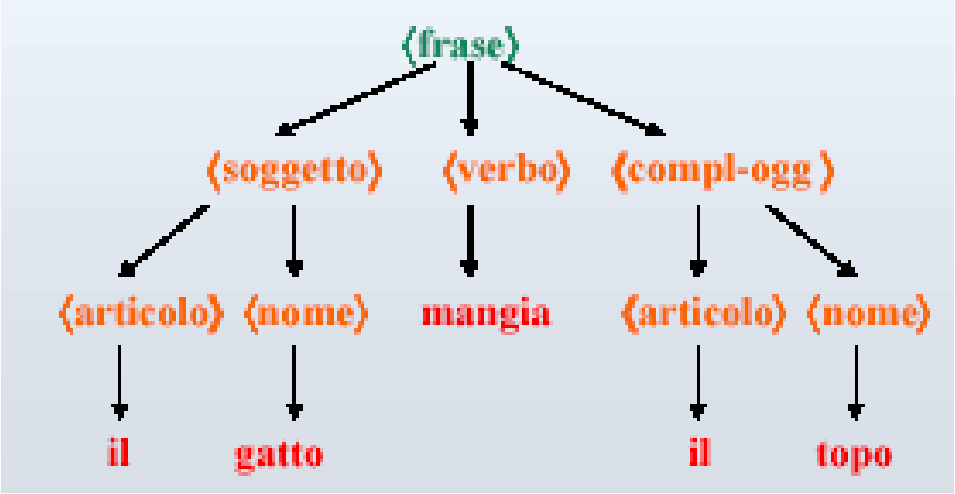
  <compl-ogg>::=<articolo><nome>

  }
- *es*.: "*il gatto mangia il topo*" è una frase del linguaggio L

Backus Naur Form

Albero di derivazione

- E' usato per comprendere come si deriva una frase
  - radice, lo scopo
  - foglie, simboli terminali
  - nodi, simboli non terminali
  - Le frasi del linguaggio si leggono nelle foglie, da sx a dx

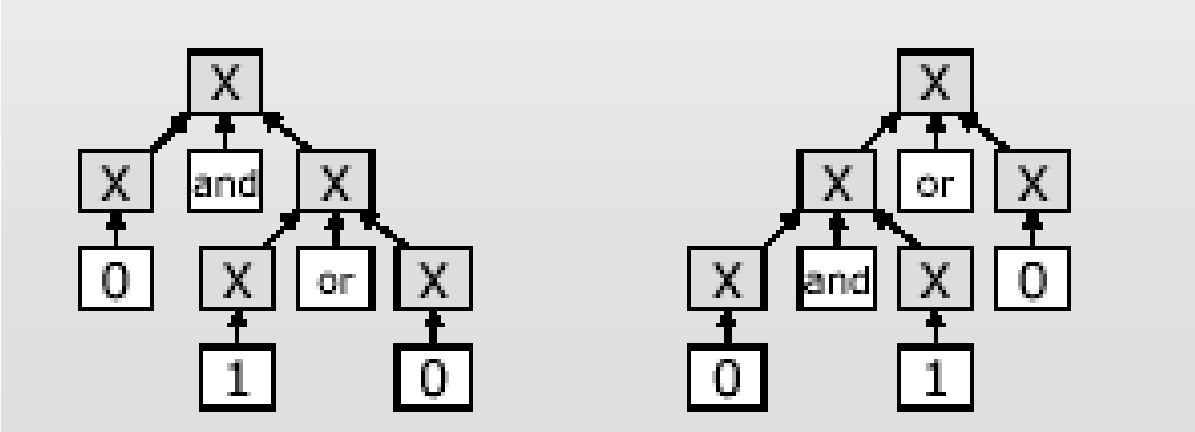


Backus Naur Form

Esempio: espressioni

- A={0,1,and,or,not}
- X ::= X and X | X or X | not X | 0 | 1

➢ Quale derivazione per: 0 and 1 or 0 ?



Per una frase potrebbero esistere due alberi
➔ grammatica *ambigua*!

Backus Naur Form

Uso di grammatiche “a precedenza”

- Un modo per garantire che ogni stringa del linguaggio sia interpretata in modo univoco è quello di introdurre una gerarchia di priorità tra gli operatori.
  - Ad esempio, seguendo la convenzione standard, possiamo stabilire che in una espressione il * ha maggiore priorità del +.
- **E’ possibile formalizzare questa regola di precedenza introducendo altri non terminali e strutturando le produzioni in base alle priorità.**
- Questa tecnica complica notevolmente la grammatica
- Nella pratica si usa spesso distinguere due grammatiche per un L. di P., una ambigua accompagnata da regole di precedenza definite a parte ed una non ambigua equivalente che viene utilizzata unicamente in fase di parsing.

Backus Naur Form

Esempio - Le espressioni

- Esempio di sintassi per i numeri interi e le principali operazioni:
  - es.: 3+4*(6/2), (3-4)*21
- I numeri:

  <cifra-non-nulla> ::= 1|2|3|4|5|6|7|8|9

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

  <num>::= 0 | <cifra-non-nulla>{<cifra>}
- Le espressioni

  <exp>::=<num>

  <exp>::=<exp><op><exp>

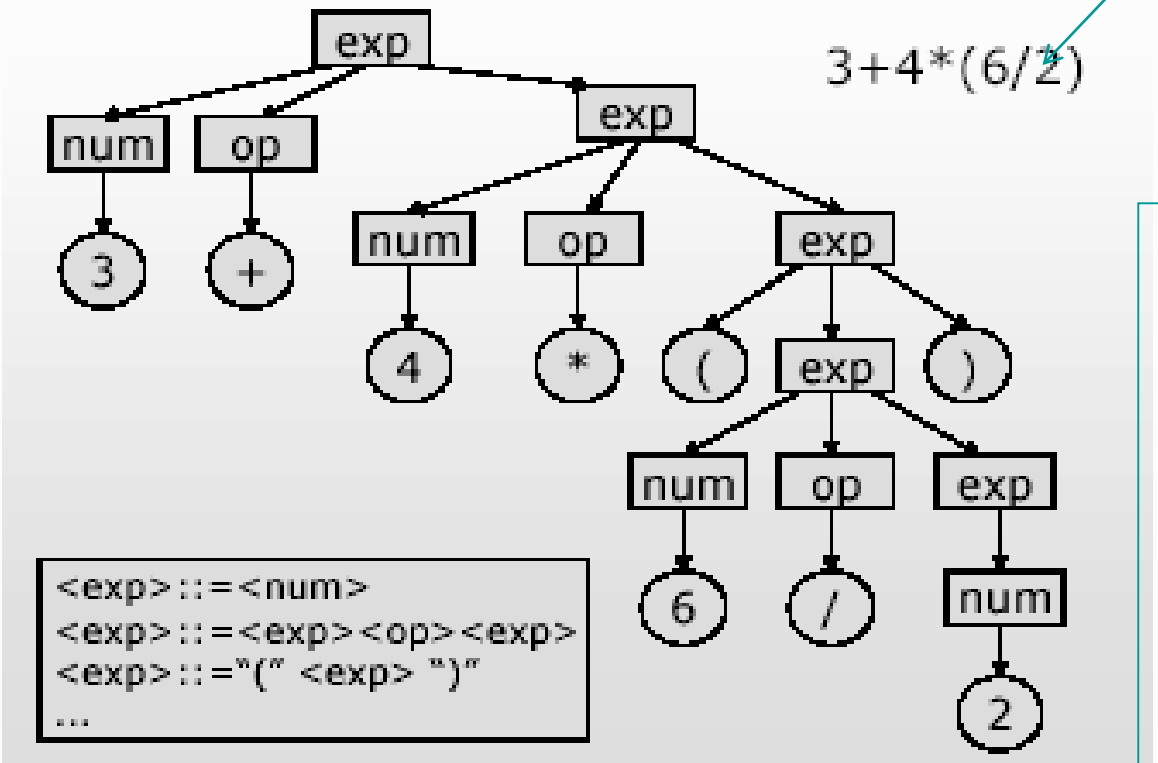
  <exp>::="("<exp>")"

  <op>::=+|-|*|/

  *, / prioritari

Specifiche dei Linguaggi di Programmazione

Albero di Derivazione



Oss. Spesso ci troviamo a dover stabilire se una frase fa parte di un linguaggio

Oss2: A volte è difficile capire se una grammatica è ambigua. Esistono linguaggi che sono inerentemente ambigui, cioè ogni grammatica che li genera è ambigua. *Problema per i parser automatici*

Extended Backus Naur Form (EBNF)

- Sono usate per avere ancora più espressività
  - Maggiore leggibilità e sinteticità
- Usano dei simboli aggiuntivi rispetto alle BNF
  - permettono di racchiudere forme di frase tra parentesi tonde, graffe o quadrate
  - Non sono del tutto standard, spesso qualcuno potrebbe usare simboli diversi...
  - ***Un dialetto EBNF è utilizzato dal W3C per definire la sintassi XML***

Descrizione	Con la sola BNF[3]	Con l'EBNF (1)[4]	Con l'EBNF (2)	Con l'EBNF (3)
Simbolo o gruppo di simboli opzionale	ε|a[5]	(a)?	{a}?	[a]
Simbolo o gruppo di simboli ripetuto una o più volte	A::=a|Aa	(a)+	{a}+	{a}
Simbolo o gruppo di simboli ripetuto zero o più volte	A::=ε|Aa	(a)*	{a}*	{[a]}

L’approccio generativista ai linguaggi (naturali e non)

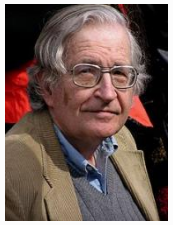
- Da dove viene?

Linguaggi formali

La teoria generativista

di Noam Chomsky

(Filadelfia, 7-12-1928)



E' un linguista, filosofo, storico, teorico della comunicazione e anarchico statunitense.
Professore emerito di linguistica al Massachusetts Institute of Technology, è riconosciuto come il fondatore della grammatica generativo-trasformazionale, spesso indicata come il più rilevante contributo alla linguistica teorica del XX secolo.
https://it.wikipedia.org/wiki/Noam_Chomsky

Il linguaggio secondo Chomsky
http://www.unife.it/letterefilosofia/comunicazione/insegnamenti/fondamenti_comunicazione_musicale/materiale_didattico/comunicazione-musicale-2013-2014-materiali-per-la-prima-parte/breve-riassunto-della-teoria-chomskyana-del-linguaggio

Linguaggi formali

La teoria generativista

- Fino oltre gli anni '50 era dominante negli Stati Uniti l'approccio della psicologia comportamentista:
- *il comportamento viene descritto in termini di stimoli e reazioni*
  - *Il bambino impara un linguaggio solo sulla base dellastimolo dei genitori*
  - *Non spiega come mai sembra che i bambini al di sotto dei 12 anni imparino lingue nuove più in fretta degli adulti*

http://www.funzioniobiettivo.it/glossadid/Caviglia/linguistica%20generativa.htm

Linguaggi formali

La teoria generativista

- La chiave all'individuazione dei processi mentali, compresi quelli preposti al linguaggio, va cercata
  - nella sperimentazione di laboratorio,
  - la mente viene considerata una scatola nera la cui struttura interna è invisibile.

- Chomsky fu tra i primi a rovesciare questa impostazione e a focalizzare lo studio sulle ***strutture mentali alla base delle abilità linguistiche***.

http://www.funzioniobiettivo.it/glossadid/Caviglia/linguistica%20generativa.htm

Linguaggi formali

La teoria generativista

- Suddivise i campi di studio della linguistica in ***competence***, la conoscenza innata delle strutture linguistiche posseduta dal parlante, e ***performance***, l'atto linguistico concreto.
- La linguistica generativa si è occupata prevalentemente della ***competence*** nell'ottica di produrre una descrizione astratta e formalizzata della conoscenza che il parlante ha della propria lingua.
- L'attività di formalizzazione considera il linguaggio come un oggetto matematico:
  - la matematica è un linguaggio astratto le cui espressioni possono essere vere o false sulla base di un insieme di assiomi e regole stabilito a priori;
  - in questo senso in algebra l'espressione x=3 può essere vera, mentre non lo è mai 2=3.

http://www.funzioniobiettivo.it/glossadid/Caviglia/linguistica%20generativa.htm

Teoria generativista (anni ’50)

- Secondo Chomsky esiste una sintassi comune a tutte le lingue naturali che viene adattata con pochi parametri per diventare una lingua concreta
- quindi la ***sintassi è la parte veramente importante nella lingua***
- ovvero le regole con cui produrre tutte le frasi di una lingua naturale attraverso la manipolazione di simboli, e chiamò questo meccanismo ***grammatica generativa***
  - L’applicazione di queste regole consente sia di generare frasi del linguaggio, sia di riconoscere la struttura di una frase e verificare se è corretta.

Oss: consente di applicare regole formali per costruire o riconoscere frasi del linguaggio, con applicazioni vastissime.

Teoria generativista (anni ’50)

- *Questa teoria ha contribuito a creare i linguaggi formali alla base dei linguaggi di programmazione*
  - 1957 , “Syntactic Structures” - Chomsky
  - per la prima volta, si propone un approccio formale alla codifica delle strutture sintattiche
  - il metodo rivoluziona completamente il trattamento dei linguaggi usando regole che consentono di generare frasi complesse.
  - identifica quindi i requisiti formali delle macchine calcolatrici in grado di riconoscere i linguaggi a vari livelli di complessità.
  - Da qui in poi sarà possibile parlare di linguaggi formali, contrapposti ai linguaggi naturali, ovvero le lingue comunemente usate dagli esseri umani

Negli anni ‘60 Chomsky ipotizzò che il cervello disponga di strutture innate espressamente dedicate al processo di codifica e decodifica del linguaggio. Questa ipotesi verrà confutata negli anni a venire nella linguistica non computazionale

Classificazione (o gerarchia)di Chomsky dei Linguaggi

- *Classificazione basata sulla complessità delle produzioni delle grammatiche che li generano*.

- Se si impongono delle restrizioni sulla forma delle regole di riscrittura, allora certe "mosse" non divengono più possibili.
- Ne consegue un sistema formale meno potente ma non per questo meno interessante ed importante dal punto di vista informatico.

Classificazione (o gerarchia)di Chomsky dei Linguaggi

- I linguaggi di tipo 0 sono generati da grammatiche a struttura di frase (non limitate)
- I linguaggi di tipo 1 sono generati da grammatiche dipendenti dal contesto (context sensitive: CS)
- I linguaggi di tipo 2 sono generati da grammatiche libere dal contesto(context free: CF)
- I linguaggi di tipo 3 sono generati da grammatiche lineari (o regolari).

➢ Oss: *restrizioni sono sulle produzioni basate sulla* ***sintassi***

Classificazione di Chomsky dei Linguaggi

- ***Tipo 0*** *grammatiche a struttura di frase*
  - sono le grammatiche in cui non viene posto alcun vincolo sulle produzioni
- ***Tipo 1*** *grammatiche dipendenti dal contesto*
  - è possibile sostituire al non terminale A il simbolo (terminale o no) γ solo quando A si trova nel contesto di α alla propria sinistra e β a destra
- ***Tipo 2*** *grammatiche libere del contesto (o algebriche)*
  - è sempre possibile sostituire al non terminale A il simbolo (terminale o non terminale) β
  - Sono di questo tipo le grammatiche che generano i linguaggi di programmazione

Classificazione di Chomsky dei Linguaggi

- ***Tipo 3:*** *grammatiche lineari (o non-counting)*
  - sono le grammatiche le cui produzioni hanno la forma A →β, dove A simbolo non terminale, β simbolo terminale o no, ma β contiene al più un simbolo non terminale.
  - Le grammatiche lineari si distinguono in
    - ***lineari destre o regolari***, quando le produzioni sono di uno dei due tipi seguenti
      - A→aB, dove A, B simboli non terminali, a terminale
      - A→ a
    - ***Lineari sinistre***, quando le produzioni sono di uno dei due tipi seguenti
      - A→Ba, dove A, B simboli non terminali, a terminale
      - A→ a

Classificazione di Chomsky dei Linguaggi

***Gerarchia delle grammatiche*:**

- Ogni grammatica di tipo *n* è anche una grammatica di tipo *n-1*

- Un linguaggio generato da una grammatica di tipo 3 è anche generabile da una di tipo 2,1,0
- Un linguaggio generato da una grammatica di tipo 2 è anche generabile da una di tipo 1,0

Classificazione di Chomsky dei Linguaggi

- Un linguaggio formale può essere quindi più o meno complesso, ed essere quindi computazionalmente più o meno esigente.

+

COMPLESSITA'

-

GERARCHIA DI CHOMSKY

Type 0 Grammars - Unrestricted

$\alpha \rightarrow \beta$, $|\alpha| \geq 0$ and $|\beta| \geq 1$.

Type 1 Grammars - Context-Sensitive

$\alpha \rightarrow \beta$, $|\alpha| \geq 1$ and $|\beta| \geq 1$ and $|\alpha| \leq |\beta|$

Type 2 Grammars - Context-Free

$\alpha \rightarrow \beta$, $|\alpha| = 1$ and $|\beta| \geq 1$

Type 3 Grammars - Regular

- left-linear regular grammar: ($A \rightarrow \alpha$ or $A \rightarrow B\alpha$)
- right-linear regular grammar: ($A \rightarrow \alpha$ or $A \rightarrow \alpha B$)

+

POTERE GENERATIVO

-

Linguaggi di tipo 0

- Sono i linguaggi generati da grammatiche di tipo 0 (cioè sono tutti i linguaggi generabili con grammatiche di Chomsky) con produzione

$\alpha \rightarrow \beta$, $\alpha \in V^* \circ VN \circ V^*$, $\beta \in V^*$ con $V = VT \cup VN$

- Le grammatiche di tipo 0 ammettono l'uso di produzioni che hanno il lato destro più corto del lato sinistro (in particolare possono anche avere il lato destro vuoto) e quindi possono dar luogo a derivazioni che accorciano stringhe.

Linguaggi di tipo 1

- Le grammatiche di Chomsky di tipo 1 (context-sensitive) sono basate su produzioni del tipo:

$\alpha \rightarrow \beta$, $\alpha \in V^* \circ V N \circ V^*$, $\beta \in V^+$

con $|\alpha| \leq |\beta|$

- Le produzioni di tipo 1 non riducono la lunghezza delle forme di frase

- ***Es*** Il linguaggio $\{a^n b^n c^n \mid n \geq 1\}$ è di tipo 1, infatti può essere generato dalla grammatica con le produzioni:

$S \rightarrow aSBC \mid aBC \quad CB \rightarrow BC \quad aB \rightarrow ab \quad bB \rightarrow bb \quad bC \rightarrow bc \quad cC \rightarrow cc$

- che soddisfano il vincolo che il lato destro è almeno lungo quanto il lato sinistro.

Linguaggi di tipo 2

- Le grammatiche di Chomsky di tipo 2 (context-free) sono basate su produzioni del tipo:

$A \rightarrow \beta$ in cui $A \in V_N$, $\beta \in V_T \cup V_N$

- una grammatica context-free è una grammatica formale in cui ogni regola sintattica è espressa sotto forma di derivazione di un simbolo a sinistra a partire da uno o più simboli a destra.
  - Una derivazione in questo tipo di grammatiche è rappresentabile con un albero di derivazione

- *Es.* Il linguaggio $\{a^n b^n \mid n \geq 0\}$ è di tipo 2 in quanto può essere generato dalla grammatica con le produzioni:

$$S \rightarrow aSb \mid ab$$

Il nome «context free» che può trarre in inganno sull'ordine della classificazione risale alla formulazione originaria di Chomsky sulle regole di produzioni

Linguaggi di tipo 2

- Es. Grammatica Context-Free che genera il linguaggio delle parentesi ben bilanciate:
  - **B →() | BB | (B)**
- Dare una derivazione per la stringa di parentesi:
  - **(()((()()))).**

B->BB->()B->()(B)->()((B))->()((B))->()((BB)) ->()((()(B)) ->()((()()))

Linguaggi di tipo 3

- Le grammatiche di Chomsky di tipo 3 (regolari) sono basate su produzioni del tipo:

$A \rightarrow aB$ oppure $A \rightarrow a$,

con $A, B \in V_N$, $a \in V_T$

- *Es* Il linguaggio $\{a^n b \mid n \geq 0\}$ è di tipo 3 in quanto si può generare con la grammatica con le produzioni:

$S \rightarrow aS$

$S \rightarrow b$

Linguaggi di tipo 3

- Le grammatiche regolari di tipo 3 sono lineari destre (LD).
- Lineari perchè nel lato destro di ogni produzione compare al più un solo metasimbolo; destre perchè il metasimbolo compare a destra dell'unico simbolo terminale
- Possono essere formalizzate anche da espressioni regolari
  - Che leggono da destra a sinistra
- Si possono anche definire grammatiche di tipo 3 Lineari Sinistre (LS) con produzioni del tipo:

  $A \rightarrow Ba$

  oppure

  $A \rightarrow a$

  con $a \in VT$, $A, B \in VN$

Classificazione di Chomsky dei Linguaggi

Riconoscibilità

- Il problema di stabilire se una stringa è frase di un linguaggio
  - è decidibile (risolubile in tempo finito) solo per le grammatiche di tipo 1
  - non è dedicibile, in generale, per le grammatiche di tipo 0

- Per i linguaggi di programmazione, si usano in genere *grammatiche di tipo 2*
  - hanno algoritmi di riconoscimento efficienti

Classificazione di Chomsky dei Linguaggi

- E per i linguaggi naturali?
- Quanto sono complessi?

  dipende da *quale* linguaggio naturale

  - **un livello alto nella gerarchia vuol dire che il linguaggio naturale è strutturalmente complesso (il più complesso è ovvio di *Tipo 0*)**

quindi un linguaggio mediamente complesso

Italiano:	Context-Free	*Tipo 2*
Inglese:	Context-Free	*Tipo 2*
Olandese:	Context-Sensitive	*Tipo 1*

(Huybregt,1976)

Classificazione di Chomsky dei Linguaggi

Linguaggi naturali di tipo 2:

- Molte frasi in linguaggio naturale hanno una struttura sintattica non contestuale

- ***Es:*** una frase è costituita da un soggetto seguito da un complemento; un soggetto è un articolo seguito da un sostantivo; il complemento è un verbo o un verbo seguito da un complemento oggetto.

→ FRASE →SOGGETTO COMPLEMENTO.

→ SOGGETTO → ART SOST

→ COMPLEMENTO → VERBO | VERBO COMP-OGGETTO

→ COMP-OGGETTO → ART SOST

ART → il SOST → gatto | topo VERBO → mangia
il gatto mangia
il gatto mangia il topo

Grammatiche generative: esercizi

Esercizio si consideri la seguente grammatica G non contestuale

- $V_T = \{a\}$, $V_N = \{S, A\}$, S = assioma **(scopo)**

- produzioni

  (1) $S \rightarrow AA$

  (2) $A \rightarrow AAA$ (4) $A \rightarrow a$

svolgere ciascuno dei seguenti punti:

a) mostrare due diverse derivazioni per "aaaaa" e due per "aaaa"

b) qual'è il linguaggio generato da G ?

c) esiste una grammatica regolare che genera lo stesso linguaggio?

http://cabibbo.dia.uniroma3.it/it/pdf/it_02_grammatiche_formali.pdf

Grammatiche generative: esercizi

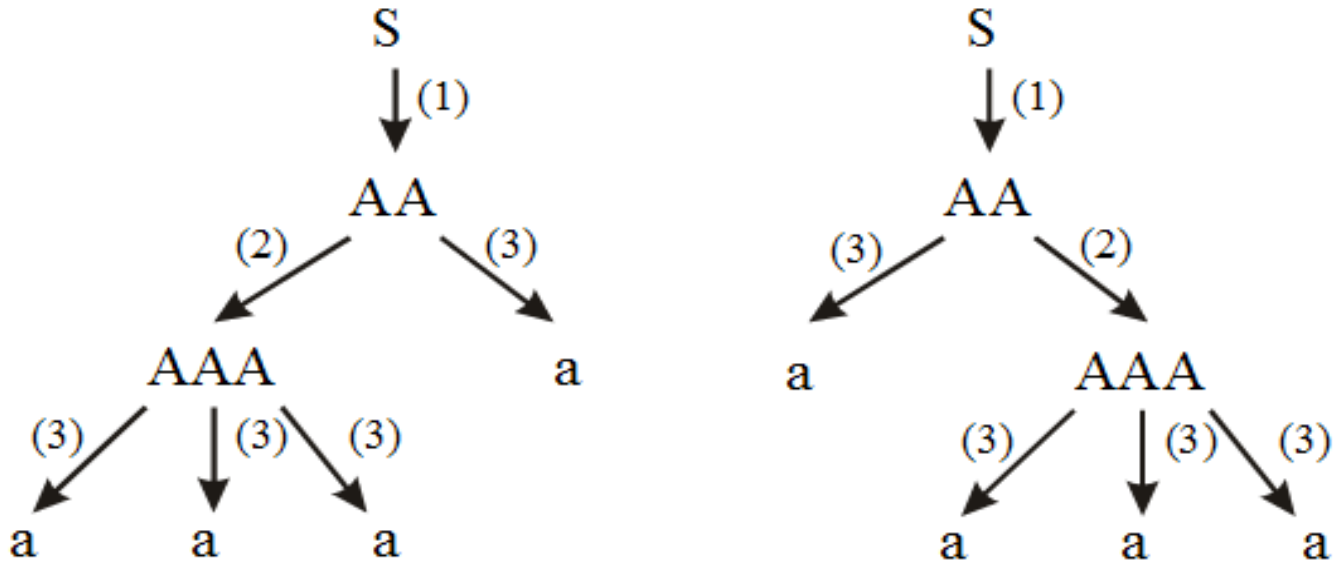
Soluzione:

a) - non esistono derivazioni per "aaaaa"

- derivazioni per "aaaa"

$\underline{S} \Rightarrow^1 \underline{A}A \Rightarrow^2 \underline{A}AAA \Rightarrow^3 a\underline{A}AA \Rightarrow^3 aa\underline{A}A \Rightarrow^3 aaa\underline{A} \Rightarrow^3 aaaa$

$\underline{S} \Rightarrow^1 \underline{A}A \Rightarrow^3 a\underline{A} \Rightarrow^2 aA\underline{A}A \Rightarrow^3 a\underline{A}Aa \Rightarrow^3 aa\underline{A}a \Rightarrow^3 aaaa$



Grammatiche generative: esercizi

Soluzione:

b) l'insieme delle stringhe su {a} di lunghezza non nulla e con un numero pari di 'a'

c) una grammatica regolare che genera lo stesso linguaggio è la seguente:

- $V_T = \{a\}$, $V_N = \{S, A, X\}$, S = assioma

- produzioni

  (1) $S \rightarrow aA$

  (2) $A \rightarrow a$ (3) $A \rightarrow aX$

  (4) $X \rightarrow aA$

Grammatiche generative: esercizi

Esercizio si consideri la seguente grammatica *G*

- $V_T = \{a, b, c\}$, $V_N = \{S, X\}$, S = assioma
- produzioni

(1) $S \rightarrow X$ (2) $S \rightarrow \varepsilon$

(3) $X \rightarrow aXa$ (4) $X \rightarrow bXb$ (5) $X \rightarrow c$

Grammatica contestuale -tipo 1
Ipotesi: ε -produzioni solo sullo scopo S e con lo stesso mai a destra

svolgere ciascuno dei seguenti punti:

mostrare alcune stringhe generate dalla grammatica

qual'è il linguaggio generato da *G* ?

Grammatiche generative: esercizi

Soluzione:

b) “c”, “aca”, “abcba”, “bcb”, “babaabcbaabab”, ...

esempio: derivazione di “abcba”:

$S \Rightarrow^1 X \Rightarrow^3 aXa \Rightarrow^4 abXba \Rightarrow^5 abcba$

c) il linguaggio delle stringhe palindrome su {a, b, c} con una ed una sola ‘c’ al centro, più la stringa vuota

Specifiche dei Linguaggi di Programmazione

- **Vocabolario**: L'insieme delle *parole* o *simboli* che possono essere utilizzate per formare frasi del linguaggio
- Noto il vocabolario, non è detto che tutte le frasi composte con parole del vocabolario siano corrette;
  - a questo punto infatti intervengono le regole grammaticali, cioè la sintassi.
- Sarà poi necessario conoscere la ***semantica*** cioè il significato delle frasi del linguaggio
- Infine, per utilizzare correttamente un linguaggio occorre conoscerne anche la ***pragmatica*** cioè *come* usare il linguaggio
  - ad esempio quali frasi è opportuno usare a seconda del contesto

Specifiche dei Linguaggi di Programmazione

- In genere la specifica dell'insieme dei programmi del Linguaggio è data in due fasi:
  - A) si definisce, attraverso una grammatica libera da contesto, un soprainsieme dei programmi del Linguaggio,
  - B) all'interno di questo insieme si definiscono alcuni vincoli contestuali che solo i programmi corretti o ben formati verificano.
    - Servono per una prima fase semantica
    - Possibile grazie all'approccio formale ai linguaggi di Chomsky, per questo padre dei linguaggi di programmazione

Specifiche dei Linguaggi di Programmazione

- Allora dividiamo in
  - **sintassi** del linguaggio
    - la descrizione fornita nella prima fase (ossia le regole della grammatica)
  - **semantica statica** del linguaggio
    - la descrizione dei vincoli contestuali fornita nella seconda fase in quanto si descrivono proprietà dei programmi osservabili staticamente, cioè prima dell'esecuzione
  - **semantica dinamica** del linguaggio
    - la descrizione dell'effetto dell'esecuzione

Specifiche dei Linguaggi di Programmazione

Commenti:

1. Un programma non ha difficoltà ad usare questo tipo di regole formali per generare frasi.
2. Analogamente un programma potrebbe analizzare una frase per ricostruire la sua struttura e verificare se essa è tra quelle ammesse dalla grammatica.

- Un ***parser*** è un programma che ha questa funzioni di analizzatore sintattico
- Operazioni di parser sono comuni nei compilatori e negli interpreti

Specifiche dei Linguaggi di Programmazione

Commenti

- Esistono tre principali tipi di descrizione di un L. di P., di cui solo i primi due forniti in genere per tutti i linguaggi:
  - I *tutorial*, che cercano di presentare efficacemente le varie caratteristiche del linguaggio anche attraverso esempi.
  - I *reference manual*, tipicamente organizzati sulla sintassi, che contengono la sintassi formale data tramite BNF più una descrizione a parole il più possibile precisa e non ambigua della semantica.
  - Le *definizioni formali*, in genere indirizzate agli specialisti.

Linguaggi formali

- **La traduzione**
  - Comprensione automatica di una frase di un linguaggio

La traduzione

- Un traduttore è un programma che effettua la traduzione automatica da un linguaggio ad un altro

- L'automazione di tale processo richiede la conoscenza della grammatica formale che definisce il linguaggio di programmazione:
  - data una frase *f*1 nel linguaggio formale *L*1 (*linguaggio sorgente*), il traduttore costruisce una frase *f2* del linguaggio formale *L2* (*linguaggio destinazione*) la frase *f2* deve "corrispondere" a *f*1

La traduzione

- La traduzione automatica dipende sia dalla sintassi che dalla semantica dei linguaggi sorgente e destinazione

- La teoria dei linguaggi formali si occupa di definire la *sintassi* di un linguaggio, ma non la sua *semantica*

  - stabilisce quali stringhe appartengono al linguaggio e quali stringhe non appartengono

- A volte è opportuno distinguere il *lessico* dalla sintassi complessiva del linguaggio

La traduzione

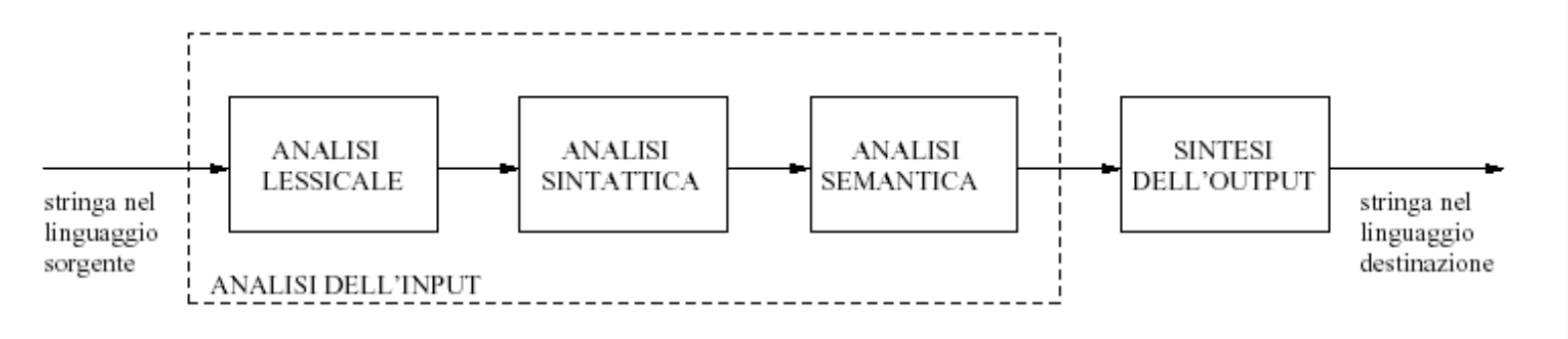
- ***Lessico*** elementi lessicali di un linguaggio <=> le parole ammesse dal linguaggio
  - esempio: lessico del linguaggio Java = tutti gli identificatori, le costanti intere, le costanti reali, le parole chiave, stringhe tra doppi apici, ecc.
  - le parole ammesse sono chiamate *token*
    - Specifica dei token lessicali avviene tramite l'uso delle ***espressioni regolari***
  - con questa distinzione, il lessico stabilisce le regole di costruzione dei token (ognuno dei quali è una sequenza di caratteri), mentre la restante parte della sintassi stabilisce le regole di costruzione delle frasi del linguaggio (ogni frase è una sequenza di token)

Struttura di un traduttore

Due fasi:

- *analisi dell'input*,
  - analisi lessicale
  - analisi sintattica
  - analisi semantica
- *sintesi dell'output*

- L'analisi *lessicale* e sintattica dipendono *soltanto* dalla sintassi del linguaggio sorgente
- L'analisi *semantica* dipende sia dalla sintassi che dalla semantica del linguaggio sorgente
- La *sintesi dell'output* dipende sia da sintassi e semantica del linguaggio sorgente che da quelle del linguaggio destinazione



Analisi lessicale (scanner)

- Si riconoscono gli elementi del linguaggio:
  - Identificatori delle variabili e delle costanti, parole chiave del linguaggio, operatori, commenti
- Verifica se sono rispettate le regole di aggregazione dei caratteri dell'alfabeto in simboli del linguaggio
  - Il sorgente è convertito da una sequenza non strutturata di caratteri in una sequenza di token
  - Es: **printf** è una parola chiave **rintf** non ha senso

Analisi lessicale (scanner)

- in genere si utilizzano dei programmi chiamati ***generatori di analizzatori lessicali*** che, a partire dalla specifica formale del lessico, generano automaticamente l'analizzatore lessicale corrispondente

Analisi sintattica (parser)

- Esamina la struttura sintattica del programma, cioèlo riconosce come stringa del linguaggio di programmazione
- Individua una sequenza di regole di produzione la cui applicazione permette di generare il programma.
- Riconosce le stringhe del linguaggio, considerando come simboli dell'alfabeto terminale, i token individuati prima
- Individua eventuali errori sintattici.
- Costruisce un albero sintattico che rappresenta la struttura del programma.

Analisi sintattica (parser)

- In genere si utilizzano dei programmi (***generatori di analizzatori sintattici***) che, a partire dalla specifica della sintassi, generano automaticamente l'analizzatore sintattico corrispondente.

Analisi semantica e sintesi dell'output

- Data la complessità della definizione della semantica di un linguaggio, le fasi di analisi semantica dell'input e di sintesi dell'output *non* possono essere automatizzate completamente, come invece avviene per le fasi di analisi lessicale e analisi sintattica
  - *pertanto, i moduli che realizzano le fasi di analisi semantica e sintesi dell'output devono essere realizzati manualmente (non esistono strumenti di generazione automatica)*

Analisi semantica e sintesi dell'output

- ***Analisi semantica***: Effettua controlli ed assegna una interpretazione alle frasi individuate nel precedente livello di analisi
  - Controllo di compatibilità di tipi ecc.

- ***Sintesi dell'output*** può essere *guidata dalla sintassi*, cioè la decomposizione del processo di traduzione delle frasi del linguaggio sorgente sulla base della struttura sintattica di tali frasi

Analisi semantica e sintesi dell'output

- Per la sintesi dell'output è possibile estendere i formalismi di specifica della sintassi al fine di catturare alcuni aspetti "semantici"
- in particolare, vengono utilizzate le *grammatiche ad attributi*, che estendono le grammatiche non contestuali mediante *azioni semantiche*, collegate alle regole di produzione della grammatica
- tramite le azioni semantiche, è possibile specificare la fase di traduzione in parallelo alla specifica della sintassi del linguaggio sorgente

E per i linguaggi naturali?

Cioè per il Natural Language Processing (ovvero Linguistica Computazionale)

Natural Language Processing (NLP)

- Le origini della linguistica computazionale si perdono nella preistoria dei programmi di traduzione automatica.
- Costruire macchine in grado di tradurre automaticamente un testo di una lingua in un'altra lingua e' un'ambizione che data perlomeno da Cartesio nel XVII, che propose un dizionario "meccanico" nel quale ogni termine della lingua venisse fatto corrispondere a un codice numerico.
- Dizionari di questo genere vennero effettivamente pubblicati da Cave Becker nel 1657, da Johann Becher nel 1661 e da Athanasius Kircher nel 1663.
- Nel 1668 John Wilkins estese queste idee alla definizione di una vera e propria "interlingua" basata sull'enumerazione di tutti i concetti elementari.

Natural Language Processing (NLP)

- Nel 1933 che divenne possibile progettare la macchina capace di elaborare codici di questo genere: il "***Cervello Meccanico" di Georges Artsrouni***, propulso da un motore elettrico, era in grado di reperire su un nastro la stringa di caratteri corrispondente alla stringa di caratteri introdotta su una tastiera.
- Un memorandum di Warren Weaver, che si basava sulla teoria dell'informazione di Shannon e sui primi usi del computer per l'analisi criptografica, apri' ufficialmente nel **1949 l'era della traduzione automatica**.
  - fatta parola per parola
- **Nel 1961** Mortimer Taube riassumeva il diffuso scetticismo sulle capacita' della macchina dichiarando fallimentari tutte le ricerche in corso.

Natural Language Processing (NLP)

- L'avvento dell'Intelligenza Artificiale diede invece nuovo impulso al campo, spostando l'attenzione verso il fenomeno primario:
  - ***la comprensione del linguaggio naturale da parte di una macchina.***
- Nacquero cosi' i primi programmi: SIR, di Bertram Raphael (1964), capace di comprendere frasi che esprimono relazioni logiche; STUDENT (1965) di Daniel Bobrow, che risolveva problemi di algebra elementare formulati in inglese; ELIZA di Joseph Weizenbaum (1966), in grado di tenere una vera e propria conversazione con l'interlocutore umano.

Natural Language Processing (NLP)

Problema complesso

- Complessità del problema dovuto a:

1. Livello di ambiguità presente nei linguaggi naturali
2. Complessità della informazione semantica contenuta anche in semplici frasi
   - «Una vecchia legge la regola.»

Qual è il soggetto? e il verbo?

La frase allude:

-a un'anziana signora che scorre con gli occhi il testo di una norma

-oppure di una «fattispecie», indicata nella frase dal pronome *la*, regolamentata da un'antica legge?

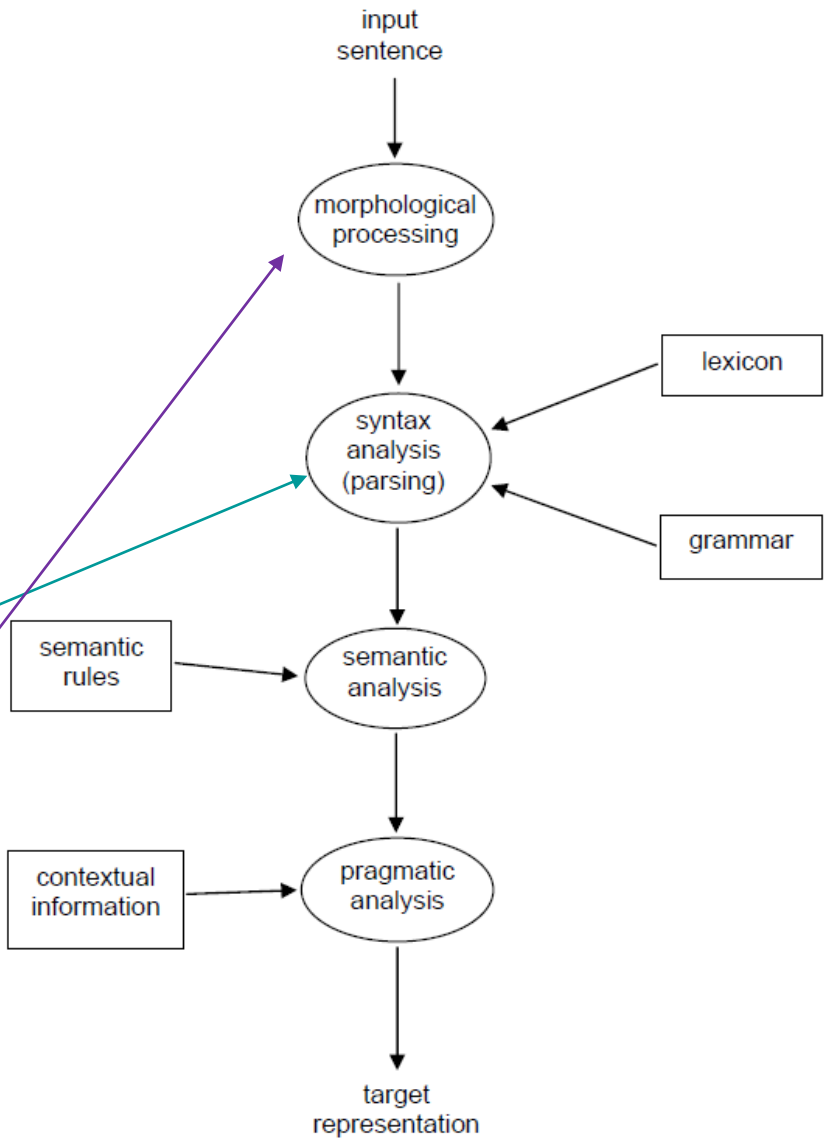
-posso definire una qualche regola di precedenza che vale in ogni caso?

I passi logici in Natural Language Processing

Raramente nei sistemi reali queste fasi sono svolte in modo strettamente separato e sequenziale

La **sintassi** italiana e quella inglese sembrano essere Context-Free e per eseguire questa parte si usano le stesse tecniche implementate per la traduzione di L.d.P.

La **morfologia**, invece, sembra essere ancora più semplice: può essere infatti rappresentata da *Grammatiche Regolari*

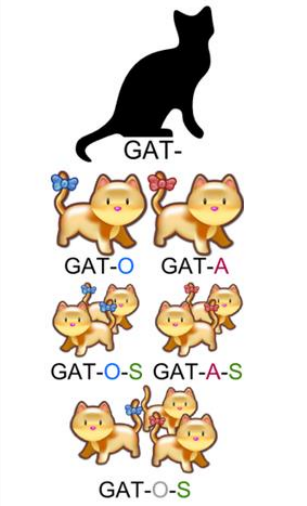


Analisi Morfologica

studio di come le parole sono costruite a partire da unità atomiche dette morfemi

definizioni

La morfologia può essere divisa in due parti principali

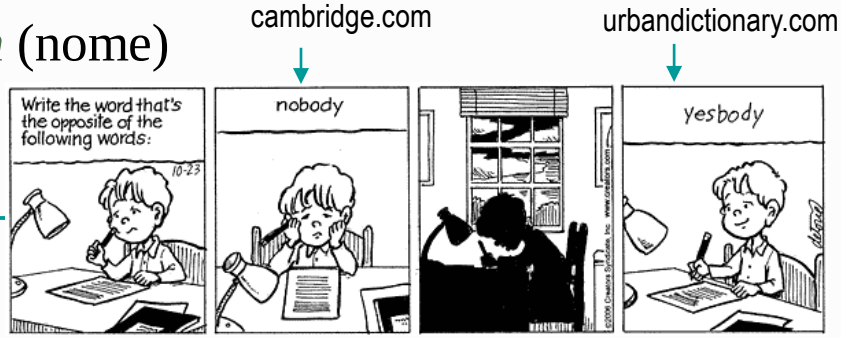


1. ***Inflectional Morphology*** combinazione di una radice con un affisso che risulta in una parola (*forma flessa*) della *stessa classe* (nome, verbo, aggettivo, ecc..) con una funzione grammaticale specifica

   - *cat* (nome sing) → *cat-**s*** (nome plur)
   - *cut* (verbo base) → *cut-**ting*** (verbo progressivo)

2. ***Derivational Morphology*** combinazione di una radice con un affisso che risulta in una parola di una *classe diversa.* Il significato della nuova parola non è facilmente prevedibile

   - *trasporto* (nome) → *trasport-**abile*** (aggettivo)
   - *computerize* (verbo) → *computeriz-ation* (nome)



Analisi Morfologica e POS (Part Of Speech)Tagging

relazione con la morfologia ?

ANALISI MORFOLOGICA

- Data una parola, trovare le sue interpretazioni morfologiche
- Può essere presente ambiguità:

  ES: talks → talk+s → talk V 3PS

  → talk N PL

POS TAGGING

- Data una parola, trovare ***la sua unica*** interpretazione morfologica
- Analisi morfologica + disambiguazione
- → metodi dell'analisi morfologica + algoritmi di disambiguazione

Analisi Morfologica e POS Tagging

Il POS tagging dovrebbe assegnare una classe ad ogni parola di un documento

- Molte parole sono **ambigue** (quinidi con più POS tag possibili)
- Un *POS tagger* deve *disambiguare*, restituendo se possibile un solo tag:
  - Utilizzando evidenze contestuali
  - Utilizzando evidenze probabilisitiche da corpora annotati

Esempio per la parola ***back***

–The ***back/JJ*** door

–On my ***back/NN***

–Promised to ***back/VB***

Analisi Morfologica e POS Tagging

Tagging: ambiguità

Quanto sono ambigue le parole inglesi ?

Dal Brown Corpus (De Rose, 1988)

2 tags	3,760
3 tags	264
4 tags	61
5 tags	12
6 tags	2
7 tags	1

NON AMBIGUE (1 tag): **35,340** (***88,5%***)

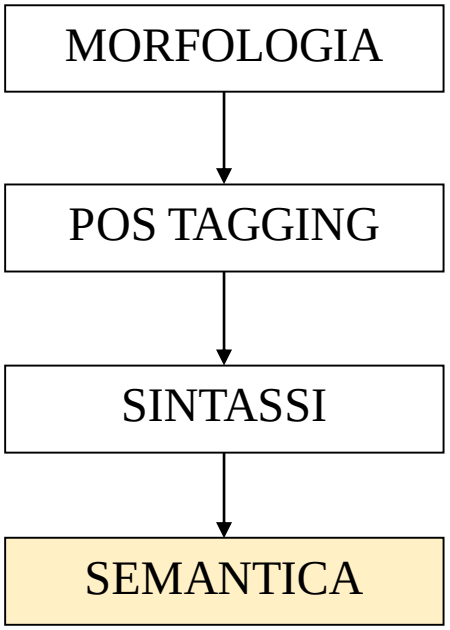
AMBIGUE (2-7 tag): **4,100** (***11,5%***)

Brown Corpus

- E’ il primo corpus elettronico di inglese “taggato” con i POS
- Il Brown University Standard Corpus of Present-Day American English è stato compilato negli anni sessanta presso la Brown University.
- Contiene 500 estratti di testo in lingua inglese ottenuti da lavori pubblicati negli Stati Uniti nel 1961, per un totale di circa un milione di parole.
- L'inserimento originale dei dati fu effettuato utilizzando macchine a schede perforate.

Analisi semantica

- *Capire il significato di una frase* serve poi per inserirla correttamente nel contesto del discorso
- *Questo compito è eseguito dall'analisi semantica* che si basa sulla rappresentazione formale del significato della frase.



Analisi semantica

- Data la ricchezza espressiva del linguaggio naturale, si possono avere più espressioni linguistiche (frasi diverse) che esprimono lo stesso fatto e di conseguenza si avranno rappresentazioni formali diverse per lo stesso concetto
  - Esempio:

Il calcio è uno sport popolare in Italia

Gli italiani amano il calcio

Il calcio è molto seguito in Italia

- E per capire il significato di una frase si deve ricorrere a processi di concettualizzazione (un metalivello rispetto a quello linguistico) che sono possibili solo se il linguaggio di rappresentazione del significato supporta l'uso di variabili per i concetti (termini di una ontologia)

Dato: Gli europei seguono il calcio
Domanda: Gli italiani amano il calcio?