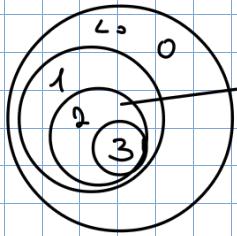


Gerarchie di Chomsky



$L = \{a^m b^n \mid m \geq 1\}$ può essere generato da grammatiche di tipo 0 perché ommette E-produzioni sia di grammatiche di tipo 2 (non contestuali) non tipo 3

L_0 è tipo 0 ma potrebbe non appartenere a tipo 1/2/3

Osservazione:

Se omelifismo grammatiche di tipo 2/3 li consentirebbe la presenza di una ^{più} E-produzione, mentre nel caso di tipo 1 è omessa una E-produzione solo sull'ominimo su S, e condizione che questo non comprema mai nelle parti destre di una produzione

Grammatica

Tipo 3!

esistente: "ogni omelismo deve fermarsi"

$$G: S \rightarrow eS \mid \& A$$

$$A \rightarrow \& A \mid \varepsilon$$

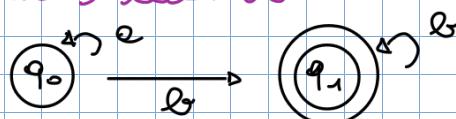
Togliamo la E-produzione per fare esercizio

$$G: S \rightarrow eS \mid \& A$$

relativa espressione regolare

$$e^* \& \&^* \text{ oppure } e^* \&^*$$

ASD relativo



CAPITOLO 6 : Linguaggi non contestuali

Ovviamente queste sono state introdotte come strumenti per definire le strutture del linguaggio naturale.

Possiamo caratterizzare le basi di un linguaggio con soggetto e predicato e quindi comporre una frase

$F \rightarrow SP$ Frase \rightarrow soggetto + predicato

$P \rightarrow V | VC$ predicho \rightarrow verbo | verbo + complemento grammaticale

Queste strutture sono sufficientemente rempliche e quindi un
alcuni contesti modellate come strumento per il linguaggio:
modulari.

g) Limpugni: pomeroti non contestuali rappresentano un settore sempre più vasto dei limpugni:

I linguaggi di programmazione supportano un sottoinsieme
proprio dei linguaggi non contestuali e li generalizziamo.
Come osservi.

Alberi di decisione o alberi sintetici

$\mathcal{S} : S \rightarrow \wp(S \cup A) \setminus \{\emptyset\}$

A → cA d l e d

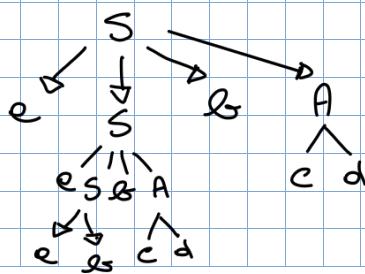
$$\rightarrow eSBe \rightarrow ee\bar{e}bA \rightarrow ee\bar{e}bbcd \in \mathcal{L}(G)$$

La grammatica introduce il linguaggio $L(G)$ celebrando $L(G)$

un esempio di albero di

derivazione di una

$$stomfe \in \mathcal{L}(G_1)$$



Queste grammatiche sono una memoria gestita e puoi
e nel caso di linguaggi di programmazione queste corrispondono
ad un'analisi sintattica "forsimp" relativa al programma di
"tradurre" le stringhe in un'altra stringa più elementare

capibile del PC

CAPITOLO 5:

Machine di Turing e calcolabilità secondo Turing

Le macchine di Turing sono il modello di calcolo di riferimento fondamentale
sia nell'ambito delle teorie di calcolabilità sia in quella
delle teorie della complessità computazionale

L'obiettivo è quello di formalizzare il concetto di calcolo
allo scopo di stabilire l'esistenza di metodi effettuativi:
per il riconoscimento dei teoremi nell'ambito dell'aritmetica

- Elenca semplici strutture
- Poteva computazionale

Le macchine di Turing sono definite come un dispositivo
operante su stringhe contenute su une memorie esterne
(mostre) mediante passi elementari definite da funz.
di transizione

Macchine di Turing e mostre simbolo

Nelle MT si è un mostro potenzialmente infinito diviso in celle contenenti ciascuna un simbolo appartenente ad un doto alfabeto Γ ampliato con il carattere speciale blank (blank) che rappresenta le situazioni di celle non contenente caratteri.

Le M^t Lavoro su un mostro, o vi è una testime che scrive
su di uno in entrambe le dicesioni.

Si può leggere o scrivere correttamente dell'alfabeto Γ oppure

The diagram shows a stack structure. A pointer variable **Q** at address **1000** points to the first node of a linked list. The list consists of four nodes, each with a value of **5**. The pointer **S** is located at address **1004**, just after the fourth node.

Ad ogni momento le HT si trovano in uno stato appartenente ad un insieme finito Q e il meccanismo che fa evolvere la computazione delle macchine è la funzione di transizione

S posta de uno stoto e de um corotore → stoto

Definizione MT

Une machine de Turing déterministe (MTD) est une sextuple $M = \langle \Gamma, \Sigma, Q, q_0, F, \delta \rangle$

Γ = alfabeto "è insieme" →

$$\overline{\Gamma} = \Gamma \cup \{ \bar{b} \}$$

5 = blank € T

Q = insieme degli stati

q_0 = stato iniziale

$F = \text{stati finiti}$ $F \subseteq Q$

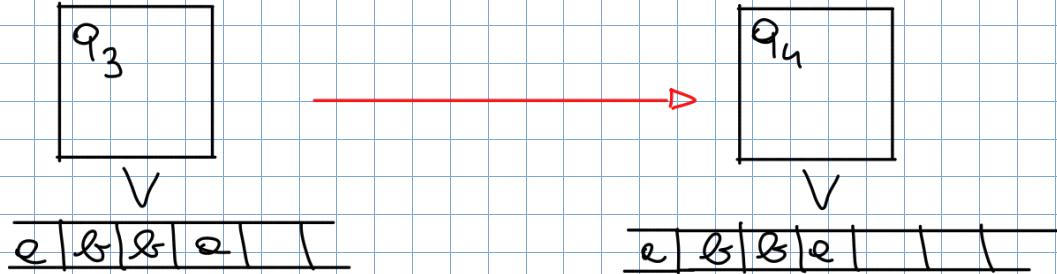
S = funzione di transizione definite così:

$\{d_{1,i}, i\}$ moltiplica lo spostamento e $d_{1,i} \neq 0$ insieme
di spostamento delle testine

$\bar{F} = F \cup \{\bar{e}\}$ per estensione
e soprattutto $\Sigma \subseteq \bar{F}$

Esempio:

$$f : (Q - F) \times (\bar{F} \cup \{\bar{e}\}) \rightarrow Q \times (\bar{F} \cup \{\bar{e}\}) \times \{d_{1,i}, i\}$$



$$(q_3, b) \mapsto (q_4, e, d)$$

$$f(q_3, b) = (q_4, e, d)$$

Le macchine usate per accettare stringhe vengono dette
recettori, mentre quelle usate per calcolare funzioni
vengono dette di tipo trasduttore

Funzione coesistenziale del linguaggio: che è definita da
 Σ^* e $\{q_1\}$ ed è ovviamente una funzione che assume
valore 1 per ogni stringa del linguaggio o soltanto 0
altrimenti.

Forme normale di Grzegorczyk

$$\text{"A} \rightarrow \lambda e" \quad A \in V_N \quad e \in V_+ \\ \lambda \in V_N^*$$