

- Semantica Operazionale - strumenti e applicazioni.

Thstroeni - Prianni ~~CEDAN~~

- Structural Operational Semantics

~~Gordon Plotkin~~

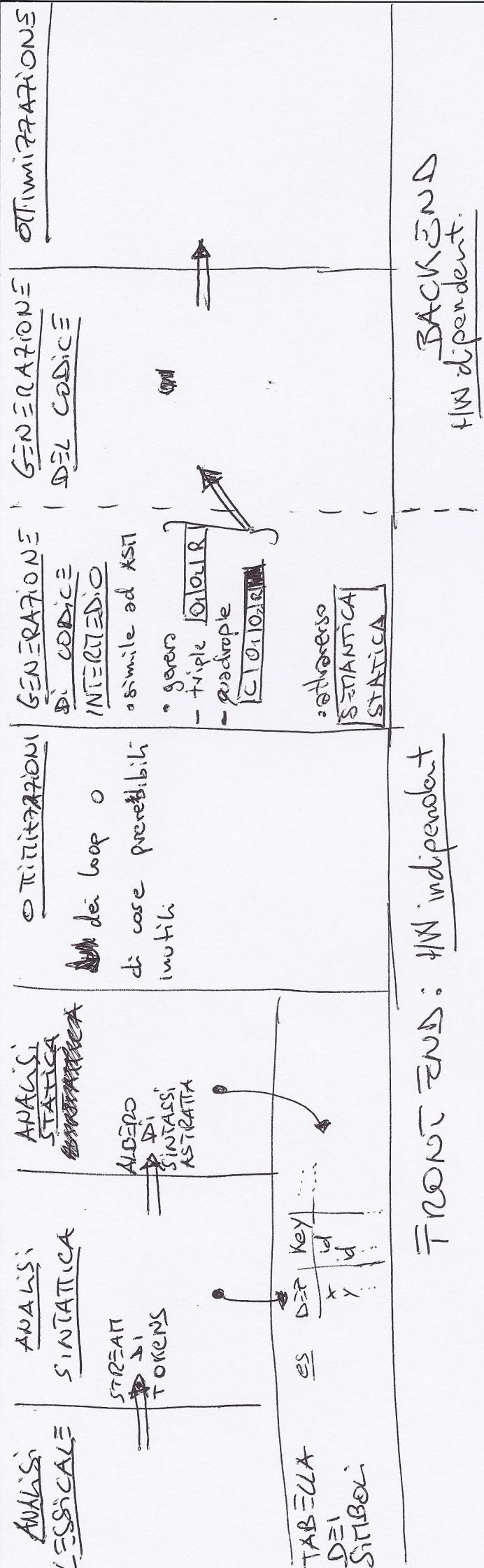
Linguaggio:

- Lessico : "caratteri" utilizzati (simboli)
- grammatica: regole di composizione dei simboli
- Semantica: associa una serie di simboli con l'oggetto "reale" (simbolo semantico)

→ Sintassi ~~f.~~ → significato ; SEMANTICA

Prototipi di linguaggi:

- ITP
- λ -calcolo
- FOL
- π -calcolo



ANALISI LESSICALE: legge il file di input e produce in output uno stream di token
ANALISI SINTATTICA: (o parsing) verifica che i token soddisfino la grammatica scelta e produce in output un albero che ha per nodi i token
 $\Sigma = (a+b)^n \Rightarrow$ viene sfoltito da informazioni
 $\Sigma = \{a, b\}^n \Rightarrow$ aggregative \Rightarrow ALBERO DI SINTASSI ASTRAITA

ANALISI SINTATTICA: vengono risolti problemi di scopi e tipo delle variabili.
TABELLA DEI SIMBOLI: dare sempre medesimate tipo e key delle variabili. I 3 passaggi:
 prima sanno quali sono questi tabella.

LINGOAGGI FORTALI

Alfabeto: Insieme finito A , non vuoto, di simboli

Stringa: sequenza finita, non vuota, di simboli di un alfabeto

Stringa vuota ϵ = zero simboli

$|S|$ = lunghezza di una stringa, n° di simboli in una stringa

$$|\epsilon| = 0$$

$$\begin{cases} S^0 = \epsilon & \text{potenza di una stringa} \\ S^n = S^{n-1}S \end{cases}$$

Prefisso di S: è ottenuto rimuovendo 0 o più simboli dalla coda di S

Suffisso di S: è ottenuto rimuovendo 0 o più simboli dalla testa di S

Substring di S: è ottenuto togliendo un prefisso o un suffisso da S

Considero questa struttura:

$$(A, \circ, e) \xleftarrow{\text{es}} \begin{array}{l} \text{MONOID} \\ (S, \circ, \epsilon) \end{array}$$

concatenazione

A : insieme di simboli

\circ : un'operazione chiusa su A : $A \times A \rightarrow A$

e : elemento neutro : ~~a op e = e op a = a~~

LINGUAGGIO: insieme \mathcal{L} di stringhe generate da un alfabeto \mathcal{A}

Linguaggi particolare: $\mathcal{L} = \emptyset$ \leftarrow linguaggio vuoto

$\mathcal{L} = \{\epsilon\}$ \leftarrow " con stringa con zero simboli

Linguaggio concatenazione: se $\mathcal{L} \in \Pi$ sono

Linguaggi \rightarrow ~~con~~ $\mathcal{L}\Pi$

$$\mathcal{L}\Pi = \{xy \mid x \in \mathcal{L}, y \in \Pi\}$$

es: $\mathcal{L} = \{\rightarrow\otimes, \otimes\otimes, -\}$

$$\Pi = \{I, II, 2\}$$

$$\mathcal{L}\Pi = \{\rightarrow QI, \rightarrow QII, \rightarrow Q2, \\ \otimes QI, \otimes QII, \otimes Q2, \\ -I, -II, -2\}$$

Quindi:

$$(\mathcal{L}, \cdot, \{\epsilon\}) \text{ è } \underline{\text{monoido}}$$

\mathcal{L} : insieme di tutti i linguaggi

\cdot : concatenazione dei linguaggi

$\{\epsilon\}$: elemento neutro dei linguaggi

STELLA DI KLEENE [$*$] operazione sui linguaggi.

$L^* = \sum_{i=0}^{\infty} L^i \rightarrow$ ~~consistente~~ ~~unione di~~
~~infinita~~ unione di L , i volte

$L^+ = \sum_{i=1}^{\infty} L^i \rightarrow$ per non contenere il linguaggio
con le stringhe vuote.

$\Rightarrow L \subseteq A^*$: un linguaggio è un sottoinsieme
di tutte le stringhe che posso
generare da un alfabeto.

PROPRIETÀ INSIEMISTICHE DEI LINGUAGGI

$$\bullet L_1 \cdot (L_2 \cup L_3) = L_1 L_2 \cup L_1 L_3$$

$$(L_2 \cup L_3) \cdot L_4 = L_2 L_4 \cup L_3 L_4$$

$$\bullet L_1 \cdot L_2 \neq L_2 \cdot L_1$$

$$\bullet L \cdot \emptyset = \emptyset L = \emptyset$$

LINGUAGGIO COMPLEMENTO

$$\bar{L} = \{ w \in A^* \mid w \notin L \}$$

Per rappresentare tutte le possibili stringhe in λ quando c'è infinito esistono 3 metodi.

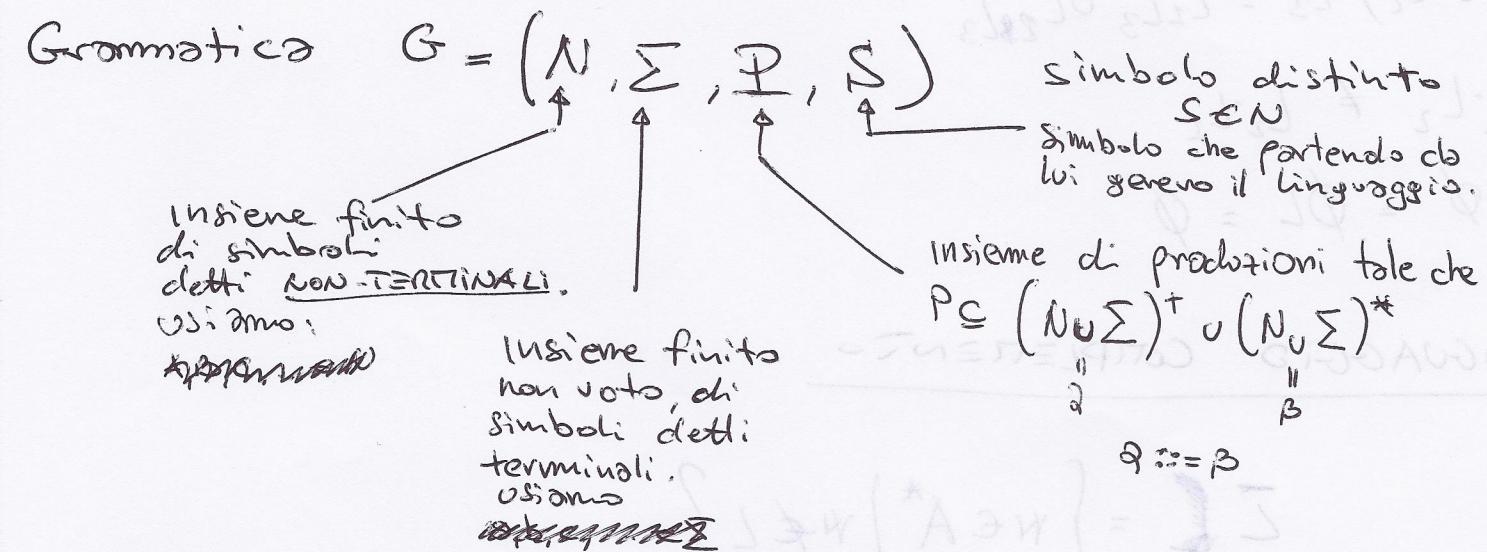
Axiomatico: Insieme di equazioni algebriche la cui soluzione è λ

Riconoscitivo: prende una stringa s e ~~usa~~ un automa a stati finiti

Generativo: è dato da una grammatica (struttura finita) che ha regole per generare tutte le stringhe di λ



a partire da una grammatica si può sempre costruire un parser, ma non il viceversa.



$L(G)$: linguaggio generato da una grammatica $G \subseteq \Sigma^*$

NOTAZIONI

$$A, B, C, D, \dots \in N$$

$$a, b, c, d, \dots \in \Sigma$$

$$\dots, x, y, z \in (N \cup \Sigma)$$

$$\dots, x, y, z \in \Sigma^*$$

$$\alpha, \beta, \gamma, \dots \in (N \cup \Sigma)^*$$

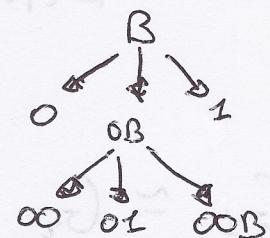
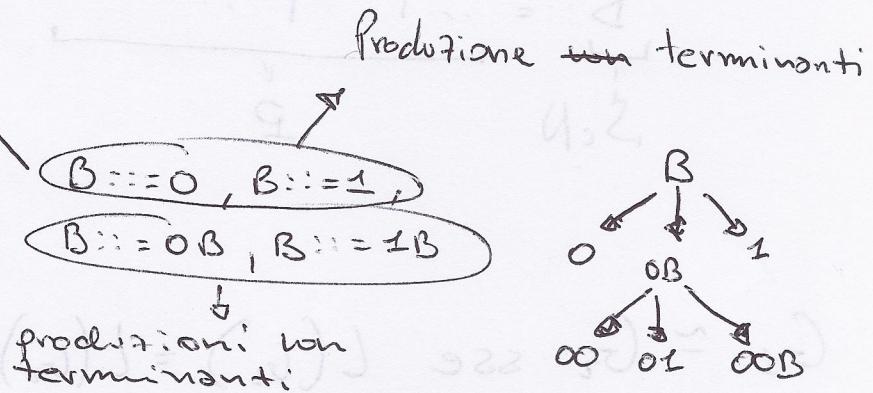
Usiamo questi simboli per gli insiemi considerati.

ESERCIZIO: Linguaggio dei numeri Binari.

$$L = \{0, 1, 00, 01, \dots\}$$

Piamo a definire G per L :

$$G_B = (\Sigma, N, P, S)$$



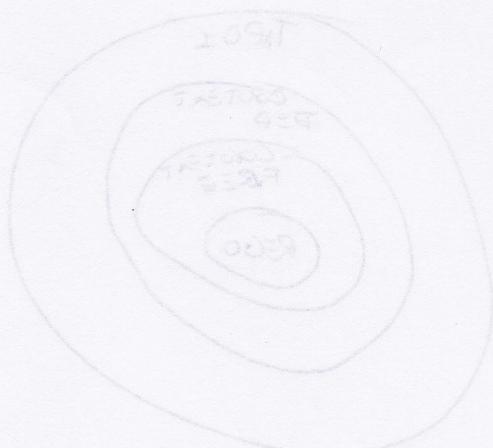
DERIVAZIONE

Suppongo: $f \in (N \cup \Sigma)^*$, $g \in (N \cup \Sigma)^*$, $\alpha ::= \beta$

Si dice che f è derivato derivativo immediato di g
vuole dire che posso trasformare f in g usando una
derivazione della grammatica

f derivato da g se

$$g \xrightarrow{*} f$$



$$L(G) = \{ w \in \Sigma^* \mid S \xrightarrow{*} w \}$$

BNF

modo diverso per rappresentare la grammatica:

Si parte dal simbolo distinto:

$$\begin{aligned} B &::= 0 \sqcup 1 \sqcup 0B \sqcup 1B \sqcup 0A \\ A &::= \dots \sqcup \dots \sqcup \dots \\ D &::= \dots \sqcup \dots \sqcup \dots \end{aligned} \quad \left. \begin{array}{l} \text{Viesce a ricostruire} \\ \text{la grammatica} \end{array} \right\}$$

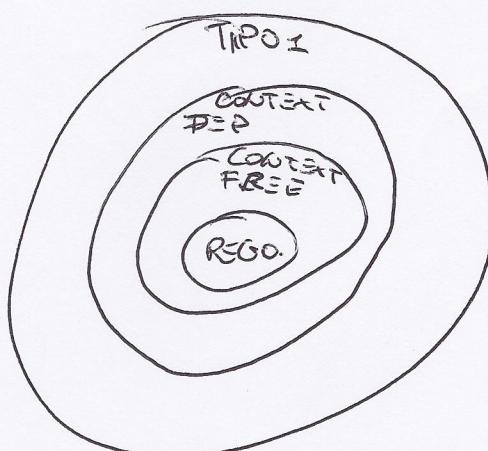
\sum

$S \in N \qquad P$

$$G_1 \approx G_2 \text{ sse } L(G_1) = L(G_2)$$

GERARCHIA DEI LINGUAGGI SECONDO IL POTERE ESPRESSIVO
DEI LINGUAGGI

- TIPO I : $\alpha \in (N \cup \Sigma)^+$, $\beta \in (N \cup \Sigma)^*$
- DIPENDENTI DAL CONTESTO : $\alpha = \gamma A \delta$, $\beta = \gamma \epsilon \delta$, $(\gamma, \delta) \in (N \cup \Sigma)^*$, $\epsilon \in (N \cup \Sigma)^+$
- LIBERI DAL CONTESTO : $\alpha \in N$, $\beta \in (N \cup \Sigma)^+$
- REGOLARI : $\alpha \in N$, $\beta \in \Sigma$, $\beta = aB$



DERIVAZIONI CANONICHE SINISTRE

Sostituisco ~~il~~ sempre il non terminale più a sinistra.
Lo ha una regola deterministica per generare le stringhe.

G è ambigua se esistono almeno 2 derivazioni canoniche sinistre della stessa stringa $L(G)$

$L(G)$ è ambiguo se tutte le grammatiche che lo generano sono ambigue

25/02/14

$$G = (N, \Sigma, P, S)$$

$$L(G) = \{w \in \Sigma^* \mid S \xrightarrow{*} w\}$$

BNF (Context free)



$$E ::= \underbrace{E+E \mid E*E \mid (E) \mid e}_{\text{elenco delle produzioni}} \Rightarrow \Sigma = \{+, *, (,), e\}$$

~~TERMINALE~~
~~NON TERMINALE~~

simboli distinti
 simboli non terminali.
 $N = \{E\}$

DERIVAZIONI CANONICHE SINISTRE

Si sostituisce il non terminale più a sinistra non terminale.

$e+e*e \rightarrow$ appartiene al linguaggio generato dalla grammatica G ?

$$E \rightarrow E+E \rightarrow \cancel{E} + \cancel{E} \rightarrow e + \cancel{E}*E \rightarrow e + e * E \rightarrow \underbrace{e + e * e}$$

N.B.: ma non è l'unico modo per generare la stringa perché al posto di $E+E$ potrei scrivere $\cancel{E}*\cancel{E}$

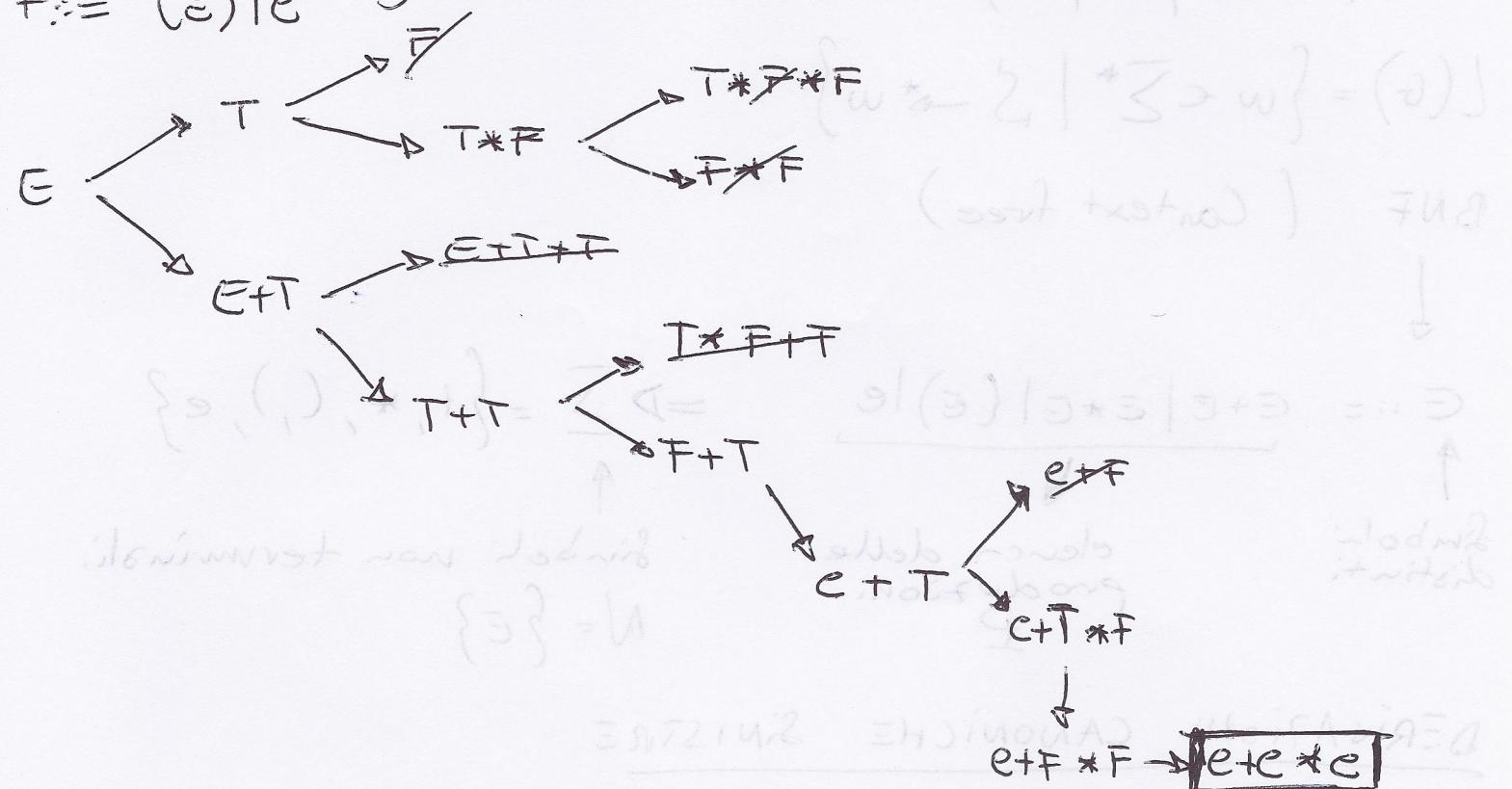
$$\cancel{E}*\cancel{E} \rightarrow E+G*E$$

finisco quando trova la stringa e appartiene a Σ^*

G è ambigua se esiste $\exists s \in L(G)$ e due derivazioni canoniche sinistre che generano s

$L(\text{linguaggio})$ è ambiguo se tutte le grammatiche G che lo generano sono ambigue.

$$\begin{array}{l} E := E + T \mid T \\ T := T * F \mid F \\ F := (E) \mid e \end{array} \quad \Rightarrow \text{NON } \equiv \text{ AMBIGUOUS.}$$

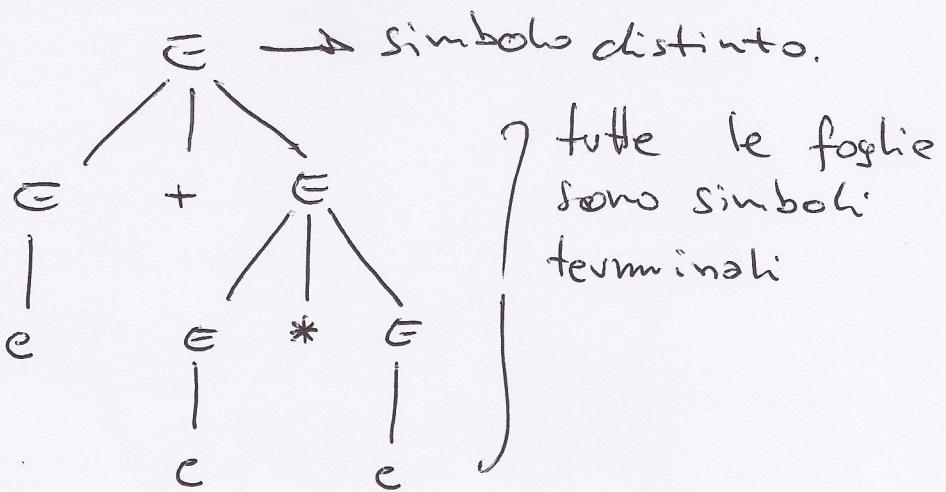


Det som hänt här är att det finns två olika sätt att tolka uttrycket. Den ena tolkningen är att man först räknar ihop 2 * 9 * 3 = 54 och sedan gör 54 - 2 = 52. Den andra tolkningen är att man först gör 2 - 2 = 0 och sedan räknar ihop 0 * 9 * 3 = 0.

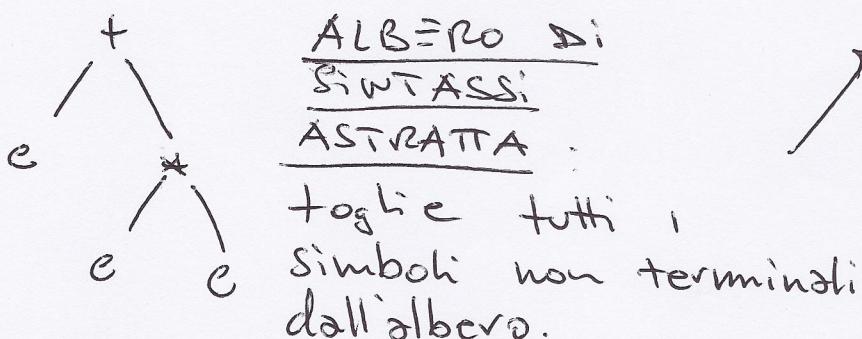
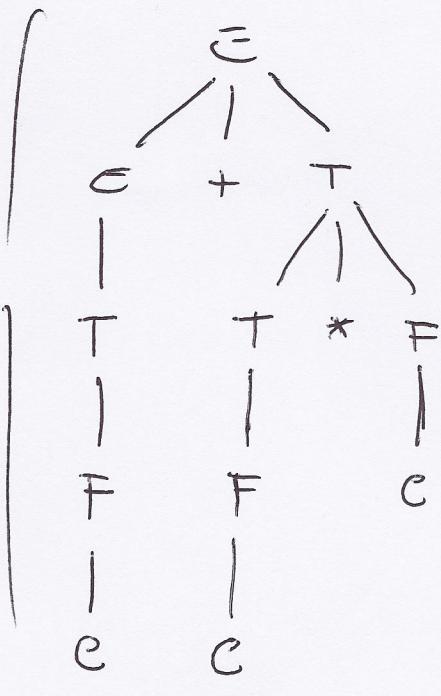
Detta visar att det finns två olika sätt att tolka uttrycket, vilket är det som gör det ambivalent.

RAPPRESENTAZIONE DIVERSA DELLA DERIVAZIONE

ALBERI DI DERIVAZIONE



ma con questa rappresentazione non ho la priorità degli operatori



da qui si definisce la SEMANTICA

Δ , \leq (relazione) $\subseteq A \times A$

binaria

insieme

Δ è ben fondata se non esistono catene così:
 $\dots a_1 \Delta \dots \Delta a_2 \Delta a_1 \Delta a_0$

(A, Δ) è ben fondata se Δ è ben fondata.

es

$(\mathbb{N}, \text{pred})$ è ben fondata

$(\mathbb{Z}, \text{pred})$ non è ben fondata

$a \Delta a = \text{riflessiva}$
 $a_0 \Delta a_1, a_1 \Delta a_2 \Rightarrow a_0 \Delta a_2$
riflessiva

~~Δ~~ Δ è una chiusura riflessiva, transitiva, ben fondata
su A

LEMMA: Δ non è mai ben fondata

DIM: $a_i \in A$ posso costruire

$\dots \Delta a_1 \Delta a_2 \Delta \dots \Delta a_i \rightarrow$ non è ben fondata

DEF: $a \in A$, ~~che non ha elementi~~ a è minima in A
rispetto a Δ se ~~non~~ a non appartengono a A

CERTIT: Δ è ben fondata su A se ogni $B \subseteq A$ ha
un elemento minima rispetto a Δ

DIM: $b_1 \Delta \dots \Delta b_n \Delta \dots \Delta b_0$

→ deve esistere un elemento che chiude la catena
perché è ben fondata.

$\forall b < b_n, b \in B$

↑ minima di B rispetto a Δ

PRINCIPIO DI INDUZIONE NOETHERIANA

P proprietà che voglio dimostrare su (A, \triangleleft) ben fondato.

$\forall a \in A$ vale $P(a) \Leftrightarrow \forall a \in A$ vale $\left(\left[\forall b \triangleleft a \text{ vale } P(b) \right] \Rightarrow P(a) \right)$

$\frac{\text{Hyp induttiva}}{\text{Passo induttivo.}}$

DIM: ~~per assurdo~~

(\Leftarrow Per assurdo.)

$\forall a \in A$ vale $\left(\left[\forall b \triangleleft a \text{ vale } P(b) \right] \Rightarrow P(a) \right)$

$\wedge \exists c \in A$ tale che $\neg P(c)$

$$C = \{c \in A \mid \neg P(c)\} \subseteq A$$

$\exists a \in C$ tale che $\forall b \triangleleft a$ vale $P(b)$

stabilire

$A \cap C = \emptyset$ (che è impossibile)

$A \neq \emptyset$ e $C \neq \emptyset$ (che è impossibile)

$\Rightarrow A \cap C = \emptyset$ (che è impossibile)

contradizione

contradizione

$\Rightarrow A \cap C = \emptyset$

$\Rightarrow \neg \exists c \in A \mid \neg P(c)$

PRINCIPIO DI INDUZIONE GENERALIZZATO

(A, Δ) $\forall a \in A : P(a) \Leftrightarrow \forall a \in \text{Base}_\Delta : P(a)$,
 $\forall a \in \text{BASE}_\Delta : ([\forall b < a : P(b)] \Rightarrow P(a))$
 $(A - \text{BASE}_\Delta)$

Da questo principio si possono definire funzioni.

(A, Δ) $\delta^A : \{a' \in A \mid a' \in \Delta\}$

$\forall a \in A, \forall h : \delta^A \rightarrow B \mid \forall (a, b) \in B \Rightarrow$

$\exists ! f : A \rightarrow B \mid \forall a \in A \mid f(a) = F(a, f|_{\delta^A})$

dove F = operatore di composizione.

CS

$$\text{Fact}(0) = 1$$

$$\text{Fact}(n) = n \cdot \text{Fact}(n-1) = \bullet(n, \text{Fact}|_{\delta^n})$$

OP di composizione

$$(w)g + (x)h \leq [(w)g + (x)h] \wedge$$

$$(w)g + (x)h \geq [(w)g + (x)h]$$

Considero questa grammatica:

$$G = (N, \Sigma, P, S) \quad \text{con} \quad \Delta \subseteq \overbrace{(N \cup \Sigma)^* \times (N \cup \Sigma)^*}^{\text{strings non vuote}} \quad \text{+}$$

$$\forall A ::= \underbrace{x_1 \dots x_n}_{\text{seguente di } n \text{ simboli}} \in P \quad \text{tc.} \quad \underbrace{x_1 < x_2 \dots x_n}_{\text{preso un qualunque simbolo della stringa, non posso generare una serie infinita decrescente}} \quad \textcircled{1}$$

con la seguente grammatica:

$$B ::= 0 \mid 1 \mid B0 \mid B1 \rightarrow \text{applico la relazione } \textcircled{1} :$$

$$0 < B, 1 < B, 0 < B0, 1 < B1, B < B0, B < B1$$

Voglio dimostrare una proprietà di $L(B)$:

$\forall w \in L(B) \quad \text{tc.} \quad w' = w0, w' \in L(B) \Rightarrow$ rappresenta un numero pari.

$$B ::= 0$$

$$B ::= 1$$

$$B ::= B0 \quad \begin{matrix} \text{non rientrano} \\ \text{nella mia} \end{matrix}$$

$$B ::= B1 \quad \leftarrow \text{Proprietà}$$

$B ::= 0 \rightarrow$ caso base perché non posso scomporre la parte dx della produzione

$$\textcircled{1} \quad 0 < B0$$

$$B < B0$$

$$\textcircled{2} \quad B ::= \frac{x_1 x_2}{B0}$$

$$\left(\left[\bigwedge_{i=1}^z \forall w \in L(x_i) \quad \text{tc.} \quad P(w) \right] \Rightarrow \forall w \in L(B0) \quad \text{tc.} \quad P(w) \right)$$

$$\left(\left[\forall w \in L(B) \quad \text{tc.} \quad P(w) \right] \Rightarrow \forall w \in L(B0) \quad \text{tc.} \quad P(w) \right)$$

$A_1 := \beta_1^1 | \dots | \beta_n^1$
 \vdots
 $A_m := \beta_1^m | \dots | \beta_n^m$

Prendo tutti gli elementi
 che hanno un costituente immediato [?]

$G = (N, \Sigma, P, S)$ $\alpha := \beta, \beta \in N, \beta(N \cup \Sigma)^*$

$\exists! g: L(G) \rightarrow B$ t.c. $\forall A := x_1 \dots x_n \in P$ t.c.
 $g(x_1 \dots x_n) = F(x_1, \dots, x_n, g(x_1), \dots, g(x_n))$

$\epsilon: L(G_{\text{bin}}) \rightarrow N$: semantica che associa a ogni linguaggio binario ai numeri naturali

$$\epsilon[0] = F(0) = 0$$

$$\epsilon[1] = F(1) = 1$$

$$\epsilon[B0] = F(B, 0, \epsilon[0], \epsilon[0]) = \epsilon[B] \circ 0$$

$$\epsilon[B1] = F(B, 1, \epsilon[B], \epsilon[1]) = \epsilon[B] \circ 1$$

es

$$\epsilon[\underbrace{\overbrace{B}^{101}}_{B}] = \underbrace{\epsilon[\overbrace{B}^1]}_1 \cdot 2 + 1 = (\underbrace{\epsilon[1]}_1 \cdot 2) \star 2 + 1 = 2 \cdot 2 + 1 = 5$$

PRINCIPIO DI COMPOSIZIONALITÀ:

Significato (semantica) di ciascuna stringa deve essere funzione solo dei componenti della stringa.

PRINCIPIO DI MODULARITÀ:

Se aggiungo costrutti ad un linguaggio L , non devo ridefinire la semantica di L .

$$z := 2, y := z; y := y + 1; z := y \quad [z=0, y=0]$$

SINTASSI DEL PROGRAMMA

non è sintassi

$$f: \text{Var}(P) \rightarrow \text{Val} + \perp \quad \begin{array}{l} \text{funzione che associa ogni} \\ \text{vare delle variabile al} \\ \text{suo valore.} \end{array}$$

\downarrow \downarrow \rightarrow
 vare della valore della bottom
 variabile variabile (non definita)

3 tipi di semantica:

OPERAZIONALE: descrive come eseguire un programma

(A)

$$P = E + S \cdot S = E + S \cdot (S - \boxed{S}) = E + S \cdot [S] = [S] \cdot [E]$$

DENOTAZIONALE: descrive cosa otteniamo dalla esecuzione del programma.

(B)

ASSIOMATICA: descrive se il programma è corretto rispetto a qualche proprietà P

(C)

A

$\langle z := 2; y := z; y := y + 1; z := y, [z=0, y=0] \rangle \rightarrow$

$\langle y := z; y := y + 1; z := y, [z=2, y=0] \rangle \rightarrow$

$\langle y := y + 1; z := y, [z=2, y=2] \rangle \rightarrow$

$\langle z = y, [z=2, y=3] \rangle \rightarrow$

$\langle \emptyset, [z=3, y=3] \rangle$

Sì può vedere come un grafo TG (Transition System)

$TG = (\underbrace{P \times S^0}_{\text{nodi}}, \underbrace{\rightarrow}_{\text{archi}}) + \text{""} \rightarrow \text{"} \subseteq (P \times S) \times (P \times S)$

oppure

$TG = (P \times S, \rightarrow, I, F) \quad I \subseteq (P, S) \text{ iniziali}$
 $F \subseteq (P, S) \text{ finali}$

$$\textcircled{B} \quad E_{\text{DEF}} : L(G) \times \mathcal{G}_{\text{INIT.}} \rightarrow \mathcal{G}_{\text{FIN.}}$$

es

$$E_{\text{DEF}}[z := 2](z = 0) = [z = 2]$$

$$E_{\text{DEF}}[z := y] \left(E[y := x + 1] \left(E[y := z] \left(E[z := 2](z = 0) \right) \right) \right) = [y = 3, z = 3]$$

C

Axioma: per determinare una conclusione non ho bisogno di nessuna proprietà

$$\text{defin. } (q, g) \models I \quad (I, I, \vdash, q, g) \models \text{DT}$$

$$\text{defin. } (q, g) \models I$$