

Capitolo 1

Induzione strutturale

Introduzione

Sia A un insieme, sia \triangleleft una relazione binaria definita su A ($\triangleleft : A \times A$).

Def. \triangleleft è b.f. (ben fondata) se $\nexists \dots \triangleleft a_i \triangleleft \dots \triangleleft a_1 \triangleleft a_0$
ovvero non esistono catene infinite discendenti.
(A, \triangleleft) è un insieme ben fondato se \triangleleft è b.f.

Sia \trianglelefteq la chiusura riflessiva e transitiva di \triangleleft b.f. su A .

Lemma 1. \trianglelefteq non è mai b.f.

Dimostrazione. Sia $a_i \in A$. $\exists \dots a_i \trianglelefteq a_i \trianglelefteq \dots$ □

Def. Sia $a \in A$. a è *minimale* in A rispetto a \triangleleft se $\forall b \triangleleft a, b \notin A$.

Lemma 2. \triangleleft è ben fondato su A se e solo se ogni $B \subseteq A$ ha un elemento minimale rispetto a \triangleleft .

Dimostrazione. • \Rightarrow) Da $B \subseteq A$ e (A, \triangleleft) b.f. si ha che non esiste $\dots \triangleleft b_i \triangleleft \dots \triangleleft b_0$, quindi $\exists b_n$ t.c. $b_n \triangleleft \dots \triangleleft b_i \triangleleft \dots \triangleleft b_0$ ovvero b_n è minimale in B rispetto a \triangleleft .

- \Leftarrow) Per assurdo. Supponiamo che esista $\dots \triangleleft a_i \triangleleft \dots \triangleleft a_0$, ovvero $\neg(\triangleleft \text{ b.f. })$. Allora l'insieme $B = \{a_i | i \in \mathbb{N}\}$ (insieme degli elementi della sequenza) non ha un minimale, cosa che contraddice l'ipotesi. Quindi una tale sequenza non esiste, ovvero \triangleleft b.f. □

1.1 Principio di induzione noetheriana

Teorema 1 (Principio di induzione noetheriana (prima forma)). *Sia \mathcal{P} una proprietà su (A, \triangleleft) b.f. .*

$$\forall a \in A. \mathcal{P}(a) \Leftrightarrow \forall a \in A. ([\forall b \triangleleft a. \mathcal{P}(b)] \Rightarrow \mathcal{P}(a))$$

Dimostrazione. • \Rightarrow) Ovvio.

• \Leftarrow) Per assurdo.

Supponiamo

$$\forall a \in A. ([\forall b \triangleleft a. \mathcal{P}(b)] \Leftarrow \mathcal{P}(a)) \quad (1.1)$$

e

$$\exists c \in A. \neg \mathcal{P}(c) \quad (1.2)$$

Sia $C = \{c \in A | \neg \mathcal{P}(c)\} \subseteq A$.

Per il lemma 2, $\exists \hat{c}$ minimale di C rispetto a \triangleleft , allora $\neg \mathcal{P}(\hat{c})$

Per \hat{c} minimale, $\forall b \triangleleft \hat{c}. b \notin C$, allora $\mathcal{P}(b)$, ovvero per (1.1) $\mathcal{P}(\hat{c})$

□

Def. $\text{base}_A = \{a \in A | a \text{ è minimale in } A \text{ rispetto a } \triangleleft\}$.

Osserviamo che $\forall a \in \text{base}_A, \forall b \in A. b \not\triangleleft a$.

Teorema 2 (Principio di induzione noetheriana - seconda forma). *Sia \mathcal{P} una proprietà su (A, \triangleleft) b.f. .*

$$\forall a \in A. \mathcal{P}(a) \Leftrightarrow \left(\begin{array}{c} \forall a \in \text{base}_A. \mathcal{P}(a) \\ \wedge \\ \forall a \in (A \setminus \text{base}_A). ([\forall b \triangleleft a. \mathcal{P}(b)] \Rightarrow \mathcal{P}(a)) \end{array} \right)$$

Teorema 3 (Induzione matematica). *Sia $A = \mathbb{N}$.*

Sia $n \triangleleft m \Leftrightarrow m = n + 1 \quad n, m \in \mathbb{N}$.

Osserviamo che \triangleleft è ben fondata: $0 \triangleleft 1 \triangleleft 2 \triangleleft \dots$.

Osserviamo $\text{base}_{\mathbb{N}} = \{0\}$.

$$\forall m \in \mathbb{N}. \mathcal{P}(m) \Leftrightarrow \left(\begin{array}{c} \mathcal{P}(0) \\ \wedge \\ \forall m > 0. (\mathcal{P}(m-1) \Rightarrow \mathcal{P}(m)) \end{array} \right)$$

Def. $a^\triangleleft = \{b \in A | b \triangleleft a\}$ con (A, \triangleleft) b.f. .

Def. Sia $f : A \rightarrow B$, sia $A' \subseteq A$. $f|_{A'} = \{(a, f(a)) | a \in A'\}$.

Teorema 4 (di ricorsione / delle definizioni noetheriane). *Sia (A, \triangleleft) b.f.*

$$\forall a \in A, \forall h : a^\triangleleft \rightarrow B. F(a, h) \in B$$

(F è detta operatore di composizione). Allora

$$\exists! f : A \rightarrow B \text{ t.c. } \forall a \in A. f(a) = F(a, f|_{a^\triangleleft})$$

Esempio. Sia $A = \mathbb{N}$, sia $n \triangleleft m \Leftrightarrow m = n + 1$.

$$\text{Fact}(n) = \begin{cases} 1 & \text{se } n = 0 \\ n \cdot \text{Fact}(n-1) & \text{se } n > 0 \end{cases}$$

In questo caso, $f = \text{Fact}$ e F è la moltiplicazione.

$$f(n) = F(n, f(n-1)) = n \cdot f(n-1)$$

Teorema 5 (di induzione sulle grammatiche libere dal contesto). *Dato:*

$$G = \{N, \Sigma, P, S\}$$

$$\triangleleft \subseteq (N \cup \Sigma) \times (N \cup \Sigma)^+$$

$$\forall A = x_1, \dots, x_n \in \mathcal{P}, x_1 \triangleleft x_2 \triangleleft \dots \triangleleft x_n$$

Dall'ultima possiamo osservare che la lunghezza della stringa più piccola e minore della lunghezza dell'intera stringa, quindi non posso creare catene infinitamente discendenti, per questo la relazione sulle stringhe di un linguaggio si può considerare ben fondata.

Esempio.

$$B = 0|1|B0|B1$$

$$0 \triangleleft 1 \triangleleft B0 \triangleleft B1$$

Osserviamo che abbiamo espresso un ordine per gli elementi 0, 1, quest'ordine per semplicità lo possiamo prenderla dalla semantica induttiva sui numeri naturali.

Teorema 6 (Principio d'induzione sulle grammatiche libere dal contesto).

$$\forall \omega \in L(G). \mathcal{P}(\omega) \Leftrightarrow$$

$$\forall A = x_1, \dots, x_n \in P \left(\left[\bigwedge_{i=1}^n \forall \omega \in L(x_i). \mathcal{P}(\omega) \right] \Rightarrow \forall \omega \in L(x_1, \dots, x_n). \mathcal{P}(\omega) \right)$$

Per ogni produzione, le stringhe generate da ogni simbolo del lato sinistro soddisfano \mathcal{P} implica che le stringhe generate dall'intera produzione soddisfano \mathcal{P} , questo è equivalente a dire che \mathcal{P} vale per ogni stringa appartenente al linguaggio.

Esempio. Cerco di dimostrare una proprietà \mathcal{P} su $B = 0|1|B0|B1$

$$\forall \omega \in L(B); \omega = \omega 0$$

$$\omega' \in L(B) \Rightarrow \omega \text{ è un numero pari}$$

Soluzione:

Riscrittura principio d'induzione prima in maniera compatta poi in maniera esplicita

$$\left(\left[\bigwedge_{i=1}^n \forall \omega \in L(x_i). \mathcal{P}(\omega) \right] \Rightarrow \forall \omega \in L(B0). \mathcal{P}(\omega) \right)$$

$$(\forall \omega \in L(B). \mathcal{P}(\omega)) \Rightarrow \forall \omega \in L(B0). \mathcal{P}(\omega)$$

Se riesco a dimostrare che la proprietà è vera per tutte le stringhe formate da B , allora posso assumere che per tutte le stringhe che conterranno $B0$ la proprietà sarà vera. (0 è un caso base e lo diamo per vero)

Capitolo 2

Definizione di Linguaggio

TODO

Composizionalità: la proprietà per cui ciascuna stringa deve essere funzione solo dei componenti della stringa stessa.

Modularità: la proprietà per cui, aggiungendo dei costrutti ad un linguaggio L , non devo ridefinire la semantica dello stesso.

- posso quindi definire un linguaggio in maniera incrementale, ovvero, partendo da un nucleo centrale, posso aggiungere dello zucchero sintattico senza modificare il nucleo di partenza.

Capitolo 3

Semantica

- **Operazionale:** COME eseguire un programma P .
- **Denotazionale:** CHE COSA si ottiene dall'esecuzione di P .
- **Assiomatica:** verifica SE il programma é corretto rispetto ad una data proprietà.

Def. dato un programma P :

$$\dots z := 2; y := z; y := y + 1; z := y. \dots \quad (3.1)$$

il **supporto a tempo di esecuzione** é definito come:

$$[z = 0, y = 0] \equiv \rho(z) = 0, \rho(y) = 0 \quad (3.2)$$

TODO

3.1 Definizione formale

Def. Una semantica é una quadrupla:

$$(L, M, \varepsilon_{i \in I}, \equiv_m)$$

dove:

- L é il linguaggio oggetto. $l \in L$ sono le stringhe del linguaggio a cui bisogna assegnare un significato.
- M é il meta linguaggio, che definisce gli oggetti che si usano per assegnare la semantica (o il significato) ad $L \Rightarrow \forall l \in L \exists m \in M. m(l) = \text{significato}$.
- $\varepsilon_{i \in I}$ sono funzioni di interpretazione semantica $\Rightarrow \varepsilon_{i \in I} : L \rightarrow M$.
- \equiv_m é l'equivalenza semantica $\subseteq M \times M$.

Lemma 3.

$$l, l' \in L, l \equiv_l l' \iff \varepsilon[l] \equiv_m \varepsilon[l'] \quad (3.3)$$

Dimostrazione. L'equivalenza sul linguaggio esiste solo e soltanto se gli elementi del meta linguaggio ($\varepsilon[l] \in M$) sono equivalenti. \square

Lemma 4.

$$\equiv_M = \text{id} \not\equiv \equiv_L = \text{id} \quad (3.4)$$

Dimostrazione. Se l'equivalenza su M é l'identitá cio non implica la stessa proprietá per l'equivalenza sul linguaggio L (?) \square

3.2 Semantica operativa

Def. Un **sistema di transizione** T_G é un grafo t.c.

$$T_G = (P \times \rho, \longrightarrow, I, F)$$

dove

- P sono i programmi (ad es. assegnazioni di (3.1))
- ρ sono gli ambienti del supporto a tempo di esecuzione (ad es. (3.2))
- $\longrightarrow \subseteq (P, \rho) \times (P, \rho)$
- $I \subseteq (P, \rho)$ iniziali
- $F \subseteq (P, \rho)$ finali

Def. La **semantica operativa** é una funzione

$$\mathcal{E}_{op} : L(G) \times \rho_{init} \longrightarrow T_G$$

3.3 Semantica denotazionale

Def. La **semantica denotazionale** é definita come

$$\mathcal{E}_{den} : L(G) \times \rho_{init} \longrightarrow \rho_{fin}$$

Rappresenta solo il risultato dell'esecuzione del programma, ovvero l'insieme degli ambienti finali.

3.4 Semantica assiomatica

Def. Una **regola di inferenza** stabilisce che:

$$\frac{p_1 \wedge \dots \wedge p_n}{C}$$

se $p_1 \wedge \dots \wedge p_n$ sono vere allora C é vera.

Def. Ho un **assioma** se:

$$\frac{}{C}$$

ovvero se non ho proprietá su cui basare la conclusione.

Def. La **semantica assiomatica** é definita sulla base di regole di inferenza ed assiomi, e dimostra la veridicitá di un programma costruendo un albero le cui foglie sono tutte **assiomi**.