Capitolo 1

Induzione strutturale

Introduzione

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

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

Sia $\unlhd$ la chiusura riflessiva e transitiva di $\lhd$ b.f. su A.

Lemma 1. $\unlhd$ *non è mai* b.f.

Dimostrazione. Sia $a_i \in A$. $\exists \ldots a_i \unlhd a_i \unlhd \ldots$ □

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

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

Dimostrazione.
- $\Rightarrow$) Da $B \subseteq A$ e $(A, \lhd)$ b.f. si ha che non esiste $\ldots \lhd b_i \lhd \ldots \lhd b_0$, quindi $\exists b_n$ t.c. $b_n \lhd \ldots \lhd b_i \lhd \ldots \lhd b_0$ ovvero b_n è minimale in B rispetto a $\lhd$.
- $\Leftarrow$) Per assurdo. Supponiamo che esista $\ldots \lhd a_i \lhd \ldots \lhd a_0$, ovvero $\neg(\lhd$ 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 $\lhd$ b.f.

□

1.1 Principio di induzione noetheriana

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

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

Dimostrazione.

- $\Rightarrow$) Ovvia.
- $\Leftarrow$) Per assurdo.
  Supponiamo
  $$\forall a \in A.([\forall b \lhd a.\mathcal{P}(b)] \Leftarrow \mathcal{P}(a)) \tag{1.1}$$
  e
  $$\exists c \in A.\neg\mathcal{P}(c) \tag{1.2}$$
  Sia $C = \{c \in A | \neg\mathcal{P}(c)\} \subseteq A$.
  Per il lemma 2, $\exists \hat{c}$ minimale di C rispetto a $\lhd$, allora $\neg\mathcal{P}(\hat{c})$
  Per $\hat{c}$ minimale, $\forall b \lhd \hat{c}.b \notin C$, allora $\mathcal{P}(b)$, ovvero per (1.1) $\mathcal{P}(\hat{c})$

$\square$

Def. $\texttt{base}_A = \{a \in A | a$ è minimale in A rispetto a $\lhd\}$.
Osserviamo che $\forall a \in \texttt{base}_A, \forall b \in A.b \not\lhd a$.

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

$$\forall a \in A.\mathcal{P}(a) \Leftrightarrow \begin{pmatrix} \forall a \in \texttt{base}_A.\mathcal{P}(a) \\ \wedge \\ \forall a \in (A \setminus \texttt{base}_A).([\forall b \lhd a.\mathcal{P}(b)] \Rightarrow \mathcal{P}(a)) \end{pmatrix}$$

Teorema 3 (Induzione matematica)**.** *Sia* $A = \mathbb{N}$.
Sia $n \lhd m \Leftrightarrow m = n + 1 \quad n, m \in \mathbb{N}$.
Osserviamo che $\lhd$ *è ben fondata:* $0 \lhd 1 \lhd 2 \lhd \dots$.
Osserviamo $\texttt{base}_{\mathbb{N}} = \{0\}$.

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

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

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

Teorema 4 (di ricorsione / delle definizione noetheriane)**.** *Sia* $(A, \lhd)$ b.f.

$$\forall a \in A, \forall h : a^{\lhd} \to B.F(a, h) \in B$$

(F è detta operatore di composizione*). Allora*

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

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

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

In questo caso, $f = \texttt{Fact}$ 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:*

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

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

$$\forall A = x_1, ..., x_n \in \mathcal{P}, x_1 \lhd x_2 \lhd ... \lhd x_n$$

Dall'ultima possiamo osservare che la funghezza della stringa piu piccola e mionore della lunghezza dell'intera stringa, quindi non posso creare catente infinitamente discendenti, per questo la relazione sulle stringhe di un linguaggio si puo considerare ben fondata.

Esempio.

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

$$0 \lhd 1 \lhd B0 \lhd B1$$

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

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

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

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

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 proprieta $\mathcal{P}$ 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

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

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

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 $\mathcal{P}$.
- **Denotazionale:** CHE COSA si ottiene dall'esecuzione di $\mathcal{P}$.
- **Assiomatica:** verifica SE il programma $\mathcal{P}$ è corretto rispetto ad una data proprietà.

Def. dato un programma $\mathcal{P}$:

$$\ldots z := 2; y := z; y := y + 1; z := y \ldots \tag{3.1}$$

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

$$[z = 0, y = 0] \equiv \rho(z) = 0, \rho(y) = 0 \tag{3.2}$$

TODO

3.1 Definizione formale

Def. Una semantica è una quadrupla:

$$(L, M, \varepsilon_{i \in I}, \equiv_m) \tag{3.3}$$

dove:

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

Lemma 3.

$$l, l' \in L, l \equiv_l l' \Longleftrightarrow \varepsilon[l] \equiv_m \varepsilon[l'] \tag{3.4}$$

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

Lemma 4.

$$\equiv_M = \text{id} \not\Rightarrow \equiv_L = \text{id} \tag{3.5}$$

Dimostrazione. Se l'equivalenza su M è l'identità ció non implica la stessa proprietà per l'equivalenza sul linguaggio L (?) □

3.2 Semantica operazionale

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

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

dove

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

Def. La **semantica operazionale** è una funzione

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

utilizzando la definizione formale (3.3) definiamo:

- L : linguaggio oggetto (che definiamo)
- M: $< S, \longrightarrow >$ dove $S = \{< l, \rho : \text{Var}(l) \longrightarrow \text{Val} > | l \in L\}$ ovvero tutti i possibili stati di tutti i possibili programmi generati da L, e $\longrightarrow \subseteq S \times S$
- $\varepsilon_{i \in I} \equiv$ definizione operativa data a inizio paragrafo.
- $\equiv_M$ è l'identità sui grafi (isomorfismo)

3.3 Semantica denotazionale

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

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

utilizzando la definizione formale (3.3):

- L è il linguaggio che definiamo
- $M = \{\rho : \text{Var}(l) \longrightarrow \text{Val} | l \in L\}$ ovvero l'insieme delle funzioni che danno valore alle variabili di L.
- $\varepsilon_{i \in I} : L \longrightarrow M, l \longrightarrow \text{val}$
- $\equiv_m$ è l'identità

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:

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

se $p_1 \wedge \ldots \wedge p_n$ sono vere allora C è vera.

Def. Ho un **assioma** se:

$$\frac{}{C}$$

ovvero se non ho proprietà su cui basare la conclusione.

Def. La **semantica assiomatica** è definita sulla base di regole di inferenza ed assiomi, e dimostra la verdicità di un programma costruendo un albero le cui foglie sono tutte **assiomi**.
Utilizzando la definizione formale di semantica a (3.3):

- L è il linguaggio che definiamo
- M è l'insieme di alberi di derivazione per $l \in L$
- $\varepsilon_{i \in I}$ sono le interpretazioni degli assiomi e delle regole di inferenza
- $\equiv_M$ è l'identità sugli alberi di derivazione

3.5 Contesto

Def. Un **contesto** $C[\cdot]$ per un programma è un programma con un buco (una parte non specificata).

1. **Principio di Subsitutivity:**
   $l, l' \in L$ che possono comparire nello stesso contesto $C[\cdot]$:
   $$l \equiv_L l' \Rightarrow \forall C[\cdot], C[l] \equiv_L C[l']$$
2. **Principio di Full abstraction:**
   $$l, l' \in L, \forall C[\cdot], C[l] \equiv_L C[l'] \Longrightarrow l \equiv_L l'$$