Informatica Teorica

Massimo Perego

Contents

Introduzione **2**

1 Teoria della Calcolabilità **4**
1.1 Notazione 4
1.1.1 Funzioni 4
1.1.2 Prodotto Cartesiano 6
1.1.3 Funzione di Valutazione 7
1.2 Sistemi di Calcolo 7
1.3 Potenza Computazionale 8

Introduzione

Si "contrappone" all'informatica applicata, ovvero qualsiasi applicazione dell'informatica atta a raggiunger uno scopo, dove l'informatica è solamente lo strumento per raggiungere in maniera efficace un obiettivo.
Con "*informatica teorica*" l'oggetto è l'informatica stessa, si studiano i fondamenti della disciplina in modo rigoroso e scientifico. Può essere fatto ponendosi delle questioni fondamentali: il *cosa* e il *come* dell'informatica, ovvero cosa è in grado di fare l'informatica e come è in grado di farlo.

Cosa: L'informatica è "la disciplina che studia l'informazione e la sua elaborazione automatica", quindi l'oggetto sono l'informazione e i dispositivi di calcolo per gestirla; scienza dell'informazione. Diventa lo studio come risolvere automaticamente un problema. Ma tutti i problemi sono risolvibili in maniera automatica? Cosa è in grado di fare l'informatica?

La branca dell'informatica teorica che studia cosa è risolvibile si chiama **Teoria della Calcolabilità**, studia cosa è calcolabile per via automatica. Spoiler: non tutti i problemi sono risolvibili per via automatica, e non potranno mai esserlo per limiti dell'informatica stessa. Cerchiamo una caratterizzazione generale di cosa è calcolabile e cosa no, si vogliono fornire strumenti per capire ciò che è calcolabile. La caratterizzazione deve essere fatta matematicamente, in quanto il rigore e la tecnica matematica permettono di trarre conclusioni sull'informatica.

Come: Una volta individuati i problemi calcolabili, come possiamo calcolarli? Il dominio della **Teoria della Complessità** vuole descrivere le risoluzione dei problemi tramite mezzi automatici in termini di risorse computazionali necessarie. Una "risorsa computazionale" è qualsiasi cosa che viene consumata durante l'esecuzione per risolvere il problema, come pos-

sono essere elettricità o numero di processori, generalmente i parametri più importanti considerati sono tempo e spazio di memoria. Bisognerà definire in modo preciso cosa si intende con "tempo" e "spazio". Una volta fissati i parametri bisogna definire anche cosa si intende con "risolvere efficientemente" un problema, in termini di tempo e spazio.

La teoria della calcolabilità dice quali problemi sono calcolabili, la teoria della complessità dice, all'interno dei problemi calcolabili, quali sono risolvibili efficientemente.

Capitolo 1

Teoria della Calcolabilità

1.1 Notazione

1.1.1 Funzioni

Funzione: Una funzione f dall'insieme A all'insieme B è una legge che dice come associare a ogni elemento di A un elemento di B. Si scrive

$$f : A \to B$$

E chiamiamo A dominio e B codominio. Per dire come agisce su un elemento si usa $f(a) = b$, b è l'immagine di a secondo f (di conseguenza a è la controimmagine).
Per definizione di funzione, è possibile che elementi del codominio siano raggiungibili da più elementi del dominio, ma non il contrario. Possiamo classificare le funzioni in base a questa caratteristica:

- **Iniettiva:** $f : A \to B$ è iniettiva sse $\forall a, b \in A,\ a \neq b \implies f(a) \neq f(b)$
- **Suriettiva:** $f : A \to B$ è suriettiva sse $\forall b \in B,\ \exists a \in A : f(a) = b$: un altro modo per definirla è tramite l'insieme immagine di f, definito come
  $$\mathrm{Im}_f = \{b \in B : \exists a, f(a) = b\} = \{f(a) : a \in A\}$$
  Solitamente $\mathrm{Im}_f \subseteq B$, ma f è suriettiva sse $\mathrm{Im}_f = B$;
- **Biettiva:** $f : A \to B$ è biettiva sse è sia iniettiva che suriettiva, ovvero

$$\begin{array}{ll} \forall a, b \in A, a \neq b : & f(a) \neq f(b) \\ \forall b \in B, \exists a \in A : & f(a) = b \end{array} \implies \forall b \in B, \exists! a \in A : f(a) = b$$

Inversa: Per le funzioni biettive si può naturalmente associare il concetto di "inversa": dato $f : A \to B$ biettiva, si definisce inversa la funzione $f^{-1} : B \to A$ tale che $f^{-1}(b) = a \Leftrightarrow f(a) = b$.

Composizione di funzioni: Date $f : A \to B$ e $g : B \to C$, f composto g è la funzione $g \circ f : A \to C$ definita come $g \circ f(a) = g(f(a))$. Generalmente non commutativo, $f \circ g \neq g \circ f$, ma è associativo.

Funzione identità: Dato l'insieme A, la funzione identità su A è la funzione $i_A : A \to A$ tale che $i_A(a) = a$, $\forall a \in A$.

Un'altra possibile definizione per l'inversa diventa:

$$f^{-1} \circ f = i_A \wedge f \circ f^{-1} = i_B$$

Funzioni Parziali: Se una funzione $f : A \to B$ è definita per $a \in A$ si indica con $f(a) \downarrow$ e da questo proviene la categorizzazione: una funzione è **totale** se definita $\forall a \in A$, **parziale** altrimenti (definita solo per qualche elemento di A).

Insieme Dominio: Chiamiamo **dominio** (o campo di esistenza) di f l'insieme

$$\mathrm{Dom}_f = \{a \in A | f(a) \downarrow\} \subseteq A$$

Quindi se $\mathrm{Dom}_f = A$ la funzione è totale, se $\mathrm{Dom}_f \subsetneq A$ allora è una funzione parziale.

Totalizzazione: Si può **totalizzare una funzione parziale** f definendo una funzione a tratti $\overline{f} : A \to B \cup \{\perp\}$ tale che

$$\overline{f}(a) = \begin{cases} f(a) & a \in \mathrm{Dom}_f(a) \\ \perp & \text{altrimenti} \end{cases}$$

Dove $\perp$ è il **simbolo di indefinito**, per tutti i valori per cui la funzione di partenza f non è definita. Da qui in poi $B_\perp$ significa $B \cup \{\perp\}$.

Insieme delle funzioni: L'insieme di tutte le funzioni che vanno da A a B si denota con

$$B^A = \{f : A \to B\}$$

La notazione viene usata in quanto la cardinalità di B^A è esattamente $|B|^{|A|}$, con A e B insiemi finiti.
Volendo includere anche tutte le funzioni parziali:

$$B_\perp^A = \{f : A \to B_\perp\}$$

Le due definizioni coincidono, $B^A = B_\perp^A$, ma quest'ultima permette di mettere in evidenza che tutte le funzioni presenti sono totali o totalizzate.

1.1.2 Prodotto Cartesiano

Chiamiamo **prodotto cartesiano** l'insieme

$$A \times B = \{(a, b) | a \in A \wedge b \in B\}$$

Rappresenta l'insieme di tutte le coppie ordinate di valori in A e B. In generale non è commutativo, a meno che $A = B$.

Può essere esteso a n-uple di valori:

$$A_1 \times \cdots \times A_n = \{(a_1, \ldots, a_n) | a_i \in A_i\}$$

Il prodotto di n volte lo stesso insieme verrà, per comodità, indicato come

$$A \times \cdots \times A = A^n$$

Proiettore: Operazione "opposta", il proiettore i-esimo è una funzione che estrae l'i-esimo elemento di una tupla, quindi è una funzione

$$\pi_i : A_1 \times \cdots \times A_n \to A_i \text{ t.c. } \pi_i(a_1, \ldots, a_n) = a_i$$

La proiezione sull'asse in cui sono presenti i valori dell'insieme a_i.

1.1.3 Funzione di Valutazione

Dati A, B e $B_\perp^A$ si definisce **funzione di valutazione** la funzione

$$\omega : B_\perp^A \times A \to B \text{ t.c. } \omega(f, a) = f(a)$$

Prende una funzione f e la valuta su un elemento a del dominio. Si possono fare due tipi di analisi su questa funzione:

- Fisso a e provo tutte le f, ottenendo un *benchmark* di tutte le funzioni su a
- Fisso f e provo tutte le a del dominio, ottenendo il *grafico* di f

1.2 Sistemi di Calcolo

Vogliamo modellare teoricamente un **sistema di calcolo**; quest'ultimo può essere visto come una black box che prende in input un programma P, dei dati x e calcola il risultato y di P su input x. La macchina restituisce y se è riuscita a calcolare un risultato, $\perp$ (indefinito) se è entrata in un loop.

Quindi, formalmente, possiamo definire un sistema di calcolo come una funzione

$$\mathcal{C} : \text{PROG} \times \text{DATI} \to \text{DATI}_\perp$$

Possiamo vedere un sistema di calcolo come una funzione di valutazione:

- i dati x corrispondono all'input a
- il programma P corrisponde alla funzione f

Formalmente, un programma $P \in \text{PROG}$ è una sequenza di regole che trasformano un dato input in uno di output, ovvero l'espressione di una funzione secondo una sintassi

$$P : \text{DATI} \to \text{DATI}_\perp$$

e di conseguenza $P \in \text{DATI}_\perp^{\text{DATI}}$. In questo modo abbiamo mappato l'insieme PROG sull'insieme delle funzioni, il che ci permette di definire il sistema di calcolo come la funzione

$$\mathcal{C} : \text{DATI}_\perp^{\text{DATI}} \times \text{DATI} \to \text{DATI}$$

Analoga alla funzione di valutazione. Con $\mathcal{C}(P, x)$ indichiamo la funzione calcolata da P su x dal sistema di calcolo $\mathcal{C}$, che viene detta **semantica**, ovvero il suo "significato" su input x.

Il modello solitamente considerato quando si parla di calcolatori è quello di **Von Neumann**.

1.3 Potenza Computazionale

Indicando con

$$\mathcal{C}(P, \_) : \text{DATI} \to \text{DATI}$$

la funzione che viene calcolata dal programma P (semantica di P).

La **potenza computazionale** di un calcolatore è definita come l'insieme di tutte le funzioni che quel sistema di calcolo è in grado di calcolare, ovvero

$$F(\mathcal{C}) = \{\mathcal{C}(P, \_) | P \in \text{PROG}\} \subseteq \text{DATI}_{\perp}^{\text{DATI}}$$

Ovvero, l'insieme di tutte le possibili semantiche di funzioni calcolabili con il sistema $\mathcal{C}$. Stabilire il carattere di quest'ultima inclusione equivale a stabilire *cosa può fare l'informatica*:

- se $F(\mathcal{C}) \subsetneq \text{DATI}_{\perp}^{\text{DATI}}$ allora esistono compiti **non automatizzabili**
- se $F(\mathcal{C}) = \text{DATI}_{\perp}^{\text{DATI}}$ allora l'informatica *può fare tutto*

Calcolare funzioni vuol dire risolvere problemi *in generale*, a ogni problema è possibile associare una funzione soluzione che permette di risolverlo automaticamente.

Un possibile approccio per risolvere l'inclusione è tramite la **cardinalità** (funzione che associa ogni insieme al numero di elementi che contiene) dei due insiemi. Potrebbe però presentare dei problemi: è efficace solo quando si parla di insiemi finiti. Ad esempio, l'insieme dei numeri naturali contiene l'insieme dei numeri pari $\mathbb{P} \subsetneq \mathbb{N}$, ma $|\mathbb{N}| = |\mathbb{P}| = \infty$.
Serve una diversa definizione di cardinalità che considera l'esistenza di infiniti *più densi di altri*.