Fondamenti di Informatica

Università di Verona
Imbriani Paolo - VR500437
Professor Isabella Mastroeni

October 7, 2025

Contents

1 Cosa è l'informatica?

La domanda chiave di questo corso è: *"Cosa è l'informatica?"*. Ci sono diversi definizioni a seconda del contesto, ma in generale l'informatica è lo studio dei processi che trasformano l'informazione. Possiamo vedere, storicamente, diverse definizioni come quella in inglese come "Computer Science" che vede l'informatica come studio della calcolabilità, della computazione e dell'informazione.

1.1 Perché la calcolabilità

Si studia la calcolabilità perché ci aiuta a capire cosa possiamo fare con gli strumenti che abbiamo. Quando descriviamo un programma, quanto tempo ci mette e quanto spazio utilizza è una domanda importante per capire se il programma è efficiente o meno. Anche in senso dei linguaggi di programmazione per capire se usiamo quello giusto per il problema che stiamo cercando di risolvere. Chiaramente è un qualcosa che in continuo sviluppo perché si evolve in base alla tecnologia che abbiamo a disposizione.

Uno dei pionieri è stato **Hilbert** che si chiedeva se la matematica fosse formalizzabile come insieme finito (non contraddittorio) di assiomi? Godel dimostrò che non è possibile rappresentare la matematica come un insieme finito di assiomi in maniera non contradditoria, dicendo che in ogni sistema formale ci sono proposizioni vere che non possono essere dimostrate all'interno del sistema.

Nel tentativo di rispondere a queste (ed altre) domande si è costruito un modello che permette di comprendere profondamente ilr agionamento computazionale, permettendo di applicarlo ad ogni disciplina. Cosa è calcolabile e cosa non lo è?

1.1.1 Macchina di Turing

Anche Turing si pose questa domanda e propose la **Macchina di Turing** come modello di calcolo. Una sola macchina (programmabile) per tutti i problemi. La macchina è universale (interprete):

$$Init(P,x) = \begin{cases} P(x) & \text{se } P \text{ è un programma che termina} \\ \uparrow & \text{se } P \text{ è un programma che non termina} \end{cases}$$

Se un problema è intuitivamente calcolabile, allora esisterà una macchina di Turing (o un dispostivo equivalente, come il computer) in grado di risolverlo, cioè calcolarlo. I modelli equivalenti possono essere:

- Lambda calcolo
- Funzioni ricorsive

- Linguaggi di programmazione (Turing completi)

I problemi non calcolabili sono infinitamente più numerosi di quelli calcolabili.

1.1.2 Limiti dell'informatica

L'informatica ha più limiti di quanto si possa pensare, definiti dall'equivalenza di Turing e l'incompletezza di Godel che ci dicono che non possiamo risolvere tutti i problemi. Ci sono anche limiti fisici e tecnologici come:

- Dati non osservabili (teorema di Shannon)
- Dati non controllabili (velocità della luce)

1.2 Basi di logica

Alcune nozioni di logica che ci serviranno in seguito:

- Variabili
- Constanti
- f simbolo di funzione m-ario, $t1, t2, .., t_m$ termini, allora $f(t1, t2, .., tm)$ è termine
- Formula atomica $P(t1, t2, .., tm)$
- equivalenze famose
- quantificatori $\forall, \exists$

1.3 ...TODO

2 Funzioni calcolabili

Un insieme è a tutti gli effetti una proprietà che dato un oggetto stabilisce se esso all'interno di insieme o no.

$$\text{Problemi} \equiv \text{Funzioni} \quad f : \mathbb{N} \to \mathbb{N}$$

Ci chiediamo se questa funzioni siano tutte calcolabili (intuitivamente). Da il teorema che abbiamo citato nei precedenti paragrafi (quello dell'incompletezza) sappiamo che non è così. Cerchiamo di vedere insiemisticamente perché questo è giustificato.

Definizione 2.1: Intuitivamente Calcolabile

Qualcosa che è **intuitivamente calcolabile** è qualcosa è che riusciamo a descrivere attraverso un algoritmo, ovvero una sequenza finita di passi discreti elementari.

La funzione di tipo:

$$f : \mathbb{N} \to \mathbb{N}$$

è un **insieme** di associazioni input-output.

Esempio 2.1

$$\begin{aligned} f = \text{quadrato} &= \{(0,0),(1,1),(2,4),(3,9),(4,16),...\} \\ &= \{(x, x^2) \mid n \in \mathbb{N}\} \end{aligned}$$

Quindi f è un insieme di coppie in $\mathbb{N} \times \mathbb{N}$. Quindi $|\mathbb{N} \times \mathbb{N}| = |\mathbb{N}|$ (cardinalità di $\mathbb{N} \times \mathbb{N}$ Quindi

$$f \subseteq \mathbb{P}(\mathbb{N} \times \mathbb{N}) = \mathbb{P}(\mathbb{N})$$

Per esempio se:

$$A = \{1,2,3\} \text{ allora } \mathbb{P}(A) = \{\emptyset, \{1\}, \{2\}, \{3\}, \{1,2\}\}$$

dove $|\mathbb{P}(A)| = 2^{|A|}$.

$$|\mathbb{N}| = \omega < |P(\mathbb{N})|) = |\mathbb{R}|$$

e quindi l'insieme delle funzioni **non è numerabile**.

2.1 Quale funzioni numerabili ci sono?

Σ =alfabeto finito di simboli che uso per il programma/algoritmo

$$\Sigma = s_1, s_2, s_3, \dots$$

quindi un programma non è nient'altro che un sottoinsieme finito di Σ^* (tutte le stringhe finite che posso formare con l'alfabeto).

$$\Sigma = a, b, c$$

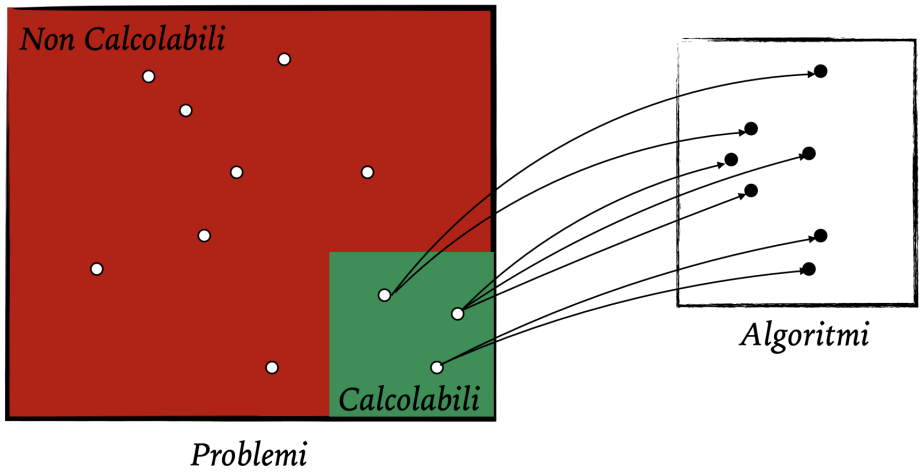
$$\Sigma^* = \{\emptyset, a, b, c, ab, ba, ac, cd, bc, cb, \dots\}$$

in questo caso, la sequenza di simboli in Σ^* è numerabile, perché

$$|\Sigma^*| = |\mathbb{N}|$$

$$|\text{Programmi in } \mathbb{N}| = |\Sigma^*| = |\mathbb{N}|$$

e di conseguenza l'insieme dei programmi è numerabile. Una veloce constatazione che possiamo fare è vedere quindi che l'insieme delle funzioni calcolabili è numerabile, perché ogni funzione calcolabile è associata ad almeno un programma che la calcola.



3 Principio di induzione

Il principio di induzione ha senso solo su insieme infiniti e serve per dimostrare che una proprietà vale per tutti gli elementi. Esistono due metodi di induzione:

- Induzione matematica
- Induzione strutturale

Tratteremo nello specifico caso in questo corso **l'induzione matematica.** Partiamo da un insieme A infinito con una relazione $< : (A, <)$ è una relazione d'ordine non riflessiva perché non è minore stretto. Utilizziamo l'induzione matematica e quindi $A = \mathbb{N}$ e $<$ è l'ordinamento stretto tra numeri. Una relazione d'ordine deve essere *ben fondata* vuol dire che non esistono catene discendenti infinite. Una catena discendente è una sequenza infinita di elementi

$$a_0 > a_1 > a_2 > a_3 > \dots$$

In un tipo di relazione riflessiva è sicuramente non ben fondata perché posso fare continuare ad inserire lo stesso numero all'infinito.

$$m \text{ minimale in } A \ : \ b \in A \text{ è minimale se } \forall b' < b . b' \notin A$$

Esempio 3.1

Se prendiamo $\{1, 2, 3\}$ con relazione d'ordine di contenimento allora esistono più minimali come $\{1, 2\}$ o $\{2, 3\}$. Quindi se $A = \mathbb{N} \implies \exists b$ minimo $\forall x \subseteq \mathbb{N}$.

Quindi preso A insieme ben fondato con ordinamento $<$ allora: π proprietà definita sugli elementi di

$$A : \pi \subseteq A \text{ allora } \forall a \in A\,.\pi(a) \iff \forall a \in A.[[\forall b < a.\pi(b)] \to \pi(a)]$$

Se dimostriamo π per ogni elemento più piccolo di a allora π vale per a.

$$Base_A = \{a \in A | a \text{ minimale}\}$$

Quindi

$$\overbrace{\forall A \in Base_A\,.\,\pi(a)}^{\text{Base}}\ \underbrace{\forall a \in A.Base_A}_{\text{passo induttivo}}.\ \overbrace{\forall b < a.\pi(b)}^{\text{ipotesi induttiva}}\ \to \underbrace{\pi(a)}_{\text{tesi}}$$

Esempio 3.2

Prendiamo come esempio il seguente enunciato:

$$\forall n \in \mathbb{N},\ \sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

Base: $n = 1$

$$A = \mathbb{N}\backslash\{0\}\ \ Base_A = \{1\} \to \sum_{i=1}^{1} i = 1$$

$$\begin{aligned}\sum_{i=1}^{1} i &= 1 \\ &= n(n+1)/2 \\ &= \frac{1(1+1)}{2} = 1\end{aligned}$$

Caso base dimostrato.

Passo induttivo: prendo $n \in \mathbb{N}$ e applico l'ipotesi induttiva: per ogni $k < n$ vale la tesi.

$$\text{Tesi da dimostrare: } \sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

$$\sum_{i=1}^{n} i = \sum_{i=1}^{n-1} i + n$$

Sappiamo sicuramente che $n - 1 < n$ e per ipotesi induttiva:

$$\sum_{i=1}^{n-1} i + n = \frac{(n-1)(n-1+1)}{2} + n = \frac{n(n-1)}{2} + n =$$

$$\frac{n(n-1)+2n}{2} = \frac{n(n-1+2)}{2} = \frac{n(n+1)}{2} \quad \square$$

e quindi la tesi vale perché siamo arrivati alla stessa espressione che volevamo dimostrare.

3.1 Linguaggi formali

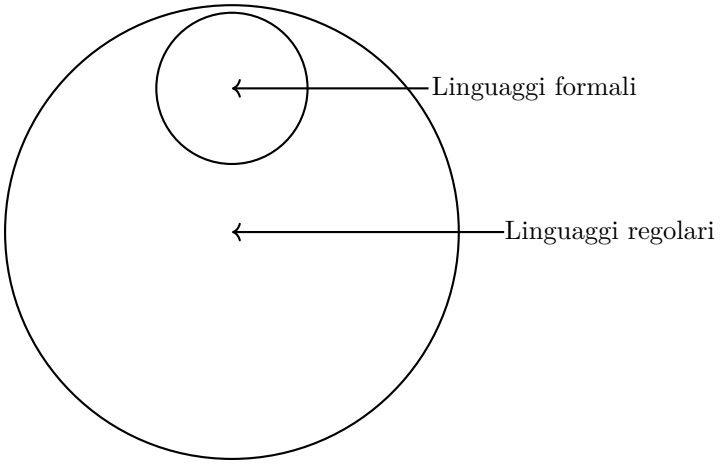
Definizione 3.1: Linguaggio formale

Un linguaggio formale è un insieme di stringhe composte da simboli in un alfabeto finito Σ.

Σ^* denota il linguaggio di tutte le stringhe dell'alfabeto Σ, se Σ non è vuota allora Σ^* è infinito e numerabile. Solitamente un linguaggio formale $\mathcal{L}$ è un sottoinsieme di Σ^* tipicamente infiniti ma non è necessario:

$$\mathcal{L} \subseteq \Sigma^*$$

I linguaggi finiti sono sicuramente regolari perché *posso sempre costruire un automa a stati finiti* che li riconosce.



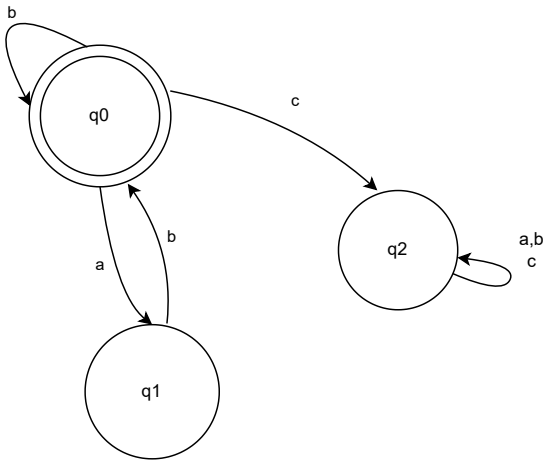


Figure 1: Esempio di automa a stati finiti