Domande Informatica Teorica

1. Cos'è un problema di decisione?

   Solution: Un problema di decisione è una domanda a cui rispondere *Sì* o *No*.
   Formalmente, è costituito da 3 elementi:

   - Nome del problema
   - Istanza degli oggetti considerati
   - Domanda, ovvero proprietà che gli oggetti possono o meno soddisfare

2. Come dimostrare l'esistenza di problemi non decidibili, senza mostrarne un esempio?

   Solution: Per studiare la decidibilità di un problema Π ci sono due possibili approcci:

   - Trovare un programma P_Π che calcola la funzione soluzione

     $$\Phi_\Pi : D \to \{0,1\} \quad \text{t.c.} \quad \Phi_\Pi(x) = \begin{cases} 1 & \text{se } p(x) \\ 0 & \text{se } \neg p(x) \end{cases}$$

     di conseguenza $\Phi_\Pi \in \mathcal{T}$
   - Se $\Phi_\Pi \in \mathcal{T}$, allora esiste un programma che la calcola

   Di conseguenza, i problemi risolvibili PROG sono isomorfi alle funzioni calcolabili, quindi numerabili, mentre tutti i possibili problemi sono rappresentati dalle funzioni da $\mathbb{N}$ a $\mathbb{N}$ (dato quanto già visto), isomorfe a $\mathbb{N}_\perp^\mathbb{N}$, quindi:

   $$\text{DATI} \sim \text{PROG} \sim \mathbb{N} \nsim \mathbb{N}_\perp^\mathbb{N}$$

   Per dimostrare che DATI $\sim \mathbb{N}$: serve una funzione tale che permetta una biezione tra dati e $\mathbb{N}$, come la funzione coppia di Cantor

   $$\langle \_, \_ \rangle : \mathbb{N} \times \mathbb{N} \to \mathbb{N}^+$$

   (banalmente estendibile a tutto $\mathbb{N}$).

   Per dimostrare che PROG $\sim \mathbb{N}$: una volta definito un sistema di calcolo e i relativi comandi, si può mostrare una codifica per questi che porta a una biezione con $\mathbb{N}$.

   Alternativamente, un problema di decisione può essere comparato al riconoscimento di un linguaggio $L \subseteq \Sigma^*$, per un alfabeto Σ di conseguenza:

- Il numero possibile di linguaggi è $P(\Sigma^*) \sim \mathbb{R}$
- I sistemi di calcolo sono in quantità PROG $\sim \mathbb{N}$

Per forza devono esistere linguaggi non decidibili.

3. Il problema dell'arresto: definizione e dimostrazione.

Solution: Definizione del problema dell'arresto:

- Nome: AR
- Istanza: $x, y \in \mathbb{N}$
- Domanda: $\varphi_y(x) \downarrow$?

Teorema 0.1. *AR è indecidibile.*

Dimostrazione. Assumiamo per assurdo che AR sia decidibile, allora esiste una funzione soluzione

$$\Phi_{\text{AR}}(x, y) = \begin{cases} 0 & \text{se } \varphi_y(x) \uparrow \\ 1 & \text{se } \varphi_y(x) \downarrow \end{cases}$$

Valutando il caso in cui $x = y$

$$\Phi_{\text{AR}}(x, x) = \begin{cases} 0 & \text{se } \varphi_x(x) \uparrow \\ 1 & \text{se } \varphi_x(x) \downarrow \end{cases}$$

Visto che $\Phi_{\text{AR}} \in \mathcal{T}$, anche la funzione

$$f(x) = \begin{cases} 0 & \text{se } \Phi_{\text{AR}}(x) = 0 \equiv \varphi_x(x) \uparrow \\ \varphi_x(x) + 1 & \text{se } \Phi_{\text{AR}}(x) = 1 \equiv \varphi_x(x) \downarrow \end{cases} \in \mathcal{T}$$

Sia $\alpha \in \mathbb{N}$ la codifica di A tale che $\varphi_\alpha = f$. Valutiamo φ_α in α:

$$\varphi_\alpha(\alpha) = \begin{cases} 0 & \text{se } \varphi_\alpha(\alpha) \uparrow \\ \varphi_\alpha(\alpha) + 1 & \text{se } \varphi_\alpha(\alpha) \downarrow \end{cases}$$

Ma tale funzione non può esistere:

- Nel primo caso $\varphi_\alpha(\alpha) = 0$ se $\varphi_\alpha(\alpha) \uparrow$, ma è una contraddizione
- Nel secondo caso $\varphi_\alpha(\alpha) = \varphi_\alpha(\alpha) + 1$, ma tale relazione non vale per nessun naturale

Siamo a un assurdo, AR è indecidibile.

□

4. Sistemi di calcolo visti in Teoria della Calcolabilità.

Solution: I sistemi di calcolo visti sono:

- Sistema RAM: infiniti registri, R_0 contiene l'output, R_1 l'input, si ha un program counter L, le istruzioni sono
  - Incremento: $R_k \leftarrow R_k + 1$
  - Decremento: $R_k \leftarrow R_k \dot{-} 1$
  - Salto condizionato: `if` $R_k = 0$ `goto` m, con $m \in \{1, \ldots, |P|\}$
- Sistema WHILE: 21 registri, x_0 output, x_1 input, sono presenti dei comandi base:
  - $x_k := x_j + 1$
  - $x_k := x_j \dot{-} 1$
  - $x_k := 0$

  e dei comandi definiti induttivamente:
  - Comando composto

    `begin` $C_1, \ldots, C_n$ `end`

    dove ogni C_i è un qualsiasi comando
  - Comando while

    `while` $x_k \neq 0$ `do` C

    dove C è un qualsiasi comando

  Di conseguenza, un programma WHILE è un comando composto

5. Approfondimento su RAM (struttura, istruzioni, stato prossimo, computazione)

Solution: Il sistema RAM permette infiniti registri, tra i quali R_0 contiene l'output e R_1 l'input, si ha inoltre un program counter L per tenere traccia dell'istruzione da eseguire. Un programma P è un insieme ordinato di istruzioni.

Le istruzioni sono:

- Incremento: $R_k \leftarrow R_k + 1$
- Decremento: $R_k \leftarrow R_k \dot{-} 1$

- Salto condizionato: `if` $R_k = 0$ `goto` m, con $m \in \{1, \ldots, |P|\}$

Ogni istruzione fa passare la macchina da uno stato a un altro; la semantica operazione di un'istruzione è formata dalla coppia degli stati prima e dopo l'istruzione.

La computazione del programma P è una sequenza di stati $\mathcal{S}_i$, infinita se non termina, altrimenti si ha uno stato finale $\mathcal{S}_{fin}$ in cui viene posto in R_0 il risultato della computazione.

Lo stato è una funzione

$$\mathcal{S} : \{L, R_i\} \to \mathbb{N}$$

ovvero, che dato un registro e un valore del program counter L, restituisce il contenuto del registro.

Uno stato finale $\mathcal{S}_{fin}$ è un qualsiasi stato tale che $\mathcal{S}(L) = 0$.

Lo stato iniziale è tale che

$$\mathcal{S}_{init}(R_i) = \begin{cases} 1 & \text{se } R_i = L \\ n & \text{se } R_i = R_1 \\ 0 & \text{altrimenti} \end{cases}$$

Per definire l'esecuzione del programma si usa la funzione stato prossimo

$$\delta : \text{STATI} \times \text{PROG} \to \text{STATI}_\perp$$

tale che

$$\delta(\mathcal{S}, P) = \mathcal{S}'$$

Dove $\mathcal{S}$ rappresenta lo stato in seguito all'esecuzione del comando P.

La funzione è tale che:

- Se $\mathcal{S}(L) = 0$, $\mathcal{S}' = \perp$ in quanto l'esecuzione è terminata
- Se $\mathcal{S}(L) > |P|$ non si ha una terminazione esplicita, quindi

$$\mathcal{S}'(R) = \begin{cases} 0 & \text{se } R = L \\ \mathcal{S}(R_i) & \text{altrimenti} \end{cases}$$

- Se $1 \leq \mathcal{S}(L) \leq |P|$, si considera l'istruzione $\mathcal{S}(L)$-esima:
  - incremento/decremento su R_k:

$$\mathcal{S}'(R) = \begin{cases} \mathcal{S}(R) + 1 & \text{se } R = L \\ \mathcal{S}(R) \pm 1 & \text{se } R = R_k \\ \mathcal{S}(R) & \text{altrimenti} \end{cases}$$

– salto condizionato su R_k

$$\mathcal{S}'(R) = \begin{cases} m & \text{se } R = L \wedge R_k = 0 \\ \mathcal{S}(L) + 1 & \text{se } R = L \wedge R_k \neq 0 \\ \mathcal{S}(R) & \text{altrimenti} \end{cases}$$

L'esecuzione di un programma genera una sequenza di stati, definita secondo la funzione δ.

6. Come abbiamo definito la potenza computazionale $F(\text{RAM})$.