

Appunti di Informatica Teorica

Parte 1: computabilità

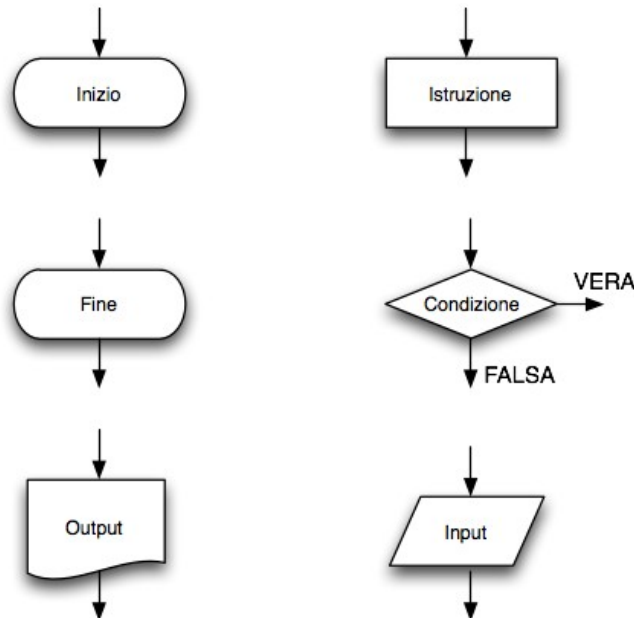
Definizione (intuitiva): un **algoritmo** è un procedimento che, forniti opportuni dati in input, genera in output la soluzione a un problema. Perché un procedimento sia un algoritmo deve:

- essere composto da una sequenza di istruzioni assunte come note e primitive;
- essere eseguito in un numero finito di applicazioni delle istruzioni, se la soluzione esiste;
- essere **deterministico**, ossia al termine dell'esecuzione di ogni istruzione deve essere univocamente determinata la scelta dell'istruzione successiva.

Algoritmo (di Euclide): computa il MCD fra due numeri a e b tali che $a < b$.

1. se a divide b , a è il risultato, fine;
2. altrimenti, $a:=b$, $b:=$ resto della divisione intera tra a e b ;
3. vai all'1.

Definizione (informale): la **rappresentazione a diagrammi di flusso** di un algoritmo prevede i seguenti simboli:



Definizione: siano D e C insiemi. Una **funzione** (totale) φ da D in C è un sottoinsieme di $D \times C$ tale che per ogni x in D esiste unica la coppia (x, y) in φ dove y in C . Ovvero:
$$\varphi \subseteq D \times C \mid \forall x \in D \exists ! (x, y) \in \varphi$$

Si usa indicare la funzione con $\varphi: D \rightarrow C$, l'elemento y corrispondente a un certo x con $\varphi(x)$ o **immagine**, D con **dominio** e C con **codominio**.

Definizione: sia $\varphi: D \rightarrow C$ una funzione. L'insieme:

$$R = \{y \mid y = \varphi(x), x \in D\}$$

è detto **rango** della funzione.

Definizione: sia $\varphi: D \rightarrow C$ una funzione, R il suo rango. φ si dice **suriettiva** sse $R=C$.

Definizione: sia $\varphi: D \rightarrow C$ una funzione. φ si dice **iniettiva** sse

$$\forall (x_1, x_2) \in D^2 \mid x_1 \neq x_2, \varphi(x_1) \neq \varphi(x_2)$$

Definizione: sia φ una funzione. φ si dice **biiettiva** sse è iniettiva e suriettiva.

Definizione: sia $\varphi: D \rightarrow C$ una funzione biiettiva. La sua **funzione inversa** $\varphi^{-1}: C \rightarrow D$ è la funzione per cui vale:

$$\forall x \in D \varphi^{-1}(\varphi(x)) = \varphi(\varphi^{-1}(x)) = x$$

Teorema: siano $\varphi: D \rightarrow B$ e $\psi: B \rightarrow C$ funzioni. La funzione composta $\varphi \circ \psi: D \rightarrow C$ è biiettiva solo se φ è iniettiva e ψ è suriettiva.

Definizione: sia A un algoritmo e $\varphi: D \rightarrow C$ una funzione. Si dice che A **computa** φ sse, $\forall (x, y) \in \varphi$, allora A con x in input produce y in output.

Definizione: sia $\varphi: D \rightarrow C$ una funzione. Si dice che φ è **computabile** sse esiste un algoritmo che la computa.

Definizione (intuitiva): sia D un insieme. Un **predicato** P(x) è un'affermazione che può essere vera o falsa per ogni elemento x di D.

Definizione (intuitiva): sia D un insieme. Una **relazione** \mathcal{R} un'affermazione che può essere vera o falsa per ogni coppia di elementi $a, b \in D^2$. Se è vera, si indica con $a \mathcal{R} b$.

Definizione: una proprietà P su un insieme D è **decidibile** sse esiste un algoritmo in grado di stabilire, per ogni $x \in D$, se è vera o falsa.

Definizione: una relazione \mathcal{R} su un insieme D è **decidibile** sse esiste un algoritmo in grado di stabilire, per ogni $a, b \in D^2$, se è vera o falsa.

Definizione: sia D un insieme, A un suo sottoinsieme. Il **predicato di appartenenza** di A $P_A(x)$ è vero sse $x \in A \forall x \in D$.

Definizione: sia D un insieme, A un suo sottoinsieme. La **funzione di appartenenza** di A $\varphi_A(x): D \rightarrow \{0,1\}$ vale 0 sse $x \in A$, 1 altrimenti.

Definizione: un insieme A è **decidibile** sse il suo predicato di appartenenza è decidibile.

Teorema: un insieme A è decidibile sse la sua funzione di appartenenza è computabile.

Definizione: siano D e C insiemi, $E \subset D$. Una **funzione parziale** φ da D in C è un sottoinsieme di $E \times C$ tale che per ogni x in E esiste unica la coppia (x,y) in φ dove y in C.

Ovvero:

$$\varphi \subseteq E \times C \mid \forall x \in E \exists! (x, y) \in \varphi$$

Si indica:

- la funzione con $\varphi: D \rightarrow C$,
- l'elemento y corrispondente a un certo $x \in E$ con $\varphi(x)$ o **immagine**,
- il fatto che a un elemento $x \in D - E$ non è associato alcun elemento in C con la scrittura $\varphi(x) = \perp$ (si dice che la funzione **non è definita** in x),
- D con **dominio**,
- E con **campo di esistenza** e
- C con **codominio**.

Definizione: sia A un algoritmo e $\varphi: D \rightarrow C$ una funzione parziale con campo di esistenza E. Si dice che A **computa** φ sse:

- $\forall x \in E$ A produce $\varphi(x)$ in output fornendo x in input;
- $\forall x \in D - E$ A non termina (o non produce alcun output) fornendo x in input;

Definizione: sia $\varphi: D \rightarrow C$ una funzione parziale. Si dice che φ è **computabile** sse esiste un algoritmo che la computa.

Definizione: un insieme A si dice **enumerabile** o **semidecidibile** sse è vera una delle seguenti condizioni:

- $A = \emptyset$ oppure
- A è il rango di una funzione computabile totale $\varphi: \mathbb{N} \rightarrow C$.

Nota: A si dice semidecidibile perché esiste un algoritmo che termina se ha in input $x \in A$, viceversa non termina.

Teorema: Sia A enumerabile. Si riesce a ottenere una funzione computabile totale $\varphi: \mathbb{N} \rightarrow C$ con rango A iniettiva solo se A è infinito.

Dimostrazione: se A è finito e ha cardinalità n $\varphi(x)$ potrà assumere al più n valori distinti.

Teorema: sia $A \subseteq \mathbb{N}$ un insieme. Se A è decidibile, allora è anche enumerabile.

Dimostrazione: se A è vuoto, allora è enumerabile. Se non è vuoto allora contiene almeno un elemento a e ha una funzione di appartenenza $f_A: \mathbb{N} \rightarrow \{0,1\}$ computabile. Posso costruire una funzione computabile $\varphi: \mathbb{N} \rightarrow A$ come segue:

$$\varphi(x) = \begin{cases} x & \text{se } f_A(x) = 0 \\ a & \text{se } f_A(x) = 1 \end{cases}$$

Essendo φ computabile e con rango A, allora A è enumerabile.

Teorema: sia $A \subseteq \mathbb{N}$ un insieme. A è decidibile sse $\bar{A} = \mathbb{N} - A$ è decidibile.

Dimostrazione: le funzioni di appartenenza si ricavano l'una dall'altra dalla relazione

$$f_A(x) = 1 - f_{\bar{A}}(x) \quad \text{e in particolare } f_A \text{ è computabile sse } f_{\bar{A}} \text{ lo è.}$$

Lemma: se $A \subseteq \mathbb{N}$ è il rango di una funzione computabile parziale $\varphi: \mathbb{N} \rightarrow C$ con campo di esistenza E, allora A è semidecidibile.

Dimostrazione: se $\forall x \varphi(x) = \perp$, allora $A = \emptyset$ e quindi è enumerabile. Se φ è definita per qualche x, si definisce l'algoritmo A_ψ eseguendo l'algoritmo A_φ che computa φ come segue:

- siano $A_{\varphi,1}, A_{\varphi,2}, \dots, A_{\varphi,n}$ i passi dell'algoritmo A_φ ;
- sia $A_{\varphi,i}(x)$ l'output dell'esecuzione dell'i-esimo passo di A_φ con input $x \in E$, per $i \leq n$;
- sia $A_{\varphi,i}(x) = A_{\varphi,n}(x)$ per $i > n$;
- A_ψ è l'algoritmo che esegue A_φ "per controdiagonali successive" ossia $A_{\varphi,1}(0), A_{\varphi,1}(1), A_{\varphi,2}(0), A_{\varphi,1}(2), A_{\varphi,2}(1), A_{\varphi,3}(0) \dots$

A_ψ computa tutti i valori per cui A_φ termina, ossia tutti i valori di A. E' possibile quindi definire la funzione computabile totale $\psi: \mathbb{N} \rightarrow A$, quindi A è semidecidibile.

Teorema: $A \subseteq \mathbb{N}$ è decidibile sse A e \bar{A} sono semidecidibili.

Dimostrazione:

- sia A decidibile, allora A è semidecidibile. D'altra parte, se A è decidibile anche \bar{A} lo è, quindi anche \bar{A} è semidecidibile.
- siano A e \bar{A} semidecidibili. Sono possibili tre casi:
 - se $A = \emptyset$, allora A è decidibile poiché $f_A(x) = 1$ è computabile;
 - se $\bar{A} = \emptyset$, allora A è decidibile poiché $f_A(x) = 0$ è computabile;
 - se A e \bar{A} non sono vuoti, si ha che A è il rango di una funzione φ_A computabile totale, inoltre \bar{A} è il rango di una funzione $\varphi_{\bar{A}}$ computabile totale. Allora posso costruire una $f_A(x)$ computabile calcolando $\varphi_A(n)$ e $\varphi_{\bar{A}}(n)$ per n crescenti da 0 in avanti; quando una delle due computazioni ritorna x (e lo farà sicuramente in un tempo finito) si è in grado di stabilire se l'elemento sta in A oppure no.

Dimostrazione (alternativa):

- sia A decidibile, ossia esiste la funzione di appartenenza f_A computabile. Sono possibili due casi:
 - se $A = \emptyset$, allora A è semidecidibile poiché vuoto e anche $\bar{A} = \mathbb{N}$ è semidecidibile poiché $\varphi: \mathbb{N} \rightarrow \bar{A} | \varphi(x) = x$, è computabile;
 - sia $A \neq \emptyset$. Sia $\varphi = \begin{cases} x & \text{se } f_A(x) = 0 \\ \perp & \text{se } f_A(x) = 1 \end{cases}$, si ha che φ è computabile, parziale e con rango A. Per il lemma, A è semidecidibile. D'altra parte definendo $\bar{\varphi} = \begin{cases} x & \text{se } f_{\bar{A}}(x) = 0 \\ \perp & \text{se } f_{\bar{A}}(x) = 1 \end{cases}$ si prova analogamente che \bar{A} è semidecidibile;
- siano A e \bar{A} semidecidibili. Sono possibili tre casi:
 - se $A = \emptyset$, allora A è decidibile poiché $f_A(x) = 1 \forall x$ è computabile;
 - se $\bar{A} = \emptyset$, allora A è decidibile poiché $f_A(x) = 0 \forall x$ è computabile;
 - se A e \bar{A} non sono vuoti, si ha che A è il rango di una funzione φ_A computabile totale, inoltre \bar{A} è il rango di una funzione $\varphi_{\bar{A}}$ computabile totale. Allora posso costruire una $f_A(x)$ computabile calcolando $\varphi_A(n)$ e $\varphi_{\bar{A}}(n)$ per n crescenti da 0 in avanti; quando una delle due computazioni ritorna x (e lo farà sicuramente in un

tempo finito) si è in grado di stabilire se l'elemento sta in A oppure no.

Teorema (triangolare): sono equivalenti le seguenti affermazioni:

1. A è enumerabile;
2. A è il campo di esistenza di una funzione computabile parziale
3. A è il rango di una funzione computabile parziale

Dimostrazione: uso il procedimento triangolare, ossia dimostro $1 \Rightarrow 2$, $2 \Rightarrow 3$ e $3 \Rightarrow 1$.

- $1 \Rightarrow 2$: ci sono due casi:
 - $A = \emptyset$, A è il campo di esistenza di $\varphi(x) = \perp$;
 - $A \neq \emptyset$, esiste quindi la funzione computabile totale $\varphi : \mathbb{N} \rightarrow A$. Sia $\psi(x) = \min_z [\varphi(z) = x]$; ψ ha campo di esistenza esattamente A ed è anch'essa computabile.
- $2 \Rightarrow 3$: esiste $\varphi(x)$ parziale, con campo di esistenza A e computabile. Sia $\psi(x) = \begin{cases} x & \text{se } \varphi(x) \neq \perp \\ \perp & \text{se } \varphi(x) = \perp \end{cases}$; ψ ha rango A ed è computabile parziale.
- $3 \Rightarrow 1$: vero per il lemma.

Definizione: siano A e B due insiemi. Si dice che:

- A e B hanno **stessa cardinalità**, in simboli $Card(A) = Card(B)$, sse esiste una funzione $\varphi : A \rightarrow B$ biiettiva;
- A e B hanno **cardinalità diversa**, in simboli $Card(A) \neq Card(B)$, sse non esiste una funzione $\varphi : A \rightarrow B$ biiettiva;
- A ha **cardinalità inferiore** a B, in simboli $Card(A) \leq Card(B)$, sse esiste una funzione $\varphi : A \rightarrow B$ iniettiva;
- A ha **cardinalità strettamente inferiore** a B, in simboli $Card(A) < Card(B)$, sse $Card(A) \leq Card(B)$ e $Card(A) \neq Card(B)$;
- A ha **cardinalità n**, in simboli $Card(A) = n$, sse A è finito ed ha n elementi.
- A è **numerabile**, in simboli $Card(A) = \aleph_0$, sse esiste una funzione $\varphi : \mathbb{N} \rightarrow A$ biiettiva.

Definizione: siano A e B insiemi. B è un **sottoinsieme proprio** di A sse $B \subset A \wedge B \neq \emptyset$.

Teorema: sia A un insieme. A è infinito se esiste un suo sottoinsieme proprio numerabile.

Corollario: sia A un insieme infinito. Allora $Card(A) \geq \aleph_0$.

Teorema: l'insieme QP dei quadrati perfetti è numerabile.

Dimostrazione: $\varphi : \mathbb{N} \rightarrow QP$, $\varphi(x) = x^2$ è biiettiva.

Teorema: l'insieme P dei numeri primi è numerabile.

Dimostrazione: si dimostra che P è infinito, quindi $Card(P) \geq \aleph_0$. D'altra parte $P \subset \mathbb{N}$, quindi $Card(P) \leq Card(\mathbb{N}) = \aleph_0$. Quindi $Card(P) = \aleph_0$.

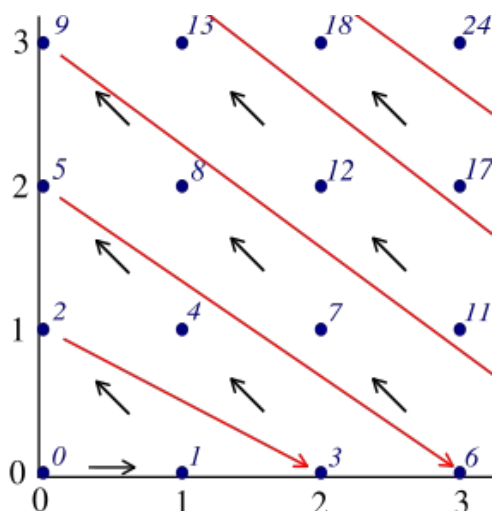
Teorema: \mathbb{Z} è numerabile.

Dimostrazione: $\varphi : \mathbb{N} \rightarrow \mathbb{Z}$, $\varphi(x) = \begin{cases} -\frac{x}{2} & \text{se } x \text{ è pari} \\ \frac{x+1}{2} & \text{se } x \text{ è dispari} \end{cases}$ è biiettiva.

Teorema: siano A e B insiemi numerabili. Allora, $A \times B$ è numerabile.

Dimostrazione (intuitiva): dal momento che B è numerabile, esiste una funzione biunivoca $\varphi : B \rightarrow A$, quindi è sufficiente dimostrare che $A \times A$ è numerabile.

D'altra parte esiste sempre una funzione biiettiva da A in $A \times A$ detta funzione coppia di Cantor la cui rappresentazione grafica (nel caso $A = \mathbb{N}$) è la seguente:



Corollario: sia A un insieme numerabile. Allora, A^n è anch'esso numerabile.

Teorema: \mathbb{Q} è numerabile.

Dimostrazione: \mathbb{Z} e \mathbb{N}^0 sono numerabili, quindi $\mathbb{Q} = \mathbb{Z} \times \mathbb{N}^0$ è numerabile.

Teorema: \mathbb{R} non è numerabile.

Dimostrazione: sia $G \subset \mathbb{R}$ l'intervallo aperto da 0 a 1. Si ha che $Card(G) = Card(\mathbb{R})$, in quanto esiste una funzione biettiva da G in \mathbb{R} ($y = \tan(\pi(x - \frac{1}{2}))$). Quindi \mathbb{R} è numerabile sse G lo è.

Ogni $g \in G$ è un numero del tipo 0,13432... con infiniti decimali, sia S l'insieme dei numeri g moltiplicati per 10 ed espressi in binario, ossia $S = \{s | s = \text{binario}(10g)\}$ con binario opportuna funzione biettiva. Quindi \mathbb{R} è numerabile sse S lo è.

Si suppone per assurdo che \mathbb{R} è numerabile, ne deriva che anche G ed S sono numerabili.

Dato che S è numerabile, si può pensare di mettere i suoi elementi in sequenza e quindi indicarli con s_i . Preso un elemento s_i , sia $s_{i,j}$ il suo j -esimo bit. Sia ora s_k l'elemento costruito con i bit con lo stesso indice degli elementi in S , ossia $s_k = s_{1,1}s_{2,2}s_{3,3}...$ Sia infine $s_m = \bar{s}_k$ il numero ottenuto tramite complemento a uno di s_k .

Accade che il bit $s_{m,m}$ dovrebbe essere contemporaneamente uguale all' m -esimo bit di s_k , per definizione di s_k , ma d'altra parte dovrebbe essere il suo complemento per definizione di s_m . Questa è una contraddizione, quindi S , G ed \mathbb{R} non sono insiemi numerabili.

Teorema: $Card(\mathbb{R}) = Card(\wp(\mathbb{N}))$

Dimostrazione: sia S l'insieme definito al paragrafo precedente, vale $Card(\mathbb{R}) = Card(S)$. Per ogni sottoinsieme L di \mathbb{N} , la sua funzione di appartenenza $f_L(j)$ assume valore 0 o 1 a seconda che l'elemento j di \mathbb{N} sia o meno incluso in L . Fissato un sottoinsieme L , sia allora s_L la sequenza dei valori assunti da $f_L(x)$ per x crescenti, ossia sia $s_{L,j} = f_L(j)$. Si può quindi definire la funzione

$$\varphi: \wp(\mathbb{N}) \rightarrow S | \varphi(L) = (s_{L,j} | s_{L,j} = f_L(j))$$

φ è biettiva, infatti:

- è suriettiva poiché per ogni sequenza s_i esiste un sottoinsieme L avente per elementi tutti e soli i naturali che nella sequenza hanno valore 0;
- è iniettiva perché detti L_1, L_2 due sottoinsiemi distinti di \mathbb{N} le loro sequenze s_{L_1} e s_{L_2} saranno necessariamente diverse.

Quindi le cardinalità di \mathbb{R} e $\wp(\mathbb{N})$ sono uguali.

Teorema (di Cantor): sia A un insieme. Allora, $Card(A) < Card(\wp(A))$.

Dimostrazione: se A è finito e $Card(A) = n$, $Card(\wp(A)) = 2^n$ quindi il teorema è dimostrato.

Se A è infinito sia φ la funzione:

$$\varphi: A \rightarrow \wp(A) | \varphi(a) = \{a\}$$

Si osserva che φ esiste sempre iniettiva, dunque vale $Card(A) \leq Card(\varphi(A))$.
 Sia per assurdo $Card(A) = Card(\varphi(A))$, esiste quindi una funzione $\psi: A \rightarrow \varphi(A)$ biiettiva.
 Sia B l'insieme degli elementi che non appartengono alla loro immagine in ψ :

$$B = \{x | x \notin \psi(x)\}$$

Sia $b \in A$ tale che $\psi(b) = B$, un tale elemento deve esistere per la suriettività di ψ . Allora:

- se $b \in \psi(b)$ allora $b \in B$, ma per definizione di B questa è una contraddizione;
- se $b \notin \psi(b)$ allora $b \notin B$, ma questo è di nuovo in contraddizione con la definizione di B.

Quindi non può esistere una funzione $\psi: A \rightarrow \varphi(A)$ suriettiva, quindi neanche biiettiva, quindi $Card(A) \neq Card(\varphi(A))$ ma dato che $Card(A) \leq Card(\varphi(A))$ deve essere $Card(A) < Card(\varphi(A))$.

Definizione: le cardinalità degli insiemi infiniti si indicano con:

- $Card(\mathbb{N}) = \aleph_0$;
- $Card(\varphi(\mathbb{N})) = Card(\mathbb{R}) = \aleph_1$;
- $Card(\varphi(\varphi(\mathbb{N}))) = \aleph_2$;
- $Card(\varphi(\varphi(\varphi(\dots_n \text{ volte } \mathbb{N} \dots)))) = \aleph_n$.

I simboli \aleph_i sono collettivamente indicati con il nome di **numeri cardinali**.

Teorema: L'insieme T delle tuple di tutte le lunghezze di numeri naturali è numerabile.

Dimostrazione: sia F l'insieme delle stringhe infinite di bit che hanno solo zeri da una certa posizione in poi. F è numerabile, poiché detto f_i l'i-esimo bit di f si può definire la funzione:

$$\varphi: F \rightarrow \mathbb{N} | \varphi(f) = \sum_{i=0}^{\infty} 2^i f_i$$

ed essa è biiettiva.

A questo punto è possibile creare una funzione iniettiva $\psi: T \rightarrow F$ che, presa una tupla, restituisce la seguente codifica:

- per ogni numero della tupla t, una sequenza di t_i "1" dove t_i è l'i-esimo elemento della tupla;
- uno "0" di separazione tra le sequenze di "1" corrispondenti a diversi elementi della tupla;
- infiniti "0" in coda.

La funzione è iniettiva perché a tuple diverse fa corrispondere codifiche diverse. Si ha quindi che:

$$Card(T) \leq Card(F) = \aleph_0$$

ma T è infinito, dunque è numerabile.

Definizione: un **linguaggio** E è l'insieme delle sequenze di simboli dette **stringhe** prese da un insieme finito o numerabile Σ detto **alfabeto**.

Teorema: un linguaggio E è numerabile.

Dimostrazione: Σ è al più numerabile, quindi $\exists \varphi: \Sigma \rightarrow \mathbb{N}$ iniettiva. Sia inoltre:

$$\psi: E \rightarrow T | \psi(e) = (\varphi(e_1), \varphi(e_2), \dots)$$

dove gli e_i sono i simboli di una tupla e. ψ è iniettiva, quindi:

$$Card(E) \leq Card(T) = \aleph_0$$

ma E è infinito, dunque è numerabile.

Teorema: l'insieme α degli algoritmi è numerabile.

Dimostrazione: ogni algoritmo deve essere espresso da una stringa finita in un linguaggio E, quindi $\alpha \subseteq E$. $Card(E) = \aleph_0$ ed α è infinito, dunque α è numerabile.

Teorema: sia $\psi: \alpha \rightarrow \mathbb{N}$ la funzione biiettiva che mette in corrispondenza gli algoritmi con i numeri naturali (essa esiste sempre poiché α è numerabile). Allora non è possibile che, contemporaneamente:

- ψ sia computabile e
- tutti gli algoritmi computino funzioni totali.

Dimostrazione: sia per assurdo ψ computabile e ϕ_i le funzioni totali computate dagli

algoritmi. Sia inoltre $f: \mathbb{N} \rightarrow \mathbb{N}$ la funzione $f(x) = \phi_x(x) + 1$, $f = \phi_k$. Allora $f(k) = \phi_k(k)$, ma questo è in contraddizione con la definizione di f che risulterebbe in $\phi_k(k) + 1$. Quindi almeno una delle due condizioni deve essere falsa.

Teorema: l'insieme \mathcal{F} delle funzioni da \mathbb{N} in \mathbb{N} ha cardinalità \aleph_1 .

Dimostrazione: perché sia $\text{Card}(\mathcal{F}) = \aleph_1$ deve essere che $\text{Card}(\mathcal{F}) \leq \aleph_1$ e $\text{Card}(\mathcal{F}) \geq \aleph_1$.

Sia \mathcal{R} una relazione appartenente all'insieme R delle relazioni. Sia $\varphi: R \rightarrow \wp(\mathbb{N} \times \mathbb{N})$ la funzione che fa corrispondere ad ogni relazione l'insieme di coppie per le quali è verificata. φ è iniettiva, quindi $\text{Card}(R) \leq \aleph_1$. Sia $\psi: \mathcal{F} \rightarrow R$ la funzione che fa corrispondere agli insiemi di coppie che definiscono le funzioni gli insiemi di coppie che definiscono le relazioni. ψ è iniettiva quindi $\text{Card}(\mathcal{F}) \leq \text{Card}(R)$, in conclusione $\text{Card}(\mathcal{F}) \leq \aleph_1$.

Sia ora \mathcal{F}_{app} l'insieme delle funzioni di appartenenza da \mathbb{N} . Si ha che $\mathcal{F}_{app} \subseteq \mathcal{F}$, quindi

$\text{Card}(\mathcal{F}_{app}) \leq \text{Card}(\mathcal{F})$. Sia $f: \mathcal{F}_{app} \rightarrow \wp(\mathbb{N})$ la funzione che fa corrispondere a ogni funzione di appartenenza l'insieme dei naturali per i quali vale 0. La funzione è biiettiva, quindi

$\text{Card}(\mathcal{F}_{app}) = \aleph_1$, in conclusione $\text{Card}(\mathcal{F}) \geq \aleph_1$.

Quindi $\text{Card}(\mathcal{F}) = \aleph_1$.

Parte 2: Macchine di Turing

Definizione: un **alfabeto** $\Sigma = \{s_0, s_1, \dots\}$ è un insieme finito o numerabile di elementi detti **simboli** che contiene almeno l'elemento s_0 , detto **spazio vuoto**.

Definizione: un **insieme di stati** $Q = \{q_0, q_1, \dots\}$ è un insieme finito o numerabile di elementi detti **stati** che contiene almeno gli elementi q_0 e q_1 detti rispettivamente **stato iniziale** e **stato finale**.

Definizione: sia Σ un alfabeto e Q un insieme di stati. Una **configurazione** è una coppia $C = (q_i, s_j)$ dove $q_i \in Q$ è detto **stato corrente** e $s_j \in \Sigma$ è detto **simbolo letto**.

Definizione: sia Σ un alfabeto e Q un insieme di stati. Un'**operazione** è una tripla $O = (s_k, Sp, q_l)$ dove $s_k \in \Sigma$ è detto **simbolo scritto**, $Sp \in \{S, D, C\}$ è detto **spostamento** e $q_l \in Q$ è detto **stato successivo**.

Definizione: sia Σ un alfabeto e Q un insieme di stati, $C = (q_i, s_j)$ una configurazione e $O = (s_k, Sp, q_l)$ un'operazione su Σ e Q . Un'**istruzione** è una coppia $I = (C, O)$, indicata anche con la quintupla $I = (q_i, s_j, s_k, Sp, q_l)$.

Definizione: sia Σ un alfabeto e Q un insieme di stati. Un **programma** P è un'insieme di istruzioni definite su Σ e Q per cui non esistono due istruzioni I_1 e I_2 aventi la stessa configurazione.

Definizione: sia Σ un alfabeto, $Q = \{q_0, q_1, \dots\}$ un insieme di stati e P un programma su Σ e Q . Una macchina di Turing è una quintupla $MT = (\Sigma, Q, P, q_0, q_1)$.

Algoritmo (esecuzione di una macchina di Turing): sia $MT = (\Sigma, Q, P, q_0, q_1)$, sia N una sequenza infinita di simboli detta **nastro** indicizzata per numeri interi. La macchina di Turing è munita di una **testina che si trova in una posizione sul nastro** i su un simbolo N_i , inoltre ha uno **stato corrente** q_j in Q . La macchina opera come segue:

1. inizialmente la testina è posizionata in $i=0$, sopra il simbolo N_0 e nello stato iniziale $q_j = q_0$;
2. viene scelta da P l'istruzione I avente configurazione $C = (q_j, N_i)$. Se una tale istruzione non esiste il programma si dice **scorretto** e l'algoritmo termina.
3. viene eseguita l'operazione $O = (s_k, Sp, q_l)$ di I , ossia:
 1. N_i viene sostituito con s_k (si dice che avviene una **sovrascrittura**);
 2. i viene incrementato di 1 se $Sp = D$ (si dice che **la testina si sposta a destra**), oppure decrementato di 1 se $Sp = S$ (si dice che **la testina si sposta a sinistra**) oppure viene lasciato invariato se $Sp = C$ (si dice che **la testina resta ferma**);
 3. lo stato corrente muta in q_l (si dice che avviene un passaggio di stato);
4. se lo stato corrente è finale l'algoritmo termina, altrimenti torna al punto 2.

Nota: per convenzione, ove non diversamente specificato si suppone per tutte le macchine di Turing descritte che:

1. l'input possa essere un numero naturale o una tupla di numeri naturali;
2. un numero naturale n è codificato da una sequenza di $n+1$ simboli $|$ detti **barrette** consecutivi sul nastro;
3. una tupla è codificata da una sequenza di barrette come al punto due separate da uno spazio vuoto;

4. l'alfabeto sia quindi $\Sigma = \{s_0, |\}$;
5. la macchina si trovi inizialmente con la testina sulla barretta più a sinistra dell'input;
6. non sia rilevante la posizione finale della testina (ma solo l'output sul nastro).

Definizione: sia P un programma e $\varphi: D \rightarrow C$ una funzione totale. Si dice che P **computa** φ sse, se $x \in D$ e $\varphi(x) = y$, allora l'esecuzione di P produce la codifica di y sul nastro fornendo la codifica di x in input.

Definizione: sia P un programma e $\varphi: D \rightarrow C$ una funzione parziale con campo di esistenza E. Si dice che P **computa** φ sse, se $x \in E$ e $\varphi(x) = y$, allora l'esecuzione di P produce la codifica di y sul nastro fornendo la codifica di x in input altrimenti la MT non termina mai la computazione.

Definizione (programmi notevoli per MT):

- Programma che computa la funzione $zero: \mathbb{N} \rightarrow \{0\} | zero(x) = 0$:
 - $(q_0, |, s_0, D, q_0)$
 - $(q_0, s_0, |, C, q_1)$
- Programma che computa la funzione $doppio': \mathbb{N} \rightarrow \{0\} | doppio'(x) = 2x + 1$:
 - $(q_1, |, s_0, D, q_2)$
 - $(q_2, |, |, D, q_2)$
 - (q_2, s_0, s_0, D, q_3)
 - $(q_3, |, |, D, q_3)$
 - $(q_3, s_0, |, D, q_4)$
 - $(q_4, s_0, |, S, q_5)$
 - $(q_5, |, |, S, q_5)$
 - (q_5, s_0, s_0, S, q_6)
 - $(q_6, |, |, S, q_6)$
 - (q_6, s_0, s_0, D, q_1)
 - (q_1, s_0, s_0, C, q_0)
- Programma che computa la funzione $doppio: \mathbb{N} \rightarrow \{0\} | doppio(x) = 2x$:
 - $(q_1, |, s_0, D, q_2)$
 - $(q_2, |, |, D, q_2)$
 - (q_2, s_0, s_0, D, q_3)
 - $(q_3, |, |, D, q_3)$
 - $(q_3, s_0, |, D, q_4)$
 - $(q_4, s_0, |, S, q_5)$
 - $(q_5, |, |, S, q_5)$
 - (q_5, s_0, s_0, S, q_6)
 - $(q_6, |, |, S, q_6)$
 - (q_6, s_0, s_0, D, q_1)

- (q_1, s_0, s_0, D, q_7)
- $(q_7, |, s_0, C, q_0)$
- Programma che computa la funzione $\varphi: \mathbb{N} \rightarrow \{0\} | \varphi(x) = \begin{cases} 0 & \text{se } x \text{ è pari} \\ 1 & \text{se } x \text{ è dispari} \end{cases}$:
 - $(q_1, |, s_0, D, q_2)$
 - $(q_2, |, s_0, D, q_3)$ q_2 : stato dei numeri pari
 - $(q_3, |, s_0, D, q_2)$ q_3 : stato dei numeri dispari
 - $(q_2, s_0, |, C, q_0)$ unica barretta
 - $(q_3, s_0, |, D, q_4)$ prima barretta
 - $(q_4, s_0, |, C, q_0)$ seconda barretta
- Programma che computa la funzione $\varphi: \mathbb{N} \rightarrow \{0\} | \varphi(x) = \begin{cases} 0 & \text{se } x \text{ è pari} \\ \perp & \text{se } x \text{ è dispari} \end{cases}$:
 - $(q_1, |, s_0, D, q_2)$
 - $(q_2, |, s_0, D, q_3)$ q_2 : stato dei numeri pari
 - $(q_3, |, s_0, D, q_2)$ q_3 : stato dei numeri dispari
 - $(q_2, s_0, |, C, q_0)$ unica barretta
 - $(q_3, s_0, |, D, q_4)$ prima barretta
 - (q_4, s_0, s_0, C, q_4) loop

Definizione: una funzione $\varphi: \mathbb{N}^n \rightarrow \mathbb{N}^m$ è detta T-computabile sse esiste un programma che la computa.

Definizione: la funzione **zero** è definita come $zero: \mathbb{N} \rightarrow \mathbb{N} | zero(x) = 0$.

Definizione: la funzione **successore** è definita come $s: \mathbb{N} \rightarrow \mathbb{N} | s(x) = x + 1$.

Definizione: le funzioni **proiezione** sono definite come $P_i^n: \mathbb{N}^n \rightarrow \mathbb{N} | P_i^n(x_1, x_2, \dots, x_n) = x_i$.

Definizione: zero, s e P_i^n sono dette **funzioni base**.

Teorema: le funzioni base sono computabili e totali.

Definizione: siano $\varphi: \mathbb{N} \rightarrow \mathbb{N}$ e $\psi: \mathbb{N} \rightarrow \mathbb{N}$ due funzioni. La **composta** di φ e ψ è la funzione $\varphi \circ \psi: \mathbb{N} \rightarrow \mathbb{N} | \varphi \circ \psi(x) = \varphi(\psi(x))$.

Teorema: la composizione conserva la computabilità ossia, se φ e ψ sono computabili allora anche $\varphi \circ \psi$ è computabile. Inoltre, la composizione conserva la totalità.

Definizione: siano $\psi: \mathbb{N}^{n+2} \rightarrow \mathbb{N}$ e $\chi: \mathbb{N}^n \rightarrow \mathbb{N}$ funzioni. La **ricorsione** $\varphi: \mathbb{N}^{n+1} \rightarrow \mathbb{N}$ di ψ e χ è:

$$\varphi(x_1, \dots, x_n, y) = \begin{cases} \chi(x_1, \dots, x_n) & \text{se } y = 0 \\ \psi(x_1, \dots, x_n, z, \varphi(x_1, \dots, x_n, z)) & \text{se } y = s(z) \end{cases}$$

Nota: per brevità di scrittura la ricorsione si indica anche nel modo seguente

$$\varphi(x_1, \dots, x_n, y) = \begin{cases} \varphi(x_1, \dots, x_n, 0) = \chi(x_1, \dots, x_n) \\ \varphi(x_1, \dots, x_n, s(y)) = \psi(x_1, \dots, x_n, y, \varphi(x_1, \dots, x_n, y)) \end{cases}$$

Teorema: la ricorsione conserva la computabilità ossia, se $\psi: \mathbb{N}^{n+2} \rightarrow \mathbb{N}$ e $\chi: \mathbb{N}^n \rightarrow \mathbb{N}$ sono computabili allora anche $\varphi: \mathbb{N}^{n+1} \rightarrow \mathbb{N}$ ottenuta per ricorsione è computabile. Inoltre, la ricorsione conserva la totalità.

Definizione: una funzione si dice **ricorsiva primitiva** o **RP-derivabile** se può essere ottenuta tramite l'applicazione di una sequenza finita di composizioni e/o ricorsioni dalle funzioni base.

Definizione: l'insieme delle funzioni ricorsive primitive è denotato con **RP**.

Teorema: ogni funzione ricorsiva primitiva è computabile e totale.

Dimostrazione (intuitiva): le funzioni base sono tutte computabili; le rimanenti funzioni sono ottenute dall'applicazione di un numero finito di trasformazioni che conservano la computabilità, quindi sono anch'esse computabili. Analogamente, si prova la totalità.

Definizione: l'**addizione intera** è definita come $+: \mathbb{N}^2 \rightarrow \mathbb{N} | x + y = \begin{cases} x + 0 = x \\ x + s(z) = s(x + z) \end{cases}$.

Teorema: l'addizione intera è ricorsiva primitiva.

Dimostrazione: la seguente successione di funzioni contiene solo funzioni base oppure applicazioni di composizione o ricorsione:

$$\varphi_1(x) = P_1^1(x)$$

$$\varphi_2(x) = s(x)$$

$$\varphi_3(x, y, z) = P_3^3(x, y, z)$$

$$\varphi_4(x, y, z) = \varphi_3 \circ \varphi_2(x) = s(P_3^3(x, y, z))$$

$$\varphi_5(x, y) = \begin{cases} \varphi_5(x, 0) = \varphi_1(x) \\ \varphi_5(x, s(z)) = \varphi_4(x, z, \varphi_5(x, z)) \end{cases} = x + y$$

Definizione: la **moltiplicazione intera** è definita come $\cdot: \mathbb{N}^2 \rightarrow \mathbb{N} | x \cdot y = \sum_{i=0}^y x$.

Teorema: la moltiplicazione intera è ricorsiva primitiva.

Dimostrazione (intuitiva): $\cdot: \mathbb{N}^2 \rightarrow \mathbb{N} | x \cdot y = \begin{cases} x \cdot 0 = 0 \\ x \cdot s(z) = x \cdot z + x \end{cases}$.

Definizione: la funzione **precedente** è definita come $pr: \mathbb{N} \rightarrow \mathbb{N} | pr(x) = \begin{cases} 0 & \text{se } x = 0 \\ x - 1 & \text{se } x > 0 \end{cases}$.

Teorema: la funzione precedente è ricorsiva primitiva.

Dimostrazione (intuitiva): $pr(x) = \begin{cases} pr(0) = zero(x) \\ pr(s(y)) = P_1^1(y) \end{cases}$.

Definizione: la **sottrazione simmetrica** è definita come $\div: \mathbb{N}^2 \rightarrow \mathbb{N} | x \div y = \begin{cases} x - y & \text{se } x \geq y \\ 0 & \text{se } x < y \end{cases}$.

Teorema: la sottrazione simmetrica è ricorsiva primitiva.

Dimostrazione (intuitiva): $\div: \mathbb{N}^2 \rightarrow \mathbb{N} | x \div y = \begin{cases} x \div 0 = x \\ x \div s(z) = pr(x \div z) \end{cases}$.

Definizione: la **funzione distanza** è definita come $dist: \mathbb{N}^2 \rightarrow \mathbb{N} | (x \div y) + (y \div x)$.

Teorema: la funzione distanza è ricorsiva primitiva.

Dimostrazione (intuitiva): la funzione distanza è ottenuta per composizione di \div e $+$, che sono ricorsive primitive.

Definizione: la funzione **segno** è definita come $sgn(x) = \begin{cases} 0 & \text{se } x=0 \\ 1 & \text{se } x>0 \end{cases}$.

Teorema: la funzione segno è ricorsiva primitiva.

Dimostrazione (intuitiva): $sgn(y) = \begin{cases} sgn(0)=0 \\ sgn(s(y))=1 \end{cases}$.

Definizione: sia $P(x_1, x_2, \dots, x_n)$ un predicato su n naturali. La **funzione caratteristica** di P è definita come $f_P: \mathbb{N}^n \rightarrow \{0,1\} | f_P(x_1, x_2, \dots, x_n) = \begin{cases} 0 & \text{sse } P(x_1, x_2, \dots, x_n) \text{ è vero} \\ 1 & \text{sse } P(x_1, x_2, \dots, x_n) \text{ è falso} \end{cases}$.

Definizione: un predicato $P(x_1, x_2, \dots, x_n)$ si dice **ricorsivo primitivo** sse la sua funzione caratteristica è ricorsiva primitiva.

Definizione: siano $R(x_1, x_2, \dots, x_n)$ ed $S(x_1, x_2, \dots, x_n)$ predicati su naturali. La **coniunzione** $R \wedge S(x_1, x_2, \dots, x_n)$ è vera sse $R(x_1, x_2, \dots, x_n)$ ed $S(x_1, x_2, \dots, x_n)$ sono vere.

Teorema: siano $R(x_1, x_2, \dots, x_n)$ ed $S(x_1, x_2, \dots, x_n)$ predicati ricorsivi primitivi, allora la loro congiunzione è ricorsiva primitiva.

Dimostrazione (intuitiva): $f_{R \wedge S} = sgn(f_R(x_1, x_2, \dots, x_n) \cdot f_S(x_1, x_2, \dots, x_n))$.

Definizione: siano $R(x_1, x_2, \dots, x_n)$ ed $S(x_1, x_2, \dots, x_n)$ predicati su naturali. La **disgiunzione** $R \vee S(x_1, x_2, \dots, x_n)$ è vera sse $R(x_1, x_2, \dots, x_n)$ oppure $S(x_1, x_2, \dots, x_n)$ è vera.

Teorema: siano $R(x_1, x_2, \dots, x_n)$ ed $S(x_1, x_2, \dots, x_n)$ predicati ricorsivi primitivi, allora la loro disgiunzione è ricorsiva primitiva.

Dimostrazione (intuitiva): $f_{R \vee S} = 1 \div sgn[(1 \div f_R()) + (1 \div f_S())]$.

Definizione: sia $R(x_1, x_2, \dots, x_n)$ una relazione su n naturali. La **funzione caratteristica** di R è definita come $f_R: \mathbb{N}^n \rightarrow \{0,1\} | f_R(x_1, x_2, \dots, x_n) = \begin{cases} 0 & \text{sse } x_1, x_2, \dots, x_n \text{ sono in relazione R} \\ 1 & \text{altrimenti} \end{cases}$.

Definizione: una relazione $R(x_1, x_2, \dots, x_n)$ si dice **ricorsiva primitiva** sse la sua funzione caratteristica è ricorsiva primitiva.

Teorema: $R(x, y) = (x = y)$ è ricorsiva primitiva.

Dimostrazione (intuitiva): $f_{=} = sgn(dist(x, y))$.

Teorema: $P(x, y) \neq (x = y)$ è ricorsiva primitiva.

Dimostrazione (intuitiva): $f_{\neq} = 1 \div sgn(dist(x, y))$.

Definizione: un insieme A si dice **ricorsivo primitivo** sse la sua funzione caratteristica è ricorsiva primitiva.

Teorema: A è ricorsivo primitivo sse \bar{A} lo è.

Dimostrazione (intuitiva): $f_{\bar{A}}(x) = 1 \div f_A(x)$.

Teorema: A è ricorsivo generale sse \bar{A} lo è.

Dimostrazione (intuitiva): $f_{\bar{A}}(x) = 1 \div f_A(x)$.

Teorema: esistono funzioni computabili non ricorsive primitive.

Dimostrazione: RP è enumerabile, infatti è possibile dare una numerazione alle funzioni ottenute

tramite $N=0,1,2,\dots$ applicazioni di composizione e ricorsione. Quindi esiste una successione di funzioni $\varphi_1, \varphi_2, \dots$ che enumera le funzioni in RP. Sia $f(x) = \varphi_x(x) + 1$, essa è evidentemente computabile ma non appartiene a RP. Se per assurdo appartenesse a RP allora dovrebbe esistere un $k \in \mathbb{N}$ tale che $f(k) = \varphi_k(k)$, calcolando $f(k)$ si giungerebbe a una contraddizione:
 $f(k) = \varphi_k(k) + 1 = \varphi_k(k) + 1$.

Definizione: sia $P(x_1, x_2, \dots, x_n, y)$ un predicato su $n+1$ naturali. La **minimizzazione** di P dati x_1, x_2, \dots, x_n è la quantità $\mu_y P(x_1, x_2, \dots, x_n, y) = \text{minimo } y \text{ che rende vero il predicato}$.

Definizione: sia $R(x_1, x_2, \dots, x_n, y)$ una relazione su $n+1$ naturali. La **minimizzazione** di R dati x_1, x_2, \dots, x_n è la quantità $\mu_y R(x_1, x_2, \dots, x_n, y) = \text{minimo } y \text{ in relazione con } x_1, x_2, \dots, x_n$.

Definizione: sia $\lambda: \mathbb{N}^{n+1} \rightarrow \mathbb{N}$ una funzione. La **minimalizzazione** $\varphi: \mathbb{N}^n \rightarrow \mathbb{N}$ di λ è la funzione $\varphi(x_1, x_2, \dots, x_n) = \mu_y [\lambda(x_1, x_2, \dots, x_n, y) = 0]$.

Teorema: la minimalizzazione conserva la computabilità ma non la totalità.

Definizione: una funzione si dice **ricorsiva generale** o **RG-derivabile** se può essere ottenuta tramite l'applicazione di una sequenza finita di composizioni, ricorsioni e/o minimalizzazioni dalle funzioni base.

Definizione: l'insieme delle funzioni ricorsive generali è denotato con **RG**.

Teorema: $RP \subset RG$.

Teorema: la funzione $\varphi(x, y) = \begin{cases} \frac{x}{y} & \text{se } x \text{ è multiplo di } y \\ \perp & \text{altrimenti} \end{cases}$ è ricorsiva generale.

Dimostrazione (intuitiva): Sia $\frac{x}{y} = z$, ossia $x = yz$; allora $\varphi(x, y) = \mu_z [\text{dist}(x, yz) = 0]$.

Definizione: un predicato $P(x_1, x_2, \dots, x_n)$ si dice **ricorsivo generale** sse la sua funzione caratteristica è ricorsiva generale.

Definizione: una relazione $R(x_1, x_2, \dots, x_n)$ si dice **ricorsiva generale** sse la sua funzione caratteristica è ricorsiva generale.

Definizione: un insieme A si dice **ricorsivo generale** sse la sua funzione caratteristica è ricorsiva generale.

Definizione: un insieme A si dice **ricorsivamente enumerabile** sse:

- A è vuoto oppure
- A è il rango di una funzione ricorsiva generale totale.

Teorema: A è ricorsivo generale sse A e \bar{A} sono ricorsivamente enumerabili.

Dimostrazione:

- sia A ricorsivo generale. Allora la sua funzione caratteristica f_A è ricorsiva generale.
 - se $A = \emptyset$ allora A è ricorsivamente enumerabile per definizione, \bar{A} è il rango della funzione identità $f: \mathbb{N} \rightarrow \mathbb{N} | f(x) = x$;
 - se $A \neq \emptyset$ allora A è il rango di $f(x) = [1 \div f_A(x)]x + f_A(x)\mu_y(f_A(P_2^2(x, f_A(y))))$ (la funzione di cui \bar{A} è il rango si ottiene analogamente);
- siano A e \bar{A} ricorsivamente enumerabili, sono il rango di due funzioni ricorsive generali

totali (rispettivamente φ e ψ). Quindi:

- se $A = \emptyset$ allora $f_A(x) = 1, f_A \in RP$;
- se $A \neq \emptyset$. Allora ogni x appartiene al rango di φ o al rango di ψ (poiché $A \cup \bar{A} = \mathbb{N}$). Sia $\lambda(x) = \mu_y(\text{dist}(\varphi(y), x), \text{dist}(\psi(y), x))$ (il più piccolo indice di enumerazione che mette x in A o \bar{A}), allora $h(x) = \text{sgn}(\text{dist}(\varphi(\lambda(x)), x))$, è la funzione caratteristica di A .

Teorema (triangolare): sono equivalenti le seguenti affermazioni:

1. A è ricorsivamente enumerabile;
2. A è il campo di esistenza di una funzione ricorsiva generale parziale;
3. A è il rango di una funzione ricorsiva generale parziale.

Algoritmo (di Godelizzazione): sia $\Sigma = s_0, s_1, \dots, s_n$ un alfabeto e sia $MT = (\Sigma, Q, P, q_0, q_1)$ una macchina di Turing con quell'alfabeto ad una certa iterazione della sua esecuzione. Siano N_i i simboli sul nastro. L'algoritmo dà una configurazione della situazione come numero naturale:

1. siano p_0, p_1, \dots, p_n i primi $n+1$ numeri primi;
2. sia u la codifica dei simboli a sinistra della testina definito come $u = \sum_{i=-1}^{-\infty} p_{-i}^{N_i}$;
3. sia v la codifica dei simboli a destra della testina definito come $v = \sum_{i=1}^{\infty} p_i^{N_i}$;
4. sia w la codifica dell'intera situazione della macchina di Turing definita come $w = 2^u 3_0^N 5_{corrente}^q 7^v$.

Definizione: sia w una situazione di una macchina di Turing ottenuta per Godelizzazione, si definisce la funzione ρ_{MT} :

$$\rho_{MT}: \mathbb{N} \rightarrow \mathbb{N} = \begin{cases} w', & \text{situazione successiva per Godelizzazione} \\ w & \text{sse } w \text{ è una situazione non finale} \\ & \text{altrimenti} \end{cases}$$

Teorema: $\rho_{MT}: \mathbb{N} \rightarrow \mathbb{N}$ è ricorsiva primitiva.

Definizione: sia $MT = (\Sigma, Q, P, q_0, q_1)$ una macchina di Turing, si definisce la funzione θ_{MT} :

$$\theta_{MT}: \mathbb{N}^2 \rightarrow \mathbb{N} \mid \theta_{MT}(w, y) = \begin{cases} \theta_{MT}(w, 0) = w \\ \theta_{MT}(w, s(z)) = \rho_{MT}(\theta_{MT}(w, z)) \end{cases}$$

θ_{MT} calcola la codifica della situazione che si ottiene dopo y passi partendo dalla codifica della situazione w .

Teorema: $\theta_{MT}: \mathbb{N} \rightarrow \mathbb{N}$ è ricorsiva primitiva.

Dimostrazione (intuitiva): θ_{MT} si ottiene per ricorsione da ρ_{MT} .

Teorema (di Turing): sono T-computabili tutte e sole le funzioni ricorsive generali parziali.

Dimostrazione (le funzioni ricorsive generali parziali sono T-computabili, intuitiva): le funzioni base sono tutte T-computabili; inoltre composizione, ricorsione e minimalizzazione conservano la T-computabilità. Infatti:

- la funzione zero è implementabile in una macchina di Turing con il seguente programma:
 - $(q_1, |, s_0, D, q_1)$
 - $(q_1, s_0, |, C, q_0)$
- la funzione successore è implementabile in una macchina di Turing con il seguente programma:

- $(q_1, |, |, D, q_1)$
- $(q_1, s_0, |, C, q_0)$
- tutte le funzioni proiezione sono implementabili in una Macchina di Turing. Ad esempio P_2^2 :
 - $(q_1, |, s_0, D, q_1)$
 - (q_1, s_0, s_0, D, q_0)
- la composizione conserva la T-computabilità. Supponendo $\psi = \varphi \circ \chi$, φ T-computabile tramite una macchina $MT\varphi$, χ T-computabile tramite una macchina $MT\chi$ si può costruire una macchina $MT\psi$ che computa ψ con un programma strutturato come segue:
 - esegue $MT\varphi$;
 - riposiziona la testina alla sinistra dell'output;
 - esegue $MT\chi$;

- la ricorsione conserva al T-computabilità. Sia, ad esempio $\varphi(x) = \begin{cases} \varphi(0) = k \\ \varphi(s(x)) = \psi(x, \varphi(x)) \end{cases}$, ψ T-computabile tramite una macchina $MT\psi$.

Allora un programma strutturato come il seguente computa φ :

- scrive sul nastro la configurazione $\$1k+1$ barre $\$2x$ barre $\$3$;
- cancella una barra tra $\$2$ e $\$3$. Se non ci sono più barre, l'output è tra $\$1$ e $\$2$;
- modifica il nastro in modo che risulti $\$1|s_0k$ barre $\$2x-1$ barre $\$3|s_0k$ barre. Si applica $MT\psi$ a destra di $\$3$, viene calcolato $\varphi(1)$;
- ripeto il ciclo, cancellando una barra tra $\$2$ e $\$3$ se non ci sono più barre l'algoritmo termina. Altrimenti aggiungo una barra tra $\$1$ e $\$2$, aggiorno il risultato, copio il contenuto oltre $\$3$ e rilancio $MT\psi$.
- la minimalizzazione conserva al T-computabilità. Sia $\varphi(x) = \mu_y(\lambda(x, y) = 0)$, λ T-computabile tramite una macchina $MT\lambda$:
 - si scrive sul nastro la configurazione $\$1x+1$ barre $s_0|$ barre $\$2\3 ;
 - copia tra $\$2\3 il contenuto tra $\$1\2 ;
 - applica $MT\lambda$ all'input tra $\$2\3 ;
 - se tra $\$2\3 resta una sola barra, l'output sono le barre tra s_0 ed $\$2$. Altrimenti si aggiunge una barra prima di $\$2$ e ripete il ciclo.

Per induzione, tutte le funzioni in RG sono T-computabili.

Dimostrazione (le funzioni T-computabili sono ricorsive generali parziali, intuitiva): sia φ una funzione T-computabile, allora esiste $MT\varphi$ che la computa. Quindi esistono $\rho_{MT\varphi}$ e

$\theta_{MT\varphi}$, posso calcolare $z^* = \mu_z[\theta_{MT\varphi}(w_1, z)]$ è la codifica di una situazione finale. Trovato z^* si può decodificare $\theta_{MT\varphi}(w_1, z^*)$ ottenendo l'output della funzione; si dimostra che questo procedimento produce una funzione in RG.

Tesi (di Church): ogni funzione è computabile sse è T-computabile.

Corollario: ogni funzione è computabile sse è ricorsiva generale.

Nota: da questo punto in avanti si assume che la tesi sia vera.

Teorema: dato un alfabeto Σ è sempre possibile trovare una funzione biunivoca

$c: \Sigma \rightarrow \{ |, s_0 \}^n$, con n intero opportuno, che ne codifica i simboli in sequenze di barre e spazi.

Dimostrazione: essendo Σ numerabile è possibile associare ad ogni simbolo un naturale k.

Quindi, è sempre codificare ogni simbolo con un certo numero di barre (ad esempio k+1).

Teorema: data una macchina di Turing $MT = (\Sigma, Q, P, q_0, q_1)$ è sempre possibile trovare una macchina di Turing "equivalente" $MT' = (\{s_0, |\}, Q, P, q_0, q_1)$ che, data in input una codifica dell'input in barre computa una codifica dell'output di MT in barre.

Teorema: l'insieme IMT delle macchine di Turing con alfabeto $\{s_0, |\}$ è enumerabile.

Dimostrazione (intuitiva): ogni MT è completamente determinata dall'insieme delle sue tuple, la cui cardinalità è indicata con n. E' possibile enumerare, ad esempio in ordine alfabetico, le rappresentazioni in forma di stringa di tutti i possibili insiemi di tuple di lunghezza n crescente da 1 in poi, quindi è possibile costruire un algoritmo che enumeri gli elementi di IMT.

Teorema: fissato un alfabeto, è possibile determinare una macchina di Turing in modo univoco tramite una sua codifica per Godelizzazione o equivalentemente tramite il suo indice i nell'enumerazione MT_i .

Nota: da questo punto in avanti si assume sempre alfabeto $\{s_0, |\}$.

Definizione: la macchina di Turing Universale prende in input l'indice i di una macchina di Turing MT_i e la codifica di un input, e fornisce in output lo stesso output che la macchina MT_i produrrebbe con quell'input.

Teorema: la macchina di Turing Universale esiste.

Teorema (indecidibilità dell'arresto): fissato un input, l'insieme delle macchine di Turing che terminano con quell'input non è decidibile. Ossia, non esiste un algoritmo che dato un input e una macchina di Turing sia in grado di stabilire se essa termina in un numero finito di passi.

Dimostrazione: sia per assurdo H l'algoritmo che, data una codifica C_{MT} di macchina di Turing MT e una codifica C_I di un input I, fornisce in output 0 se MT con input I termina, 1 altrimenti, ossia:

$$H(C_{MT}, C_I) = \begin{cases} 0 & \text{se } MT(I) \text{ termina} \\ 1 & \text{se } MT(I) \text{ non termina} \end{cases}$$

Siano inoltre:

$$H'(C_{MT}) = H(C_{MT}, C_{MT}) = \begin{cases} 0 & \text{se } MT(C_{MT}) \text{ termina} \\ 1 & \text{se } MT(C_{MT}) \text{ non termina} \end{cases}$$

e

$$Z(C_{MT}) = \begin{cases} \text{non termina} & \text{se } H'(C_{MT}) = 0 \\ 0 & \text{se } H'(C_{MT}) = 1 \end{cases}$$

A questo punto:

1. se $Z(C_Z)$ termina allora $H'(C_Z) = 1$, allora $Z(C_Z)$ non termina, il che è assurdo;
2. se $Z(C_Z)$ non termina allora $H'(C_Z) = 0$, allora $Z(C_Z)$ termina, il che è assurdo.

Pertanto si può concludere che un tale algoritmo H non esista.

Dimostrazione (alternativa): sia $T(m, k, y)$ il predicato vero sse MT_m con input k si ferma dopo y iterazioni di computazione.

Si ha che T è computabile, infatti è sufficiente che MT_T che esegua i seguenti passi:

- scrittura di k sul nastro;
- esecuzione di y iterazioni della macchina MT_m ;
- controllo dello stato (se finale o meno).

Sia $T'(m, k) = \exists y | T(m, k, y)$ il predicato vero sse MT_m con input k termina.

Sia $T''(x) = \exists y | T(x, x, y)$ è il predicato vero sse MT_x con input x termina.

Si ha che T'' non è ricorsiva generale.

Infatti, sia per assurdo T'' ricorsiva generale. Allora la sua funzione caratteristica:

$$f_{T''}(x) = \begin{cases} 0 & \text{se } T''(x) \text{ è vero} \\ 1 & \text{altrimenti} \end{cases}$$

ossia:

$$f_{T''}(x) = \begin{cases} 0 & \text{se } MT_x(x) \text{ termina} \\ 1 & \text{altrimenti} \end{cases}$$

è computabile. Quindi si può trovare l'indice h della macchina di Turing MT_h tale che:

$$MT_h(x) = \begin{cases} \text{non termina} & \text{se } f_{T''}(x) = 0 \\ 1 & \text{se } f_{T''}(x) = 1 \end{cases}$$

Allora:

- se $f_{T''}(h) = 0$ allora $MT_h(h)$ termina ma allora $f_{T''}(h) = 1$, il che è assurdo;
- se $f_{T''}(h) = 1$ allora $MT_h(h)$ non termina ma allora $f_{T''}(h) = 0$, il che è assurdo.

Essendo T'' un caso particolare di T' , neanche essa è ricorsiva generale. Ma allora, per la tesi di Church, non esiste algoritmo in grado di determinare se una macchina di Turing con input fissato termina.

Teorema: fissato un input, l'insieme delle macchine di Turing che terminano con quell'input è semidecidibile. Ossia, fissato un input k , l'insieme delle funzioni ricorsive generali definite per quell'input è semidecidibile.

Dimostrazione: sia φ_m l'enumerazione delle funzioni ricorsive generali. Sia $\Phi(m, k) = \varphi_m(k)$

$$\text{e } zero \circ \Phi(m, k) = \begin{cases} 0 & \text{se } \Phi(m, k) \text{ è definita} \\ \perp & \text{se } \Phi(m, k) \text{ non è definita} \end{cases}$$

Quest'ultima funzione è ricorsiva generale per costruzione, quindi le coppie (m, k) sono il campo di esistenza di una funzione ricorsiva generale parziale. Per il teorema triangolare, quindi, vale che l'insieme $\{(m, k) \mid \Phi(m, k) \text{ è definita}\}$ è enumerabile, ossia semidecidibile.

Teorema: l'insieme delle funzioni ricorsive generali totali non è decidibile.

Dimostrazione: sia φ_k l'enumerazione delle funzioni ricorsive generali e sia $\psi(k)$ la funzione caratteristica delle funzioni ricorsive generali totali. Per assurdo, sia ψ computabile.

$$\text{Allora, sia } \chi(x) = \begin{cases} \varphi_x(x) + 1 & \text{se } \varphi_x \text{ è totale, ossia } \psi(x) = 0 \\ 0 & \text{se } \varphi_x \text{ è parziale, ossia } \psi(x) = 1 \end{cases}$$

χ è chiaramente computabile, inoltre è anche totale poiché $\varphi_x(x)$ viene calcolata solo se φ_x è totale.

Essendo computabile, esiste k tale che $\varphi_k = \chi$, ed essendo totale $\chi(k)$ è definito. Si ha che $\chi(k) = \varphi_k(k)$ perché χ è nella sequenza, inoltre $\chi(k) = \varphi_k(k) + 1$ per definizione di χ , il che è assurdo.

Teorema: l'insieme delle funzioni ricorsive generali totali non è neanche semidecidibile.

Dimostrazione: sia $T = \{x \in \mathbb{N} \mid \varphi_x \text{ è totale}\}$ l'insieme degli indici delle funzioni totali.

Per assurdo, sia T semidecidibile. Allora, dal momento che T non è vuoto, T deve essere il rango di una funzione computabile totale $\psi: \mathbb{N} \rightarrow T$. Quindi, deve essere possibile enumerare le funzioni computabili totali come segue: $\varphi_{\psi(0)}, \varphi_{\psi(1)}, \varphi_{\psi(2)}, \dots$

Sia $\chi(x) = \varphi_{\psi(x)} + 1$, χ è totale, quindi esiste k tale che $\chi = \varphi_{\psi(k)}$. Allora

$$\chi(k) = \varphi_{\psi(k)} = \varphi_{\psi(k)} + 1, \text{ che è assurdo.}$$