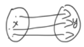
$\Phi = \{ f : \underline{N} \to N \}$



(x, y)
$(\underline{x}, \underline{f(x)})$

$f \to \{ (0, f(0)) ; (1, f(1)) ; (2, f(2)) ; (3, f(3)) ; \dots \} \in \mathcal{P}(\mathbb{N} \times \mathbb{N})$

Per ogni funzione $f : \mathbb{N} \to \mathbb{N}$ esiste un elemento di $\mathcal{P}(\mathbb{N}\times\mathbb{N})$ univocamente determinato.
$f_1 \neq f_2$ allora $\underset{\mathcal{P}(\mathbb{N}\times\mathbb{N})}{e_1} \neq \underset{\mathcal{P}(\mathbb{N}\times\mathbb{N})}{e_2}$

Si crea cioè una f. INIETTIVA da Φ a $\mathcal{P}(\mathbb{N}\times\mathbb{N})$

$Card(\Phi) \leq Card\,\mathcal{P}(\mathbb{N}\times\mathbb{N}) = \aleph_1$

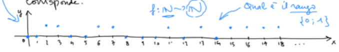
$Card(\Phi)$ < finita / $\aleph_0$ (ALEPH) / $\aleph_1$? $Card(\Phi) \leq \aleph_1$ (A)

Giriamo il discorso:
consideriamo di nuovo l'insieme G (stringhe binarie infinite) e confrontiamolo con Φ (funzioni $\mathbb{N} \to \mathbb{N}$)

Prendiamo $g \in G$
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 ....
g = 0 0 1 1 0 0 1 1 0 0 0 1 0 1 0 1 1 1 1 0 1 0 0 0 1 1 1 1 1 1 0 . . .

Per ogni elemento g, esiste una funzione $\mathbb{N} \to \mathbb{N}$ che gli corrisponde.

$f : \underline{\mathbb{N}} \to \mathbb{N}$ Qual è il range {0; 1}



$g = f(0) f(1) f(2) f(3) f(4) f(5) f(6) \dots$
Per ogni stringa binaria infinita g esiste una funzione f univocamente determinata. Ossia esiste una funzione da G a Φ iniettiva
$g_1 \neq g_2 \to f_1 \neq f_2$

$\aleph_1 = Card(G) \leq Card(\Phi)$ (B)

$\underbrace{\aleph_1 \leq \overbrace{Card(\Phi) \leq \aleph_1}^{A}}_{B}$

L'unico modo perché i due risultati coesistano senza contraddizione è che
$\boxed{Card(\Phi) = \aleph_1}$

Esistono tante funzioni da $\mathbb{N}$ a $\mathbb{N}$ quanti sono i punti geometrici di una retta (del segmento (0,1)).

Computabilità

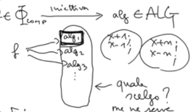
Una f si dice computabile quando esiste un algoritmo che ci fornisce f(x) dato x.

Data una funzione f, essa è < computabile : esiste un ALG / oppure ↕ MUTUAMENTE ESCLUSIVI / non computabile : NON esiste un ALG
saperlo / scoprirlo / ... → EPISTEMOLOGIA ← noi lo sappiamo?
esistenza / non esistenza / ... → ONTOLOGIA ←

$\Phi = \Phi_{COMP} \cup \Phi_{NC}$
Φ: ha cardinalità $\aleph_1$; Φ_{COMP}: computabili; Φ_{NC}: non computabili

$\Phi_{COMP} \cap \Phi_{NC} = \emptyset$

$Card(\Phi_{COMP})$
Per ogni f. computabile, per definizione esiste un algoritmo che la computa.
Quindi $f \in \Phi_{COMP} \xrightarrow{\text{iniettiva}} alg \in ALG$



← quale scelgo? me ne serve uno e uno solo.
· Scelgo quello più breve (in termini di lunghezza)
· e se lunghezza uguale... ordine lessicografico.

Con questi criteri per ogni funzione esiste un solo algoritmo univocamente determinato.
Cioè esiste una $f : \Phi_{COMP} \to ALG$ iniettiva
cioè $Card(\Phi_{COMP}) \leq Card(ALG) = \aleph_0$

$Card(\Phi_{COMP}) \leq \aleph_0$
finita / $\aleph_0$

per ogni $n \in \mathbb{N}$
$f(x) = n$
esistono $\underbrace{\aleph_0}_{\text{almeno}}$ funzioni computabili

$\aleph_0 \leq Card(\Phi_{COMP}) \leq \aleph_0$
$\hookrightarrow Card(\Phi_{COMP}) = \aleph_0$

$\Phi = \Phi_{COMP} \cup \Phi_{NC}$
$\aleph_1$ $\aleph_0$? $\aleph_1$

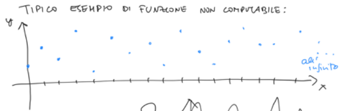
Φ_{NC} non può essere finito né $\aleph_0$, perché potremmo creare una sequenza numerabile di funzioni.

Due insiemi A e B numerabili
$a_0 b_0\ a_1 b_1\ a_2 b_2\ a_3 b_3 \dots$
la loro unione è numerabile

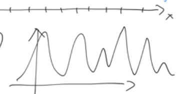
Perciò Φ_{NC} ha cardinalità $\aleph_1$.

$Card(\Phi_{COMP}) = \aleph_0$ $Card(\Phi_{NC}) = \aleph_1$

TIPICO ESEMPIO DI FUNZIONE NON COMPUTABILE:



$y = f(x)$?



$E = mc^2$

$\cancel{f = g\frac{m_1 m_2}{r^2}}$

$\Phi = \{ f : \underline{N} \to N \}$
f ha come input un solo numero.
MA! $f : \mathbb{N}^n \to \mathbb{N}$
$f(x_1, x_2, \dots, x_n) = y$

Abbiamo studiato un caso particolare? NO!
Non perdiamo in generalità lavorando su $f : \mathbb{N} \to \mathbb{N}$ perché:
$\to (k_1, \dots, k_n)$
$t(k_1, \dots, k_n) = 0k_1^* 0k_2^* \cdots 0k_n^*$
k_j^* : $k_j + 1$ caratteri '1'

mappo le n-uple su stringhe binarie finite.

Una volta che ho una stringa binaria
$b = b_0 b_1 \dots b_m$
ottengo subito e univocamente, un numero naturale:
$n(b_0 b_1 \dots b_m) = \sum^{m} (2^i \cdot b_i)$