

# Tre passi in Teoria della Calcolabilità

Andrea Asperti

Department of Computer Science, University of Bologna  
Mura Anteo Zamboni 7, 40127, Bologna, ITALY  
aspersi@cs.unibo.it

# Content

- 1 Numerabilità e diagonalizzazione
  - Dovetailing
  - Sequenze e parti finite
  - Il teorema di Cantor
- 2 Divagazioni sul tema della diagonalizzazione
  - Breve Escursus in Deduzione Naturale
  - Il problema della definibilità
- 3 La tesi di Church
  - Incompletezza dei formalismi totali
  - Le funzioni parziali calcolabili
  - Macchine di Turing
- 4 Funzioni calcolabili ... o meno
  - Il problema della terminazione
  - La proprietà s.m.n.
  - Il problema della equivalenza di programmi
  - Tre funzioni simili

# Content

- 1 Numerabilità e diagonalizzazione
  - Dovetailing
  - Sequenze e parti finite
  - Il teorema di Cantor
- 2 Divagazioni sul tema della diagonalizzazione
  - Breve Escursus in Deduzione Naturale
  - Il problema della definibilità
- 3 La tesi di Church
  - Incompletezza dei formalismi totali
  - Le funzioni parziali calcolabili
  - Macchine di Turing
- 4 Funzioni calcolabili ... o meno
  - Il problema della terminazione
  - La proprietà s.m.n.
  - Il problema della equivalenza di programmi
  - Tre funzioni simili

# Content

- 1 Numerabilità e diagonalizzazione
  - Dovetailing
  - Sequenze e parti finite
  - Il teorema di Cantor
- 2 Divagazioni sul tema della diagonalizzazione
  - Breve Escursus in Deduzione Naturale
  - Il problema della definibilità
- 3 La tesi di Church
  - Incompletezza dei formalismi totali
  - Le funzioni parziali calcolabili
  - Macchine di Turing
- 4 Funzioni calcolabili ... o meno
  - Il problema della terminazione
  - La proprietà s.m.n.
  - Il problema della equivalenza di programmi
  - Tre funzioni simili

# Content

- 1 Numerabilità e diagonalizzazione
  - Dovetailing
  - Sequenze e parti finite
  - Il teorema di Cantor
- 2 Divagazioni sul tema della diagonalizzazione
  - Breve Escursus in Deduzione Naturale
  - Il problema della definibilità
- 3 La tesi di Church
  - Incompletezza dei formalismi totali
  - Le funzioni parziali calcolabili
  - Macchine di Turing
- 4 Funzioni calcolabili ... o meno
  - Il problema della terminazione
  - La proprietà s.m.n.
  - Il problema della equivalenza di programmi
  - Tre funzioni simili

# Content

- 1 Numerabilità e diagonalizzazione
  - Dovetailing
  - Sequenze e parti finite
  - Il teorema di Cantor
- 2 Divagazioni sul tema della diagonalizzazione
  - Breve Escursus in Deduzione Naturale
  - Il problema della definibilità
- 3 La tesi di Church
  - Incompletezza dei formalismi totali
  - Le funzioni parziali calcolabili
  - Macchine di Turing
- 4 Funzioni calcolabili ... o meno
  - Il problema della terminazione
  - La proprietà s.m.n.
  - Il problema della equivalenza di programmi
  - Tre funzioni simili

# Content

- 1 Numerabilità e diagonalizzazione
  - Dovetailing
  - Sequenze e parti finite
  - Il teorema di Cantor
- 2 Divagazioni sul tema della diagonalizzazione
  - Breve Escursus in Deduzione Naturale
  - Il problema della definibilità
- 3 La tesi di Church
  - Incompletezza dei formalismi totali
  - Le funzioni parziali calcolabili
  - Macchine di Turing
- 4 Funzioni calcolabili ... o meno
  - Il problema della terminazione
  - La proprietà s.m.n.
  - Il problema della equivalenza di programmi
  - Tre funzioni simili

# Content

- 1 Numerabilità e diagonalizzazione
  - Dovetailing
  - Sequenze e parti finite
  - Il teorema di Cantor
- 2 Divagazioni sul tema della diagonalizzazione
  - Breve Escursus in Deduzione Naturale
  - Il problema della definibilità
- 3 La tesi di Church
  - Incompletezza dei formalismi totali
  - Le funzioni parziali calcolabili
  - Macchine di Turing
- 4 Funzioni calcolabili ... o meno
  - Il problema della terminazione
  - La proprietà s.m.n.
  - Il problema della equivalenza di programmi
  - Tre funzioni simili

# Outline

- 1 Numerabilità e diagonalizzazione
  - Dovetailing
  - Sequenze e parti finite
  - Il teorema di Cantor
- 2 Divagazioni sul tema della diagonalizzazione
  - Breve Escursus in Deduzione Naturale
  - Il problema della definibilità
- 3 La tesi di Church
  - Incompletezza dei formalismi totali
  - Le funzioni parziali calcolabili
  - Macchine di Turing
- 4 Funzioni calcolabili ... o meno
  - Il problema della terminazione
  - La proprietà s.m.n.
  - Il problema della equivalenza di programmi
  - Tre funzioni simili

# Numerabilità

Un insieme  $A$  si dice *numerabile* se esiste una funzione suriettiva  $f$  dall'insieme dei numeri naturali  $\mathcal{N}$  in  $A$ .  
 $f$  è detta *funzione di enumerazione*.

- $\mathcal{N}$  è numerabile
- Ogni sottoinsieme di un insieme numerabile è ancora numerabile.

Che possiamo dire dei soprainsiemi di  $\mathcal{N}$ ?

# Numerabilità

Un insieme  $A$  si dice *numerabile* se esiste una funzione suriettiva  $f$  dall'insieme dei numeri naturali  $\mathcal{N}$  in  $A$ .  
 $f$  è detta *funzione di enumerazione*.

- $\mathcal{N}$  è numerabile
- Ogni sottoinsieme di un insieme numerabile è ancora numerabile.

Che possiamo dire dei soprainsiemi di  $\mathcal{N}$ ?

# Numerabilità

Un insieme  $A$  si dice *numerabile* se esiste una funzione suriettiva  $f$  dall'insieme dei numeri naturali  $\mathcal{N}$  in  $A$ .  
 $f$  è detta *funzione di enumerazione*.

- $\mathcal{N}$  è numerabile
- Ogni sottoinsieme di un insieme numerabile è ancora numerabile.

Che possiamo dire dei soprainsiemi di  $\mathcal{N}$ ?

# Aggiungere un elemento

## Lemma

*Sia  $A$  numerabile. Allora  $\{*\} \oplus A$  è ancora numerabile.*

Sia  $f : \mathcal{N} \rightarrow A$  la funzione di enumerazione di  $A$ . Definiamo  $g : \mathcal{N} \rightarrow \{*\} \oplus A$  nel modo seguente:

$$g(x) = \begin{cases} g(0) = * \\ g(x+1) = f(x) \end{cases}$$

Chiaramente  $g$  è suriettiva.

**Corollario:** Sia  $A$  numerabile e  $D$  finito. Allora  $D \oplus A$  è ancora numerabile.

# Aggiungere un numero infinito di elementi

## Lemma

*L'unione di due insiemi numerabili  $A$  e  $B$  è ancora numerabile.*

Siano  $f$  e  $g$  le due funzioni di enumerazione di  $A$  e  $B$ .  $A \oplus B$  è allora enumerato dalla seguente funzione  $h$ :

$$h(x) = \begin{cases} h(2x) = f(x) \\ h(2x + 1) = g(x) \end{cases}$$

**Corollario:** Un'unione finita di insiemi numerabili è numerabile.

**Corollario:** Se  $D$  è finito e  $A$  è numerabile allora  $D \times A$  è numerabile.

# Moltiplicare per un numero infinito di elementi

Cerchiamo una funzione biunivoca da  $\mathcal{N} \times \mathcal{N}$  in  $\mathcal{N}$ .

Un modo alternativo di descrivere il problema consiste nella definizione di una politica di visita delle coppie  $\langle i, j \rangle$  del piano in modo tale che ogni coppia venga ispezionata una ed una sola volta al passo  $k$ .  $k$  è la *codifica* della coppia  $\langle i, j \rangle$ .

## dovetailing

Una semplice politica che risolve il problema è la seguente:

	0	1	2	3	4	...	i
0	0	1	3	6	10		
1		↙	4	↙	7	↙	11
2			↙	8	↙	12	
3				↙	9	13	
4					↙	14	
⋮							
j							

## Codifica delle coppie

La somma delle componenti  $i$  ed  $j$  per i punti su di una stessa diagonale è costante e pari al numero di diagonali già interamente percorse. Il numero di punti del piano già visitati in queste diagonali è

$$\sum_{k=0}^{i+j} k = \frac{(i+j)(i+j+1)}{2}$$

Per visitare l'elemento  $\langle i, j \rangle$  dovrò ancora percorrere  $j$  passi lungo l'ultima diagonale, per cui

$$\langle i, j \rangle = j + \sum_{k=0}^{i+j} k = \frac{(i+j)^2 + i + 3j}{2}$$

# Corollari

## Lemma

*Il prodotto cartesiano di due insiemi numerabili è numerabile.*

## Lemma

*L'unione di un insieme numerabile di insiemi numerabili è ancora numerabile.*

Sia  $A$  un insieme numerabile e sia  $f$  la sua funzione di enumerazione.  $\{B_a | a \in A\}$  una collezione di insiemi numerabili, ciascuno enumerato da una funzione  $g_a$ . La funzione  $h : \mathcal{N} \times \mathcal{N} \rightarrow \bigcup_{a \in A} B_a$  definita da

$$h(n, m) = g_{f(n)}(m)$$

è suriettiva.

# Sequenze e Linguaggi

## Lemma

*Se  $A$  numerabile, anche  $\bigcup_{i \in \mathcal{N}} A^i$  è numerabile.*

**Osservazione:** Nel caso in cui  $A$  sia finito, l'insieme  $\bigcup_{i \in \mathcal{N}} A^i$  non è altro che l'insieme di tutte le stringhe definibili sull'alfabeto  $A$ . Dunque ogni linguaggio, inteso come sottoinsieme di stringhe definite su di un dato alfabeto, è sempre numerabile.

**Corollario:** L'insieme delle parti finite di un insieme numerabile è numerabile.

# Il teorema di Cantor

## Theorem

*(Cantor) Dato un insieme arbitrario  $A$ , non può esistere una funzione suriettiva da  $A$  in  $\mathcal{P}(A)$ .*

Supponiamo che esista una funzione suriettiva  $g : A \rightarrow \mathcal{P}(A)$ . Consideriamo il seguente insieme:

$$\Delta = \{a \in A \mid a \notin g(a)\}$$

$\Delta \subseteq A$  e siccome abbiamo supposto che  $g$  sia suriettiva deve esistere  $\delta$  tale che  $g(\delta) = \Delta$ . Abbiamo allora:

$$\begin{aligned} \delta \in \Delta &\Leftrightarrow \delta \notin g(\delta) && \text{per definizione di } \Delta \\ &\Leftrightarrow \delta \notin \Delta && \text{per definizione di } \delta \end{aligned}$$

Siccome questo è assurdo,  $g$  non può essere suriettiva.

# Outline

- 1 Numerabilità e diagonalizzazione
  - Dovetailing
  - Sequenze e parti finite
  - Il teorema di Cantor
- 2 Divagazioni sul tema della diagonalizzazione
  - Breve Escursus in Deduzione Naturale
  - Il problema della definibilità
- 3 La tesi di Church
  - Incompletezza dei formalismi totali
  - Le funzioni parziali calcolabili
  - Macchine di Turing
- 4 Funzioni calcolabili ... o meno
  - Il problema della terminazione
  - La proprietà s.m.n.
  - Il problema della equivalenza di programmi
  - Tre funzioni simili

# Il paradosso di Russel

Sia  $U = \{x \mid x \notin x\}$  Allora

$$U \in U \Leftrightarrow U \notin U$$

Principio di comprensione

$$P(t) \Leftrightarrow t \in \{x \mid P(x)\}$$

# Il paradosso di Russel

Sia  $U = \{x \mid x \notin x\}$  Allora

$$U \in U \Leftrightarrow U \notin U$$

**Principio di comprensione**

$$P(t) \Leftrightarrow t \in \{x \mid P(x)\}$$

# Breve Escursus in Deduzione Naturale

- le prove hanno la forma di un albero.
- la radice dell'albero è la conclusione della prova
- le foglie dell'albero sono le ipotesi
- i nodi dell'albero sono regole logiche della forma

$$\frac{A_1 \dots A_n}{B}$$

che connettono un insieme di premesse  $A_1 \dots A_n$  alla conclusione (locale)  $B$ .

- alcune regole possono cancellare (discharge) ipotesi.

# La Deduzione Naturale - regole di inferenza (1)

## congiunzione

$$(\wedge i) \frac{A \quad B}{A \wedge B}$$

$$(\wedge e_l) \frac{A \wedge B}{A}$$

$$(\wedge e_r) \frac{A \wedge B}{B}$$

Legenda: i = introduction, e = elimination, l = left, r = right

## implicazione

$$(\rightarrow i) \frac{\begin{array}{c} [A] \\ \vdots \\ B \end{array}}{A \rightarrow B}$$

$$(\rightarrow e) \frac{A \rightarrow B \quad A}{B}$$

# La Deduzione Naturale - regole di inferenza (1)

## coniunzione

$$(\wedge i) \frac{A \quad B}{A \wedge B} \quad (\wedge e_l) \frac{A \wedge B}{A} \quad (\wedge e_r) \frac{A \wedge B}{B}$$

Legenda: i = introduction, e = elimination, l = left, r = right

## implicazione

$$(\rightarrow i) \frac{\begin{array}{c} [A] \\ \vdots \\ B \end{array}}{A \rightarrow B} \quad (\rightarrow e) \frac{A \rightarrow B \quad A}{B}$$

# La Deduzione Naturale - regole di inferenza (1)

## congiunzione

$$(\wedge i) \frac{A \quad B}{A \wedge B} \quad (\wedge e_l) \frac{A \wedge B}{A} \quad (\wedge e_r) \frac{A \wedge B}{B}$$

Legenda: i = introduction, e = elimination, l = left, r = right

## implicazione

$$(\rightarrow i) \frac{\begin{array}{c} [A] \\ \vdots \\ B \end{array}}{A \rightarrow B} \quad (\rightarrow e) \frac{A \rightarrow B \quad A}{B}$$

# La Deduzione Naturale - regole di inferenza (2)

**falso**

$$(\perp e) \frac{\perp}{A}$$

**negazione**

$$(\neg i) \frac{\begin{array}{c} [A] \\ \vdots \\ \perp \end{array}}{\neg A}$$

$$(\neg e) \frac{\neg A \quad A}{\perp}$$

$$(RAA) \frac{\begin{array}{c} [\neg A] \\ \vdots \\ \perp \end{array}}{A}$$

# La Deduzione Naturale - regole di inferenza (2)

**falso**

$$(\perp e) \frac{\perp}{A}$$

**negazione**

$$(\neg i) \frac{\begin{array}{c} [A] \\ \vdots \\ \perp \end{array}}{\neg A}$$

$$(\neg e) \frac{\neg A \quad A}{\perp}$$

$$(RAA) \frac{\begin{array}{c} [\neg A] \\ \vdots \\ \perp \end{array}}{A}$$

# La Deduzione Naturale - regole di inferenza (2)

**falso**

$$(\perp e) \frac{\perp}{A}$$

**negazione**

$$(\neg i) \frac{\begin{array}{c} [A] \\ \vdots \\ \perp \end{array}}{\neg A}$$

$$(\neg e) \frac{\neg A \quad A}{\perp}$$

$$(RAA) \frac{\begin{array}{c} [\neg A] \\ \vdots \\ \perp \end{array}}{A}$$

# La Deduzione Naturale - tagli (1)

Un taglio e' una sequenza composta da una regola di introduzione immediatamente seguita da una regola di eliminazione per lo stesso connettivo.

Tali configurazioni possono essere **semplificate** (eliminazione dei tagli).

**coniunzione:**

$$\frac{\frac{A \quad B}{A \wedge B}}{A} \rightsquigarrow A$$

$$\frac{\frac{A \quad B}{A \wedge B}}{B} \rightsquigarrow B$$

# La Deduzione Naturale - tagli (2)

## implicazione

$$\begin{array}{c}
 [A] \\
 \vdots \\
 B \quad \vdots \\
 \hline
 A \rightarrow B \quad A \\
 \hline
 B
 \end{array}
 \rightsquigarrow
 \begin{array}{c}
 \vdots \\
 A \\
 \vdots \\
 B
 \end{array}$$

## negazione

$$\begin{array}{c}
 [A] \\
 \vdots \\
 \perp \quad \vdots \\
 \hline
 \neg A \quad A \\
 \hline
 \perp
 \end{array}
 \rightsquigarrow
 \begin{array}{c}
 \vdots \\
 A \\
 \vdots \\
 \perp
 \end{array}$$

# Regole di comprensione

$$\frac{P(t)}{t \in \{x|P(x)\}} (\text{set} - i) \qquad \frac{t \in \{x|P(x)\}}{P(t)} (\text{set} - e)$$

$$\frac{\frac{P(t)}{t \in \{x|P(x)\}}}{P(t)} \rightsquigarrow P(t)$$

# Il paradosso di Russel again

Sia  $U = \{x \mid x \notin x\}$ .

$$\begin{array}{c}
 \frac{\frac{[U \in U]}{U \notin U} \quad [U \in U]}{\perp} \quad \frac{\frac{[U \in U]}{U \notin U} \quad [U \in U]}{\perp} \\
 \frac{\perp \quad \frac{\perp}{U \in U}}{\perp}
 \end{array}$$

Eliminando i tagli si ottiene...la prova stessa.

# Il paradosso di Russel again

Sia  $U = \{x \mid x \notin x\}$ .

$$\begin{array}{c}
 \frac{\frac{[U \in U]}{U \notin U} \quad [U \in U]}{\perp} \\
 \frac{\frac{[U \in U]}{U \notin U} \quad [U \in U]}{\perp} \quad \frac{\frac{[U \in U]}{U \notin U} \quad [U \in U]}{\perp}}{\perp}
 \end{array}$$

Eliminando i tagli si ottiene...la prova stessa.

## Il problema della definibilità

Le funzioni *definibili* da  $\mathcal{N}$  in  $\mathcal{N}$  sono una quantità numerabile.

Fissiamo un qualche enumerazione  $f_j$ .

Definiamo una funzione  $g$  nel modo seguente:

$$g(x) = f_x(x) + 1$$

$g$  è ben definita, dunque dovrebbe comparire nel nostro elenco di funzioni definibili. Supponiamo che  $g = f_k$ . Abbiamo allora:

$$f_k(k) = g(k) = f_k(k) + 1$$

La nozione di *definibilità* è dunque sempre relativa (dipendente dal linguaggio) e incompleta; comunque essa venga fissata esistono “buone definizioni” definite che sfuggono alla definizione adottata.

## Il problema della definibilità

Le funzioni *definibili* da  $\mathcal{N}$  in  $\mathcal{N}$  sono una quantità numerabile.

Fissiamo un qualche enumerazione  $f_j$ .

Definiamo una funzione  $g$  nel modo seguente:

$$g(x) = f_x(x) + 1$$

$g$  è ben definita, dunque dovrebbe comparire nel nostro elenco di funzioni definibili. Supponiamo che  $g = f_k$ . Abbiamo allora:

$$f_k(k) = g(k) = f_k(k) + 1$$

La nozione di *definibilità* è dunque sempre relativa (dipendente dal linguaggio) e incompleta; comunque essa venga fissata esistono “buone definizioni” definite che sfuggono alla definizione adottata.

## Il problema della definibilità

Le funzioni *definibili* da  $\mathcal{N}$  in  $\mathcal{N}$  sono una quantità numerabile.

Fissiamo un qualche enumerazione  $f_j$ .

Definiamo una funzione  $g$  nel modo seguente:

$$g(x) = f_x(x) + 1$$

$g$  è ben definita, dunque dovrebbe comparire nel nostro elenco di funzioni definibili. Supponiamo che  $g = f_k$ . Abbiamo allora:

$$f_k(k) = g(k) = f_k(k) + 1$$

La nozione di *definibilità* è dunque sempre relativa (dipendente dal linguaggio) e incompleta; comunque essa venga fissata esistono “buone definizioni” definite che sfuggono alla definizione adottata.

## Il problema della definibilità

Le funzioni *definibili* da  $\mathcal{N}$  in  $\mathcal{N}$  sono una quantità numerabile.

Fissiamo un qualche enumerazione  $f_j$ .

Definiamo una funzione  $g$  nel modo seguente:

$$g(x) = f_x(x) + 1$$

$g$  è ben definita, dunque dovrebbe comparire nel nostro elenco di funzioni definibili. Supponiamo che  $g = f_k$ . Abbiamo allora:

$$f_k(k) = g(k) = f_k(k) + 1$$

La nozione di *definibilità* è dunque sempre relativa (dipendente dal linguaggio) e incompleta; comunque essa venga fissata esistono “buone definizioni” definite che sfuggono alla definizione adottata.

# Outline

- 1 Numerabilità e diagonalizzazione
  - Dovetailing
  - Sequenze e parti finite
  - Il teorema di Cantor
- 2 Divagazioni sul tema della diagonalizzazione
  - Breve Escursus in Deduzione Naturale
  - Il problema della definibilità
- 3 La tesi di Church
  - Incompletezza dei formalismi totali
  - Le funzioni parziali calcolabili
  - Macchine di Turing
- 4 Funzioni calcolabili ... o meno
  - Il problema della terminazione
  - La proprietà s.m.n.
  - Il problema della equivalenza di programmi
  - Tre funzioni simili

# L'interprete

Sia dato un linguaggio di programmazione  $L$  e sia  $P_i$  una enumerazione dei suoi programmi.

Nel seguito denoteremo con  $\varphi_i$  la funzione calcolata dal programma  $P_i$  ( $\varphi_i : \mathcal{N}^k \rightarrow \mathcal{N}$ ).

## Definition

Un **interprete** per  $L$  è una funzione binaria  $u$  per cui

$$u(i, j) = \varphi_i(j)$$

# Incompletezza dei formalismi totali

## Theorem

*Nessun linguaggio  $L$  (con un minimo di espressività) contiene il proprio interprete (ammesso che  $L$  permetta di calcolare solo funzioni totali).*

Supponiamo che le funzioni esprimibili in  $L$  contengano le proiezioni, la funzione successore, e siano chiuse per composizione. Consideriamo la seguente funzione

$$g(x) = u(x, x) + 1$$

Se l'interprete è in  $L$ , allora lo è anche  $g$ . Dunque dovrebbe esistere  $a$  tale che  $\varphi_a = g$ . Ma in tal caso:

$$\varphi_a(a) = g(a) = u(a, a) + 1 = \varphi_a(a) + 1$$

# Incompletezza dei formalismi totali

## Theorem

*Nessun linguaggio  $L$  (con un minimo di espressività) contiene il proprio interprete (ammesso che  $L$  permetta di calcolare solo funzioni totali).*

Supponiamo che le funzioni esprimibili in  $L$  contengano le proiezioni, la funzione successore, e siano chiuse per composizione. Consideriamo la seguente funzione

$$g(x) = u(x, x) + 1$$

Se l'interprete è in  $L$ , allora lo è anche  $g$ . Dunque dovrebbe esistere  $a$  tale che  $\varphi_a = g$ . Ma in tal caso:

$$\varphi_a(a) = g(a) = u(a, a) + 1 = \varphi_a(a) + 1$$

# Le funzioni parziali calcolabili

Che succede se ammettiamo di considerare programmi potenzialmente divergenti (while, ricorsione generale, ...)?

Sarebbe naturale attendersi, anche in questo caso, l'esistenza di una gerarchia infinita di linguaggi con espressività crescente.

Questo non accade: sbattiamo contro un limite superiore

## le funzioni parziali calcolabili

(una nozione assoluta della matematica, indipendente dal formalismo).

# Le funzioni parziali calcolabili

Che succede se ammettiamo di considerare programmi potenzialmente divergenti (while, ricorsione generale, ...)?

Sarebbe naturale attendersi, anche in questo caso, l'esistenza di una gerarchia infinita di linguaggi con espressività crescente.

Questo non accade: sbattiamo contro un limite superiore

## le funzioni parziali calcolabili

(una nozione assoluta della matematica, indipendente dal formalismo).

# Le funzioni parziali calcolabili

Che succede se ammettiamo di considerare programmi potenzialmente divergenti (while, ricorsione generale, ...)?

Sarebbe naturale attendersi, anche in questo caso, l'esistenza di una gerarchia infinita di linguaggi con espressività crescente.

Questo non accade: sbattiamo contro un limite superiore

## **le funzioni parziali calcolabili**

(una nozione assoluta della matematica, indipendente dal formalismo).

# La Tesi di Church

La natura assoluta della nozione di funzione (parziale) calcolabile, corrispondente all'idea intuitiva di un procedimento effettivo di calcolo (algoritmo) è nota come **Tesi di Church**.

- La Tesi di Church non è dimostrabile.
- La Tesi di Church è avvalorata da una infinità di ricerche compiute riguardo al potere espressivo di diversi linguaggi (e costrutti) di programmazione.
- La Tesi di Church potrebbe essere confutata esibendo un procedimento di calcolo *intuitivamente effettivo* ma non esprimibile in nessuno dei moderni linguaggi di programmazione.

# La Tesi di Church

La natura assoluta della nozione di funzione (parziale) calcolabile, corrispondente all'idea intuitiva di un procedimento effettivo di calcolo (algoritmo) è nota come **Tesi di Church**.

- La Tesi di Church non è dimostrabile.
- La Tesi di Church è avvalorata da una infinità di ricerche compiute riguardo al potere espressivo di diversi linguaggi (e costrutti) di programmazione.
- La Tesi di Church potrebbe essere confutata esibendo un procedimento di calcolo *intuitivamente effettivo* ma non esprimibile in nessuno dei moderni linguaggi di programmazione.

# La Tesi di Church

La natura assoluta della nozione di funzione (parziale) calcolabile, corrispondente all'idea intuitiva di un procedimento effettivo di calcolo (algoritmo) è nota come **Tesi di Church**.

- La Tesi di Church non è dimostrabile.
- La Tesi di Church è avvalorata da una infinità di ricerche compiute riguardo al potere espressivo di diversi linguaggi (e costrutti) di programmazione.
- La Tesi di Church potrebbe essere confutata esibendo un procedimento di calcolo *intuitivamente effettivo* ma non esprimibile in nessuno dei moderni linguaggi di programmazione.

# La Tesi di Church

La natura assoluta della nozione di funzione (parziale) calcolabile, corrispondente all'idea intuitiva di un procedimento effettivo di calcolo (algoritmo) è nota come **Tesi di Church**.

- La Tesi di Church non è dimostrabile.
- La Tesi di Church è avvalorata da una infinità di ricerche compiute riguardo al potere espressivo di diversi linguaggi (e costrutti) di programmazione.
- La Tesi di Church potrebbe essere confutata esibendo un procedimento di calcolo *intuitivamente effettivo* ma non esprimibile in nessuno dei moderni linguaggi di programmazione.

# La Macchina di Turing: hardware

## Hardware

- Un nastro di memoria illimitato, diviso in celle di dimensione fissata. Ogni cella può contenere un carattere di un alfabeto dato (compreso un carattere “bianco” di inizializzazione).
- una testina di lettura mobile.
- Un automa a stati finiti (concettualmente simile al moderno microprocessore).

# La Macchina di Turing: software

## Software

Il comportamento della macchina procede per *passi discreti*, ed è regolato da un insieme  $P$  di quintuple della forma

$$(q, a, q', a', m)$$

dove

- $a$  e  $a'$  sono caratteri del nastro,
- $q$  e  $q'$  sono stati dell'automa
- $m \in \{R, L\}$ , è un mossa della testina.

Se  $(q, a, q', a', m) \in P$ , l'automa si trova nello stato  $q$  e la cella in lettura contiene il carattere  $a$ , allora l'automa transita nello stato  $q'$ , il carattere  $a'$  sovrascrive  $a$  e la testina si sposta di una posizione verso sinistra o verso destra a seconda della mossa  $m$ .

# L'essenza della Macchina di Turing

Agente di calcolo con potenzialità **finite**, che procede per passi **discreti**.

Ad ogni passo posso

- prendere visione di una porzione finita dello spazio (discretizzato) circostante (compreso un eventuale stato interno)
- modificare una porzione finita di tale spazio,
- spostarmi di una distanza finita in tale spazio

# La Macchina di Turing Universale

Una macchina di Turing è descritta dall'insieme delle sue quintuple. Posso definire una **Macchina Universale** che prende in input la sequenza di quintuple di una macchina  $M$ , un input  $m$  e simula il comportamento di  $M$  su input  $m$ .

Se leggo l'insieme delle quintuple come la codifica numerica della macchina  $M$ , la Macchina Universale è un interprete nel senso precedentemente esposto.

La Macchina Universale permette di **eseguire tutti i programmi con un unico hardware**: è a tutti gli effetti l'antesignano del moderno elaboratore elettronico.

# La Macchina di Turing Universale

Una macchina di Turing è descritta dall'insieme delle sue quintuple. Posso definire una **Macchina Universale** che prende in input la sequenza di quintuple di una macchina  $M$ , un input  $m$  e simula il comportamento di  $M$  su input  $m$ .

Se leggo l'insieme delle quintuple come la codifica numerica della macchina  $M$ , la Macchina Universale è un interprete nel senso precedentemente esposto.

La Macchina Universale permette di **eseguire tutti i programmi con un unico hardware**: è a tutti gli effetti l'antesigano del moderno elaboratore elettronico.

# La Macchina di Turing Universale

Una macchina di Turing è descritta dall'insieme delle sue quintuple. Posso definire una **Macchina Universale** che prende in input la sequenza di quintuple di una macchina  $M$ , un input  $m$  e simula il comportamento di  $M$  su input  $m$ .

Se leggo l'insieme delle quintuple come la codifica numerica della macchina  $M$ , la Macchina Universale è un interprete nel senso precedentemente esposto.

La Macchina Universale permette di **eseguire tutti i programmi con un unico hardware**: è a tutti gli effetti l'antesignano del moderno elaboratore elettronico.

# Outline

- 1 Numerabilità e diagonalizzazione
  - Dovetailing
  - Sequenze e parti finite
  - Il teorema di Cantor
- 2 Divagazioni sul tema della diagonalizzazione
  - Breve Escursus in Deduzione Naturale
  - Il problema della definibilità
- 3 La tesi di Church
  - Incompletezza dei formalismi totali
  - Le funzioni parziali calcolabili
  - Macchine di Turing
- 4 Funzioni calcolabili ... o meno
  - Il problema della terminazione
  - La proprietà s.m.n.
  - Il problema della equivalenza di programmi
  - Tre funzioni simili

# Come procedere

- fissare un formalismo e studiare le funzioni esprimibili in quel formalismo
- lavorare a un livello più astratto, evidenziando alcuni principi chiave (calcolabilità di determinate funzioni, chiusura rispetto a determinati costrutti, etc.) su cui si basa la trattazione teorica (approccio pseudo-assiomatico).

Seguiremo il secondo approccio (meno pedante e assai più divertente).

# Come procedere

- fissare un formalismo e studiare le funzioni esprimibili in quel formalismo
- lavorare a un livello più astratto, evidenziando alcuni principi chiave (calcolabilità di determinate funzioni, chiusura rispetto a determinati costrutti, etc.) su cui si basa la trattazione teorica (approccio pseudo-assiomatico).

Seguiremo il secondo approccio (meno pedante e assai più divertente).

## Un po' di notazione

Supponiamo avere un insieme enumerato (modello di calcolo)  $\varphi_i$  di funzioni *parziali* da interi in interi.

Useremo la notazione  $\varphi_i(n) \downarrow$  per indicare che la funzione è definita (o converge) per input  $n$ , e  $\varphi_i(n) \uparrow$  quando è indefinita (o diverge) per tale input.

Definiamo **dominio** (dom) di  $\varphi_i$  il suo insieme di convergenza, ossia

$$\text{dom}(\varphi_i) = \{n \mid \varphi_i(n) \downarrow\}$$

Il **codominio** (cod) di  $\varphi_i$  è l'insieme dei possibili output:

$$\text{cod}(\varphi_i) = \{m \mid \exists n, \varphi_i(n) = m\}$$

## Il problema della terminazione

Il test di terminazione è così definito:

$$g(i, x) = \begin{cases} 1 & \text{se } \varphi_i(x) \downarrow \\ 0 & \text{se } \varphi_i(x) \uparrow \end{cases}$$

Consideriamo ora la funzione  $f$ , definita come segue:

$$f(x) = \begin{cases} 1 & \text{se } g(x, x) = 0 \\ \uparrow & \text{se } g(x, x) = 1 \end{cases}$$

Se  $g$  è calcolabile nel nostro modello di calcolo, e il modello gode di minime proprietà di chiusura, anche  $f$  dovrebbe appartenere al modello. Dovrebbe cioè esistere  $m$  tale che  $f = \varphi_m$ .

## Il problema della terminazione - 2

Supposto  $f = \varphi_m$  ci chiediamo quanto vale  $\varphi_m(m)$ :

$$\varphi_m(m) = \begin{cases} 1 & \text{se } g(m, m) = 0 \\ \uparrow & \text{se } g(m, m) = 1 \end{cases}$$

ma applicando la definizione di  $g$  notiamo che

$$\varphi_m(m) = \begin{cases} 1 & \text{se } \varphi_m(m) \uparrow \\ \uparrow & \text{se } \varphi_m(m) \downarrow \end{cases}$$

il che è assurdo. Questo dimostra che la funzione  $g$  di terminazione *non è esprimibile* in nessun modello di calcolo che ammetta divergenza, e con minime proprietà di chiusura: **il problema della terminazione non è algoritmicamente risolubile.**

# Terminazione “diagonale”

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

$K = \text{dom}(k)$ .

La funzione  $k$  non è calcolabile.

Corollario (della dimostrazione) del teorema precedente.

# La proprietà s.m.n.

Una classe di funzioni parziali gode della proprietà s.m.n. se, per ogni funzione  $f^{m+n}$  appartenente alla classe, esiste una funzione *totale*  $s_n^m$  tale che, per ogni  $\vec{x}_m, \vec{y}_n$

$$\varphi_{s_n^m(\vec{x}_m)}(\vec{y}_n) = f(\vec{x}_m, \vec{y}_n)$$

- 1 una istanza parziale di una funzione calcolabile è calcolabile
- 2 il programma dell'istanza si ottiene in modo uniforme ed effettivo a partire dal programma generale e dai parametri di istanziazione

## Il problema della totalità

$$\text{total}(i) = \begin{cases} 1 & \text{se } \varphi_i \text{ è totale} \\ 0 & \text{altrimenti} \end{cases}$$

Consideriamo una funzione ausiliaria definita nel modo seguente:

$$f(x, y) = \begin{cases} 0 & \text{se } \varphi_x(x) \downarrow \\ \uparrow & \text{altrimenti} \end{cases}$$

La funzione  $f$  è calcolabile.

## Il problema della totalità

$$total(i) = \begin{cases} 1 & \text{se } \varphi_i \text{ è totale} \\ 0 & \text{altrimenti} \end{cases}$$

Consideriamo una funzione ausiliaria definita nel modo seguente:

$$f(x, y) = \begin{cases} 0 & \text{se } \varphi_x(x) \downarrow \\ \uparrow & \text{altrimenti} \end{cases}$$

La funzione  $f$  è calcolabile.

## Il problema della totalità - 2

Per il teorema s.m.n. esiste  $h$  totale calcolabile tale che

$$\varphi_{h(x)}(y) = f(x, y) = \begin{cases} 0 & \text{se } \varphi_x(x) \downarrow \\ \uparrow & \text{altrimenti} \end{cases}$$

Avremmo allora

$$\text{total}(h(x)) = \begin{cases} 1 & \text{se } \varphi_x(x) \downarrow \\ 0 & \text{se } \varphi_x(x) \uparrow \end{cases}$$

Se  $\text{total}$  fosse calcolabile, anche  $\text{total} \circ h$  lo sarebbe.

Contraddizione, dunque  $\text{total}$  non è calcolabile.

# Il problema della equivalenza estensionale di programmi

$$eq(i, j) = \begin{cases} 1 & \text{se } \varphi_i \approx \varphi_j \\ 0 & \text{altrimenti} \end{cases}$$

Consideriamo la funzione costante 0, e sia  $m$  un indice per essa.  
 Consideriamo la funzione  $h$  della sezione precedente:

$$\varphi_{h(x)}(y) = \begin{cases} 0 & \text{se } \varphi_x(x) \downarrow \\ \uparrow & \text{se } \varphi_x(x) \uparrow \end{cases}$$

$$f(x) = eq(h(x), m) = \begin{cases} 1 & \text{se } \varphi_x(x) \downarrow \\ 0 & \text{altrimenti} \end{cases}$$

$f$  non è calcolabile e dunque neppure  $eq$ .

# Il problema della equivalenza estensionale di programmi

$$eq(i, j) = \begin{cases} 1 & \text{se } \varphi_i \approx \varphi_j \\ 0 & \text{altrimenti} \end{cases}$$

Consideriamo la funzione costante 0, e sia  $m$  un indice per essa.  
 Consideriamo la funzione  $h$  della sezione precedente:

$$\varphi_{h(x)}(y) = \begin{cases} 0 & \text{se } \varphi_x(x) \downarrow \\ \uparrow & \text{se } \varphi_x(x) \uparrow \end{cases}$$

$$f(x) = eq(h(x), m) = \begin{cases} 1 & \text{se } \varphi_x(x) \downarrow \\ 0 & \text{altrimenti} \end{cases}$$

$f$  non è calcolabile e dunque neppure  $eq$ .

# Tre funzioni simili

$$g(i) \equiv \begin{cases} 1 & \text{se } \exists n, \varphi_i(n) \downarrow \\ 0 & \text{altrimenti} \end{cases}$$

$$g'(i) \equiv \begin{cases} 1 & \text{se } \exists n, \varphi_i(n) \downarrow \\ \uparrow & \text{altrimenti} \end{cases}$$

$$g''(i) \equiv \begin{cases} \text{minimo } n, \varphi_i(n) \downarrow & \text{se } \exists n, \varphi_i(n) \downarrow \\ \uparrow & \text{altrimenti} \end{cases}$$

# La funzione $g$

Consideriamo la solita funzione calcolabile  $h$ :

$$\varphi_{h(x)}(y) = \begin{cases} 0 & \text{se } \varphi_x(x) \downarrow \\ \uparrow & \text{se } \varphi_x(x) \uparrow \end{cases}$$

$$f(x) = g(h(x)) = \begin{cases} 1 & \text{se } \varphi_x(x) \downarrow \\ 0 & \text{altrimenti} \end{cases}$$

$f$  non è calcolabile e dunque neppure  $g$ .

# La funzione $g'$

## Il predicato T di Kleene

Esiste una funzione calcolabile totale  $t$  tale che

$$\varphi_i(j) \downarrow \Leftrightarrow \exists c, t(x, y, c) = 1$$

**La forma normale di Kleene** Esistono  $p$  e  $t$  totali calcolabili tali che

$$\varphi_z(x) = p(\mu y. (t(z, x, y) = 1))$$

# La funzione $g'$ - 2

$$g'(i) \equiv \begin{cases} 1 & \text{se } \exists n, \varphi_i(n) \downarrow \\ \uparrow & \text{altrimenti} \end{cases}$$

$$g'(i) = \mu \langle n, c \rangle . t(i, n, c) = 1; \text{ return } 1$$

# La funzione $g''$

$$\varphi_{h(i)}(y) = f(i, y) = \begin{cases} 0 & \text{se } y > 0 \vee \varphi_i(i) \downarrow \\ \uparrow & \text{altrimenti} \end{cases}$$

$\varphi_{h(i)}(0) \downarrow$  se e solo se  $\varphi_i(i) \downarrow$ . Inoltre  $\varphi_i(1) \downarrow$ . Dunque

$$g''(h(i)) \equiv \begin{cases} 0 & \varphi_i(i) \downarrow \\ 1 & \text{altrimenti} \end{cases}$$

Se  $g''$  fosse calcolabile risolveremmo il problema della terminazione diagonale.

# Outline

- 1 Numerabilità e diagonalizzazione
  - Dovetailing
  - Sequenze e parti finite
  - Il teorema di Cantor
- 2 Divagazioni sul tema della diagonalizzazione
  - Breve Escursus in Deduzione Naturale
  - Il problema della definibilità
- 3 La tesi di Church
  - Incompletezza dei formalismi totali
  - Le funzioni parziali calcolabili
  - Macchine di Turing
- 4 Funzioni calcolabili ... o meno
  - Il problema della terminazione
  - La proprietà s.m.n.
  - Il problema della equivalenza di programmi
  - Tre funzioni simili

# Insiemi ricorsivi e ricorsivamente enumerabili

## Definition

Un insieme si dice **ricorsivo** (o decidibile) se la sua funzione caratteristica è calcolabile.

## Esempi

- 1 l'insieme vuoto e l'insieme  $\mathcal{N}$  di tutti i numeri naturali
- 2 ogni insieme finito
- 3 l'insieme dei numeri pari
- 4 l'insieme dei numeri primi

## Non ogni insieme è ricorsivo

(contro) Esempio:

$$K = \{x \mid \varphi_x(x) \downarrow\}$$

# Insiemi ricorsivi e ricorsivamente enumerabili

## Definition

Un insieme si dice **ricorsivo** (o decidibile) se la sua funzione caratteristica è calcolabile.

## Esempi

- 1 l'insieme vuoto e l'insieme  $\mathcal{N}$  di tutti i numeri naturali
- 2 ogni insieme finito
- 3 l'insieme dei numeri pari
- 4 l'insieme dei numeri primi

Non ogni insieme è ricorsivo

(contro) Esempio:

$$K = \{x \mid \varphi_x(x) \downarrow\}$$

# Insiemi ricorsivi e ricorsivamente enumerabili

## Definition

Un insieme si dice **ricorsivo** (o decidibile) se la sua funzione caratteristica è calcolabile.

## Esempi

- 1 l'insieme vuoto e l'insieme  $\mathcal{N}$  di tutti i numeri naturali
- 2 ogni insieme finito
- 3 l'insieme dei numeri pari
- 4 l'insieme dei numeri primi

## Non ogni insieme è ricorsivo

(contro) Esempio:

$$K = \{x \mid \varphi_x(x) \downarrow\}$$

# Proprietà di chiusura degli insiemi ricorsivi

## Theorem

*Gli insiemi ricorsivi sono chiusi rispetto alle operazioni di unione, intersezione e complementazione.*

Siano  $A$  e  $B$  insiemi ricorsivi, e siano  $c_A$  e  $c_B$  le loro funzioni caratteristiche. Allora:

- $c_{\overline{A}}(n) = 1 - c_A(n)$
- $c_{A \cap B}(n) = \min\{c_A(n), c_B(n)\}$
- $c_{A \cup B}(n) = \max\{c_A(n), c_B(n)\}$

Gli insiemi ricorsivi, ordinati rispetto alla relazione di inclusione insiemistica, formano quindi un reticolo booleano (*Algebra di Boole*).

# Insiemi ricorsivamente enumerabili

## Definition

Un insieme si dice **ricorsivamente enumerabile (r.e.)** se o è vuoto oppure è il codominio di una funzione *totale* calcolabile (detta funzione di enumerazione).

Ogni insieme ricorsivo è anche r.e.

Sia  $A$  ricorsivo e sia  $c_A$  la sua funzione caratteristica.

Il caso in cui  $A$  è finito è banale.

Supponiamo  $A$  infinito:

$$\begin{cases} f(0) = \mu y, c_A(y) = 1 \\ f(x+1) = \mu y, c_A(y) = 1 \wedge y > f(x) \end{cases}$$

# Proprietà degli insiemi r.e.

## Theorem

*Un insieme  $A$  è ricorsivo se e solo se sia  $A$  che  $\bar{A}$  sono r.e.*

$\Rightarrow$ : ovvio.

$\Leftarrow$ : Supponiamo che  $A$  e  $\bar{A}$  siano rispettivamente enumerati da  $f$  e  $g$ . Poniamo

$$\begin{cases} h(2x) = f(x) \\ h(2x + 1) = g(x) \end{cases}$$

$h$  è suriettiva su  $N$ . Sia  $pari(n)$  la funzione caratteristica dell'insieme dei numeri pari.

$$c_A(n) = pari(\mu y (h(y) = n))$$

$c_A$  è calcolabile e *totale*.

# Semidecidibilità

Sia  $A$  un insieme di numeri naturali. Le seguenti affermazioni sono equivalenti:

- 1  $A = \emptyset \vee \exists f : A = \text{cod}(f)$ ,  $f$  totale calcolabile
- 2  $\exists g : A = \text{dom}(g)$ ,  $g$  parziale calcolabile
- 3  $\exists h : A = \text{cod}(h)$ ,  $h$  parziale calcolabile

# 1 $\Rightarrow$ 2

Se  $A = \emptyset$  basta considerare la funzione  $g$  ovunque divergente.

Sia  $A = \text{cod}(f)$  per  $f$  totale calcolabile, poniamo

$$g(x) = \mu y (f(y) = x)$$

Chiaramente  $g$  è calcolabile e  $g(x) \downarrow$  se e solo se  $x \in \text{cod}(f)$ .

2  $\Rightarrow$  3

Sia  $A = \text{dom}(g)$ .

Poniamo

$$h(x) = x + 0 * g(x)$$

che converge a  $x$  se e solo se  $g(x)$  converge  
(la funzione  $h$  è l'identità ristretta al dominio di convergenza di  $g$ ).

# 3 $\Rightarrow$ 1

Sia  $A = \text{cod}(h)$ , per  $h$  parziale calcolabile.

Distinguiamo due casi.

- se  $A$  è vuoto, non c'è nulla da dimostrare.
- Sia  $a \in A$  e supponiamo inoltre che  $h = \varphi_i$ ;

$$f(\langle x, c \rangle) = \begin{cases} p(c) & \text{se } t(i, x, c) = 1 \\ a & \text{altrimenti} \end{cases}$$

dove  $t$  e  $p$  sono le due funzioni della forma normale di Kleene.  
 $f$  è totale e calcolabile e  $\text{cod}(f) = A$ .

# $K$ è r.e.

Esistono insiemi r.e. ma non ricorsivi.

Sia

$$k(x) = \varphi_x(x)$$

$k$  è calcolabile e siccome  $K = \text{dom}(k)$ ,  $K$  è r.e.

# Proprietà di chiusura degli insiemi r.e.

## Gli insiemi r.e. sono chiusi rispetto a unione e intersezione

**unione** Sia  $A = \text{dom}(f)$  e  $B = \text{dom}(g)$  per  $f, g$  parz. calc.  
 $A \cap B = \text{dom}(h)$  per  $h(x) = f(x) * g(x)$ .

**intersezione** Sia  $A = \text{cod}(f')$  e  $B = \text{cod}(g')$  con  $f'$  e  $g'$  parz. calc.  
 $A \cup B = \text{cod}(h')$  dove

$$\begin{cases} h'(2x) & = f'(x) \\ h'(2x + 1) & = g'(x) \end{cases}$$

## Proprietà di chiusura degli insiemi r.e. (2)

**Gli insiemi r.e. non sono chiusi per complementazione**

$\bar{K}$  non è r.e. (altrimenti  $K$  sarebbe ricorsivo).

**Corollario** Esistono insiemi che non sono nè ricorsivi nè ricorsivamente enumerabili.

## Proprietà di chiusura degli insiemi r.e. (2)

**Gli insiemi r.e. non sono chiusi per complementazione**

$\bar{K}$  non è r.e. (altrimenti  $K$  sarebbe ricorsivo).

**Corollario** Esistono insiemi che non sono nè ricorsivi nè ricorsivamente enumerabili.

# Outline

- 1 Numerabilità e diagonalizzazione
  - Dovetailing
  - Sequenze e parti finite
  - Il teorema di Cantor
- 2 Divagazioni sul tema della diagonalizzazione
  - Breve Escursus in Deduzione Naturale
  - Il problema della definibilità
- 3 La tesi di Church
  - Incompletezza dei formalismi totali
  - Le funzioni parziali calcolabili
  - Macchine di Turing
- 4 Funzioni calcolabili ... o meno
  - Il problema della terminazione
  - La proprietà s.m.n.
  - Il problema della equivalenza di programmi
  - Tre funzioni simili

## Insiemi estensionali

Un insieme (proprietà)  $A \subseteq \mathcal{N}$  si dice estensionale (w.r.t  $\varphi$ ) se per ogni  $i, j$

$$i \in A \wedge \varphi_i \equiv \varphi_j \Rightarrow j \in A$$

Ovvero, se  $c_A$  è la funzione caratteristica di  $A$ ,

$$\varphi_i \equiv \varphi_j \Rightarrow c_A(i) = c_A(j)$$

Una proprietà estensionale di (indici di) programmi è una proprietà relativa all'estensione dei programmi: la **funzione calcolata** dai programmi e non il **modo** in cui questa viene calcolata.

## Insiemi estensionali

Un insieme (proprietà)  $A \subseteq \mathcal{N}$  si dice estensionale (w.r.t  $\varphi$ ) se per ogni  $i, j$

$$i \in A \wedge \varphi_i \equiv \varphi_j \Rightarrow j \in A$$

Ovvero, se  $c_A$  è la funzione caratteristica di  $A$ ,

$$\varphi_i \equiv \varphi_j \Rightarrow c_A(i) = c_A(j)$$

Una proprietà estensionale di (indici di) programmi è una proprietà relativa all'estensione dei programmi: la **funzione calcolata** dai programmi e non il **modo** in cui questa viene calcolata.

## Esempi di proprietà estensionali

- $P(i) = \varphi_i$  è totale
- $P(i) = 5 \in \text{cod}(\varphi_i)$
- $P(i) = \text{dom}(\varphi_i)$  è finito
- $P(i) \varphi_i(0) \uparrow$
- $P(i) \exists n, \varphi_i(n) \downarrow \wedge \varphi_i(n+1) \downarrow$

Il complementare di un insieme estensionale è estensionale

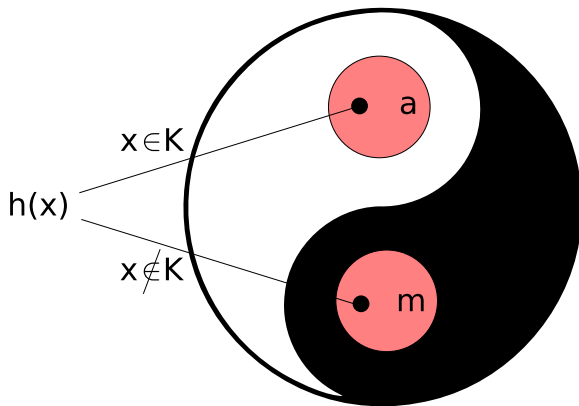
# Il teorema di Rice

## Rice 1953

Una proprietà estensionale di programmi è decidibile se e solo se è triviale.

## Il teorema di Rice e lo Yin Yang

$$\forall x, \phi_m(x) \uparrow$$



# La funzione $h$

Sia  $K = \text{dom}(\phi_k)$ . Poniamo

$$\phi_{h(x)}(y) = \phi_k(x); \phi_a(y)$$

Per s.m.n  $h$  è totale e calcolabile.

Se  $\phi_m$  è la funzione ovunque divergente,

$$\phi_{h(x)} \approx \begin{cases} \phi_a & \text{if } x \in K \\ \phi_m & \text{if } x \notin K \end{cases}$$

# La funzione $h$

Sia  $K = \text{dom}(\phi_k)$ . Poniamo

$$\phi_{h(x)}(y) = \phi_k(x); \phi_a(y)$$

Per s.m.n  $h$  è totale e calcolabile.

Se  $\phi_m$  è la funzione ovunque divergente,

$$\phi_{h(x)} \approx \begin{cases} \phi_a & \text{if } x \in K \\ \phi_m & \text{if } x \notin K \end{cases}$$

# Lo Yin Yang

- Yin e yang sono forze apparentemente disgiunte e opposte ma in realtà interconnesse e interdipendenti: ognuna origina l'altra.
- Ogni entità possiede entrambi gli aspetti yin e yang, benchè questi possano manifestarsi in modi e tempi differenti.
- Yin e yang interagiscono costantemente, e non possono esistere in modo statico, ma sono in continua trasformazione l'uno nell'altro.
- Yin e yang non corrispondono a una divisione manichea in buono/cattivo, considerata superficiale dalla filosofia taoista.

## Uso del teorema di Rice

- Uso diretto, per dimostrare che determinate proprietà (essendo estensionali) non sono decidibili.
- Uso indiretto, per dimostrare che determinate proprietà estensionali non sono neppure semidecidibili (basta dimostrare che il complementare è r.e.)

Esempio:

$$A = \{ i \mid \varphi_i(0) \downarrow \}$$

$A$  non è banale, e dunque, per Rice, non può essere ricorsivo (uso diretto); d'altra parte  $A$  è r.e., dunque

$$\bar{A} = \{ i \mid \varphi_i(0) \uparrow \}$$

non è neppure r.e. altrimenti sia  $A$  che  $\bar{A}$  sarebbero ricorsivi, contraddicendo il risultato di Rice (uso indiretto).

# Outline

- 1 Numerabilità e diagonalizzazione
  - Dovetailing
  - Sequenze e parti finite
  - Il teorema di Cantor
- 2 Divagazioni sul tema della diagonalizzazione
  - Breve Escursus in Deduzione Naturale
  - Il problema della definibilità
- 3 La tesi di Church
  - Incompletezza dei formalismi totali
  - Le funzioni parziali calcolabili
  - Macchine di Turing
- 4 Funzioni calcolabili ... o meno
  - Il problema della terminazione
  - La proprietà s.m.n.
  - Il problema della equivalenza di programmi
  - Tre funzioni simili

# Aritmetica

Il linguaggio dell'aritmetica è il linguaggio del primo ordine basato sulla seguente segnatura:

$$0, S, +, \cdot, =$$

Indichiamo con  $\bar{n}$  il termine che rappresenta il numero  $n$ .

$A \subseteq \mathcal{N}^k$  si dice aritmetico se esiste una formula aritmetica  $\psi(x_1, \dots, x_k)$  tale che

$$(n_1, \dots, n_k) \in A \Leftrightarrow \mathcal{N} \models \psi[\bar{n}_1/x_1, \dots, \bar{n}_k/x_k]$$

ovvero  $\psi(x_1, \dots, x_k)$  è vera sui numeri naturali.

Diremo in questo caso che  $\psi$  è una *descrizione aritmetica* di  $A$ .

## Insiemi/funzioni aritmetici

Gli insiemi aritmetici sono chiusi rispetto alle operazioni di unione, intersezione e complementazione.

Una funzione  $f$  si dice aritmetica se il suo grafo è un insieme aritmetico.

Le seguenti funzioni sono aritmetiche:

$f$	arietà	descrizione
0	0	$y = 0$
$S$	1	$y = S(x)$
+	2	$y = x_1 + x_2$
$\cdot$	2	$y = x_1 \cdot x_2$
$\pi_i^k$	$k$	$y = x_k$

# composizione di funzioni

La composizione di funzioni aritmetiche è aritmetica. Sia

$f(x) = g(h(x))$  e siano  $\psi_g(x, y)$  e  $\psi_h(x, y)$  le descrizioni aritmetiche di  $g$  e  $h$ .

Definiamo

$$\psi_f(x, z) = \exists y, g(x, y) \wedge h(y, z)$$

è facile dimostrare che  $\psi_f$  è una descrizione aritmetica di  $f$ .

# composizione di funzioni

La composizione di funzioni aritmetiche è aritmetica. Sia

$f(x) = g(h(x))$  e siano  $\psi_g(x, y)$  e  $\psi_h(x, y)$  le descrizioni aritmetiche di  $g$  e  $h$ .

Definiamo

$$\psi_f(x, z) = \exists y, g(x, y) \wedge h(y, z)$$

è facile dimostrare che  $\psi_f$  è una descrizione aritmetica di  $f$ .

# minimizzazione

Una funzione definita per minimizzazione di una funzione aritmetica è ancora aritmetica.

Sia

$$f(x) = \mu y.(g(x, y) = 0)$$

e supponiamo che  $\psi_g$  sia una descrizione aritmetica di  $g$ .

$f$  è descritta dalla seguente formula:

$$\psi_f(x, y) = \psi_g(x, y, 0) \wedge \forall i, i < y \rightarrow \exists m, (\psi_g(x, i, m) \wedge m \neq 0)$$

# minimizzazione

Una funzione definita per minimizzazione di una funzione aritmetica è ancora aritmetica.

Sia

$$f(x) = \mu y.(g(x, y) = 0)$$

e supponiamo che  $\psi_g$  sia una descrizione aritmetica di  $g$ .

$f$  è descritta dalla seguente formula:

$$\psi_f(x, y) = \psi_g(x, y, 0) \wedge \forall i, i < y \rightarrow \exists m, (\psi_g(x, i, m) \wedge m \neq 0)$$

# funzioni calcolabili

## Theorem

*Tutte le funzioni calcolabili sono aritmetiche.*

Conseguenza del fatto che ogni funzione calcolabile può essere espressa in un formalismo che contiene somma, prodotto, costanti, proiezioni ed è chiuso per composizione e ricorsione primitiva.

# Insiemi ricorsivamente enumerabili

## Theorem

*Ogni insieme ricorsivamente enumerabile è aritmetico.*

Se  $A$  è r.e. esiste  $f$  totale e calcolabile tale che  $A = \text{dom}(f)$ . Sia  $\psi_f(x, y)$  una descrizione aritmetica di  $f$ . Allora

$$n \in A \Leftrightarrow \mathcal{N} \models \exists y, \psi_f(\bar{n}, y)$$

e dunque  $\psi_A(x) = \exists y, \psi_f(x, y)$  è una descrizione aritmetica di  $A$ .

**corollario** L'insieme  $K$  è aritmetico.

# Insiemi ricorsivamente enumerabili

## Theorem

*Ogni insieme ricorsivamente enumerabile è aritmetico.*

Se  $A$  è r.e. esiste  $f$  totale e calcolabile tale che  $A = \text{dom}(f)$ . Sia  $\psi_f(x, y)$  una descrizione aritmetica di  $f$ . Allora

$$n \in A \Leftrightarrow \mathcal{N} \models \exists y, \psi_f(\bar{n}, y)$$

e dunque  $\psi_A(x) = \exists y, \psi_f(x, y)$  è una descrizione aritmetica di  $A$ .

**corollario** L'insieme  $K$  è aritmetico.

# Indecidibilità della verità

## Theorem

*L'insieme delle formule aritmetiche vere non è ricorsivamente enumerabile.*

Sia  $\{\psi_n\}_{n \in \mathcal{N}}$  una enumerazione effettiva delle formule aritmetiche in una variabile. Consideriamo l'insieme  $A$  così definito

$$n \in A \Leftrightarrow \models \neg \psi_n(\bar{n})$$

Se la verità aritmetica fosse semidecidibile, allora  $A$  sarebbe r.e.

Dunque  $A$  dovrebbe essere aritmetico e dovrebbe esistere una formula  $\psi_a$  per cui

$$n \in A \Leftrightarrow \models \psi_a(\bar{n})$$

Ma per  $n = a$  otteniamo una contraddizione.

# Incompletezza

## Theorem

*Ogni sistema formale aritmetico, se consistente, è incompleto, nel senso che esistono formule aritmetiche valide ma non dimostrabili.*

Un sistema formale è definito da un insieme ricorsivo di assiomi e un insieme di regole di inferenza che permettono di dedurre nuovi teoremi a partire dagli assiomi in un numero finito di applicazioni.

Pertanto le formule aritmetiche dimostrabili costituiscono un insieme ricorsivamente enumerabile. Poichè le formule vere non sono r.e. esistono necessariamente delle formule vere ma non dimostrabili.

# bibliography



A. Asperti, A. Ciabattoni.

*Logica a Informatica.*

McGraw-Hill, 1997.



N. J. Cutland.

*Computability: An Introduction to Recursive Function Theory.*

Cambridge University Press, Cambridge, UK, 1986.



P. G. Odifreddi.

*Classical Recursion Theory: the Theory of Functions and Sets of Natural Numbers*, volume 125 of *Studies in Logic and the Foundations of Mathematics*.

Elsevier, 1997.



H. Rogers.

*Theory of Recursive Functions and Effective Computability.*

MIT press, 1987.