

# Calcolabilità e Complessità

Andrea Asperti

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

## Calcolabilità

Capire quali problemi possono essere risolti in modo **algoritmico**

- ▶ Studiare e confrontare diversi meccanismi e costrutti computazionali
- ▶ Fornire una definizione precisa delle nozioni di calcolabilità e di decidibilità
- ▶ Fornire metodi e strumenti per capire se un problema è o meno decidibile
- ▶ Comprendere le implicazioni logiche (Teorema di Incompletezza di Gödel)

## Complessità

Capire quali problemi possono essere risolti in modo **efficiente**

- ▶ Fornire una nozione di costo computazionale
- ▶ Classificare i problemi in base alla loro efficienza
- ▶ Studiare la relazione tra calcolo deterministico e nondeterministico
- ▶ Studiare la relazione tra risorse differenti (tempo e spazio)
- ▶ Discutere i principali problemi aperti di teoria della complessità

## Calcolabilità

- ▶ N. J. Cutland. *Computability: An Introduction to Recursive Function Theory*. Cambridge University Press, Cambridge, UK, 1986.
- ▶ S. Barry Cooper, *Computability Theory*, Chapman & Hall, 2004.
- ▶ 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.

## Complessità

- ▶ Michael Sipser. *Introduzione alla teoria della computazione*. Maggioli editore, 2016.
- ▶ Christos H. Papadimitriou, *Computational Complexity*, Addison Wesley, 1994
- ▶ Sanjeev Arora and Boaz Barak. *Computational Complexity*. Cambridge University Press, 2009

# Calcolabilità

## Introduzione

- ▶ un risultato banale?
  - quante sono le funzioni da  $\mathcal{N}$  in  $\mathcal{N}$ ?
  - quanti sono i programmi?
- ▶ definibilità vs. calcolabilità
- ▶ la tesi di Church
  - formalismi subricorsivi
  - incompletezza dei formalismi totali
  - universalità della nozione di calcolabilità

## Risultato centrale    non tutte le funzioni sono calcolabili

Ripensare **ogni nozione** in ottica costruttiva

Esempi:

## Sviluppi

- ▶ ogni insieme ha una funzione caratteristica (f.c.).  
Ma non ogni funzione caratteristica è calcolabile.  
Quali insiemi hanno una f.c. **calcolabile**?
- ▶ ogni sottoinsieme di  $\mathcal{N}$  è enumerabile.  
Quali sottoinsiemi possono essere enumerati  
mediante una funzione **calcolabile**?
- ▶ ...

# Numerabilità e diagonalizzazione - Lezioni 1 e 2

---

- ▶ Enumerare, contare, calcolare
- ▶ Quali insiemi sono enumerabili?
- ▶ Dovetailing
- ▶ Alfabeti, stringhe, linguaggi
- ▶ Diagonalizzazione - Teorema di Cantor
- ▶ Diagonalizzazione e paradossi (Russel, Berry)

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  $\mathcal{N}$ ?

## 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 una quantità numerabile 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.

# Muoversi tra due infiniti: dovetailing

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  $\langle i, j \rangle$ .

	0	1	2	3	4	...	$i$
0	0	1	3	6	10		
1	2	4	7	11			
2	5	8	12				
3	9	13					
4	14						
$\vdots$							
$j$							

# Codifica delle coppie del piano

---

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

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

## 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.

## 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

## 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.

# Il paradosso di Russel

Russel

Sia  $U = \{x | x \notin x\}$ . Allora

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

**Principio di comprensione**

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

# Il problema della definibilità

---

Le funzioni *definibili* da  $\mathcal{N}$  in  $\mathcal{N}$  sono una quantità numerabile. Fissiamo un qualche enumerazione  $f_i$ .

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 paradosso di Berry

## Berry

“Sia  $n$  il più piccolo intero positivo non definibile con meno di 100 caratteri”.

- ▶ i numeri definibili con meno di 100 caratteri sono una quantità finita, dunque esistono (infiniti) numeri che soddisfano la proprietà suddetta, e tra questi ne deve esistere uno minimo.
- ▶ tale numero è dunque univocamente determinato dalla definizione di Berry.
- ▶ ma la definizione di Berry è più breve di 100 caratteri: contraddizione.

La “definizione” di Berry non è ben posta (la nozione di *definibilità* non è ben definita).

- la nozione di (e)numerabilità
- dovetailing: muoversi tra più infiniti
- diagonalizzazione
- “indefinibilità” della nozione di definibilità

# Formalismi totali

- ▶ funzioni ricorsive
- ▶ ricorsione primitiva
- ▶ somme e prodotti limitati
- ▶ predicati ricorsivi
- ▶ minimizzazione limitata
- ▶ ricorsione multipla
- ▶ ricorsione primitiva vs. iterazione
- ▶ la funzione di Ackermann
- ▶ ricorsione di ordine superiore
- ▶ incompletezza algoritmica dei formalismi totali

La definizione di una funzione  $f$  si dice *ricorsiva* se il valore di  $f$  su alcuni argomenti dipende dal valore di  $f$  su altri argomenti, solitamente più “semplici” rispetto ad una opportuna metrica da definire.

$$\begin{cases} fatt(0) = 1 \\ fatt(n+1) = (n+1) * fatt(n) \end{cases}$$

**Fattoriale**

$$\begin{cases} fibo(0) = 1 \\ fibo(1) = 1 \\ fibo(n+2) = fibo(n) + fibo(n+1) \end{cases}$$

**Fibonacci**

Le seguenti funzioni sono funzioni iniziali del linguaggio primitivo ricorsivo  $\mathcal{L}$ :

1. le **funzioni costanti**

$$c_m^k(x_1, x_2, \dots, x_k) = m \quad \text{con } m \in \mathcal{N}$$

2. le **proiezioni** (identità generalizzate)

$$\pi_i^k(x_1, x_2, \dots, x_k) = x_i \quad \text{per qualche } 1 \leq i \leq k$$

3. la funzione **successore**

$$s(x) = x + 1$$

## ► Composizione

Se  $h : \mathcal{N}^n \rightarrow \mathcal{N}$  e  $g_1, g_2, \dots, g_n : \mathcal{N}^k \rightarrow \mathcal{N}$  appartengono ad  $\mathcal{L}$ , anche la seguente funzione  $f : \mathcal{N}^k \rightarrow \mathcal{N} \in \mathcal{L}$ :

$$f(\vec{x}) = h(g_1(\vec{x}), g_2(\vec{x}), \dots, g_n(\vec{x}))$$

con  $\vec{x} = (x_1, x_2, \dots, x_k)$

## ► Ricorsione primitiva

Se  $g : \mathcal{N}^k \rightarrow \mathcal{N}$ ,  $h : \mathcal{N}^{k+2} \rightarrow \mathcal{N} \in \mathcal{L}$ , anche la seguente funzione  $f : \mathcal{N}^{k+1} \rightarrow \mathcal{N} \in \mathcal{L}$ :

$$f : \begin{cases} f(0, \vec{x}) = g(\vec{x}) \\ f(y+1, \vec{x}) = h(y, f(y, \vec{x}), \vec{x}) \end{cases}$$

con  $\vec{x} = (x_1, x_2, \dots, x_k)$

Una funzione  $f$  è *primitiva ricorsiva* se e solo se esiste una sequenza di funzioni  $f_1, f_2, \dots, f_n = f$  tale che ogni  $f_i$  o è una funzione base oppure è ottenuta da funzioni  $f_j$  con  $j < i$  mediante composizione o ricorsione primitiva.

## codice

$$f_1(x) = x$$

$$f_2(x) = x + 1$$

$$f_3(x, y, z) = y$$

$$f_4(x, y, z) = f_2(f_3(x, y, z))$$

$$f_5(y, x) = \begin{cases} f_5(0, x) = f_1(x) \\ f_5(y + 1, x) = f_4(y, f_5(y, x), x) \end{cases}$$

$$f \equiv f_6(x) = f_5(f_1(x), f_1(x))$$

## esecuzione

$$\begin{aligned} f(2) &= f_6(2) \\ &= f_5(f_1(2), f_1(2)) \\ &= f_5(f_1(2), 2) \\ &= f_5(2, 2) \\ &= f_4(1, f_5(1, 2), 2) \\ &= f_4(1, f_4(0, f_5(0, 2), 2), 2) \\ &= f_4(1, f_4(0, f_1(2), 2), 2) \\ &= f_4(1, f_4(0, 2, 2), 2) \\ &= f_4(1, f_2(f_3(0, 2, 2)), 2) \\ &= f_4(1, f_2(2), 2) \\ &= f_4(1, 3, 2) \\ &= f_2(f_3(1, 3, 2)) \\ &= f_2(3) \\ &= 4 \end{aligned}$$



# Abusi notazionali (inlining)

---

$$f_1(x) = x$$

$$f_2(x) = x + 1$$

$$f_3(x, y, z) = y$$

$$f_4(x, y, z) = f_2(f_3(x, y, z)) = y + 1$$

$$f_5(y, x) = \begin{cases} f_5(0, x) = f_1(x) = x \\ f_5(y + 1, x) = f_5(y, x) + 1 \end{cases}$$

$$f \equiv f_6(x) = f_5(f_1(x), f_1(x)) = f_5(x, x)$$

Abuseremo la notazione per ragioni di leggibilità

# Alcune funzioni primitive ricorsive

---

## Moltiplicazione

$$\begin{aligned}\text{mult}(0, x) &= 0 \\ \text{mult}(y + 1, x) &= \text{add}(x, \text{mult}(y, x))\end{aligned}$$

## Predecessore

$$\begin{aligned}\text{pred}(0) &= 0 \\ \text{pred}(y + 1) &= y\end{aligned}$$

## Zero

$$\begin{aligned}\text{zero}(0) &= 1 \\ \text{zero}(y + 1) &= 0\end{aligned}$$

## Fattoriale

$$\begin{aligned}\text{fatt}(0) &= 1 \\ \text{fatt}(y + 1) &= \text{mult}(s(y), \text{fatt}(y))\end{aligned}$$

## Sottrazione

$$\begin{aligned}\text{sub}(0, m) &= m \\ \text{sub}(n + 1, m) &= \text{pred}(\text{sub}(n, m))\end{aligned}$$

## Confronto

$$\text{eq}(n, m) = \text{zero}(\text{add}(\text{sub}(n, m), \text{sub}(m, n)))$$

$$\sigma_f(z, \vec{x}) \equiv \sum_{y \leq z} f(y, \vec{x}) : \begin{cases} \sigma_f(0, \vec{x}) = f(0, \vec{x}) \\ \sigma_f(z+1, \vec{x}) = \sigma_f(z, \vec{x}) + f(z+1, \vec{x}) \end{cases}$$

$$\pi_f(z, \vec{x}) \equiv \prod_{y \leq z} f(y, \vec{x}) : \begin{cases} \pi_f(0, \vec{x}) = f(0, \vec{x}) \\ \pi_f(z+1, \vec{x}) = \pi_f(z, \vec{x}) * f(z+1, \vec{x}) \end{cases}$$

# Predicati primitivi ricorsivi

Un predicato  $P$  è *primitivo ricorsivo* se e solo se lo è la sua funzione caratteristica  $\mathbf{c}_P$ . **Es:** zero e eq.

## Lemma

*I predicati primitivi ricorsivi sono chiusi rispetto ai connettivi logici.*

$$\begin{aligned}\mathbf{c}_{\neg P}(\vec{x}) &= 1 - \mathbf{c}_P(\vec{x}) \equiv \text{sub}(\mathbf{c}_P(\vec{x}), 1) \\ \mathbf{c}_{P \wedge Q}(\vec{x}) &= \mathbf{c}_P(\vec{x}) * \mathbf{c}_Q(\vec{x}) \equiv \text{mult}(\mathbf{c}_P(\vec{x}), \mathbf{c}_Q(\vec{x}))\end{aligned}$$

## Lemma

*I predicati primitivi ricorsivi sono chiusi rispetto alla quantificazione limitata. Ovvero, se  $P(z, \vec{x})$  è un predicato primitivo ricorsivo lo sono anche  $\forall_{z \leq y} P(z, \vec{x})$  e  $\exists_{z \leq y} P(z, \vec{x})$ .*

$$\mathbf{c}_{\forall_{z \leq y} P(z, \vec{x})} \equiv \prod_{z \leq y} \mathbf{c}_P(z, \vec{x})$$

## Divisibilità

$$\text{divide}(x, y) \iff \exists_{n \leq y} \text{eq}(\text{mult}(n, x), y)$$

## Primalità

$$\text{prime}(x) \iff x \geq 2 \wedge \forall_{y \leq x} (\text{divide}(y, x) \Rightarrow y = 1 \vee y = x)$$

Posso calcolare l'n-esimo numero primo?

$$\begin{cases} Pr(0) = 2 \\ Pr(n+1) = \min\{p \mid \text{prime}(p) \wedge (p > Pr(n))\} \end{cases}$$

# Minimizzazione limitata

Per *minimizzazione* ( $\mu$ -ricorsione) si intende la ricerca del minimo intero positivo  $\mu_y P(y)$  che soddisfa il predicato  $P$ .

La minimizzazione si dice *limitata* se viene fornito un bound alla ricerca.

## Lemma

*Le funzioni primitive ricorsive sono chiuse rispetto alla minimizzazione limitata.*

Dobbiamo dimostrare che se  $R(y, \vec{x})$  è un predicato primitivo ricorsivo, allora lo è anche  $\mu_{y < z} R(y, \vec{x})$ .

Consideriamo il predicato primitivo ricorsivo

$$h(y, \vec{x}) = \forall_{w \leq y} \neg R(w, \vec{x})$$

Supponiamo che  $y_0 = \mu_{y < z} R(y, \vec{x})$ . Allora per ogni valore di  $y < y_0$  la funzione  $h(y, \vec{x})$  assume valore 1 e per ogni valore  $y_0 \leq y < z$  la funzione  $h(y, \vec{x})$  assume valore 0.

Dunque,  $y_0$  è pari al numero di interi inferiori a  $z$  per cui  $h(y, \vec{x})$  assume valore 1, ovvero:

$$\mu_{y < z} R(y, \vec{x}) = y_0 = \sum_{y < z} \forall_{w \leq y} \neg R(w, \vec{x})$$

Le funzioni di codifica e decodifica delle coppie sono primitive ricorsive:

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

$$\text{fst}(p) = \mu_{x \leq p} \exists y \leq p \langle x, y \rangle = p$$

$$\text{snd}(p) = \mu_{y \leq p} \exists x \leq p \langle x, y \rangle = p$$

Mediante l'uso di coppie posso restituire in output **agglomerati di risultati**, cosa che permette di simulare la **ricorsione multipla**.

La seguente definizione non è primitiva ricorsiva:

$$\begin{cases} \text{fib}(0) = 1 \\ \text{fib}(1) = 1 \\ \text{fib}(n+2) = \text{fib}(n) + \text{fib}(n+1) \end{cases}$$

Idea: definire una funzione ausiliaria `fib'` che, su input  $x$  restituisca in output la *coppia* dei valori  $\langle \text{fib}(x), \text{fib}(x+1) \rangle$ :

$$\begin{aligned} \text{fib}'(0) &= \langle 1, 1 \rangle \\ \text{fib}'(x+1) &= \langle \text{fib}(x+1), \text{fib}(x+2) \rangle \\ &= \langle \text{snd}(\text{fib}'(x)), \text{fst}(\text{fib}'(x)) + \text{snd}(\text{fib}'(x)) \rangle \end{aligned}$$

La funzione `fib'` è primitiva ricorsiva, e dunque lo è anche `fibonacci`:

$$\text{fib}(n) = \text{fst}(\text{fib}'(n))$$



$$\langle 1, 1 \rangle$$

$$\langle 1, 2 \rangle$$

$$\langle 2, 3 \rangle$$

$$\langle 3, 5 \rangle$$

...

$$\langle \textit{fib}(n), \textit{fib}(n + 1) \rangle$$

# Calcolo di Fibonacci - prima componente

---

$\langle 1, 1 \rangle$



$\langle 1, 2 \rangle$



$\langle 2, 3 \rangle$




$\langle 3, 5 \rangle$





...


$\langle \text{fib}(n), \text{fib}(n+1) \rangle$

# Calcolo di Fibonacci - seconda componente

$$\langle 1, 1 \rangle$$


$$\langle 1, 2 \rangle$$


$$\langle 2, 3 \rangle$$


$$\langle 3, 5 \rangle$$


...



$$\langle \textit{fib}(n), \textit{fib}(n + 1) \rangle$$

# Ricorsione multipla

Si dice *storia* di  $f(y, \vec{x})$  la funzione  $\hat{f}(y, \vec{x})$  così definita:

$$\begin{aligned}\hat{f}(y, \vec{x}) &= \langle f(0, \vec{x}), f(1, \vec{x}), \dots, f(y, \vec{x}) \rangle_{y+1} \\ &\equiv \langle \langle f(0, \vec{x}), f(1, \vec{x}) \rangle, \dots \rangle, f(y, \vec{x}) \rangle_{\underbrace{>_2 >_2 \dots >_2}_y}\end{aligned}$$

La storia di  $f$  permette di definire il seguente *schema di ricorsione multipla*:

$$\begin{cases} f(0, \vec{x}) &= g(\vec{x}) \\ f(y+1, \vec{x}) &= h(y, \hat{f}(y, \vec{x}), \vec{x}) \end{cases}$$

**Problema:**  $f$  è primitiva ricorsiva?

Sì, infatti la storia di  $f$  è esprimibile mediante ricorsione primitiva:

$$\begin{cases} \hat{f}(0, \vec{x}) = f(0, \vec{x}) = g(\vec{x}) \\ \hat{f}(y+1, \vec{x}) = \langle \hat{f}(y, \vec{x}), f(y+1, \vec{x}) \rangle = \langle \hat{f}(y, \vec{x}), h(y, \hat{f}(y, \vec{x}), \vec{x}) \rangle \end{cases}$$

Si noti che la definizione precedente non dipende da  $f$  ma solo da  $g$  e  $h$ . Infine:

$$\begin{cases} f(0, \vec{x}) = \hat{f}(0, \vec{x}) \\ f(y+1, \vec{x}) = \text{snd}(\hat{f}(y+1, \vec{x})) \end{cases}$$

# Ricorsione primitiva e iterazione

Le funzioni primitive ricorsive sono **for-calcolabili**.

Esempio:

$$\begin{cases} f(0, \vec{x}) = g(\vec{x}) \\ f(y+1, \vec{x}) = h(y, f(y, \vec{x}), \vec{x}) \end{cases}$$
$$\begin{aligned} f(y, \vec{x}) &:= \\ &\quad acc = g(\vec{x}) \\ &\quad \text{for } i := 0 \text{ to } y \\ &\quad \quad acc := h(i, acc, \vec{x}) \\ &\quad \text{return } acc \end{aligned}$$

È possibile dimostrare il seguente risultato:

## Teorema

Le funzioni primitive ricorsive sono **tutte e solo** quelle **for-calcolabili**, cioè esprimibili in un linguaggio imperativo (del primo ordine) utilizzando come unici costrutti di controllo di flusso l'if-then-else, il for e la chiamata di funzione non ricorsiva.

**Domanda:** Ogni funzione calcolabile totale è for-calcolabile?

# La funzione di Ackermann

# Funzionali di ricorsione e iterazione

Una funzione definita mediante uno schema ricorsivo primitivo o iterativo è univocamente determinata dalle sue componenti  $g$  e  $h$ . Questo suggerisce la seguente notazione compatta, che utilizzeremo nelle prossime slides

$$f = \mathbf{Rec} \ g \ h \quad \text{per} \quad f = \begin{cases} f(0, \vec{x}) = g(\vec{x}) \\ f(y + 1, \vec{x}) = h(y, f(y, \vec{x}), \vec{x}) \end{cases}$$

$$f = \mathbf{Ite} \ g \ h \quad \text{per} \quad f = \begin{cases} f(0, \vec{x}) = g(\vec{x}) \\ f(y + 1, \vec{x}) = h(f(y, \vec{x})) \end{cases}$$

Rec e Ite sono esempi significativi di *funzionali*, ovvero funzioni di ordine superiore che operano su altre funzioni.

# La funzione di Ackermann

---

$$\begin{aligned}\text{ack}(0, 0, y) &= y \\ \text{ack}(0, x + 1, y) &= \text{ack}(0, x, y) + 1 \\ \text{ack}(1, 0, y) &= 0 \\ \text{ack}(z + 2, 0, y) &= 1 \\ \text{ack}(z + 1, x + 1, y) &= \text{ack}(z, \text{ack}(z + 1, x, y), y)\end{aligned}$$

## Funzione di Ackermann

La funzione di Ackermann è ben definita, implementabile, e terminante; tuttavia **non è esprimibile** nel formalismo primitivo ricorsivo.



# Istanze della funzione di Ackermann

Consideriamo le istanze parziali

$$\text{ack}_{z,y}x = \text{ack}(z, x, y)$$

In base alle prime due equazioni

$$\text{ack}_{0,y}(x) = x + y$$

In base all'ultima equazione abbiamo inoltre che

$$\begin{aligned}\text{ack}_{z+1,y}(x) &= \text{ack}_{z,y}(\text{ack}_{z+1,y}(x-1)) \\ &= \text{ack}_{z,y}(\text{ack}_{z,y}(\text{ack}_{z+1,y}(x-2))) \\ &= \underbrace{\text{ack}_{z,y}(\text{ack}_{z,y}(\dots \text{ack}_{z,y}(\text{ack}_{z+1,y}(0)) \dots))}_{x \text{ volte}}\end{aligned}$$

Dunque il risultato di  $\text{ack}_{z+1,y}(x)$  si ottiene iterando  $x$  volte la funzione del livello sottostante  $\text{ack}_{z,y}$  a partire dal valore base  $\text{ack}_{z+1,y}(0)$ . Le equazioni 3 e 4 fissano rispettivamente tale valore base a 0 per  $z = 1$  e a 1 per  $z \geq 2$ .

Dunque

$$\text{ack}_{1,y} = \text{lte } 0 \text{ ack}_{0,y}$$

$$\text{ack}_{z+2,y} = \text{lte } 1 \text{ ack}_{z+1,y}$$

# Crescita della funzione di Ackermann

Esplicitando i primi livelli della funzione abbiamo:

$$\begin{aligned}\text{ack}(0, x, y) &= x + y \\ \text{ack}(1, x, y) &= x * y \\ \text{ack}(2, x, y) &= y^x \\ \text{ack}(3, x, y) &= \left. y^{y^{\dots^y}} \right\} x \text{ volte}\end{aligned}$$

La funzione di Ackermann cresce dunque *molto* velocemente; ad esempio

$$\text{ack}(3, 3, 3) = 3^{27} > 10^{14}$$

La *complessità* della funzione di Ackermann trascende il potere espressivo del linguaggio primitivo ricorsivo.

# Iterare un iteratore

Posto

$$\text{next } f = \text{Ite } 1 \ f$$

abbiamo, per  $z > 1$

$$\text{ack}_{z,y} = \text{next } \text{ack}_{z-1,y}$$

e dunque

$$\begin{aligned}\text{ack}_{z+1,y} &= \underbrace{\text{next} \dots \text{next}}_{z \text{ volte}} \text{ack}_{1,y} \\ &= \text{Ite } \text{ack}_{1,y} \text{ next } z\end{aligned}$$

Se per calcolare Ackermann è sufficiente iterare, perchè non è primitiva ricorsiva?

Perchè si richiede di **iterare un funzionale** (funzione di ordine superiore).

# Iterare un iteratore

Posto

$$\text{next } f = \text{Ite } 1 \ f$$

abbiamo, per  $z > 1$

$$\text{ack}_{z,y} = \text{next } \text{ack}_{z-1,y}$$

e dunque

$$\begin{aligned} \text{ack}_{z+1,y} &= \underbrace{\text{next} \dots \text{next}}_{z \text{ volte}} \text{ack}_{1,y} \\ &= \text{Ite } \text{ack}_{1,y} \text{ next } z \end{aligned}$$

Se per calcolare Ackermann è sufficiente iterare, perchè non è primitiva ricorsiva?

Perchè si richiede di **iterare un funzionale** (funzione di ordine superiore).

# Gerarchie di linguaggi totali

## Il problema dell'interprete

# Estensioni del formalismo primitivo ricorsivo

È possibile aumentare il potere espressivo del formalismo primitivo ricorsivo in modo da permettere di implementare la funzione di Ackermann?

Si. Due direzioni sono possibili:

- ▶ indebolire il principio di ricorsione strutturale, ad esempio operando su più argomenti e accettando la ricorsione se l'**ordinamento lessicografico** di questi diminuisce. Ad esempio, accettiamo la chiamata

$$\text{ack}(z + 1, x + 1, y) = \text{ack}(z, \text{ack}(z + 1, x, y), y)$$

poichè in tale ordinamento, qualunque sia  $m = \text{ack}(z + 1, x, y)$ ,

$$(z, m, y) < (z + 1, x + 1, y)$$

- ▶ manteniamo il principio strutturale, ma estendiamo il sistema di tipi in modo da considerare **funzioni di ordine superiore**. Questo permette di iterare funzionali aggirando il problema di cui alla pagina precedente. Il linguaggio che si ottiene in questo modo è il **Sistema T** di Gödel (in appendice).  
Gödel ha dimostrato che le funzioni del Sistema T sono tutte e solo quelle **dimostrabilmente totali** nell'aritmetica di Peano.

# Estensioni del formalismo primitivo ricorsivo

È possibile aumentare il potere espressivo del formalismo primitivo ricorsivo in modo da permettere di implementare la funzione di Ackermann?

Sì. Due direzioni sono possibili:

- ▶ indebolire il principio di ricorsione strutturale, ad esempio operando su più argomenti e accettando la ricorsione se l'**ordinamento lessicografico** di questi diminuisce. Ad esempio, accettiamo la chiamata

$$\text{ack}(z + 1, x + 1, y) = \text{ack}(z, \text{ack}(z + 1, x, y), y)$$

poichè in tale ordinamento, qualunque sia  $m = \text{ack}(z + 1, x, y)$ ,

$$(z, m, y) < (z + 1, x + 1, y)$$

- ▶ manteniamo il principio strutturale, ma estendiamo il sistema di tipi in modo da considerare **funzioni di ordine superiore**. Questo permette di iterare funzionali aggirando il problema di cui alla pagina precedente. Il linguaggio che si ottiene in questo modo è il **Sistema T** di Gödel (in appendice).

Gödel ha dimostrato che le funzioni del Sistema T sono tutte e solo quelle **dimostrabilmente totali** nell'aritmetica di Peano.

- i vincoli sintattici che garantiscono la terminazione dei programmi consentono la dimostrabilità della totalità della funzione calcolata da detti programmi in sistemi logici sufficientemente espressivi.

*Ad esempio la totalità delle funzioni primitive ricorsive può essere dimostrata nell'aritmetica di Peano.*

- viceversa, ogni sistema logico permette di dimostrare la totalità di una certa classe di funzioni, che possono tipicamente essere caratterizzate in forma sintattica ed espresse in un opportuno linguaggio di programmazione

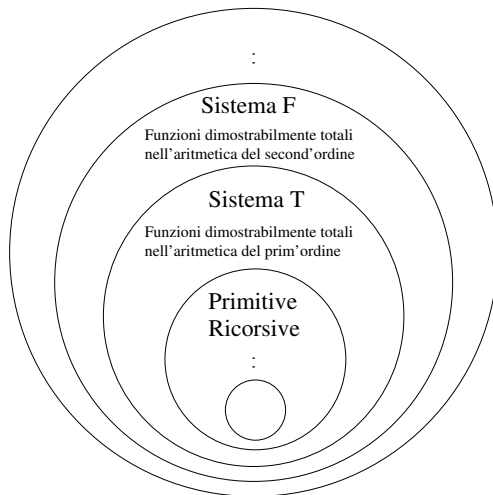
*Ad esempio le funzioni dimostrabilmente totali nell'aritmetica di Peano possono essere espresse nel sistema  $T$ .*

- data l'incompletezza dei sistemi logici, è naturale attendersi che in qualunque sistema logico esistano funzioni totali non dimostrabilmente totali nel sistema in esame.

*È quindi naturale aspettarsi una gerarchia infinita di formalismi totali con espressività crescente.*



# Gerarchie di Linguaggi di programmazione totali



# Il problema dell'Interprete

---

Sia data una enumerazione effettiva (e.g. lessicografica)  $P_n$  dei programmi del linguaggio  $\mathcal{L}$ , e sia  $\varphi_n$  la funzione calcolata dal programma  $P_n$ .

Un **interprete** per  $\mathcal{L}$  è una funzione che preso in input l'indice  $n$  di un programma ed un input  $m$  simula il comportamento di  $P_n$  su tale input, cioè una funzione  $I$  tale che

$$I(n, m) = \varphi_n(m)$$

Se la numerazione dei programmi è effettiva,  $I$  è **intuitivamente calcolabile**.

Ci chiediamo se l'interprete per  $\mathcal{L}$  può essere scritto in  $\mathcal{L}$ , cioè se esiste  $u$  tale che  $I = \varphi_u$ .

# Incompletezza algoritmica dei Formalismi Totali

Supponiamo che esista  $u$  tale che  $I(n, m) = \varphi_u(n, m) = \varphi_n(m)$ .  
Consideriamo la funzione

$$f(x) = \varphi_u(x, x) + 1 = \varphi_x(x) + 1$$

Se il linguaggio  $\mathcal{L}$  è chiuso per composizione  $f \in \mathcal{L}$ , e dunque deve esistere un programma  $i$  tale che  $\varphi_i = f$ .

Ma allora

$$\varphi_i(i) = f(i) = \varphi_i(i) + 1$$

che è assurdo.

## Teorema

Nessun formalismo totale è in grado di esprimere il proprio interprete.

- “for” v.s. “while”
- la funzione di Ackermann
- gerarchia infinita di linguaggi di programmazione per funzioni totali
- il problema dell'interprete

# La tesi di Church

# Le Macchine di Turing e la Tesi di Church - Lezioni 6, 7

---

- ▶ Costrutti computazionali potenzialmente divergenti
- ▶ La tesi di Church
- ▶ Le Macchine di Turing
- ▶ La Macchina universale

# Alcuni importanti costrutti potenzialmente divergenti

---

- ▶ goto
- ▶ cicli while
- ▶ minimizzazione ( $\mu$ -ricorsione illimitata)
- ▶ ricorsione generale
- ▶ costrutti autoreferenziali (auto-applicazione, auto-interpretazione)
- ▶ ...

I linguaggi parziali ammettono tipicamente la definizione del loro proprio interprete.

**Abbiamo una gerarchia infinita di linguaggi parziali con potere espressivo crescente?**

# Sulla auto-applicazione

L'auto applicazione **induce divergenza**. Posto

$$f\ x = x\ x$$

la computazione di  $f\ f$  non termina.

L'auto applicazione permette di **simulare la ricorsione**.

Supponiamo di voler definire una funzione ricorsiva  $f$  tale che

$$f\ x = M[f, x]$$

Consideriamo la funzione ausiliaria

$$h\ y\ x = M[y\ y, x]$$

dove le occorrenze di  $f$  in  $M$  sono state rimpiazzate dalla auto applicazione  $y\ y$ .

Posto  $f = h\ h$  abbiamo

$$f\ x = h\ h\ x = M[h\ h, x] = M[f, x]$$

**Remark:** L'auto applicazione è il tipico esempio di costruito mal tipato.



- ▶ (Term) rewriting systems (Thue 1914, Post 1920)
- ▶ Combinatory Logic (Schönfinkel, Curry 1924)
- ▶ Funzioni definite da equazioni ricorsive (Herbrand 1931, Gödel 1934, Kleene 1936)
- ▶  $\lambda$ -calcolo (Church 1936)
- ▶ Turing machines (Turing 1936)
- ▶ Funzioni  $\mu$ -ricorsive (Kleene 1938)
- ▶ Markov algorithms (Markov 1954)
- ▶ Register machines (Shepherdson, Sturgis 1963)
- ▶ ...

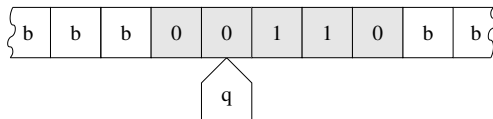
Tutti questi modelli (e molti altri) sono computazionalmente equivalenti.  
Le funzioni esprimibili in questi modelli sono dette *funzioni calcolabili*.

## Church

Le funzioni calcolabili sono esattamente quelle intuitivamente calcolabili mediante una procedura effettiva di calcolo.

*"Tarski has stressed in his lecture (and I think justly) the great importance of the concept of general recursiveness (or Turing's computability). It seems to me that this importance is largely due to the fact that with this concept one has for the first time succeeded in giving an absolute notion to an interesting epistemological notion, i.e., one not depending on the formalism chosen." (Gödel 1946)*

**Remark:** La tesi di Church non può essere dimostrata.



## 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 *b* (bianco) di inizializzazione.
- ▶ una testina di lettura mobile.
- ▶ un automa a stati finiti.

## Operazioni elementari

- ▶ leggere e scrivere il carattere individuato dalla testina
- ▶ spostare la testina di una posizione verso destra o verso sinistra
- ▶ modificare lo stato interno dell'automa

# La Macchina di Turing: definizione formale

Una Macchina di Turing (one-tape, deterministica) è una tupla  $\langle Q, \Gamma, b, \Sigma, \delta, q_0, F \rangle$  dove

- ▶  $Q$  è un insieme finito di **stati**
- ▶  $\Gamma$  è l'**alfabeto** finito del nastro
- ▶  $b \in \Gamma$  è il **carattere bianco**
- ▶  $\Sigma \subseteq \Gamma \setminus \{b\}$  è l'insieme dei **caratteri di input/output**
- ▶  $q_0 \in Q$  è lo **stato iniziale**
- ▶  $F \subseteq Q$  è l'insieme degli **stati finali** (o di accettazione)
- ▶  $\delta : Q \setminus F \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$  è la **funzione di transizione**.

$L$  (left) e  $R$  (right) denotano le possibili mosse della testina.

**Remark:** la funzione di transizione ha un grafo finito. Ogni elemento del grafo è una **quintupla**  $(q, a, q', a', M)$  tale che  $\delta(q, a) = (q', a', M)$ . L'insieme finito delle quintuple può essere visto come l'insieme delle istruzioni della macchina (programma), e la determina univocamente.

# Convenzioni di input-output

---

## input

- ▶ si suppone che il nastro sia inizializzato con la stringa di input (un carattere per ogni cella)
- ▶ la testina è posizionata sul primo carattere dell'input.
- ▶ tutte le altre celle del nastro sono inizializzate col carattere speciale  $b$ .

## output

- ▶ nel momento in cui la macchina si arresta l'output è la più lunga stringa di caratteri in  $\Gamma$  (in particolare, senza  $b$ ) alla destra della testina

# Configurazioni istantanee

Una **configurazione** è una descrizione dello stato della computazione ad un dato istante della computazione. Questa è definita come una tripla

$$\sigma, q, \tau$$

dove  $q$  è lo stato dell'automa e  $\sigma$  e  $\tau$  sono due stringhe di caratteri che descrivono la porzione non (definitivamente) bianca del nastro alla sinistra e alla destra della testina. Il carattere in lettura è il primo carattere di  $\tau$ .

La computazione avviene per **passi discreti**: una transizione tra due configurazioni è una relazione  $\vdash$  governata dalla funzione di transizione.

$$\begin{array}{ll} \sigma, q, a\tau \vdash \sigma a', q', \tau & \text{se } \delta(q, a) = (q', a', R) \\ \sigma c, q, a\tau \vdash \sigma, q', ca'\tau & \text{se } \delta(q, a) = (q', a', L) \end{array}$$

Nelle regole precedenti si suppone che il nastro possa essere esteso “on demand” con caratteri bianchi qualora se ne abbia necessità.

La relazione  $\vdash^*$  denota la chiusura transitiva e riflessiva della relazione  $\vdash$ .

Una funzione  $f : \Sigma^* \rightarrow \Sigma^*$  è calcolata da una macchina di Turing  $M$  se per ogni  $\alpha$  esiste  $q_f \in F$  tale che

$$\epsilon, q_0, \alpha \vdash^* \sigma, q_f, \tau$$

e  $f(\alpha)$  è il più lungo prefisso di  $\tau$  appartenente a  $\Sigma^*$

# 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

Esiste una **MdT Universale** che prende in input la sequenza di quintuple di una macchina  $M$ , un input  $m$  e simula il comportamento di  $M$  su input  $m$ .

- ▶ divido il nastro in una parte superiore ed una inferiore. La parte superiore è utilizzata per memorizzare le quintuple della MdT da simulare  $M$ , quella inferiore è il nastro di lavoro, su cui viene inizialmente copiato il dato di input  $m$ ;
- ▶ lo stato interno comprende tre “registri” speciali  $Q$ ,  $A$  e  $S$  utilizzati per memorizzare rispettivamente lo stato della macchina  $M$ , il carattere di lettura/scrittura e lo shift (R/L) della testina; inizialmente  $Q = q_0^M$ ,
- ▶ letto il carattere  $a$  sul nastro inferiore, rimpiozzo  $a$  con un carattere speciale  $\#$  per ricordare la posizione della testina e memorizzo  $a$  nel registro  $A$ ;
- ▶ procedo dunque a cercare nella parte superiore del nastro una quintupla relativa ai valori correnti di  $Q$  e  $A$ ; quando la trovo, rimpiozzo i contenuti dei registri con il valore del nuovo stato  $q'$ , del carattere da scrivere  $a'$  e della mossa da eseguire  $s$ ; se  $q'$  è finale la computazione si arresta;
- ▶ torno quindi a cercare il carattere  $\#$  sul nastro inferiore, lo rimpiozzo con il contenuto di  $A$ , eseguo la mossa  $S$  e ricomincio il ciclo;

Se leggo l'insieme delle quintuple come la codifica numerica (indice) di  $M$ , la MdT 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.

# Numerazioni di Gödel - Lezione 8

---

- ▶ Numerazioni accettabili
- ▶ Proprietà  $utm$  e  $smn$
- ▶ Riducibilità di enumerazioni
- ▶ Il teorema di equivalenza di Roger
- ▶ Il predicato  $T$  di Kleene

# Numerazione delle funzioni calcolabili

---

Dato un formalismo turing completo, possiamo enumerare i programmi  $P_i$  definibili nel formalismo dato. Detta  $\varphi_i$  la funzione calcolata da  $P_i$ , la numerazione dei programmi **induce dunque una enumerazione di tutte le funzioni calcolabili**.

Cercheremo nel seguito di caratterizzare alcune **delle proprietà salienti** della enumerazione  $\varphi_i$ , in modo da lavorare ad un livello sufficiente di astrazione rispetto al modello di calcolo sottostante.

# Chiusura per composizione

---

La composizione di funzioni calcolabili è una funzione calcolabile.

Ci aspettiamo quindi che l'insieme delle funzioni enumerate da  $\varphi$  sia chiuso rispetto alla composizione.

Se la numerazione dei programmi è effettiva, ci aspettiamo che esista un interprete:

## Proprietà utm (Universal Turing Machine)

Esiste un indice  $u$  tale che per ogni  $x, y$

$$\varphi_u(x, y) = \varphi_x(y)$$

# Proprietà smn

Se una funzione  $f(x, y)$  è calcolabile, allora per ogni possibile valore  $a$  di  $x$ , l'istanza

$$f_a(y) = f(a, y)$$

è calcolabile. Dunque esiste un qualche indice che dipende da  $a$ , diciamo  $h(a)$  per cui

$$\varphi_{h(a)}(y) = f_a(y) = f(x, y)$$

La proprietà smn asserisce che questa  $h$  è **totale e calcolabile**, cioè che (l'indice per) il programma che calcola l'istanza relativa ad  $a$  può essere ottenuto in modo effettivo a partire da  $a$ .

## Proprietà smn - caso binario

Per ogni funzione binaria parziale calcolabile  $f$  esiste una funzione *totale calcolabile*  $h$  tale che, per ogni  $x, y$

$$\varphi_{h(x)}(y) = f(x, y)$$

# Proprietà smn - caso generale

La proprietà precedente può essere generalizzata considerando  $m$  variabili come parametri per l'istanza, e  $n$  variabili aggiuntive.

## Proprietà smn

Per ogni funzione parziale calcolabile  $f^{m+n}$  esiste una funzione *totale* calcolabile  $s_n^m$  tale che, per ogni  $\vec{x}_m, \vec{y}_n$

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

# Numerazioni accettabili

---

Fissato un **qualunque** linguaggio di programmazione (ad esempio Macchine di Turing), le proprietà utm e smn **possono essere dimostrate**.

Noi le assumeremo “assiomaticamente”, come punto di partenza della teoria della calcolabilità che vogliamo sviluppare.

Una enumerazione che gode delle proprietà utm e smn è detta **accettabile**.

D'ora in avanti supporremo di lavorare con una numerazione accettabile delle funzioni parziali calcolabili.



# Riducibilità di enumerazioni

Siano date due funzioni di enumerazione  $\varphi, \psi : \mathcal{N} \rightarrow A$ .

1.  $\psi$  è **riducibile** a  $\varphi$  (in simboli:  $\psi \leq \varphi$ ) se esiste una funzione totale calcolabile  $f$  tale che, per ogni numero naturale  $n$ ,  $\psi_n = \varphi_{f(n)}$
2.  $\psi$  è **equivalente** a  $\varphi$  (in simboli:  $\psi \equiv \varphi$ ) se  $\psi \leq \varphi$  e  $\varphi \leq \psi$ .

È possibile dimostrare il seguente risultato:

**Roger**

Tutte le enumerazioni accettabili di funzioni parziali ricorsive sono equivalenti tra loro.

Quindi tutta la teoria che andremo a sviluppare è sostanzialmente indipendente dalla numerazione.

# Interpretazione di s.m.n.

---

funzioni binaria v.s. famiglia di funzioni unarie (curryficazione)

$$A \times B \rightarrow C \approx A \rightarrow (B \rightarrow C)$$

```
def currfify(f):  
    def fa(a): return (lambda b: f(a,b))  
    return fa
```

```
def mul(a,b): return a*b
```

```
mulc = currfify(mul)  
mul4 = mulc(4)  
mul4(9)
```

$$A \times B \rightarrow C \approx A \rightarrow (B \rightarrow C)$$

$$a, b \vdash f(a, b) \quad a \vdash s(a)$$

$$\begin{array}{l} A \times B \rightarrow C \approx A \rightarrow (B \rightarrow C) \\ a, b \vdash f(a, b) \quad a \vdash \text{index of } s(a) \end{array}$$

$$\begin{array}{l} A \times B \rightarrow C \approx A \rightarrow (B \rightarrow C) \\ a, b \vdash f(a, b) \quad a \vdash \text{index of } s(a) \end{array}$$

$$\varphi_{s(a)}(b) = f(a, b)$$

# Interprete bound e predicato T di Kleene

---

L'interprete, dovendo interpretare programmi potenzialmente divergenti è esso stesso **potenzialmente divergente**.

È possibile tuttavia fornire una **"approssimazione" terminante** (bound) dell'interprete, limitando il consumo delle risorse di calcolo (passate in input all'interprete stesso).

Posso cioè chiedere all'interprete bound di simulare la computazione della funzione  $i$  su input  $x$  con risorse limitate (e.g. timeout)  $t$  e osservare lo stato della computazione all'esaurimento delle risorse.

Per formalizzare questa idea si fa uso del **predicato T di Kleene**.

# Il predicato T di Kleene

## Il predicato T di Kleene

Esiste un predicato  $T(i, n, k, t)$  tale che

1.  $\varphi_i(n) = k \Leftrightarrow \exists t \geq k, T(i, n, k, t)$
2. la funzione caratteristica di T è calcolabile

T è il predicato (intuitivamente calcolabile) che afferma che la computazione del programma di indice  $i$  su input  $n$  termina **con risorse fissate** (e.g. tempo o spazio)  $t$  e fornisce come risultato  $k$ . Si suppone che le risorse necessarie a produrre  $k$  siano almeno pari a  $k$  (fissando opportunamente l'unità di misura).

Si osservi che anche la funzione caratteristica del predicato

$$T^3(i, n, t) = \exists k, T(i, n, k, t) \equiv \exists k \leq t, T(i, n, k, t)$$

è ancora calcolabile, ovvero

**È possibile decidere se una computazione si arresta in un tempo dato**

# La forma normale di Kleene

Il predicato  $T$  di Kleene fornisce una visione più fine della nozione di interprete, in cui si evidenzia la nozione di **costo computazionale**.

In particolare

## Teorema della forma normale di Kleene

Per ogni  $i, n$

$$\varphi_i(n) = fst(\mu_{\langle k, t \rangle} T(i, n, k, t))$$



# Funzioni non calcolabili - Lezione 9

---

- ▶ Notazione
- ▶ Terminazione
- ▶ Totalità
- ▶ Equivalenza
- ▶ Tre funzioni “simili”

# Un po' di notazione

Sia  $\varphi_i$  una enumerazione accettabile delle funzioni parziali calcolabili.

Useremo la notazione  $\varphi_i(n) \downarrow$  per indicare che la funzione è **definita** (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:

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

Le funzioni parziali sono **ordinate parzialmente** rispetto all'inclusione insiemistica dei loro grafi:

$$\varphi_i \subseteq \varphi_j \Leftrightarrow \forall n \in \text{dom}(\varphi_i), \varphi_i(n) = \varphi_j(n)$$

# Il problema della terminazione

Il **test di terminazione** è così definito:

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

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

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

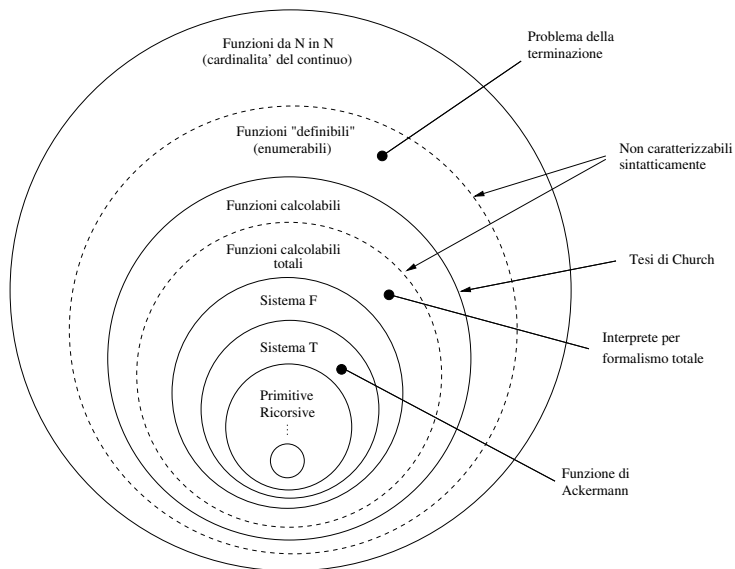
Se  $h$  è calcolabile lo è anche  $f$ . Dovrebbe dunque esistere  $m$  tale che  $f = \varphi_m$ .  
Supposto  $f = \varphi_m$ , ci chiediamo quanto vale  $\varphi_m(m)$ :

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

il che è assurdo. Questo dimostra che

**il test di terminazione non è una funzione calcolabile!**

# The big picture



# 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.

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

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

Sappiamo che  $total \circ h$  non è calcolabile, e dunque non lo è neppure  $total$ .

# 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:

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

Poniamo

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

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

# Tre funzioni simili

---

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

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

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

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  $g$ .



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

**Accortezza:** muoversi con una tecnica di dovetailing tra i due infiniti dei possibili valori di input e del tempo di esecuzione.

Sia  $t(i, n, s)$  la funzione caratteristica del predicato (ternario)  $T$  di Kleene.

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

# La funzione $g''$

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

Consideriamo la seguente funzione  $h$  totale calcolabile

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

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

- ▶ Insiemi ricorsivi
- ▶ Insiemi ricorsivamente enumerabili (r.e)
- ▶ Enumerabilità crescente
- ▶ Caratterizzazioni equivalenti
- ▶ Il Teorema di proiezione
- ▶ Proprietà di chiusura

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
5. tutti gli insiemi definiti da predicati primitivi ricorsivi

## Non ogni insieme è ricorsivo

(contro) Esempio:

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

## Lemma

*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*).

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

## Lemma

*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:

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

## Lemma

*Un insieme  $A$  è ricorsivo se e solo se può essere enumerato in modo crescente.*

$\Rightarrow$  si veda la prova del lemma precedente

$\Leftarrow$  Se  $A$  è finito l'asserto è ovvio. Possiamo dunque supporre  $A$  infinito; sia  $f$  la funzione di enumerazione. Definiamo la seguente funzione:

$$c_A(x) = f(\mu_y f(y) \geq x) = x$$

(scandisco tutti i dati enumerati da  $f$  finché non ne trovo uno non inferiore all'input  $x$ : a questo punto arresto la ricerca e verifico se il dato coincide con l'input cercato).

Il fatto (non ovvio) che  $c_A$  è totale è una conseguenza del fatto che  $A$  è infinito e dunque  $\text{cod}(f)$  contiene valori arbitrariamente grandi (superiori ad ogni  $x$ ).

# Teorema di complementazione

## Teorema

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*.



# Caratterizzazioni equivalenti degli insiemi r.e.

## Teorema

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

- $1 \Rightarrow 2$  Il caso  $A = \emptyset$  è ovvio. Sia  $A = \text{cod}(f)$  per  $f$  tot. calc., 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  $A = \text{dom}(g)$ ; basta considerare

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

- $3 \Rightarrow 1$  Sia  $A = \text{cod}(h)$ , per  $h$  parziale calcolabile.

Il caso  $A = \emptyset$  è triviale. Posto  $a \in A$  e  $h = \varphi_i$  consideriamo

$$f(\langle x, k, s \rangle) = \begin{cases} k & \text{se } T(i, x, k, s) = 1 \\ a & \text{altrimenti} \end{cases}$$

dove  $T$  è il predicato di Kleene.  $f$  è totale e calcolabile e  $\text{cod}(f) = A$ .

$A$  decidibile

$$f(x) = \begin{cases} 1 & \text{se } x \in A \\ 0 & \text{se } x \notin A \end{cases}$$

$A$  semidecidibile

$$f(x) = \begin{cases} \downarrow & \text{se } x \in A \\ \uparrow & \text{se } x \notin A \end{cases}$$

per  $f$  calcolabile

Esistono insiemi semidecidibili (r.e) ma non decidibili (ricorsivi):

## Teorema

L'insieme  $K = \{x \mid \varphi_x(x) \downarrow\}$  è r.e.

La funzione  $k(x) = \varphi_x(x)$  è calcolabile e  $K = \text{dom}(k)$ .

# Enumerazione degli insiemi r.e.

---

Possiamo definire una **enumerazione**  $W : \mathcal{N} \rightarrow \mathcal{RE}$  dell'insieme  $\mathcal{RE}$  di tutti gli insiemi r.e. ponendo

$$W_i = \text{dom}(\varphi_i)$$

Si noti che

$$K = \{x \mid \varphi_x(x) \downarrow\} = \{x \mid x \in \text{dom}(\varphi_x)\} = \{x \mid x \in W_x\}$$

# Il teorema di proiezione

## Teorema di proiezione

Un insieme  $A$  è r.e. se e solo se esiste un insieme  $B$  ricorsivo tale che

$$A = \{m \mid \exists n, \langle n, m \rangle \in B\}$$

- $\Leftarrow$  Sia  $c_B$  la funzione caratteristica di  $B$ . Allora  $A = \text{dom}(f)$  dove

$$f(m) = \mu n, c_B(\langle n, m \rangle) = 1$$

- $\Rightarrow$  Sia  $A = \text{dom}(\varphi_i)$ . Dunque  $m \in A$  se e solo se  $\varphi_i(m) \downarrow$ , se e solo se  $\exists n, T(i, n, m)$ , dove  $T$  è il predicato ternario di Kleene.

Basta dunque porre

$$B = \{\langle n, m \rangle \mid T(i, n, m)\}$$

# Proprietà di chiusura degli insiemi r.e.

## Teorema

Gli insiemi r.e. sono chiusi rispetto a unione e intersezione, ma non rispetto alla complementazione.

**unione** Sia  $A = \text{cod}(f')$  e  $B = \text{cod}(g')$  con  $f'$  e  $g'$  parziali calc.  
 $A \vee B = \text{cod}(h')$  dove

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

**intersezione** Sia  $A = \text{dom}(f)$  e  $B = \text{dom}(g)$  per  $f, g$  parziali calc.  
 $A \wedge B = \text{dom}(h)$  per  $h(x) = f(x) * g(x)$ .

**complementazione**  $\overline{K}$  non è r.e. (altrimenti  $K$  sarebbe ricorsivo).

**Corollario** Esistono insiemi che non sono nè ricorsivi nè ricorsivamente enumerabili (e.g.  $\overline{K}$ ).

## Teorema

Siano  $A, B \subseteq \mathcal{N}$  e  $f : \mathcal{N} \rightarrow \mathcal{N}$

1. se  $A$  è ricorsivo e  $f$  è totale calcolabile, allora  $f^{-1}(A)$  è ricorsivo
2. se  $A$  è r.e. e  $f$  è calcolabile, allora  $f^{-1}(A)$  è r.e
3. se  $A$  è r.e. e  $f$  è calcolabile, allora  $f(A)$  è r.e

1.  $c_{f^{-1}(A)} = c_A \circ f$
2. se  $A = \text{dom}(g)$ , allora  $f^{-1}(A) = \text{dom}(g \circ f)$
3. se  $A = \text{cod}(g)$ , allora  $f(A) = \text{cod}(f \circ g)$

# Unioni e intersezioni infinite

## Lemma

- 1) una unione r.e. di insiemi r.e. è ancora r.e.
- 2) una intersezione r.e. di insiemi r.e. non è necessariamente r.e.

$$1) \text{ per ogni } x \bigcup_{i \in W_x} W_i \text{ è r.e.} \quad 2) \text{ esiste } x \bigcap_{i \in W_x} W_i \text{ non è r.e.}$$

- 1) Per dovetailing.
- 2) Per smn esiste  $h$  totale calcolabile tale che

$$W_{h(i)} = \{x \mid \neg T(x, x, i)\}$$

dove  $T$  è il predicato di Kleene. Si dimostra facilmente che

$$\bigcap_{i \in \mathcal{N}} W_{h(i)} = \overline{K}$$

Poichè  $\bigcap_{i \in \mathcal{N}} W_{h(i)} = \bigcap_{i \in \text{cod}(h)} W_i$  l'asserto è dimostrato.

**Remark:** gli insiemi  $W_{h(i)}$  sono *ricorsivi* e  $h$  può essere scelta monotona crescente, quindi una intersezione *ricorsiva* di insiemi *ricorsivi* non è in generale r.e.

# I teoremi di Rice e Rice-Shapiro - Lezioni 12,13

---

- ▶ Insiemi estensionali
- ▶ Il teorema di Rice
- ▶ Il teorema di Rice-Shapiro (monotonia)
- ▶ Il teorema di Rice-Shapiro (compattezza)



# 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 alla **funzione calcolata** (estensione) e non alla **forma** o al **modo** (intensione) in cui questa viene calcolata.

$P(i)$ estensionale	$P(i)$ intensionale
$\varphi_i$ è totale	$\forall n \exists k, T(i, n, k, i^2)$
$\varphi_i \equiv f$	$\varphi_i \equiv f \wedge i \leq 100$
$5 \in \text{cod}(\varphi_i)$	$i \in \text{cod}(\varphi_i)$
$\text{dom}(\varphi_i)$ è finito	$ \text{dom}(\varphi_i)  > i$
$\varphi_i(0) \uparrow$	$\varphi_i(i) \uparrow$
$\exists n, \varphi_i(n) \downarrow \wedge \varphi_i(n+1) \downarrow$	$\varphi_i \equiv \varphi_{i+1}$

**Remark:** Il complementare di un insieme estensionale è estensionale

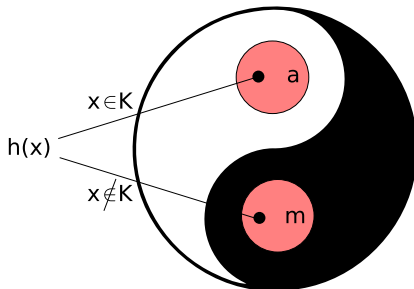
# Il teorema di Rice

Rice 1953

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

Sia  $c$  la funzione caratteristica del predicato. Sia  $m$  un indice per la funzione ovunque divergente, e sia  $a$  tale  $c(a) \neq c(m)$ . Cerco  $h$  calcolabile tale che

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



# Definizione di $h$

Consideriamo la funzione

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

dove “;” denota la composizione sequenziale.  
Per la proprietà smn  $h$  è totale e calcolabile.

È banale verificare che

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

Dunque, utilizzando l'ipotesi di estensionalità, avremmo

$$c(h(x)) = \begin{cases} c(a) & \text{if } x \in K \\ c(m) & \text{if } x \notin K \end{cases}$$

che permetterebbe di decidere l'appartenenza a  $K$ .

# 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

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

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

# Monotonia e compattezza

Sia  $A$  un insieme estensionale (rispetto a  $\varphi$ ) di numeri naturali

- $A$  è detto **monotono** se per ogni  $i$  e  $j$

$$i \in A \wedge \varphi_i \subseteq \varphi_j \rightarrow j \in A$$

- $A$  è detto **compatto** se per ogni  $i \in A$  esiste  $j \in A$  tale che

1. il grafo di  $\varphi_j$  è finito
2.  $\varphi_j \subseteq \varphi_i$

<i>insieme</i>	<i>monotonia</i>	<i>compattezza</i>
$\{ i \mid \varphi_i(0) \downarrow \}$	si	si
$\{ i \mid \varphi_i \text{ è totale} \}$	si	no
$\{ i \mid \text{cod}(\varphi_i) \text{ è finito} \}$	no	si
$\{ i \mid \text{dom}(\varphi_i) \text{ e } \overline{\text{dom}(\varphi_i)} \text{ sono infiniti} \}$	no	no

**Remark:** se  $A$  e  $\bar{A}$  sono entrambi monotoni allora sono banali.

# Rice-Shapiro (monotonia)

## Rice-Shapiro (monotonia)

Ogni insieme estensionale  $A$  ricorsivamente enumerabile é monotono.

Supponiamo che esistano due indici  $i$  e  $j$  tali che  $i \in A$ ,  $j \notin A$  and  $\varphi_i \leq \varphi_j$ .  
Consideriamo la funzione

$$\varphi_{f(x)}(y) = \varphi_i(y) \parallel (\varphi_x(x); \varphi_j(y))$$

dove " $\parallel$ " denota la composizione parallela (l'output é quello del primo thread terminante). É facile vedere che

$$\phi_{f(x)} \approx \begin{cases} \phi_j & \text{if } x \in K \\ \phi_i & \text{if } x \notin K \end{cases}$$

(nel caso  $x \in K$  uso l'ipotesi  $\varphi_i \leq \varphi_j$ ). Dunque

$$f(x) \in A \Leftrightarrow x \in \overline{K}$$

e  $\overline{K}$  sarebbe r.e., il che é assurdo.

# Rice-Shapiro (compattezza)

## Rice-Shapiro (compattezza)

Ogni insieme estensionale  $A$  ricorsivamente enumerabile é compatto.

Sia  $A$  un insieme estensionale ricorsivamente enumerabile. Supponiamo che  $i \in A$  e che per ogni  $j$  tale che  $\varphi_j \subseteq \varphi_i$  e  $\varphi_j$  è finito si abbia  $j \notin A$ .

Consideriamo la funzione  $f$  totale calcolabile definita come segue (per smn)

$$\varphi_{f(x)}(y) = \begin{cases} \uparrow & \text{if } \varphi_x(x) \downarrow \text{ in meno di } y \text{ passi} \\ \varphi_i(y) & \text{otherwise} \end{cases}$$

Se  $x \in \overline{K}$  allora  $\varphi_{f(x)} \approx \varphi_i$  e dunque  $f(x) \in A$ .

Se  $x \in K$  allora la computazione di  $\varphi_x(x)$  terminerà in un numero finito  $t$  di passi, e la funzione  $\varphi_{f(x)}$  convergerà solo per valori di input  $y \leq t$ .

Dunque  $f(x)$  è un indice per una sottofunzione finita di  $\varphi_i$ , e per ipotesi  $f(x) \notin A$ .

In conclusione

$$f(x) \in A \Leftrightarrow x \in \overline{K}$$

e  $\overline{K}$  sarebbe r.e., il che é assurdo.

I teoremi di Rice-Shapiro permettono di dimostrare facilmente che determinati insiemi estensionali **non sono** r.e. Ad esempio:

$\{ i \mid \varphi_i \text{ è totale} \}$  non è r.e. in quanto non è compatto

$\{ i \mid \text{cod}(\varphi_i) \text{ è finito} \}$  non è r.e. in quanto non è monotono

**Warning:** Esistono insiemi estensionali monotoni e compatti ma non r.e., ad esempio  $\{ i \mid \text{dom}(\varphi_i) \cap \overline{K} \neq \emptyset \}$ .



# Il teorema del punto fisso di Kleene - Lezione 14

---

- ▶ primo teorema di ricorsione
- ▶ secondo teorema di ricorsione
- ▶ applicazioni

# Il teorema del punto fisso di Kleene

## Teorema del punto fisso (Kleene)

Per ogni funzione totale calcolabile  $f$  esiste  $m$  tale che

$$\varphi_{f(m)} \approx \varphi_m$$

Per  $smn$  esiste  $h$  totale e calcolabile tale che

$$\varphi_{h(x)}(y) = g(x, y) = \varphi_{f(\varphi_x(x))}(y)$$

Sia  $p$  un indice per  $h$  e poniamo  $m = \varphi_p(p) = h(p)$  (che è sicuramente definito in quanto  $h$  è totale). Allora, per ogni  $y$

$$\varphi_m(y) = \varphi_{h(p)}(y) = g(p, y) = \varphi_{f(\varphi_p(p))}(y) = \varphi_{f(m)}(y)$$

Ogni funzione ricorsiva è il punto fisso di un opportuno funzionale (funzione tra funzioni).

Esempio:

$$\text{fact}(x) = \text{if } x==0 \text{ then } 1 \text{ else } x*\text{fact}(x-1)$$

usando la lambda notazione

$$\text{fact} = \text{lambda } x: \text{if } x==0 \text{ then } 1 \text{ else } x*\text{fact}(x-1)$$

e dunque posto

$$F(g) = \text{lambda } x: \text{if } x==0 \text{ then } 1 \text{ else } x*g(x-1)$$

abbiamo che

$$\text{fact} = F(\text{fact})$$

Se  $F$  è algoritmica, il teorema di Kleene assicura l'esistenza del punto fisso.

## Punti fissi e ricorsione (2)

---

Voglio cercare

$$\varphi_m = F(\varphi_m)$$

Se  $F$  è algoritmica, posso calcolare effettivamente l'indice della funzione trasformata, cioè esiste  $f$  totale e calcolabile tale che

$$\varphi_{f(i)} = F(\varphi_i)$$

Preso dunque un punto fisso di  $f$ ,

$$\varphi_m = \varphi_{f(m)} = F(\varphi_m)$$

**Remark** Il teorema del punto fisso richiede solo smn e interprete:

**L'interprete permette di simulare la ricorsione!**

# Applicazioni del teorema del punto fisso

L'uso del teorema del punto fisso per simulare ricorsione è particolarmente pulito, in quanto la funzione  $f$  che “implementa”  $F$  è estensionale, nel senso che

$$\varphi_i \equiv \varphi_j \Rightarrow \varphi_{g(i)} \equiv \varphi_{g(j)}$$

Tuttavia il teorema è vero per **qualunque trasformazione effettiva**.

Ad esempio:

- ▶ in ogni enumerazione accettabile di programmi esistono sicuramente due programmi consecutivi con comportamenti identici, ovvero esiste  $i$  tale che

$$\varphi_{i+1} \equiv \varphi_i$$

*Dimostrazione:* si prenda il punto fisso del successore.

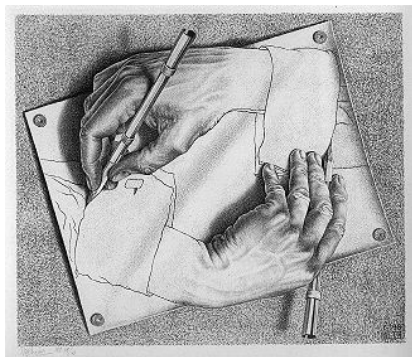
- ▶ Esiste un programma che “stampa se stesso”, cioè esiste  $i$  tale che

$$\varphi_i(0) = i$$

*Dimostrazione:* Per smn esiste  $h$  tale che  $\varphi_{h(x)}(y) = x$ ; se ne prenda un punto fisso.

# Programmi che stampano se stessi

---



Famoso esercizio di programmazione: **Quine**

# Il secondo teorema di ricorsione

## Secondo Teorema di ricorsione di Kleene

Per ogni funzione binaria totale calcolabile  $f$  esiste una funzione calcolabile  $s$  tale che, per ogni  $y$

$$\varphi_{f(s(y),y)} \approx \varphi_{s(y)}$$

Per smn esistono  $r, h$  totali e calcolabili tale che

$$\varphi_{\varphi_{r(y)}(x)}(z) = \varphi_{h(x,y)}(z) = g(x, y, z) = \varphi_{f(\varphi_x(x),y)}(z)$$

Posto  $s(y) = \varphi_{r(y)}(r(y))$  abbiamo, per ogni  $z$

$$\varphi_{s(y)}(z) = \varphi_{\varphi_{r(y)}(r(y))}(z) = \varphi_{h(r(y),y)}(z) = \varphi_{f(\varphi_{r(y)}(r(y)),y)}(z) = \varphi_{f(s(y),y)}(z)$$

# Dimostrazione alternativa del teorema di Rice

---

Supponiamo per assurdo che  $A$  sia ricorsivo, ma non banale.  
Esistono dunque  $i$  e  $j$  tali che  $i \in A$  e  $j \in \bar{A}$ .

Considero la seguente funzione:

$$h(x) = \begin{cases} 1 & \text{se } x \in \bar{A} \\ j & \text{se } x \in A \end{cases}$$

Per definizione

$$h(x) \in A \Leftrightarrow x \in \bar{A}$$

Inoltre, se  $A$  è ricorsivo  $h$  è totale calcolabile e per il teorema del punto fisso di Kleene, esiste un indice  $b$  tale che  $\varphi_b = \varphi_{h(b)}$ . Avremmo quindi

$$b \in A \Leftrightarrow h(b) \in A \Leftrightarrow b \in \bar{A}$$

che è una contraddizione.



- ▶ La nozione di  $(m)$ -riducibilità
- ▶  $m$ -completezza
- ▶ Insiemi produttivi e creativi
- ▶ Il problema di Post
- ▶ Insiemi immuni e semplici
- ▶ Complessità di Kolmogorov

# Definizione di riducibilità

Siano  $A, B \subseteq \mathcal{N}$ ;  $A$  si dice **riducibile** ( $m$ -riducibile) a  $B$  (in simboli  $A \leq_m B$ ), se esiste una funzione totale e calcolabile  $f$  tale che

$$x \in A \Leftrightarrow f(x) \in B$$

Due insiemi si dicono **equivalenti** ( $m$ -equivalenti) (in simboli  $A =_m B$ ), se  $A \leq_m B$  e  $B \leq_m A$ ;

## Osservazioni:

- ▶ la relazione  $\leq_m$  è un preordine (i.e. è riflessiva e transitiva)
- ▶ la relazione  $=_m$  è una relazione di equivalenza
- ▶  $A \leq_m B$  se e solo se  $\overline{A} \leq_m \overline{B}$
- ▶ se  $A \leq_m B$  e  $B$  è ricorsivo (risp. r.e.) allora  $A$  è ricorsivo (risp. r.e.)

$$K_0 =_m K$$


---

Sia  $K_0 = \{\langle i, n \rangle \mid n \in W_i\}$

►  $K \leq_m K_0$ . Siccome

$$i \in K \Rightarrow i \in W_i \Rightarrow \langle i, i \rangle \in K_0$$

la funzione  $f(x) = \langle x, x \rangle$  permette di ridurre  $K$  a  $K_0$ .

►  $K_0 \leq_m K$ . Consideriamo la funzione totale calcolabile  $h$  per cui

$$\varphi_{h(i,x)}(y) = g(i, x, y) = \varphi_i(x)$$

Abbiamo

$$\langle i, n \rangle \in K_0 \Leftrightarrow n \in W_i \Leftrightarrow \forall y, \varphi_{h(i,n)}(y) \downarrow \Leftrightarrow \varphi_{h(i,n)}h(i, n) \downarrow \Leftrightarrow h(i, n) \in K$$

Quindi la funzione  $h$  riduce  $K_0$  a  $K$ .

Un insieme si dice **m-completo** se è r.e. ed ogni insieme r.e. è riducibile ad esso.

## Lemma

$K_0$  e  $K$  sono insiemi completi.

Dato che  $K_0 =_m K$  è sufficiente dimostrare la proprietà per  $K_0$ .

Abbiamo già dimostrato che se  $A \leq_m K_0$  allora  $A$  è r.e. e dunque esiste  $i$  tale che  $A = W_i$ . Allora, per ogni  $n$

$$n \in A \Leftrightarrow n \in W_i \Leftrightarrow \langle i, n \rangle \in K_0$$

## Lemma

$A$  è completo se e solo se  $A =_m K$ .

Se  $A =_m K$  allora  $A$  è r.e. e m-completo perché lo è  $K$ .

Viceversa se  $A$  è m-completo, allora è r.e. e per la completezza di  $K$ ,  $A \leq_m K$ ; inoltre, siccome  $K$  è r.e.,  $K \leq_m A$  per la m-completezza di  $A$ .

# Insiemi produttivi e creativi

Sia  $A \subseteq \mathcal{N}$ .

1.  $A$  si dice **produttivo** se esiste  $f$  totale e calcolabile tale che per ogni  $i$

$$W_i \subseteq A \Rightarrow f(i) \in A \setminus W_i$$

2.  $A$  si dice **creativo** se è r.e. ed il suo complemento  $\bar{A}$  è produttivo.

Si osservi che un insieme produttivo non può essere r.e. Infatti, se  $A = W_i$  allora preso  $W_i \subseteq A$  avremmo che  $A \setminus W_i = \emptyset$  e quindi  $f(i) \notin A \setminus W_i$ .

## Teorema

$K$  è creativo (e la funzione di produzione è l'identità).

Sappiamo che  $K$  è r.e.; dimostriamo che  $\bar{K}$  è produttivo, ed in particolare che

$$W_i \subseteq \bar{K} \Rightarrow i \in \bar{K} \setminus W_i$$

- ▶  $i \in \bar{K}$ . Infatti, se  $i \in K$  allora  $i \in W_i$  e siccome  $W_i \subseteq \bar{K}$  avremmo  $i \in \bar{K}$  che è una contraddizione. Dunque  $i \in \bar{K}$ .
- ▶  $i \notin W_i$ . Infatti, se  $i \in W_i$  allora  $i$  dovrebbe appartenere a  $K$ , ma abbiamo appena dimostrato il contrario.

## Teorema

Sia  $A \subseteq \mathcal{N}$ .  $A$  è produttivo se e solo se  $\overline{K} \leq_m A$ .

( $\Leftarrow$ ) Supponiamo che  $\overline{K} \leq_m A$ , e sia  $g$  la corrispondente funzione di riduzione. Per smn esiste  $h$  totale e calcolabile tale che  $\varphi_{h(i)}(x) = \varphi_i(g(x))$ . Dunque

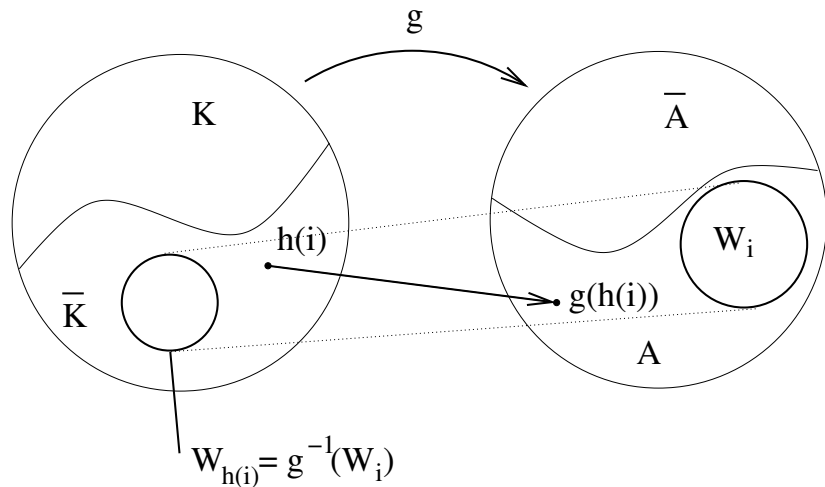
$$W_{h(i)} = g^{-1}(W_i)$$

Posto  $f(i) = g(h(i))$  vogliamo dimostrare che  $f$  è una funzione di produzione per  $A$ , ovvero che per ogni  $i$

$$W_i \subseteq A \Rightarrow f(i) \in A \setminus W_i$$

Se  $W_i \subseteq A$ ,  $W_{h(i)} = g^{-1}(W_i) \subseteq \overline{K}$  e per la produttività di  $\overline{K}$ ,  $h(i) \in \overline{K} \setminus W_{h(i)}$ . Ma allora  $f(i) = g(h(i)) \in A \setminus W_i$ .

# Caratterizzazione della produttività



# Caratterizzazione della produttività

( $\Rightarrow$ ) Sia  $A$  produttivo, e sia  $f$  la funzione di produzione. Consideriamo  $h$  t.c.

$$\varphi_{h(z,y)}(n) = \begin{cases} 0 & \text{se } f(z) = n \wedge y \in K \\ \uparrow & \text{altrimenti} \end{cases}$$

Per il secondo teorema di ricorsione esiste  $s$  totale e calcolabile tale che

$$\varphi_{s(y)}(n) = \varphi_{h(s(y),y)}(n) \begin{cases} 0 & \text{se } f(s(y)) = n \wedge y \in K \\ \uparrow & \text{altrimenti} \end{cases}$$

e dunque

$$W_{s(y)} = \begin{cases} \{f(s(y))\} & \text{se } y \in K \\ \emptyset & \text{altrimenti} \end{cases}$$

Vogliamo dimostrare che  $f \circ s$  è una funzione di riduzione da  $K$  a  $\bar{A}$ :

- ▶ Se  $y \in K$  allora  $W_{s(y)} = \{f(s(y))\}$ . Se  $f(s(y)) \in A$  allora  $W_{s(y)} \subseteq A$  e quindi per la produttività di  $A$ ,  $f(s(y)) \in A \setminus W_{s(y)}$ , ed in particolare  $f(s(y)) \notin W_{s(y)} = \{f(s(y))\}$  che è assurdo. Dunque  $f(s(y)) \in \bar{A}$ .
- ▶ Se  $y \notin K$ , allora  $W_{s(y)} = \emptyset \subseteq A$  e dunque, per la produttività di  $A$ ,  $f(s(y)) \in A \setminus W_{s(y)}$ , ed in particolare  $f(s(y)) \in A$ .



## Teorema

Sia  $A \subseteq \mathcal{N}$ .  $A$  è creativo se e solo se  $A =_m K$ .

Per definizione,  $A$  è creativo se e solo se è r.e. e  $\bar{A}$  è produttivo.

Ma  $A$  è r.e se e solo se  $A \leq_m K$ .

Inoltre, per il teorema precedente,  $\bar{A}$  è produttivo se e solo se  $\bar{K} \leq_m \bar{A}$ , che equivale a  $K \leq_m A$ .

## Il problema di Post

la riduzione di  $K$  ad  $A$  è una **tecnica generale** per dimostrare l'indcidibilità di  $A$ ? (almeno limitatamente agli insiemi r.e.)

Ovvero,

per ogni  $A$  r.e. non ricorsivo vale  $A =_m K$ ?

Equivalentemente, ci possiamo chiedere se ogni insieme r.e.  $A$  non ricorsivo è creativo (o completo).

## Lemma

*Ogni insieme produttivo  $A$  contiene un sottoinsieme infinito r.e.*

Sia  $f$  una funzione di produzione per  $A$ . Per smn esiste  $r$  totale e calcolabile tale che, per ogni  $i$ :

$$W_{r(i)} = W_i \cup \{f(i)\}$$

Si noti che siccome  $f(i) \notin W_i$ ,  $W_{r(i)}$  estende strettamente  $W_i$ . Inoltre, se  $W_i \subseteq A$  anche  $W_{r(i)} \subseteq A$ . Preso dunque  $m$  tale che  $W_m = \emptyset$ , l'unione

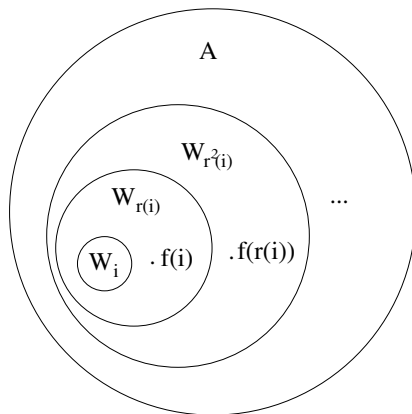
$$\bigcup_{n \in \mathcal{N}} W_{r^n(m)}$$

è infinita, r.e. e contenuta in  $A$ .

Il risultato precedente motiva la seguente **definizione**: sia  $A \subseteq \mathcal{N}$ .

- ▶  $A$  si dice **immune** se è infinito e nessun suo sottoinsieme infinito è r.e.
- ▶  $A$  si dice **semplice** se è r.e. e  $\bar{A}$  è immune.

# Illustrazione del risultato precedente



La funzione di produzione  $f$  per l'insieme produttivo  $A$  permette di *costruirne* una approssimazione infinita.

Andiamo alla ricerca di un insieme semplice (dunque non creativo).

La **complessità di Kolmogòrov** di un numero  $n$ , indicata con  $k(n)$  è il più piccolo indice  $i$  tale che  $\varphi_i(0) = n$ .

**Intuizione:** invece di dare una descrizione esplicita del numero  $n$ , si può fornire una procedura  $i$  che permetta di generarlo.

## Lemma

Per ogni  $i$ , se  $\varphi_i(0) \downarrow$ ,  $k(\varphi_i(0)) \leq i$ .

Ovvio, per la definizione di  $k$ .

# Numeri random

Un numero  $n$  si dice **random** se  $n \leq k(n)$ .  
Indicheremo con  $\mathcal{R}$  l'insieme dei numeri random.

**Intuizione:** un numero random è un numero di cui non è possibile fornire una descrizione più succinta: lui stesso è la sua descrizione minima.

## Lemma

$\overline{\mathcal{R}} \neq \emptyset$ .

Utilizzando il teorema del punto fisso si dimostra l'esistenza di un  $i$  tale che  $\varphi_i(0) = i + 1$ . Dunque  $i + 1$  non è random.

## Lemma

L'insieme  $\overline{\mathcal{R}}$  è r.e.

Un numero  $a$  non è random se e solo se esiste un indice  $i < a$  tale che  $\varphi_i(0) = a$ . Dunque  $\overline{\mathcal{R}}$  è codominio della funzione parziale calcolabile

$$g(i) = \begin{cases} \varphi_i(0) & \text{se } i < \varphi_i(0) \\ \uparrow & \text{altrimenti} \end{cases}$$

## Teorema

L'insieme  $R$  dei numeri random è immune.

Sia  $A$  un insieme infinito r.e. e sia  $f$  una sua funzione di enumerazione.

Per smn esiste  $h$  totale e calcolabile tale che

$$\varphi_{h(i)}(x) = f(\mu n. f(n) > i)$$

Si osservi che la minimizzazione  $\mu n. f(n) > i$  termina per ogni  $i$  in quanto  $f$  è totale e  $\text{cod}(f) = A$  è infinito. Per definizione,  $f(\mu n. f(n) > i) \in A$ , e se supponiamo  $A \subseteq R$  deve trattarsi di un numero random; inoltre

$$(*) \quad \varphi_{h(i)}(x) = f(\mu n. f(n) > i) > i$$

Per il teorema del punto fisso, esiste  $m$  tale che  $\varphi_m \approx \varphi_{h(m)}$ . Dunque avremmo

- ▶  $\varphi_m(0) = \varphi_{h(m)}(0) \in R$ , che implica  $\varphi_m(0) \leq k(\varphi_m(0))$ ; inoltre abbiamo dimostrato che  $k(\varphi_m(0)) \leq m$ , e per transitività  $\varphi_m(0) \leq m$ .
- ▶ d'altra parte, per  $(*)$ ,  $\varphi_m(0) = \varphi_{h(m)}(0) > m$ , che porta a contraddizione.

- ▶ Aritmetica del prim'ordine
- ▶ Insiemi e funzioni aritmetiche
- ▶ Indecidibilità della verità aritmetica
- ▶ il Teorema di incompletezza di Gödel



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  $A$ .

Ad esempio, l'insieme dei numeri primi è aritmetico, in quanto può essere descritto dalla seguente formula aritmetica:

$$\psi(n) = 2 \leq n \wedge \forall y, 1 < y \wedge y < n \Rightarrow \forall z, y \cdot z \neq n$$

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	$y = x_1 \cdot x_2$
$\pi_i^k$	$k$	$y = x_k$

## Lemma

*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

$$\psi_f(x, z) = \exists y, \psi_g(x, y) \wedge \psi_h(y, z)$$

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

## Lemma

*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:

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

# Le funzioni calcolabili sono aritmetiche

---

## Teorema

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 minimizzazione.

# Gli insiemi r.e. sono aritmetici

## Teorema

Ogni insieme ricorsivamente enumerabile è aritmetico.

Se  $A$  è r.e. esiste  $f$  (parziale) 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.

## Teorema

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

Sia  $\{\psi_n\}_{n \in \mathbb{N}}$  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.

## Teorema di incompletezza di Gödel

Ogni sistema formale aritmetico, se consistente, è incompleto (i.e. 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.



- ▶ Le classi  $\Sigma_n^0, \Pi_n^0, \Delta_n^0$
- ▶ Operazioni sui quantificatori
- ▶ Insiemi completi
- ▶ Il teorema della gerarchia aritmetica

# Le classi $\Sigma_n^0, \Pi_n^0, \Delta_n^0$

Sia  $n$  un numero naturale.

- ▶  $\Sigma_n^0$  è la classe di tutti gli insiemi  $A \subseteq \mathcal{N}^k$  per cui esiste un insieme **ricorsivo**  $R \subseteq \mathcal{N}^{n+k}$  tale che

$$A = \{x \in \mathcal{N}^k \mid \exists y_1 \forall y_2 \exists y_3 \dots Q y_n, (x, y_1, y_2, \dots, y_n) \in R\}$$

dove  $Q = \forall$  per  $n$  pari, e  $Q = \exists$  per  $n$  dispari.

- ▶  $\Pi_n^0$  è la classe di tutti gli insiemi  $A \subseteq \mathcal{N}^k$  per cui esiste un insieme **ricorsivo**  $R \subseteq \mathcal{N}^{n+k}$  tale che

$$A = \{x \in \mathcal{N}^k \mid \forall y_1 \exists y_2 \forall y_3 \dots Q y_n, (x, y_1, y_2, \dots, y_n) \in R\}$$

dove  $Q = \exists$  per  $n$  pari, e  $Q = \forall$  per  $n$  dispari.

- ▶  $\Delta_n^0 = \Sigma_n^0 \cap \Pi_n^0$ .
- ▶  $\bigcup_{n \in \mathcal{N}} (\Sigma_n^0 \cup \Pi_n^0)$  è detta **gerarchia aritmetica**.

# Semplici proprietà della gerarchia aritmetica

Sia  $n$  un numero naturale:

- ▶ La classe  $\Sigma_0^0 = \Pi_0^0 = \Delta_0^0$  è la classe degli **insiemi ricorsivi**.
- ▶ La classe  $\Sigma_{n+1}^0$  è la classe delle **proiezioni** di insiemi in  $\Pi_n^0$ .
- ▶ La classe  $\Pi_n^0$  è la classe dei **complementi** di insiemi in  $\Sigma_n^0$ .
- ▶ In particolare,  $\Sigma_1^0$  è la classe degli **insiemi r.e** e  $\Pi_1^0$  è la classe degli **insiemi co-r.e.**
- ▶  $\Delta_1^0 = \Delta_0^0$  è la classe degli **insiemi ricorsivi**.
- ▶  $\Sigma_n^0 \cup \Pi_n^0 \subseteq \Sigma_{n+1}^0 \cap \Pi_{n+1}^0 = \Delta_{n+1}^0$

Dimostreremo in seguito che l'ultima inclusione è stretta.

# Esempio: il problema dell'equivalenza

$$\text{Equ} = \{(i, j) \in N \mid \varphi_i = \varphi_j\} \in \Pi_2^0$$

Infatti

$$\begin{aligned}(i, j) \in \text{Equ} &\Leftrightarrow \forall n((\varphi_i(n) \downarrow \wedge \varphi_j(n) \downarrow \wedge \varphi_i(n) = \varphi_j(n))) \\ &\quad \vee (\varphi_i(n) \uparrow \wedge \varphi_j(n) \uparrow) \\ &\Leftrightarrow \forall n((\exists t_1 t_2 k (T(i, n, t_1, k) \wedge T(j, n, t_2, k)) \\ &\quad \vee \forall t_1 t_2 k_1 k_2 (\neg T(i, n, t_1, k_1) \wedge \neg T(j, n, t_2, k_2))) \\ &\Leftrightarrow \forall n s_1 s_2 k_1 k_2 \exists t_1 t_2 k ((T(i, n, t_1, k) \wedge T(j, n, t_2, k)) \\ &\quad \vee (\neg T(i, n, s_1, k_1) \wedge \neg T(j, n, s_2, k_2)))\end{aligned}$$

**N.B.** Sequenze di quantificatori dello stesso tipo collassano in un solo quantificatore.

Poichè la matrice è ricorsiva, questo dimostra che Equ è (al più) in  $\Pi_2^0$ .  
Possiamo dare una classificazione più stretta?

## Il Teorema di Kuratowski

Ogni formula aritmetica è equivalente ad una con un prefisso formato da una lista alternata di quantificatori e una matrice ricorsiva.

La prova si basa sulle seguenti trasformazioni (si suppone che  $x$  non occorra libera in  $\beta$ ):

$$\begin{array}{ll} \neg(\exists x\alpha) \Leftrightarrow \forall x\neg\alpha & \neg(\forall x\alpha) \Leftrightarrow \exists x\neg\alpha \\ (\exists x\alpha) \wedge \beta \Leftrightarrow \exists x(\alpha \wedge \beta) & (\forall x\alpha) \wedge \beta \Leftrightarrow \forall x(\alpha \wedge \beta) \\ (\exists x\alpha) \vee \beta \Leftrightarrow \exists x(\alpha \vee \beta) & (\forall x\alpha) \vee \beta \Leftrightarrow \forall x(\alpha \vee \beta) \\ (\exists x\alpha) \rightarrow \beta \Leftrightarrow \forall x(\alpha \rightarrow \beta) & (\forall x\alpha) \rightarrow \beta \Leftrightarrow \exists x(\alpha \rightarrow \beta) \\ \beta \rightarrow (\exists x\alpha) \Leftrightarrow \exists x(\beta \rightarrow \alpha) & \beta \rightarrow \forall x\alpha \Leftrightarrow \forall x(\beta \rightarrow \alpha) \end{array}$$

## Lemma

*Due quantificatori di cui il primo sia limitato possono sempre essere permutati.*

$$\begin{aligned}\forall x \leq a \exists y R(x, y) &\Leftrightarrow \exists w \forall x \leq a R(x, \text{nth}(x, w)) \\ \exists x \leq a \forall y \neg R(x, y) &\Leftrightarrow \forall w \exists x \leq a \neg R(x, \text{nth}(x, w))\end{aligned}$$

dove  $\text{nth}(x, w)$  è la funzione ricorsiva che estrae l'ennesimo elemento da una tupla:

$$\begin{cases} \text{nth}(0, w) = w \\ \text{nth}(n+1, \langle y, z \rangle) = \text{nth}(n, z) \end{cases}$$

Quindi i quantificatori limitati possono essere spinti verso l'interno e assorbiti nella matrice ricorsiva:

$\forall$  e  $\exists$  **limitati** non influiscono sulla classificazione nella gerarchia aritmetica

# La gerarchia aritmetica e gli insiemi aritmetici

## Teorema

La gerarchia aritmetica contiene esattamente gli insiemi aritmetici.

Gli insiemi ricorsivi sono aritmetici (in quanto lo sono quelli r.e.). Gli insiemi della gerarchia aritmetica sono costruiti da insiemi ricorsivi utilizzando un insieme finito di quantificatori, e dunque sono tutti aritmetici.

Viceversa, la descrizione di ogni insieme aritmetico  $A$  può essere trasformata in forma normale prenessa, dove la matrice è una proposizione priva di quantificatori in cui compare solo il predicato (decidibile) di uguaglianza, ed è dunque ricorsiva. Quindi  $A$  appartiene alla gerarchia aritmetica.

# Il teorema di enumerazione

## Teorema

Per ogni  $n, m \geq 1$  esiste un insieme  $U_n^m \in \Sigma_n^0 \subseteq \mathcal{N}^{m+1}$  che enumera tutti gli insiemi  $A \in \Sigma_n^0 \subseteq \mathcal{N}^m$ , ovvero tale che  $A(\vec{x}) \Leftrightarrow U(e, \vec{x})$  per qualche  $e$ .  
Similmente per  $\Pi_n^0$ .

Per induzione su  $n$ . Se  $n = 1$

$$U_1^m(i, \vec{x}) \Leftrightarrow \vec{x} \in W_i \in \Sigma_1^0$$

enumera tutti gli insiemi r.e.; inoltre

$$\overline{U_1^m}(i, \vec{x}) \Leftrightarrow \vec{x} \in \overline{W_i} \in \Pi_1^0$$

enumera tutti gli insiemi co-r.e.

Induttivamente, sia data  $U_n^{m+1} \in \Sigma_n^0$ . Posto

$$U_{n+1}^m(e, \vec{x}) \Leftrightarrow \forall y U_n^{m+1}(e, \vec{x}, y) \in \Pi_{n+1}^0$$

$U_{n+1}^m$  enumera tutti i sottoinsiemi di  $\mathcal{N}^m$  in  $\Pi_{n+1}^0$  e  $\overline{U_{n+1}^m}$  quelli in  $\Sigma_{n+1}^0$ .



# Completezza nella gerarchia aritmetica

Sia  $A \subseteq \mathcal{N}$  e sia  $n \in \mathcal{N}$

- ▶  $A$  è detto  $\Sigma_n^0$ -completo se  $A \in \Sigma_n^0$  e  $B \leq_m A$  per ogni  $B \in \Sigma_n^0 \subseteq \mathcal{N}$
- ▶  $A$  è detto  $\Pi_n^0$ -completo se  $A \in \Pi_n^0$  e  $B \leq_m A$  per ogni  $B \in \Pi_n^0 \subseteq \mathcal{N}$

## Teorema

Per ogni  $n$ , sia

$$K_0^n = \{\langle i, x \rangle \mid U_n^1(i, x)\}$$

Allora  $K_0^n$  è  $\Sigma_n^0$ -completo (e  $\overline{K_0^n}$  è  $\Pi_n^0$ -completo).

Sia  $B \in \Sigma_n^0 \subseteq \mathcal{N}$ . Per il teorema di enumerazione deve esiste  $e$  tale che

$$B(x) \Leftrightarrow U_n^1(e, x) \Leftrightarrow K_0^n(\langle e, x \rangle)$$

e dunque la funzione  $f(x) = \langle e, x \rangle$  riduce  $B$  a  $K_0^n$ .

# Il teorema della gerarchia aritmetica

## Teorema della gerarchia

Per ogni  $n$

1.  $\Sigma_n^0 \setminus \Pi_n^0 \neq \emptyset$  (e quindi  $\Delta_n^0 \subset \Sigma_n^0$ )
2.  $\Pi_n^0 \setminus \Sigma_n^0 \neq \emptyset$  (e quindi  $\Delta_n^0 \subset \Pi_n^0$ )
3.  $\Sigma_n^0 \cup \Pi_n^0 \subset \Delta_{n+1}^0$

1. supponiamo che  $U_n(i, x)$  enumeri tutte le relazioni in  $\Sigma_n^0$  e consideriamo l'insieme  $K_n(x) = U_n(x, x)$ .  $K_n \in \Sigma_n^0$ ; se  $K_n \in \Pi_n^0$  allora  $\overline{K_n} \in \Sigma_n^0$  e dovrebbe esistere  $e$  tale che  $\overline{K_n}(e) = U_n(e, e)$ . Ma allora otterremmo la seguente contraddizione:

$$\overline{K_n}(e) = U_n(e, e) = K_n(e)$$

2. se  $A \in \Sigma_n^0 \setminus \Pi_n^0$ , allora  $\overline{A} \in \Pi_n^0 \setminus \Sigma_n^0$
3. sia  $A \in \Sigma_n^0 \setminus \Pi_n^0$  e consideriamo l'insieme

$$Q(x, z) = (A(x) \wedge z = 0) \vee (\overline{A}(x) \wedge z = 1)$$

Chiaramente  $Q \in \Delta_{n+1}^0$ . Tuttavia  $Q \notin \Pi_n^0$  poichè altrimenti lo sarebbe anche  $A(x) = Q(x, 0)$  e allo stesso modo  $Q \notin \Sigma_n^0$ .

# La classificazione di alcuni problemi indecidibili

---

- ▶  $K_1 = \{i \mid W_i \neq \emptyset\}$  è  $\Sigma_1^0$ -completo
- ▶  $\text{Fin} = \{i \mid W_i \text{ è finito}\}$  è  $\Sigma_2^0$ -completo
- ▶  $\text{Inf} = \{i \mid W_i \text{ è infinito}\}$  è  $\Pi_2^0$ -completo
- ▶  $\text{Tot} = \{i \mid W_i = \mathcal{N}\}$  è  $\Pi_2^0$ -completo
- ▶  $\text{Equ} = \{\langle i, j \rangle \mid \varphi_i = \varphi_j\}$  è  $\Pi_2^0$ -completo
- ▶  $\text{Cof} = \{i \mid W_i \text{ è cofinito}\}$  è  $\Sigma_3^0$ -completo
- ▶  $\text{Rec} = \{i \mid W_i \text{ è ricorsivo}\}$  è  $\Sigma_3^0$ -completo
- ▶  $\text{Creative} = \{i \mid W_i \text{ è creativo}\}$  è  $\Sigma_3^0$ -completo
- ▶  $\text{Simple} = \{i \mid W_i \text{ è semplice}\}$  è  $\Pi_3^0$ -completo
- ▶ ...

# Prova di completezza per il problema dell'equivalenza

Abbiamo già dimostrato che  $\text{Equ} \in \Pi_2^0$ . Dobbiamo dimostrare che ogni  $A \in \Pi_2^0$  è riducibile a  $\text{Equ}$ . Poichè  $A \in \Pi_2^0$  deve esistere  $B$  ricorsivo tale che

$$A = \{n \mid \forall x \exists y B(x, y, n)\}$$

Consideriamo la funzione totale calcolabile  $h$  tale che

$$\varphi_{h(n)}(x) = g(n, x) = 0 \cdot \mu_y(B(x, y, n))$$

Sia inoltre  $m$  un indice per la funzione costante 0, e confrontiamo  $h(n)$  con  $m$ :

$$\begin{aligned}\langle h(n), m \rangle \in \text{Equ} &\Leftrightarrow \varphi_{h(n)} = \varphi_m \\ &\Leftrightarrow \forall x \varphi_{h(n)}(x) = 0 \\ &\Leftrightarrow \forall x \exists y B(x, y, n) \\ &\Leftrightarrow n \in A\end{aligned}$$

Dunque la funzione totale calcolabile  $s(n) = \langle h(n), m \rangle$  riduce  $A$  a  $\text{Equ}$ .

# Esempi di problemi completi nella gerarchia aritmetica

